МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Матєшов Владислав Валерійович

Програмне забезпечення системи моніторингу продуктивності IoT

Спеціальність: 123 «Комп'ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

Доренський Олександр Павлович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Матєшову Владиславу Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи моніторингу продуктивності IoT*

керівник роботи *Доренський Олександр Павлович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 204-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи моніторингу продуктивності IoT*

4. 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

5. 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*

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Матєшов В.В. Програмне забезпечення системи моніторингу продуктивності IoT. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи моніторингу продуктивності IoT.

Метою розробки є програмне забезпечення системи моніторингу продуктивності IoT.

Результат роботи – програмна реалізація системи моніторингу продуктивності IoT.

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 архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі RAD Studio Delphi 10.4 Sydney.

Ключові слова: комп'ютерна інженерія, моніторинг, продуктивність, IoT

ABSTRACT

Matieshov V.V. IoT performance monitoring system software. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software which is intended for system of monitoring of productivity of IoT is developed.

The purpose of the development is the software of the IoT performance monitoring system.

The result is a software implementation of the IoT performance monitoring system.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program was developed in the environment of RAD Studio Delphi 10.4 Sydney.

Keywords: computer engineering, monitoring, performance, IoT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІТ	–	Інформаційні технології
ЦОД	–	Центри обробки даних
ШІ	–	Штучний інтелект
IaaS	–	Інфраструктура як послуга
IoT	–	Інтернет речей
PaaS	–	Платформа як послуга
SaaS	–	Програмне забезпечення як послуга

ВСТУП

Актуальність теми. У найближче десятиліття завдяки інтернету речей (IoT) очікується сплеск обсягу даних, вироблених підприємствами. Це дуже вплине на ефективність і конкурентоспроможність бізнесу. Застосунки IoT відрізняються від звичайних одноадресних комунікацій, тому підприємствам необхідно відповісти на низку важливих питань, щоб зберегти бізнес в епоху інформаційної революції.

Інтернет речей називають третьою хвилею інформаційної революції. Другий було поширення смартфонів і мобільних комунікацій. Четвертою, імовірно, буде штучний інтелект (ШІ), він візьме на себе подальший розвиток інформаційного середовища, у яке будуть інтегруватися люди.

IoT – це не тільки мільярди речей, підключених до мережі, але й новий спосіб керування бізнес-процесами. Уже зараз технології інтернету речей одержали широке поширення. Наприклад, виробники авіадвигунів, будівельної техніки можуть стежити за роботою виробів у режимі реального часу, а керуючі компанії одержувати інформацію з показань лічильників жителів району. Інтернет речей генерує величезна кількість інформації й швидко росте. За прогнозами IDC, ринок IoT до 2022 р. досягнеться $7,1 трлн.

Випливаючи у фарватері третьої хвилі основне питання, яке турбує підприємства – це питання безпеки. Так, згідно з опитуваннями, проведеними компанією HP, уразливі приблизно 70% найбільше часто використовуваних пристроїв IoT.

Крім питання безпеки існує ще п'ять ключових факторів, які потрібно врахувати при адаптації системи моніторингу продуктивності під Інтернет речей:

- великі обсяги даних;
- нові пристрої й протоколи;
- пульсуючий трафік;

– більш серйозна залежність від хмари;

– збільшення потреби в розгортанні IPv6.

Нові вимоги до моніторингу продуктивності

Розгортання IoT може порушити поточний процес моніторингу продуктивності, тому необхідно випливати стратегії адаптації IT-інфраструктури під нові вимоги.

Підключені пристрої, такі як автомобілі й датчики, не викликають проблем окремо. Але всі разом вони генерують величезні обсяги даних і різко збільшують навантаження на мережу й системи, які їх контролюють.

Одна велика компанія зі списку Fortune 100 у цей час використовує підключені по IP датчики для відстеження енергоспоживання в центрах обробки даних (ЦОД). За допомогою мережі датчиків можна виявити неефективну витрату енергії, але для цього доводиться щодня аналізувати більш 25 млрд показників.

Тому в першу чергу слід мати у виді, що розвиток інтернету речей – це питання масштабу.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи моніторингу продуктивності IoT.

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

– Огляд існуючих систем моніторингу продуктивності IoT.

– Дослідження системи моніторингу продуктивності IoT.

– Програмна реалізація системи моніторингу продуктивності IoT.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу продуктивності IoT.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу продуктивності IoT, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

У найближчі роки мережна інфраструктура підприємств переживе стрибок виробництва даних. Якщо платформа для моніторингу продуктивності не зможе економічно ефективно масштабуватися при такому збільшенні обсягу даних, вона стане причиною ряду серйозних проблем.

Так, неможливо буде передбачити поломку будь-якого пристрою, не підключеного до IoT, що робить уразливим усе виробництво. Контроль процесів буде вестися з меншою частотою, а в ряді напрямків, наприклад контролі мережного трафіку, навіть опитування із частотою у хвилину є недостатнім. У підсумку, дані для звітності й аналізу стануть неточними.

Тому необхідно вибирати платформу моніторингу, засновану на моделі розподілених обчислень. Такі системи набагато краще підготовлені до роботи з великими обсягами даних, генеруємими IoT.

Звичайно під IT-інфраструктурою підприємства мають на увазі традиційний набір рішень, наприклад: сервери, маршрутизатори, комутатори, балансувальники навантаження, файєрволи і т.д. Помилково вважати, що IT-інфраструктурі підприємства не належать пристрою IoT, датчики електроніка, що й носиться. Усе, що підключається до мережі, є частиною корпоративної IT-інфраструктури.

Багато з об'єктів Інтернету речей уже є в мережі підприємства або сучасному ЦОД:

- датчики безпеки, відеокамери й електронні замки;
- системи екологічного моніторингу;
- розумні серверні стійки;
- електрогенератори й системи живлення;

– джерела безперебійного живлення;

– системи керування будинком.

Уже зараз у світі кількість активних бездротових IoT-пристроїв перевищує 16 млрд. Через три роки їх кількість подвоїться. Це означає, що стандартний інтернет-протокол SNMP для IP-мереж може не впоратися з навантаженням і забезпеченням захисту інформації.

У цей час багато виробників створюють нові протоколи, відмінні від SNMP. Імовірно, у найближчі роки на ринок вийдуть нові розв'язки для керування IoT.

Таким чином, при виборі рішення для моніторингу продуктивності необхідно знайти рішення, який використовує апаратно-незалежний підхід до збору даних.

Простіше говорячи, платформа моніторингу повинна вміти збирати будь-які дані з будь-якими тимчасовими інтервалами незалежно від джерела. Також важливо, щоб постачальник відповідних інструментів мав гарну репутацію в області роботи з даними, що надходять зі сторонніх платформ і систем керування елементами мережі.

Десятки мільярдів пристроїв і датчиків у різних куточках планети будуть генерувати інформаційні пакети спорадично. Це реалії Інтернету речей, які мають на увазі відмова від застарілого принципу відстеження мережного трафіку з інтервалами в п'ять хвилин. Так, при аналізі мережного трафіку з інтервалами 1,5 секунди кількість сплесків трафіку може бути на 250% вище, чим при інтервалі в одну хвилину.

Потрібно використовувати платформу моніторингу, яка в стані робити збір даних з кожної необхідною частотою. Для світу IoT це є обов'язковим.

До 2022 р. для підключення величезної кількості пристроїв Інтернету речей буде потрібно запускати приблизно 340 серверів у день, або 120 тис. у рік.

Для масштабних мереж Інтернету речей будуть потрібні хмарні обчислення, тому що в найближчі десятиліття тільки вони здатні забезпечити необхідну продуктивність за прийнятною ціною.

Тому першим питанням є готовність підприємства мігрувати в хмару й вибрати один зі шляхів:

- інфраструктура як послуга (IaaS);
- платформа як послуга (PaaS);
- програмне забезпечення як послуга (SaaS).

Міграція в хмару – комплексне завдання, яке включає забезпечення безпеки й моніторинг продуктивності, щоб переконатися в надійності нової IT-інфраструктури.

Таким чином, сучасна система моніторингу продуктивності повинна мати можливість здійснювати контроль якості сервісу й продуктивності гібридної IT-інфраструктури, здійснювати кореляцію даних із власних серверів і хмарних платформ, таких як AWS.

Інтернет речей стане головним драйвером впровадження протоколу IPv6. Застарілий протокол IPv4 здатний підтримувати до 4,3 млрд адрес, чого явно недостатньо для IoT.

Тому платформа для керування й моніторингу продуктивністю повинна бути повністю сумісна з IPv6, а не мати лише підтримку цього протоколу.

Інтернет речей зажадає від підприємства особливого підходу до керування IT-інфраструктурою. Розсилання новин, підкасти, RSS потоки й маркетингові матеріали можуть заплутати й «затопити» інформацією. Тому при виборі системи моніторингу продуктивності в епоху третьої хвилі інформаційної революції кожний повинен відповісти низка питань, на які ми постаралися зробити акцент у даному матеріалі.

Ефективна система керування й моніторингу продуктивності Інтернету речей повинна відповідати наступним вимогам:

- здатна впоратися зі швидким ростом трафіку й обсягів даних;

– може працювати одночасно із протоколами IPv4 і IPv6;

– можливість керування всіма новими пристроями, незалежно від стандарту зв'язку або метрик продуктивності;

– видимість трафіку в кожному сегменті з точністю до секунд;

– керування з одного екрана гібридним хмарним середовищем, що охоплює всі рівні моніторингу, включаючи фізичні, віртуальні, ключові індикатори продуктивності.

1.2 Область застосування

Як використовується Інтернет речей у галузях економіки:

Інтернет речей у сфері виробництва:

– Удосконалюйте процеси, використовуючи галузеві розв'язки Інтернету речей. Використовуйте датчики й просунуту аналітику для прогнозування необхідності обслуговування вузлів і деталей, скорочення часу незапланованих простоїв, що негативно позначаються на виробництві.

– Створюйте нові бізнес-моделі, що реалізують, що попереджає обслуговування й моніторинг продуктивності виробленого вами устаткування, забезпечуючи більш ефективну взаємодію із клієнтами.

– Забезпечте доступ до даних датчиків для поліпшення графіка виїзного обслуговування. Це гарантує своєчасний приїзд технічних фахівців і доставку інструментів, що запобіжить перетворенню потенційних проблем у серйозні збої.

Інтернет речей для розумних міст:

– Застосовуйте відстеження використання й інтелектуальні мережі для забезпечення надійної, ефективної й економічно безпечної передачі енергії при одночасному зниженні розцінок для клієнтів.

– Підключите інфраструктуру, щоб краще регулювати рух, підвищити ефективність аварійних систем і зменшити час реагування поліції й влади.

– Підключите пристрою й системи в будинках, щоб підвищити ефективність їх функціонування й керування.

– Підвищуйте ефективність роботи комунальних служб у всіх областях – від ремонту зламаних вуличних ліхтарів і регулювання світлофорів до оптимізації маршрутів мусоровозів.

Інтернет речей для транспорту:

– Підтримуйте дорожній рух, прогнозуючи й відслідковуючи потреби в обслуговуванні – усувайте потенційні проблеми під час планових простоїв, щоб забезпечити динаміку свого бізнесу.

– Створюйте інноваційні можливості й рішення для транспортних засобів з підтримкою Інтернету речей, що дозволяють трансформувати умови водіння для клієнтів і одержувати коштовну інформацію для вашого бізнесу.

– Поліпшуйте логістику за допомогою даних і оповіщень у реальному часі для оптимізації маршрутів доставки, контролю продуктивності й прискорення відгуку на затримки або проблеми при їхнім виникненні.

– Відслідковуйте й обробляйте в режимі реального часу дані дорожнього трафіку, щоб спростити керування транспортною інфраструктурою, оцінювати дорожні умови й усувати пробки.

Інтернет речей у сфері роздрібної торгівлі:

– Інтернет речей підприємств громадського харчування. Підвищуйте ефективність, щоб скоротити операційні витрати. Відслідковуйте якість і безпека, поліпшуйте обслуговування устаткування й відслідковуйте поставки.

– Інтернет речей для супермаркетів і роздрібних магазинів. Прискорюйте розвиток і ріст бізнесу й зміцнюйте лояльність до бренда. Відслідковуйте запаси, поведінку споживачів і рекомендуйте продукти.

– Інтернет речей для готельного господарства. Оптимізуйте операції й збільшуйте доходи. Відслідковуйте використання готельних номерів, ураховуйте переваги гостей і не допускайте нестачі використовуваних матеріалів.

– Інтернет речей для стадіонів. Поліпшуйте взаємодію із глядачами, створюючи умови, що дозволяють стежити за улюбленими спортивними подіями за допомогою інструментів збору статистики в реальному часі.

Інтернет речей у сфері охорони здоров'я:

– Забезпечте пацієнтам можливість одержувати допомогу вдома, де вони почувають себе найбільше комфортно. За допомогою датчиків, що носяться, лікарі можуть віддалено відслідковувати стан здоров'я пацієнтів і реагувати в режимі реального часу.

– Допоможіть вашим співробітникам витрачати менше часу на пошук і інші допоміжні роботи й більше часу проводити з пацієнтами завдяки поліпшенню відстеження медичних активів і керування ними – і все це на хмарній платформі, відповідної до вимог HIPAA.

– Забезпечте готовність критично важливих медичних пристроїв до використання тоді, коли ваші пацієнти потребують їх найбільше, завчасно усуваючи проблеми за допомогою обслуговування, що попереджає.

– Підвищуйте загальну якість життя пацієнтів, відслідковуючи, як використовується устаткування, використовуючи датчики лікарняних ліжок для моніторингу кімнатної температури й рукомийників.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу продуктивності IoT, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Програмний комплекс Sunvizion Service Fulfillment

Рішення Sunvizion Service Fulfillment пропонує виробництво послуг на базі каталогу готових компонентів, що підвищує якість надання послуг абонентам, скорочує час виходу на ринок, піднімає рівень задоволеності клієнтів. Архітектура Sunvizion Service Fulfillment поєднує внутрішньо інтегровані й готові до зовнішньої інтеграції модулі: модуль інвентаризації послуг (Service Inventory), модуль каталогу послуг (Service Catalogue), модуль потоку робіт (Workflow), модуль паспортизації мережі (network inventory), модуль робочих ресурсів (Workforce), модуль оркестрування (Provisioning Orchestration).

Програмний комплекс Sunvizion Service Fulfillment: характеристики

– Максимально швидке моделювання й впровадження нових послуг.

– Можливість автоматичної реконфігурації процесу створення послуги в ході його виконання.

– Відсутність необхідності регресійного тестування після впровадження нової послуги.

– Обмін даними між різними учасниками процесу виконання послуги: працівниками, що виконують поставлені завдання, зовнішніми виконавцями й зовнішніми системами, інтегрованими з Sunvizion Service Fulfillment.

– Інтуїтивно зрозумілий веб-портал для відображення потоку завдань на будь-якому пристрої.

– Видимість усіх параметрів послуг, як з погляду клієнтів, так і апаратного забезпечення.

– Реєстрація послуг, використовуваних клієнтом.

– Резервування мережних елементів, необхідних для надання послуг.

– Моніторинг мережних ресурсів для контролю стану елементів мережі й оптимізації планування мережі.

– Повністю автоматична активація й модифікація послуг.

ПЗ для автоматизації й документування вимірів високочастотних кабельних складань Rohde & Schwarz VNA Suite

Програмне забезпечення Rohde & Schwarz VNA Suite дозволяє проводити аналіз технологічних параметрів кабельних складань, автоматично формувати протоколи за результатами вимірів, а також вести базу даних вимірів з можливістю їх перегляду. Rohde & Schwarz VNA Suite уже впроваджене на одному з підприємств і активно використовується відділом технічного контролю. Програмне забезпечення Rohde & Schwarz VNA Suite доступно як у вигляді опції до векторних аналізаторів ланцюгів Rohde & Schwarz, так і у вигляді окремого програмного продукту. Рішення може бути дороблене під конкретні завдання замовника.

ПЗ для автоматизації й документування вимірів високочастотних кабельних складань Rohde & Schwarz VNA Suite: характеристики

– Вимір високочастотних кабельних складань у діапазоні від 10 МГц до 67 ГГц.

– Доступ до всіх проведених раніше вимірів у вигляді протоколів вимірів і touchstone файлів.

– Зниження помилок вимірів викликаних людським фактором.

– Скорочення середнього часу вимірів, а отже збільшення продуктивності праці відділів технічного контролю.

– Середній час виміру – менш хвилини.

Програмний комплекс для моніторингу роботи мережі

Моніторинг роботи мережі – спеціально розроблений застосунок, що дозволяє визначати погіршення параметрів роботи телекомунікаційної мережі й

контролювати якість надаваних послуг. Система поліпшує робочу ефективність і значно скорочує сукупну вартість володіння (ТСО).

Дане ПЗ здійснює моніторинг мережі за допомогою аналізу параметрів KPI, обумовлених користувачем. Детальна інформація про продуктивність, зібрана системою, дозволяє підвищити наочність мережної продуктивності й поліпшити якість надаваної послуги. Тому що застосунок забезпечує доступ до даних у режимі реального часу, співробітники можуть швидко відреагувати на які-небудь відхилення від нормальної роботи мережі. Моніторинг роботи мережі Sunvizion також допомагає ефективно коректувати мережні помилки й точніше планувати ємність мережі.

Моніторинг у реальному часі

– Здійснюється в реальному масштабі часу.

– Запуск алертів у реальному часі при зниженні продуктивності мережі.

– Гнучкість і висока масштабованість.

– Готовність до інтеграції зі сторонніми веб-сервісами.

– Сумісність із різними технологіями й протоколами (UMTS, EVDO, IP/MPLS, LTE, WiMAX).

Простий і гнучкий інструмент для створення звітів

– Простий огляд звітів про KPI.

– Гнучке й просте оформлення звітів про KPI за допомогою інструмента для малювання.

– Підтримка основних звітів, що описують поведінку мережі.

– Можливість кастомизації звітів зі специфічним KPI.

Багаторівневість

– Багаторівневий аналіз.

– Сходові діаграми (багатоступінчасті схеми) забезпечують наочна вистава інформації.

– Формувач KPI не залежить від мережної технології або типів вистави зовнішніх даних (XDR).

– Умови спрацьовування алертів визначаються користувачами.

– Механізми відстеження статусу процесів, зв'язаних зі списком завдань користувача.

Переваги

– Забезпечує мережний інтелект для працівників мережного операційного центру (МОЦ), планувальників ємності й виконавців.

– Поліпшена операційна ефективність.

– Швидке виявлення мережних проблем.

– Зниження загальної вартості володіння (ТСО).

– Детальна інформація про продуктивність, що враховує всі аспекти мережі;

– Підвищення якості послуг і користувацького досвіду.

Програмний комплекс для моніторингу роботи мережі: характеристики

– Моніторинг, повідомлення й повідомлення про помилки в режимі реального часу.

– Готові мульти-послуги, мульти-вендор, мульти-технології й мульти-протокол (UMTS, EVDO, IP/MPLS, LTE, WiMAX).

– Гнучкість і висока універсальність.

– Легка інтеграція з новими послугами.

ПЗ Sunvizion для надання бездротового доступу (OTA) для WiMAX

ПЗ Sunvizion для надання бездротового доступу (OTA) для WiMAX – це унікальний рішення, який дозволяє проводити продаж і повністю автоматизувати активацію послуг, наданих за допомогою WiMAX-мережі, а також проводити виписку рахунків за послуги, надані абонентам WiMAX-мережі. Даний застосунок підтримує обидва стандарти WiMAX: 802.16d і 802.16e. На додаток до базових послуг (протокол VoIP і Data), платформа також дозволяє реалізовувати продаж і активацію так званих додаткових послуг, які розроблені на основі функціональності абонентських пристроїв.

ПЗ Sunvizion для надання бездротового доступу (OTA) для WiMAX: характеристики

– Скорочення часу активації й зміни параметри послуги.

– Зниження витрат на обслуговуючий персонал.

– Більш ефективний збір боргів і більш висока клієнтська дисципліна у зв'язку з автоматичним блокуванням послуги.

– Швидке впровадження нової послуги без необхідності дорогого внутрішнього процесу навчання.

– Проста установка пристроїв у клієнта або відсутність необхідності виїзду установника.

– Керування відносинами із клієнтами, виставляння рахунків, керування обсягом роботи й системами мережної інвентаризації дозволяє реалізовувати надання комплексних послуг клієнтам і оптимізувати бізнес-процес: продаж, модифікацію послуг, оплату клієнтських рахунків.

Аналітична платформа VistaNEO

VistaNEO – система керування якістю – являє собою платформу для аналізу корінних причин (RCA) і оптимізації мобільної мережі. Вона призначена для відділів, відповідальних за якість зв'язку в мережі. Система дозволяє оптимально розподіляти ресурси мережі для задоволення очікувань клієнтів на підставі використання даних про географічний розподіл абонентів.

VistaNEO є потужним і масштабованим рішенням із клієнт- серверною архітектурою й дозволяє інженерам RAN проводити детальний аналіз даних, переданих під час розмов абонентів з урахуванням географічної локалізації. Зібрані дані служать для усунення різних проблем у мережі, перш ніж вони відіб'ються якості зв'язку абонентів.

VistaNEO – програмна платформа, яка збирає, обробляє й зберігає дані про продуктивність радіомережі, засновані на досвіді абонента. Зібрані дані складаються з радіо реєстрацій (також називаних подіями мобільних викликів), які є багатим джерелом інформації на рівні 3, а також містять кілька коштовних

вимірів, виконуваних за допомогою мобільного пристрою під час з'єднання з мережею. У цих даних відсутні інформація про місце розташування, але за рахунок використання складних алгоритмів під час запису, можна одержати приблизне місце розташування користувацького пристрою.

Переваги використання VistaNEO

У цей час іде активний процес розвитку бездротових мереж. Оператори виводять на ринок нові технології, такі як HSPA, LTE і Lte -advance. Для того щоб забезпечити високу продуктивність мережі, їм потрібно постійно збирати велика кількість мережних даних, заснованих на досвіді абонентської бази.

Зручний процес аналізу й оптимізації

VistaNEO дозволяє поліпшити якість зв'язку в мережі, надаючи всі необхідні види даних про продуктивність мобільної мережі. Завдяки функції вбудованого аналізу впливу перешкод, взаємного впливу сусідніх базових станцій і перевантаження стільника, VistaNEO підвищує ефективність оптимізації, дозволяючи швидко й легко визначати причину виникнення проблеми. Система дозволяє здійснювати моніторинг усіх абонентів мережі без винятку й вести єдину базу даних мережних подій.

Використання досвіду для пошуку вирішення проблем

Завдяки вбудованим в VistaNEO ключовим показникам, орієнтованим на клієнта, програмний комплекс дозволяє легко швидко ідентифікувати, аналізувати й вирішувати проблеми, пов'язані з якістю обслуговування окремо взятого абонента, а також визначити, які абоненти в мережі найбільше часто зустрічаються із проблемами. Виявлення причин проблем відбувається моментально, і система відразу пропонує розв'язки для їхнього усунення. Інженери по оптимізації радіомережі можуть приділяти першорядної увагу оптимізації найбільш складних ділянок, домагаючись, в остаточному підсумку, максимального задоволення потреб клієнтів і максимальної прибутковості, при мінімізації зусиль по усуненню неполадок.

Візуалізація поведінки абонентів і мережі

VistaNEO надає відображення параметрів стану й продуктивності мережі з погляду клієнта й оператора, що допомагає одержати цілісну виставу про якість роботи мережі.

Масштабованість і гнучкість

VistaNEO є гнучкою й масштабованою платформою. Кількість мобільних операторів безупинно росте, тому система підтримує устаткування різних виробників і різні технології. Дані про стан мережі й абонентська аналітика можуть зберігатися в системі протягом певного періоду часу для складання повної картини про продуктивність мережі.

VistaNEO надає цілий ряд функцій поряд з ергономічністю. Це єдиний інструмент для аналітики, підтримки VIP клієнтів і інженерної роботи. Платформа може використовуватися як індивідуально, так і в комбінації з різними рішеннями InfoVista, щоб забезпечити максимальну ефективність аналізу. Високопродуктивний рішення VistaNEO стане ключовим у рішення повсякденних завдань по управлінню доступу абонентів у мережі серед інженерів і експлуатаційних відділів. Платформа надає спільний доступ до системи з боку різних департаментів оператора або компаній-партнерів.

Глибокий абонентський аналіз

Абоненти є ключовим активом будь -якого оператора стільниковому зв'язка, незалежно від застосовуваних технологій і виробників застосовуваного устаткування. Висока якість і продуктивність надаваної абонентам зв'язки є основним напрямком для мобільних операторів. Метою оператора є забезпечення правильних умов експлуатації, стратегічної оптимізації й пошук несправностей.

Трасування викликів дозволяє операторові виконувати глибокий аналіз продуктивності. VistaNEO є ідеальним рішенням для обробки великої кількості мережних даних, одержуваних у процесі трасування.

Економія часу й витрат на усунення неполадок

Економія часу й витрат є важливим чинником в оптимізації бездротової мережі й усуненні неполадок. З розвитком мережі, процес виявлення причин несправностей ускладнюється. Усе більше часу потрібно на пошук проблеми і її рішення. VistaNEO пропонує можливості, які допомагають операторам значно деталізувати аналіз мережі й скоротити дублювання роботи з усунення неполадок, виконуваної відділом по керуванню якістю й експлуатаційним відділом.

Платформа дозволяє задавати ключові показники для конкретного сегмента клієнтів (наприклад, для корпоративних користувачів), автоматично виявляти абонентів, стільник і мережі, з якими виникають проблеми й видавати рекомендації з їхнього усунення.

VistaNEO надає величезний набір інформації, отриманої в результаті аналізу даних трасування викликів. Уся інформація є доступною для розуміння користувача. Також обробляються дані оптимізації, «забруднення» мережі, зони покриття, можливості перебудування радіомережі. За результатами діагностики система виводить інтелектуальні пропозиції по поліпшенню якості мережі на інтерактивному дисплеї.

Для рішення локальних завдань траблшутингу мережі, може бути використаний тільки інженерний клієнт Xeus. Система дозволяє не тільки знаходити й повідомляти про проблему, але й виявляти її причини. При цьому немає необхідності в установці програмного забезпечення на мобільні термінали або сім-карти.

Аналітична платформа VistaNEO: характеристики

– Простота й гнучкість пошуку ключових абонентів, які випробовують проблеми в мережі.

– Можливість установки пріоритету ключових абонентів і областей забезпечує більш висока якість обслуговування й більш чіткої орієнтації мережі.

– Аналіз групових і індивідуальних ключових показників ефективності.

– Безшовне відстеження місця розташування абонента по областях мережі.

– Виявлення проблемних областей.

– Підтримка більшості бездротових стандартів: GSM, GPRS, EDGE, WCDMA, HSPA, HSPA+, LTE (TDD і FDD), Lte-advanced, Wi-Fi, WiMAX і т.д.

Система моніторингу Rohde & Schwarz AMMOS GX400

Система R&S®AMMOS® GX400ID – це закінчене середовище розробки, яке клієнти можуть використовувати для розробки й тестування власних модулів ВЧ-Декодерів. Нові декодери завантажуються в групи датчиків R&S®AMMOS® шляхом відновлення ПЗ. Рішення R&S®AMMOS® GX400 розроблене для виявлення, моніторингу й аналізу сигналів радіозв'язку у ВЧ і ОВЧ/УВЧ діапазонах.

Система моніторингу Rohde & Schwarz AMMOS GX400: характеристики

– Контроль ВЧ і ОВЧ/УВЧ приймачів у режимах фіксованої частоти (FFM) і сканування. Підтримуються ПЧ-спектр у реальному часі й запис/відтворення сигналів ПЧ в R&S®AMMOS® GX425 AMREC.

– Класифікація й демодуляція/декодування ВЧ-сигналів.

– Класифікація й демодуляція/декодування ОВЧ і УВЧ сигналів.

– Контроль ВЧ, ОВЧ і УВЧ приймачів у покроковому режимі й режимі фіксованої частоти (FFM). Підтримуються широкосмуговий спектр частот у реальному часі й запис/відтворення широкосмугових сигналів ПЧ в R&S®AMMOS® GX425 AMREC. Є функція масштабування для відображення широкосмугового спектра частот з дозволом до 1 Гц.

– Автоматичне виявлення сигналів з фіксованою частотою й короткочасних (пакетних) сигналів для моніторингу й спостереження.

Система керування продуктивністю застосунків

InfoVista Ipanema – це система, яка автоматично збільшує продуктивність ваших застосунків, що працюють в WAN-мережах завдяки інтеграції унікальних можливостей: прозорість застосунків і керування ними (Application Visibility and

Control), динамічна гібридна WAN, забезпечення безпеки й оптимізація WAN. Система призначена для задоволення потреб кожного користувача, незалежно від складності IT-інфраструктури (кількості застосунків, сайтів, провайдерів, користувачів і т.д.).

InfoVista Ipanema є цілеорієнтованою системою. Вона дозволяє директорам по інформаційних технологіях визначати продуктивність тих або інших застосунків відповідно до корпоративних бізнес-пріоритетами й цілями. Ipanema надає докладну візуалізацію потоку зі складними алгоритмами керування для забезпечення дотримання угоди про рівень послуг (SLA).

InfoVista Ipanema розглядає всі сценарії гібридних WAN мереж і пропонує найсучасніші розв'язки для динамічного керування всією мережею й окремо взятими потоками в гібридних WAN. Система дозволяє максимізувати використання смуги пропущення за рахунок автоматичного перенапрямку кожного потоку по певному сегменту мережі на підставі доступності каналів глобальної мережі в цей момент (MPLS, Інтернет, бездротовий зв'язок і т.д.). Таким чином, система вибирає потік, який найкраще задовольняє потреби критичності бізнес-застосунки, незалежно від хостингу, на якому вони розміщені: центри обробки даних (у тому числі частки), приватна хмара або SaaS.

Архітектура InfoVista Ipanema складається із системи централізованого керування на основі веб-інтерфейсу, яка здійснює моніторинг і надає звітність про продуктивність застосунків і дотримання заданого рівня послуг (SLA), широкого спектра фізичних і віртуальних (VNF) мережних кінцевих крапок, які відповідають потребам як маленьких філій компаній, так і найбільших центрів обробки інформації (ЦОД).

InfoVista Ipanema доступна також у якості хмарного сервісу IpaneMACloud.

Ключові особливості

Видимість застосунків

Забезпечує надання повної інформації про використання застосунків і їх продуктивності в глобальній мережі: від дрібних подробиць до керування SLA продуктивністю.

Керування застосункими

Динамічно регулює поведінку й ресурси мережі для одержання точної кількості трафіку, яка необхідно для забезпечення заданої продуктивності критично важливих застосунків у самих складних й постійно мінливих ситуаціях із трафіком.

Оптимізація WAN

Зменшує час відгуку застосунків і надає додаткову віртуальну смугу пропущення шляхом прискорення застосунків і виключення надмірності даних.

Створення динамічної WAN-мережі

Створює динамічну гібридну WAN-мережу для декількох розподілених філій компанії, здійснюючи вибір у режимі реального часу найкращого шляху передачі відповідно до фактичних характеристик продуктивності застосунків і завантаженості мережного трафіку.

Безпека WAN

Захищає від погроз галузей підключення до інтернету шляхом VPN-Шифрування й пересилання веб-трафіку через захищені веб-шлюзи SWG (Secure Web Gateways) провайдерів, які дозволяють або забороняє передачу для переходу безпосередньо в Інтернет.

Гібридна WAN

На шляху IT-трансформації потрібне збільшення пропускної здатності WAN, підвищення доступності послуг і гнучкості. При цьому вузли мережі часто виявляються перевантажені, тому що в цей час відбувається активне збільшення IT-сервісів і транзакції застосунків між користувачами й датацентрами, інтернетом і хмарними сервісами. У такому випадку ваша мережа WAN може

швидко стати перешкодою на шляху до успіху IT-трансформації. Завдяки використанню системи InfoVista Ipanema ви зможете повною мірою реалізувати можливості вашої гібридної мережі WAN за рахунок розширених IT-можливостей, які дозволяють об'єднати роботу всіх компонентів (MPLS і Інтернет).

Області застосування

Гарантія продуктивності ERP-, Crm-систем і бізнес-застосунків

Для досягнення максимальної ефективності, підприємства усе більше покладаються на свої IT-організації, що надають доступ до критично важливих бізнес-застосунків, розміщеним централізовано в приватних центрах обробки даних (ЦОД) або в хмарі, у тому числі, ERP, CRM, керування фінансами, бізнес-аналітика й інші спеціалізовані бізнес-додатки. InfoVista Ipanema допоможе вам захистити всі потоки даних у застосунках у всій мережі WAN. Ви можете задати в SLA рівень критичності певного бізнес-застосунку й пов'язаних з ним операційних процесів.

Уніфіковані комунікації й колективна робота

Персонал деяких компаній може різко збільшуватися, наприклад, у випадку роботи підприємства в міжнародному середовищі. У таких випадках у зв'язку з високою динамічністю трафіку й контенту, який припадає на різних співробітників, генеруються дуже складні матриці трафіку, які безпосередньо впливають на загальну продуктивність корпоративної мережі і її застосунки. Ipanema дозволить добитися відмінного рівня взаємодії з кінцевими користувачами. Завдяки використанню системи можна виділити окрему смугу пропущення для кожного відео- або файлового потоку, а також захистити голосові виклики від джиттера, миттєві повідомлення – від неприйнятних затримок, і в той же час зберегти продуктивність ERP або SaaS застосунків.

SaaS, хмарні сервіси

Однієї з основних проблем хмарних застосунків, з якими підприємства зустрічаються, є те, що інтернет загального користування в цей час стає частиною

часток WAN, і про дуктивність хмарних потоків може виявитися під погрозою. Наприклад, такі сервіси, як Youtube або Dropbox, можуть негативно позначатися на доступі до інтернету. Ipanema захищає важливі бізнес-застосунки й прискорює розгортання й впровадження застосунків SaaS. Так само Ipanema має чудовий досвід взаємодії з кінцевим користувачем, гарантуючи задану пропускну здатність для кожної окремо взятої передачі даних із застосування механізмів стиску.

Система моніторингу InfoVista Vista360

Ваша організація страждає від перевантаження інформацією? Доводиться сортувати більші обсяги важливої інформації про керування продуктивністю мереж і застосунків, перш ніж прийняти зважений рішення?

Vista360 – потужне інноваційне Web 2.0 застосунок від компанії InfoVista, яке забезпечує легкий у використанні, гнучкий і наочний моніторинг продуктивності мереж і застосунків, що дозволяє співробітникам, що ухвалюють рішення одержати доступ до будь-якої необхідної інформації в такому виді й у той момент, коли їм це потрібно для прийняття ефективних рішучих заходів.

InfoVista Vista360 сполучається з будь-яким мережним, серверним і прикладним рішенням InfoVista для моніторингу й створення звітів і використовує всі переваги потужної платформи InfoVista для забезпечення якості послуг. InfoVista Vista360 задовольняє потреба користувачів в одержанні в реальному часі персоналізованої інформації, на основі якої можна ухвалити рішення, допомагаючи їм використовувати інформацію про продуктивність мереж, послуг і застосунків, отриману від Vistainsight for Networks і 5View Service Data Manager.

З Vista360 відділи експлуатації операторів послуг одержують можливість створювати свої власні системи моніторингу продуктивності мереж і застосунків для аналізу й усунення неполадок на застосунок до традиційних систем і системам, що реагують на помилки. Комбінація гнучкості й простоти використання Vista360 дозволяє інженерам відділів експлуатації розробити

систему відповідно до їхнього власного бачення послуги в трьох вимірах: використання, інфраструктура й комплексна продуктивність і одержати повну візуалізацію послугу.

InfoVista Vista360 працює й на рівні застосунків, дозволяючи інженерам розуміти не тільки візуальну структуру зв'язків між компонентами інфраструктури, але й те, як послуги використовуються застосункими й продуктивність застосунків. Така прозорість дуже важлива, коли оператор надає послуги, що включають додатки.

Система керування продуктивністю мереж і дата-центрів InfoVista Vistainsight for Networks

InfoVista Vistainsight for Networks гарантує якісне, надійне функціонування мереж на основі IP і орієнтоване на інтернет-провайдерів, операторів мобільного зв'язку й великі IT-компанії з більшою кількістю підрозділів. Використання InfoVista Vistainsight for Networks дозволить їм надавати високоякісні послуги, одержувати детальні звіти про користувачів, вимірювати рівень надаваних послуг (SLA) за допомогою стандартних методик, що й налаштовуються, відслідковувати працездатність і завантаженість сервісів і ресурсів і завчасно виявляти зниження продуктивності мереж або якості роботи застосунків.

InfoVista Vistainsight for Networks – це попередньо налаштований застосунок, що поєднує переваги програм "з коробки" і можливість гнучкого налаштування для задоволення потреб операторів зв'язку. Заснована на легко масштабованій платформі вилученого збору даних і створення звітів InfoVista, Vistainsight for Networks надає операторам зв'язку можливість миттєво включати в систему нові прибуткові сервіси.

Система керування продуктивністю мереж InfoVista Vistainsight for Networks працює й з дата -центрами, включаючи підтримку серверів, сховищ і мережної інфраструктури, як з фізичної точки зору, так і з віртуальної. Є моделі вбудованих сервісів, звіти й трафік для різних мереж і послуг, включаючи:

IP/MPLS, MEF-aligned Carrier Ethernet, опорних мереж мобільного зв'язку, IP-VPN, application-aware VPNs і Infrastructure-as-a-service (IaaS).

Система керування продуктивністю мереж інтегрується із системою керування продуктивністю застосунків InfoVista 5View і панелями моніторингу InfoVista Vista360. Система також має підтримку InfoVista Mobile Knowledge Pack, що розширює можливості керування продуктивністю на мобільні мережі.

Складові KPI:

- Top N.
- Quality of Service.
- Saturation.
- Time To Capacity.
- Performance Exceptions.
- Instance availability.
- End to End metrics.

Підтримувані вендори:

- Cisco IPSLA.
- Juniper RPM.
- Cisco CbQoS and MPLS(TE).
- NBAR, Netflow.
- Alcatel 5620 SAM, SDC 5529.
- Alcatel Dslams.
- Huawei Dslams.
- Cisco OEM.
- ADVA NTU.
- Ericsson GGSN/SGSN.
- Juniper BRAS.

Система керування продуктивністю мереж і дата -центрів InfoVista Vistainsight for Networks: характеристики

– Звіт по ємності для послуг і інфраструктури; VPN Access QoS, drops / delivery ratio; service latency evolution; core, наприклад, MPLS (LSP) network planning.

– Повідомлення про перевищення використання.

– KPI для керування й планування ємності, включаючи: 95%, peak, busy hour, busy day; baselines (time of day), trending, early warning notifications.

Кросдомена система керування продуктивністю мережі для мобільних операторів InfoVista Mobile Knowledge Pack

Перехід до інфраструктури на основі IP допоміг операторам мобільних мереж задовольнити потреба в збільшенні швидкостей передачі. Але для підвищення якості послуг виникла необхідність використовувати нові складні елементи керування, що значно підвищують операційні витрати. У той же час, діапазон і природа мобільних послуг продовжують розширюватися (від 2G до 3G+, 4G LTE, а зараз – LTE Advanced) і зростаюча конкуренція на ринку змушує операторів мобільного зв'язку обертати все більша увага на задоволеність абонентів.

InfoVista Mobile Knowledge Pack озброює інженерні й експлуатаційні відділи операторів мобільному зв'язку інтелектуальною системою керування мережею, що не залежить від топології для керування мультидоменою інфраструктурою. InfoVista Mobile Knowledge Pack має відкриту й гнучку систему одержання й передачі інформації, що дозволяє відділам експлуатації збирати інформацію з безлічі простих лічильників, якими оснащують мережне устаткування виробники, і використовувати цю інформацію для визначення необхідних показників прибутки при плануванні й експлуатаційних процесах. InfoVista Mobile Knowledge Pack швидко адаптується до постійно мінливої інфраструктури, що й обновляється, і допомагає суттєво скоротити час розгортання нових систем, таких як HSPA+ або LTE.

Поряд з наскрізним моніторингом інфраструктури базових станцій, інтеграція з Vistainsight for Networks зробить прозорої й опорну мережа, дозволяючи операторам мобільного зв'язку стежити за проходженням послуги на всьому шляху й досягати небувалої ефективності в роботі інженерних і експлуатаційних підрозділів і використовувати для цього всього один інструмент.

Кроссдомена система керування продуктивністю мережі для мобільних операторів InfoVista Mobile Knowledge Pack: характеристики

– Керування продуктивністю мобільних мереж незалежно від вендора й розташування.

– Зв'язує якість надання послуги з топологією.

– Упорядковує велика кількість мережних KPI/KQI.

– Завчасне попередження про вплив на послугу.

– Механізм опитування за заявкою й складання звітів у реальному часі.

– Створення KPI на вимогу.

Програмне забезпечення VistaLink for 5620 SAM

InfoVista VistaLink for Alcatel-Lucent 5620 Service Aware Manager (SAM) – це інтеграційний рішення (сертифіковане по програмі Alcatel-Lucent OSS Connected Partner Program), що пропонує переваги кращого в класі стандартного програмного продукту під час відсутності витрат і ризиків, типових для складних інтеграційних проектів. Цей модуль розширення, що підходить для більших різнорідних мереж, забезпечує користувачів Alcatel-Lucent 5620 SAM продуманою системою керування продуктивністю мережі, яка використовує всі можливості рішення Alcatel-Lucent 5620 SAM.

InfoVista VistaLink for Alcatel-Lucent 5620 SAM забезпечує операторам зв'язку гарантовано якісну передачу послуг на основі IP. Використання InfoVista VistaLink for Alcatel-Lucent 5620 SAM дозволить їм надавати високоякісні послуги, одержувати детальні звіти про користувачів, вимірювати рівень надаваних послуг (SLA) за допомогою стандартних методик, що й

налаштовуються, відслідковувати працездатність і завантаженість сервісів і завчасно виявляти зниження продуктивності мереж.

Будучи головним партнером компанії Alcatel-Lucent у програмі Alcatel-Lucent's Connected Partner Program, рішення InfoVista підтримують усі функції новітніх релізів Alcatel-Lucent's 5620 SAM. Завдяки постійній взаємодії InfoVista з командою SAM OSS і строгої сертифікації, постачальники послуг можуть бути впевнені в тому, що їх середовище керування продуктивністю й моніторингу продовжить функціонувати бездоганно, навіть після відновлення мережі, що дозволяє їм використовувати всі новітні переваги нового релізу Alcatel-Lucent.

Програмне забезпечення VistaLink for 5620 SAM: характеристики

– Однократна підписка, звіти з Web-інтерфейсом.

– Фокус на послузі.

– Працює « з коробки», сертифікована інтеграція.

– Завчасне попередження про вплив на послугу.

– Стандартні, що й налаштовуються KPI для різних послуг.

– Може використовуватися спільно.

Хмарна платформа забезпечення гарантованого якості послуг зв'язку InfoVista Vistago

Vistago – це інноваційний хмарний застосунок від InfoVista, орієнтоване на оптимізацію двох найважливіші складові якості обслуговування операторів зв'язки й мобільних мереж (CSP/MNO): бізнес-послуги й опорна транспортна мережа.

Система надає провайдерам послуг зв'язки (CSP) і операторам мобільних мереж (MNO) наскрізну видимість продуктивності мережі й інфраструктури з погляду впливу на бізнес і забезпечує впевненість у виконанні вимог угод про якість надаваних послуг (SLA) в умовах стрімкого зростання пропускної здатності, нових технологій і мінливої архітектури.

Проблема інтелектуального підходу до бізнес-процесам при скороченні бюджетування стоїть на шляху в багатьох операторів Tier-1. Платформа Vistago

дозволяє забезпечити гарантована якість послуг зв'язки й знизити сукупну вартість володіння мережею (TCO)

Використання хмарної системи Vistago значно скорочує час, необхідне на обґрунтування бізнесу. Надавана як послуга, Vistago передбачає періодичну плату за обслуговування й не вимагає капіталовкладень. Завдяки застосуванню платформи у своїх бізнес-процесах, телекомунікаційні компанії можуть добитися максимальної впевненості як надаваних послуг.

Планування потужностей і точне визначення масштабів опорної мережі й послуг VPN

Система надає розширений аналіз можливості консолідації мережних ємностей, аналіз архівних звітів, тенденцій і прогнозів для визначення областей, що вимагають збільшення смуги пропущення, і оптимального розподілу капіталовкладень у мережу.

Багаторівнева діагностика якості обслуговування

Моніторинг усіх доменів і фізичних рівнів, включаючи оптичний, бездротової, Ethernet і IP, у режимі реального часу, дозволяє оцінити вплив використання устаткування різних вендорів на продуктивність.

Просунута звітність по відповідності заданому рівню SLA IP VPN і Carrier Ethernet послуг

Фірмовий клієнтський портал для відстеження порогів по SLA і наскрізної видимості продуктивності надає панелі, що налаштовуються, візуалізацію й аналітикові, а також можливість установки певних порогів. Служба підтримки й інші учасники бізнес-процесу можуть переглядати всіх корпоративних клієнтів і послуги для контролю відповідності заданому рівню SLA.

Робота з моделі «фріміум»

Багаторівнева звітність, надавана Vistago заснована на, так званому, «фріміуме» – бізнес моделі, згідно з якою оператори зв'язки можуть безкоштовно одержувати базові звіти, а платно надається доступ до просунутої версії, що забезпечують більш високий рівень обслуговування.

Хмарна платформа забезпечення гарантованого якості послуг зв'язки InfoVista Vistago: характеристики

– Цінова доступність дозволяє платити в міру росту й пр огнозувати щомісячні витрати.

– Доступність і безперервна робота навіть у випадку відновлення програмного забезпечення або збоїв.

– Стругаючи політика паролів, захищена віртуальна приватна копія й надмірність, створення безпечних тунелів IPsec VPN між центрами обробки даних оператора й Vistago.

– Регулярна оцінка використання обчислювальних ресурсів і сховищ даних, коректування виділених ресурсів для задоволення мінливих вимог.

– Усі надавані показники пов'язані з відповідним клієнтом і сервісом від моменту збору до візуалізації, щоб забезпечити сервіс-орієнтований зв'язок, навігацію й аналіз впливу окремих показників на бізнес-клієнта.

– Безпечна віртуальна приватна хмара відповідає необхідним нормативним і бізнес-вимогам.

– Повна реєстрація й контроль змін за допомогою вікон обслуговування, а також цілодобова глобальна підтримка керування інцидентами забезпечують високу надійність.

– Неперевершена в галузі масштабованість, гнучкість, інтерфейс користувача й інтеграція з моделлю обслуговування.

– Віртуальні пристрої, що завантажуються, різних виробників і інтеграція системи керування елементами із хмарою.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, MAC OS, iOS і Android мовою

Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для MACOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для MACOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з

підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на MACOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовується щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64 -bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів н а одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для MACOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для MACOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи моніторингу продуктивності IoT.

В процесі розробки кваліфікаційної бакалаврської роботи 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

Інтернет речей – це сукупність вбудованих систем, мереж бездротових датчиків, систем керування й засобів автоматизації. Усе це дозволяє реалізувати підключені фабрики, інтелектуальні магазини, розумні будинки й міста, а пристрої, що також носяться. За допомогою технологій Інтернету речей ви зможете перетворити свій бізнес за рахунок отриманих на основі даних аналітичних відомостей, удосконалених робочих процесів, нових сфер діяльності й більш ефективного використання матеріалів.

Технології Інтернету речей продовжують розвиватися. Щорічно з'являється незліченна кількість постачальників послуг, різноманітні платформи й мільйони нових пристроїв. Зараз для початку роботи в екосистемі Інтернету речей розроблювачам потрібно прийняти безліч рішень. Це керівництво допоможе вам скласти уявлення про розповсюджені протоколи Інтернету речей, а також про вимоги до енергоспоживання й підключенню.

Екосистема технологій Інтернету речей

Екосистема технологій Інтернету речей складається з наступних рівнів: рівень пристроїв, рівень даних, рівень підключення й рівень користувачів технологій.

Рівень пристроїв

Сукупність датчиків, виконавчих пристроїв, устаткування, програмного забезпечення, засобів підключення й шлюзів, що становлять пристрій, який з'єднується й взаємодіє з мережею.

Рівень даних

Дані, які збираються, обробляються, відправляються, зберігаються, аналізуються, представляються й використовуються в бізнесі-контексті.

Рівень бізнесу

Бізнес-функції технологій Інтернету речей, включаючи керування виставлянням рахунків і магазинами даних.

Рівень користувачів

Компоненти, які дозволяють людині взаємодіяти із пристроями Інтернету речей.

Стек технологій Інтернету речей. Пристрої Інтернету речей

Пристрої Інтернету речей

Нижче наведені деякі розповсюджені терміни, пов'язані із пристроями в стеці технологій Інтернету речей.

Вбудовані системи

Тут використовується як устаткування, так і програмне забезпечення. Такі системи управляють певними функціями, пов'язаними з більшою системою. Вбудовані системи створюються на основі мікропроцесорів або мікроконтролерів.

Інтелектуальні пристрої

Ці пристрої можуть виконувати обчислення й часто оснащені мікроконтролером.

Мікроконтролер (MCU)

Ці невеликі комп'ютери впроваджуються в мікросхеми й оснащені ЦП, ОЗП й ПЗП. Вони містять елементи, необхідні для виконання простих завдань. Але мікроконтролери мають більш обмежену потужність, чому мікропроцесори.

Мікропроцесор (MPU)

Функції ЦП можна розміщати на одній або декількох інтегральних мікросхемах. Хоча для виконання завдань мікропроцесорам потрібні периферійні пристрої, витрати на обробку значно скорочуються, тому що в них входить тільки вартість ЦП.

Пристрої, не призначені для обчислень

Ці пристрої тільки встановлюють підключення й передають дані. У них немає можливості виконувати обчислення.

Перетворювачі

Фізичні пристрої, які перетворять один вид енергії в інший. У контексті пристроїв Інтернету речей до них ставляться внутрішні датчики й виконавчі пристрої, що передають дані в міру того, як об'єкти взаємодіють із їхнім середовищем.

Виконавчі пристрої

Виконують фізичні дії, коли центр керування дає вказівки. Звичайно причиною є зміни, зафіксовані датчиками.

Датчики

Виявляють зміни у своєму середовищі й створюють е лектричні сигнали для обміну даними. Датчики звичайно виявляють зміни навколишнього середовища, наприклад температури, фізичного положення й змісту хімічних речовин.

Стек технологій Інтернету речей. Протоколи Інтернету речей і підключення

При плануванні проекту Інтернету речей важливо враховувати спосіб підключення пристрою й взаємодії з ним. Так ви зможете визначити, які протоколи Інтернету речей застосовні до нього.

Підключення пристроїв Інтернету речей

У стеці технологій Інтернету речей пристрої підключаються за допомогою шлюзів або вбудованих функцій.

Що таке шлюзи Інтернету речей?

Шлюзи відповідають за підключення пристроїв Інтернету речей до хмари. Дані, що збираються із пристроїв Інтернету речей, проходять через шлюз, попередньо обробляються в прикордонному середовищі, а потім відправляються в хмару.

Використання шлюзів Інтернету речей продовжує час роботи від акумулятора. Крім того, зменшується затримка й обсяг переданих даних. Шлюзи також дозволяють підключати пристрої без прямого доступу до Інтернету й забезпечують додатковий рівень безпеки, захищаючи дані при переміщенні в обох напрямках.

Як підключати пристрою Інтернету речей до мережі?

Необхідний тип підключення залежить від пристрою, його функцій і користувачів. Як правило, відстань, на яку повинні передаватися дані (малий діапазон або великий діапазон), визначає необхідний тип підключення Інтернету речей.

3.2 Розробка структурної схеми

У якості фрагмента архітектури Internet of Things (IoT) розглянемо мережу, що полягає з декількох обчислювальних мереж фізичних об'єктів, підключених до мережі Інтернет з допомогу одного із пристроїв: Gateway, Border router, Router.

Як випливає из архітектури IoT, мережа Internet of Things полягає: з обчислювальних мереж фізичних об'єктів, традиційної IP мережі Інтернет і різних пристроїв (Gateway, Border router і т.д.) мережі, що поєднують їх.

Обчислювальні мережі фізичних предметів складаються з "розумних" датчиків і приводів (виконавчих пристроїв), об'єднаних в обчислювальну мережу (персональну, локальну й глобальну) і керованих центральним контролером (шлюзом або IoT Habs, або платформою IoT).

В Internet of Things (IoT) застосовуються технології бездротових обчислювальних мереж фізичних предметів з низьким енергоспоживанням, до яких ставляться мережі малого, середнього й далекого радіуса дії (WPAN, WLAN, LPWAN).

Бездротові технології мереж LPWAN (Low-power Wide-area Network) Інтернету речей IoT

До розповсюджених технологій мереж далекого радіуса дії LPWAN, які представлені на мал. 1, ставляться: LoRaWAN, SIGFOX, "Стриж" і Cellular Internet of Things або скорочене CIoT (EC-GSM, LTE-M, NB-IoT). До мереж LPWAN ставляться й інші технології, наприклад, ISA-100.11.a, Wireless, DASH7, Symphony Link, RPMA і так далі,. Великий список технологій представлений на сайті link-labs.

Однією із широко розповсюджених технологій є LoRa, яка призначена для мереж далекого радіуса дії, з метою передачі даних телеметрії різних приладів обліку (датчиків води, газу і т.д.) на далекі відстані.

LoRa – це метод модуляції, який визначає протокол фізичного рівня моделі OSI. Технологія модуляція LoRa може застосовуватися в мережах з різною топологією й різними протоколами канального рівня. Ефективними мережами LPWAN є мережі LoRaWAN, які використовують протокол канального рівня LoRaWAN (MAC протокол канального рівня), а в якості протоколу фізичного рівня – модуляцію LoRa.

Мережа LoRaWAN складається з прикінцевих вузлів End Nodes (трансіверів або модулів LoRa), підключених по бездротових мережах до концентраторів/шлюзам або базовим станціям, Network Server (сервера мережі оператора) і Application Server (сервера застосунків сервіс провайдера). Мережна архітектура LoRaWAN – " клієнт-сервер". LoRaWAN працює на 2 рівні моделі OSI.

Між компонентами мережі «прикінцеві вузли – сервер» використовується двосторонній зв'язок. Взаємодія прикінцевих вузлів локальної мережі LoRaWAN із сервером відбувається на основі протоколів канального рівня. У якості адреси використовуються унікальні ідентифікатори пристроїв (прикінцевих вузлів) і унікальні ідентифікатори застосунків на сервері застосунків.

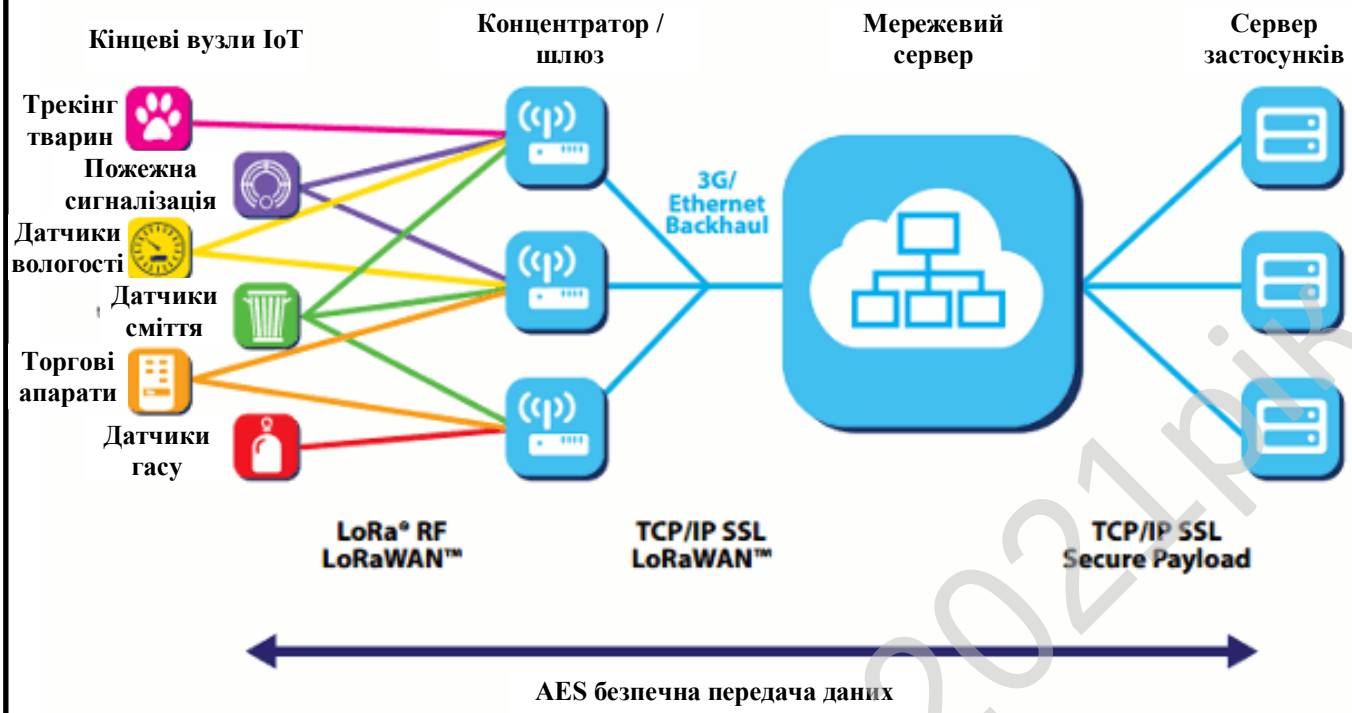

Рисунок 3.1 – Структурна схема системи

Фізичним рівнем стека протоколів LoRaMAC сегмента мережі «прикінцеві вузли – шлюз», який функціонує на другому рівні моделі OSI, є бездротова модуляція LoRa, а MAC-протоколом канального рівня є LoRaWAN. Шлюзи LoRa підключаються до сервера мережі провайдера або оператора за допомогою стандартних технологій Wi-Fi/Ethernet/3G, які ставляться до рівня інтерфейсів IP мереж (фізичним і канальним рівням стека TCP/IP).

Шлюз LoRa забезпечує міжмережева взаємодія між мережами на основі різнорідних технологій LoRa/LoRaWAN і Wi-Fi, Ethernet або 3G. На мал. 1 представлена мережа LoRa з одним шлюзом, виконана по топології «зірка», але мережа LoRa може бути й з безліччю шлюзів (стільни кова структура мережі). У мережі LoRa з безліччю шлюзів «прикінцеві вузли – шлюз» побудовані по топології «зірка», у свою чергу, "шлюзи – сервер" теж підключені по топології «зірка».

Отримані з прикінцевих вузлів дані зберігаються, відображаються й обробляються на сервері застосунків (на автономному Web сайті або в «хмарі»). Для аналізу IoT-Даних можуть застосовуватися методи Big Data. Користувачі за допомогою клієнтських застосунків, установлених на смартфон або ПК, мають можливість доступу до інформації на сервері застосунків.

Технології SIGFOX (sigfox.com) і "Стриж" (strij.net) аналогічні технології LoRaWAN (www.semtech.com), але мають деякі відмінності. Основна відмінність полягають у методах модуляції, які визначають протоколи фізичних рівнів цих мереж. Технології SIGFOX, LoRaWAN і "Стриж" є конкурентами на ринку мереж LPWAN.

Конкурентами на ринку мереж LPWAN є й технології C IoT (EC-GSM, LTE-M, NB-IoT), а також G5. Вони призначені для побудови бездротових мереж LPWAN стільниковому зв'язка на основі існуючої інфраструктури стільникових операторів. Застосування традиційних мереж стільниковому зв'язка в IoT є нерентабельним, тому в цей час нішу мереж LPWAN зайняли LoRaWAN, SIGFOX і т.д. Але якщо оператори стільникового зв'язку вчасно впровадять технології EC-GSM (Extended Coverage GCM), LTE-м (LTE для M2M-комунікацій), засновані на еволюції GSM і розвитку LTE, те вони потіснять LoRaWAN, SIGFOX і інші технології з ринку LPWAN.

До найбільш перспективних напрямків побудови бездротових мереж LPWAN ставиться вузькополосний інтернет речей NB-IoT (Narrow Band IoT) на базі LTE, який може бути розгорнуть поверх існуючих мереж LTE операторів стільникового зв'язку. Але стратегічним напрямком в CIoT є стільникові мережі нового покоління 5G, які будуть підтримувати IoT.

Технологія 5G, призначена для роботи з різнорідним трафіком, забезпечить підключення до Інтернет різноманітних пристроїв з різними параметрами (енергоспоживанням, швидкостями передачі даних і т.д.) як мобільних пристроїв (смартфонів, телефонів, планшетів і т.д.), так і Smart Objects (sensors or actuators).

Слід зазначити, що в Internet of Things (IoT) поряд з використанням хмарних технологій застосовуються технології «мрячних обчислень» (fog computing). Це обумовлене тим, що в хмарній моделі, використовуваної в IoT, слабким місцем є пропускна здатність каналів операторів зв'язки, по яких здійснюється обмін даними між "хмарою" і "розумними" пристроями обчислювальних мереж фізичних предметів.

Концепція "мрячних обчислень" припускає децентралізацію обробки даних за рахунок передачі частини роботи з обробки даних і прийняттю управлінських рішень з "хмари" безпосередньо пристроям обчислювальних мереж фізичних предметів.

Підвищення пропускної здатності каналів зв'язки Cloud computing може забезпечити новий підхід їх побудови на основі технології Software-Defined Networks (SDN). Тому впровадження SDN дозволить підвищити ефективність роботи каналів зв'язки Cloud computing і Internet of Things (IoT).

Типи мереж Інтернету речей

Мережі з низьким енергоспоживанням і малим діапазоном

Ці мережі добре підходять для будинку, офісу й інших невеликих середовищ. Для їхнього застосування досить невеликих акумуляторів, а іноді їх можна настроїти й без використання акумулятора. Як правило, вони досить економічні в експлуатації.

Нижче наведені розповсюджені приклади.

Bluetooth

Bluetooth забезпечує високошвидкісну передачу даних і відправляє сигнали голосу й даних на відстань до 10 метрів.

Wi-Fi/802.11

Завдяки низькою вартості експлуатації Wi-Fi – стандартний варіант для будинку й роботи. Але цей варіант підходить не для всіх сценаріїв через обмежений діапазон дії й постійного споживання енергії.

Z-Wave

Мережа в мережі для домашніх пристроїв. Забезпечує обмін даними за допомогою низькоенергетичних радіохвиль. Z-Wave забезпечує взаємодія між домашніми системами автоматизації на рівні застосунків.

Zigbee

Це популярний варіант для домашніх систем автоматизації й медичних пристроїв. Zigbee найкраще підходить для особистих мереж з невеликими пристроями, які споживають мало енергії, мають низьку пропускну здатність і використовуються в замкненому діапазоні.

Мережі з низьким енергоспоживанням і широкою зоною охвату (LPWAN)

Забезпечують зв'язок на відстані від 500 метрів, відрізняються мінімальним енергоспоживанням і використовуються для більшості пристроїв Інтернету речей. Наприклад, мережі з більшим діапазоном і широкою зоною охвату (LoRaWAN) забезпечують зв'язок між мобільними захищеними двонаправленними пристроями, що працюють від акумулятора.

Нижче наведені розповсюджені приклади.

Інтернет речей 4G LTE

Такі мережі забезпечують високу потужність і малу затримку. Це відмінний варіант для сценаріїв Інтернету речей, у яких потрібно одержувати відомості або відновлення в реальному часі.

Інтернет речей 5G

Мережі Інтернету речей 5G ще не реалізовані. Планується, що в майбутньому вони будуть підтримувати подальші інновації в Інтернеті речей, а також забезпечувати набагато більш високу швидкість завантаження й підключення до набагато більшого числа пристроїв у певній зоні.

Cat-0

Ці мережі на основі LTE – самі економічні. Вони забезпечують основу для технології Cat-M, яка замінить 2G.

Cat-1

Цей стандарт для мобільного Інтернету речей в остаточному підсумку замінить 3G. Мережі Cat-1 легко набудовувати. Це відмінний варіант для застосунків, яким потрібно голосовий або браузерний інтерфейс.

LTE Cat-M1

Ці мережі повністю сумісні з мережами LTE. Вони дозволяють оптимізувати вартість і потужність другого покоління мікросхем LTE, розроблених спеціально для застосунків Інтернету речей.

Вузькополосне підключення

Цей стандарт радіочастотних технологій працює на основі підмножини стандарту LTE. Він орієнтований на покриття в приміщенні, а також забезпечує низькі витрати й тривалу роботу від акумулятора.

NB-IoT/Cat-M2

Використовує модуляцію розширеного спектра з послідовним перебором (DSSS) для відправлення даних безпосередньо на сервер. Це усуває необхідність у використанні шлюзу. Налаштування мереж NB-IoT коштує дорожче, але завдяки відсутності шлюзу їх робота супроводжується меншими витратами.

Sigfox

Цей провідний глобальний постачальник мережних послуг Інтернету речей пропонує бездротові мережі для підключення об'єктів з низьким енергоспоживанням, які безупинно створюють дані.

Протоколи Інтернету речей: як пристрої Інтернету речей обмінюються даними з мережею

Пристрою Інтернету речей обмінюються даними за допомогою протоколів Інтернету речей. Протокол IP – це набір правил, що визначають спосіб відправлення даних в Інтернет. Протоколи Інтернету речей гарантують, що інформація з одного пристрою або датчика буде зчитуватися й правильно інтерпретуватися іншим. Тому що існує безліч пристроїв Інтернету речей, важливо використовувати потрібний протокол у потрібному контексті.

Який протокол Інтернету речей мені підійде?

Тип використовуваного протоколу Інтернету речей залежить від рівня архітектури системи, на якому повинні передаватися дані. Модель OSI надає карту різних рівнів, на яких виконується відправлення й одержання даних. Кожний протокол в архітектурі системи Інтернету речей забезпечує взаємодія між пристроями, між пристроєм і шлюзом, між шлюзом і центром обробки даних або між шлюзом і хмарою, а також обмін даними між центрами обробки даних.

Рівень застосунків

Рівень застосунки служить інтерфейсом для обміну даними між користувачем і пристроєм.

Розширений протокол керування чергою повідомлень (AMQP)

Рівень устаткування, на якому встановлюється взаємодія між ПЗ проміжного шару для обміну повідомленнями. Це забезпечує спільну роботу різноманітних систем і застосунків. Так формується стандартизована система обміну повідомленнями в промисловому масштабі.

Обмежений протокол застосунків (CoAP)

Мережний протокол з обмеженням пропускної здатності й доступу до мережі, призначений для пристроїв з обмеженою потужністю. Протокол забезпечує обмін даними між комп'ютерами. Крім того, CoAP – це протокол передачі документів, який працює за допомогою протоколу UDP.

Служба розподілу даних (DDS)

Універсальний одноранговий протокол зв'язку, який дозволяє виконувати будь-які дії – від запуску невеликих пристроїв до підключення високопродуктивних мереж. DDS спрощує розгортання, підвищує надійність і зменшує складність.

MQTT

Протокол обміну повідомленнями, призначений для неінтенсивного обміну даними між комп'ютерами. В основному використовується для підключень із низькою пропускною здатністю до вилучених розташувань. MQTT

використовує підписаний видавцем шаблон. Це ідеальний варіант для невеликих пристроїв, що вимагають ефективної пропускної здатності й використання акумулятора.

Рівень транспортування

Рівень транспортування забезпечує й захищає обмін даними між рівнями.

Протокол TCP

Головний протокол для більшості типів підключення до Інтернету. Забезпечує обмін даними між вузлами. Для цього великі набори даних розбиваються на окремі пакети. При необхідності пакети повторно відправляються й перекомпонуються.

Протокол UDP

Протокол зв'язку для взаємодії між процесами. Працює на основі протоколу IP. Протокол UDP підвищує швидкість передачі даних по протоколу TCP. Це кращий варіант для застосунків, що вимагають передачі даних без втрат.

Рівень мережі

Рівень мережі допомагає окремим пристроям взаємодіяти з маршрутизатором.

6LoWPAN

Версія IPv6 з меншою потужністю, яка скорочує час передачі.

IPv6

Це останнє відновлення протоколу IP для маршрутизації трафіку через Інтернет, а також визначення й виявлення пристроїв у мережі.

Канальний рівень даних

На цьому рівні дані передаються в межах архітектури системи. При цьому визначаються й виправляються помилки, виявлені на фізичному рівні.

IEEE 802.15.4

Стандарт радіочастотних пристроїв для бездротового підключення з низьким енергоспоживанням. Він використовується з Zigbee, 6LoWPAN і іншими стандартами для створення бездротових мереж.

LPWAN

Мережа такого типу забезпечує зв'язок на відстані від 500 метрів. LoRaWAN – це приклад мережі LPWAN, оптимізованої для енергозбереження.

Фізичний рівень

На фізичному рівні встановлюється комунікаційний канал, що забезпечує підключення пристроїв у зазначеному середовищі.

Bluetooth з низьким енергоспоживанням (BLE)

Дозволяє значно скоротити енергоспоживання й витрати. Підтримує такий же діапазон підключення, як і класична технологія Bluetooth. BLE працює прямо в мобільних операційних системах. Ця технологія швидко набуває популярності в сфері побутової електроніки, тому що вона економічна й забезпечує тривалу роботу від акумулятора.

Ethernet

Це економічне провідне з'єднання, яке забезпечує швидке підключення до даних, а також малу затримку.

"Довгостроковий розвиток" (LTE)

Стандарт бездротового широкосмугового зв'язку для мобільних пристроїв і терміналів даних. LTE збільшує потужність і шв идкість бездротових мереж, а також підтримує багатоадресні й широкомовні потоки.

Зв'язок близької дії (NFC)

Набір протоколів зв'язки, що стосуються використання електромагнітних полів. Дозволяє двом пристроям обмінюватися даними на відстані чотирьох сантиметрів друг від друга. Пристрою з підтримкою NFC працюють як карти ключів посвідчень і звичайно використовуються для смарт-карт, безконтактних мобільних платежів і бронювання квитків.

Радіочастотна ідентифікація (RFID)

Використовує електромагнітні поля для відстеження електронних тегів, відомості про яких не можна одержати інакше. Сумісне устаткування забезпечує

електроживлення й взаємодіє із цими тегами, зчитуючи інформацію для ідентифікації й перевірки дійсності.

Wi-Fi/802.11

Стандартний варіант для будинку й роботи. Цей варіант економічний. Але він підходить не для всіх сценаріїв через обмежений діапазон дії й постійного споживання енергії.

Стек технологій Інтернету речей. Платформи Інтернету речей

Платформи Інтернету речей дозволяють легко створювати й запускати проекти Інтернету речей. Такі платформи надають єдину службу, яка управляє розгортанням, пристроями й даними. Платформи Інтернету речей управляють апаратними й програмними протоколами, забезпечують безпека й перевірку дійсності, а також надають користувацькі інтерфейси.

Точне визначення платформи Інтернету речей сформулювати важко. Причина в тому, що є більш 400 постачальників послуг з різними пропозиціями – від програмного забезпечення й устаткування до пакетів SDK і Api-інтерфейсів. Але до складу більшості платформ Інтернету входять:

- хмарний шлюз Інтернету речей;
- засобу перевірки дійсності й керування пристроями, а також Api-інтерфейси;
- хмарна інфраструктура;
- інтеграція сторонніх застосунків.

Керовані служби

Різноманітні керовані служби Інтернету речей допомагають підприємствам працювати на попередження й обслуговувати екосистему Інтернету речей. Такі служби дозволяють реалізувати й спростити створення, розгортання, адміністрування й моніторинг проекту Інтернету речей.

Зв'язок Інтернету речей із сучасними технологіями

Віртуальна реальність і Інтернет речей

Спільне використання віртуальної реальності й Інтернету речей допомагає одержати візуальний контекст складних систем і ухвалювати розв'язки в реальному часі. Наприклад, доповнена (інша назва – змішана) реальність дозволяє створити візуальне накладення зібраних даних. Цю технологію в комбінації з Інтернетом речей також можна застосовувати на практиці в безлічі інших областей. Спільне використання віртуальної реальності й Інтернету речей забезпечує технологічні поліпшення в таких галузях, як охорона здоров'я, виїзне обслуговування, транспортні послуги й виробництво.

Квантові обчислення й Інтернет речей

Квантові обчислення – природній спосіб обробки значного обсягу даних, створюваних Інтернетом речей. Такий підхід дозволяє прискорити роботу завдяки потужним обчислювальним можливостям. Крім того, квантове шифрування допомагає додати рівень безпеки, який необхідний, але поки недосяжний через низьку обчислювальну потужність, якої має більшість пристроїв Інтернету речей.

Блокчейн і Інтернет речей

Зараз немає способу, за допомогою якого можна було б переконатися, що дані з Інтернету речей не оброблялися до їхнього продажу або спільного використання. Блокчейн у комбінації з Інтернетом речей дозволяє розділити приймачі даних і підвищити рівень довіри, щоб можна було перевіряти дані, а також виконувати їхнє трасування й забезпечувати надійність.

Рішення з відкритим кодом і Інтернет речей

Технології з відкритим кодом прискорюють роботу рішень Інтернету речей, дозволяючи розроблювачам використовувати в застосунках з технологіями Інтернету речей засобу за своїм вибором.

Безсерверні обчислення й Інтернет речей

Враховуючи непостійний трафік проектів Інтернету речей, безсерверні обчислення забезпечують економічний спосіб динамічного масштабування. І при цьому не потрібно управляти інфраструктурою.

Kubernetes і Інтернет речей

Завдяки моделі розгортання без простоїв Kubernetes допомагає обновляти проекти Інтернету речей у реальному часі без впливу на роботу користувачів. Kubernetes дозволяє легко й ефективно виконувати масштабування за допомогою хмарних ресурсів, надаючи загальну платформу для розгортання в прикордонному середовищі.

Штучний інтелект і Інтернет речей

Системи Інтернету речей збирають такі величезні обсяги даних, що часто для їхнього сортування й аналізу потрібні засоби ШІ й машинного навчання. Це дозволяє виявляти закономірності й вживати заходів на основі отриманих аналітичних відомостей. Наприклад, засобу ШІ можуть аналізувати дані, зібрані з виробничого устаткування, і прогнозувати потребу в обслуговуванні. Це дозволяє скоротити витрати й час простою через непередбачені збої.

Дані Інтернету речей і аналітика

Технології Інтернету речей створюють настільки більші обсяги даних, що для їхнього перетворення в практичні корисні відомості необхідні спеціалізовані процеси й засобу.

Розповсюджені області застосування технологій Інтернету речей

Обслуговування, що попереджає

Моделі машинного навчання Інтернету речей, розроблені й навчені для виявлення сигналів в історичних даних, можна використовувати для виявлення таких же тенденцій у поточних даних. Це дозволяє користувачам автоматизувати запити на профілактичне обслуговування й запитувати нові компоненти заздалегідь, щоб вони завжди були доступні при необхідності.

Прийняття рішень у реальному часі

Ефективні архітектури аналітики Інтернету речей у реальному часі масштабуються для роботи з великими обсягами даних і зменшення затримки. Є багато служб аналітики Інтернету речей з компонентами для комплексного створення звітів у реальному часі, у тому числі:

– Сховище для великих обсягів даних з використанням форматів, що підтримують запити засобів аналітики.

– Обробка потоків даних великого обсягу для їхньої фільтрації й агрегування перед аналізом.

– Цикли аналізу з малою затримкою з використанням засобів аналітики в реальному часі, які дозволяють створювати звіти й візуалізувати дані.

– Приймання даних у реальному часі за допомогою брокерів повідомлень.

– Розповсюджені завдання технологій Інтернету речей.

Сховище даних

Збір великих обсягів даних приводить до збільшення необхідного обсягу сховища. Є кілька служб сховища даних з відмінностями в організаційній структурі, протоколах перевірки дійсності, обмеженнях розміру та ін.

Обробка даних

Величезний обсяг даних, що збираються в Інтернеті речей, непросто очистити, обробити й інтерпретувати з високою швидкістю. Прикордонні обчислення дозволяють виконувати ці завдання шляхом переносу більшої частини оброблюваних даних із централізованої системи в прикордонну зону мережі, ближче до пристроїв, яких потрібні ці дані. Але при децентралізації обробки даних виникають нові проблеми, що зокрема стосуються надійності й масштабованості прикордонних пристроїв, а також безпеки переданих даних.

Безпека й конфіденційність Інтернету речей

Безпека й конфіденційність Інтернету речей – критично важливі аспекти в будь-якому проекті Інтернету речей. Технології Інтернету речей допоможуть перетворити ваші бізнес-операції. Але пристрою Інтернету речей можуть являти

загрози, якщо вони не захищені належним чином. Кібератаки можуть порушити безпеку даних, ушкодити устаткування й навіть заподіяти шкода здоров'ю людину.

Надійна система кібербезпеки Інтернету речей перевершує стандартні заходи забезпечення конфіденційності. Ця система дозволяє моделювати погрози. Знання різних способів, за допомогою яких зловмисники можуть порушити безпеку системи, – перший крок до запобігання атак.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Функціональна схема складається з двох великих блоків:

– Блок визначення функцій моніторингу IoT.

– Блок визначення об’єктів моніторингу IoT.

Блок визначення функцій моніторингу IoT складається з наступних блоків:

– Робота з ресурсами мережі.

– Робота з сесіями.

– Моніторинг трафіку IoT.

– Робота з пристроями IoT.

– Монітор з'єднань.

– Статистика подій.

– Функції для роботи з мережею.

Розглянемо детальніше кожний з блоків.

Робота з ресурсами мережі включає в себе:

– Визначення доступних ресурсів IoT.

– Закриття локального ресурсу.

– Відкриття локального ресурсу.

– Приховання й показ ресурсів IoT.

Система відображає наявні у мережі ресурси у вигляді дерева. Пошук ресурсів IoT можна здійснювати по заданим умовам: локальні чи глобальні ресурси; всі ресурси, тільки файли, чи тільки принтери, тощо.

Можна додавати до загальних ресурсів IoT мережі свої власні, а також закривати їх потім.

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 IoT включає в себе:

– Визначення підключених інтерфейсів.

– Визначення вхідного та вихідного трафіку IoT.

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 пристроями IoT включає в себе:

– Одержання списку відкритих пристроїв IoT.

– Закриття відкритого пристрою IoT.

Можна переглянути, які з Ваших пристроїв IoT, що Ви відкрили для загального доступу, переглядають по мережі. Програма відобразить список пристроїв IoT 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.

З рисунку видно, що блок визначення об’єктів моніторингу IoT локальної мережі складається з трьох блоків:

– Моніторинг трафіку IoT.

– Моніторинг обладнання IoT.

– Моніторинг ресурсів IoT.

Моніторинг трафіку IoT використовується для контролю вхідного та вихідного трафіку IoT. Він включає у себе контроль підключених інтерфейсів, статистику подій по основним мережним протоколам: TCP, UDP, IP та ICMP.

TCP – один з основних мережних протоколів Інтернету, призначений для управління передачею даних в мережах і підмережах TCP/IP.

Рисунок 3.2 – Функціональна схема системи

UDP – один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання. UDP – це один з найпростіших протоколів транспортного рівня моделі OSI, котрий виконує обмін дейтаграмами без підтвердження та гарантії доставки.

IP – найбільш широко розповсюджена реалізація ієрархічної схеми мережної адресації. Використовуваний в мережі Інтернет, протокол відповідає за адресацію пакетів, але не відповідає за встановлення з'єднань, не є надійним і дозволяє реалізувати тільки негарантовану доставку даних.

ICMP – мережний протокол, що входить в стек протоколів TCP/IP. В основному ICMP використовується для передачі повідомлень про помилки й інші виняткові ситуації, що виникли при передачі даних. Також на ICMP покладають деякі сервісні функції, зокрема на основі цього протоколу заснована дія таких загальновідомих утиліт як ping та traceroute.

Моніторинг обладнання IoT включає в себе побудову списку наявного обладнання IoT та здійснення його контролю. До мережного обладнання IoT, що підлягає моніторингу IoT, відносяться: персональні комп'ютери, ноутбуки, сервери, принтери, IP-телефони.

Моніторинг ресурсів IoT дозволяє переглядати та завантажувати наявні в мережі ресурси, а також розміщувати чи приховувати для загального доступу свої ресурси. До ресурсів IoT локальної мережі відносяться: файли, мультимедіа, бази даних, сервіси інформаційної безпеки, список користувачів.

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

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

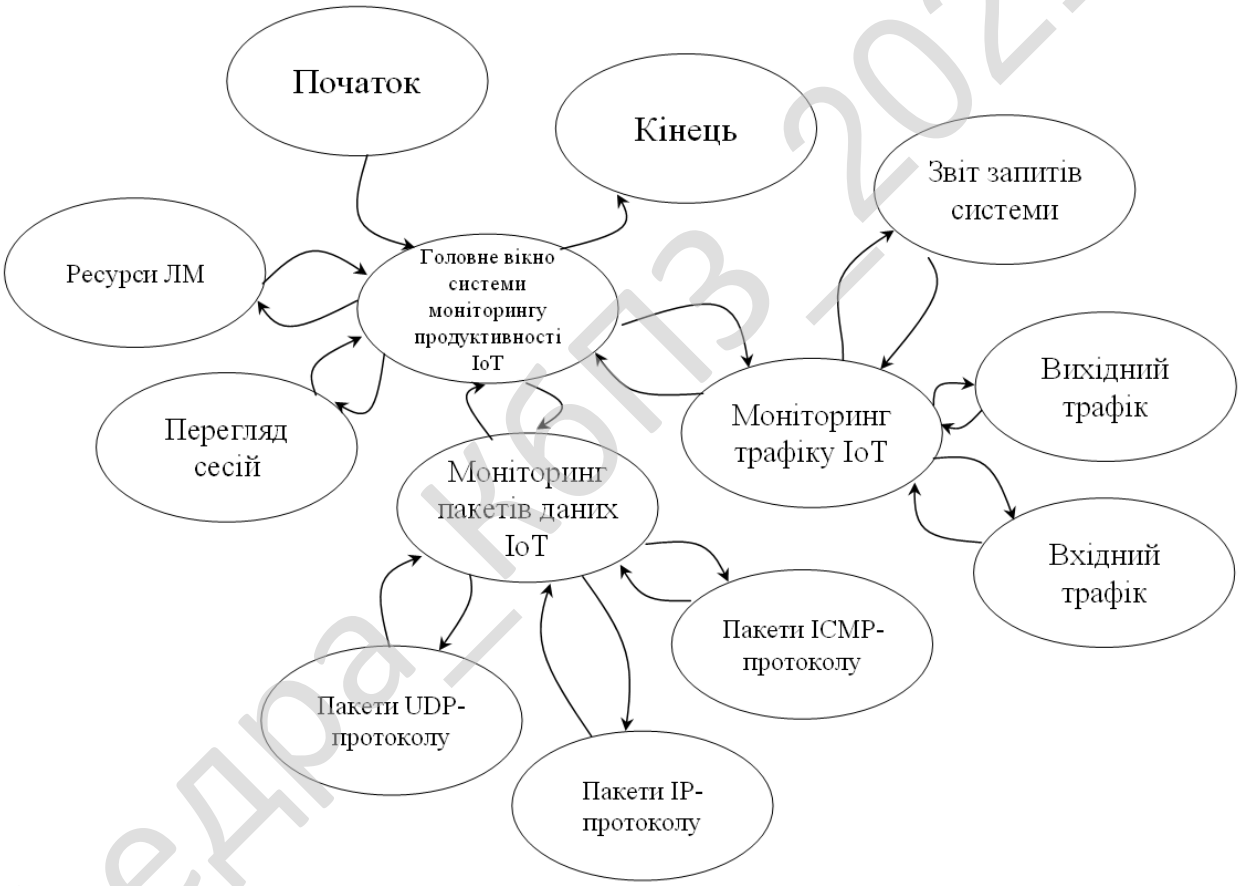

Рисунок 3.3 – Діаграма взаємодії процесів

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

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

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над програмою 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.

Розглянемо алгоритм роботи основної програми у вигляді блок-схеми зображеної на рисунку 4.1. Вона складається з наступних функціональних блоків:

– Виведення вікна системи моніторингу продуктивності IoT.

– Пошук IoT пристроїв ЛМ.

– Побудова дерева ресурсів IoT ЛМ.

– Виведення на екран дерева IoT ресурсів ЛМ.

– Виведення відкритих IoT пристроїв.

– Виведення відкритих сесій IoT пристроїв.

– Запит відкрити IoT пристрій.

– Відкриття ресурсу пристрою, обчислення продуктивності.

– Запит закрити ресурс пристрою.

– Закриття ресурсу пристрою.

– Підпрограма моніторингу (рис 4.2).

– Виведення вихідного трафіку.

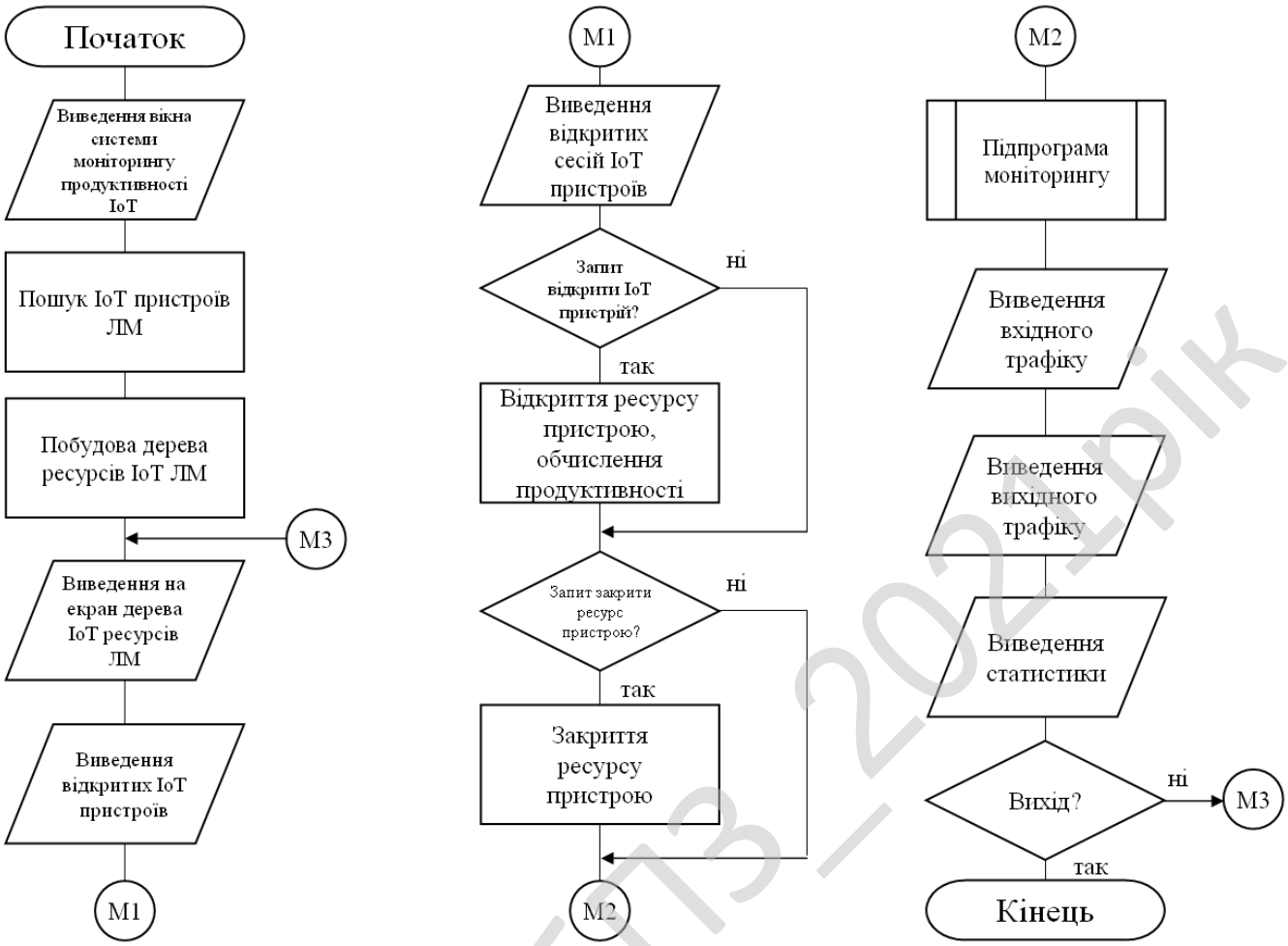

Рисунок 4.1 – Блок-схема основної програми

– Виведення статистики.

– Вихід (кінець циклу).

На рисунку 4.2 зображено роботу підпрограми з наступними функціональними блоками:

– Додати ресурс IoT.

– Введення параметрів IoT ресурсу.

– Введення імені IoT ресурсу.

– Встановлення прав доступу до IoT ресурсу.

– Встановлення максимальної кількості підключень до IoT ресурсу.

– Додавання IoT ресурсу.

– Захистити IoT ресурс паролем.

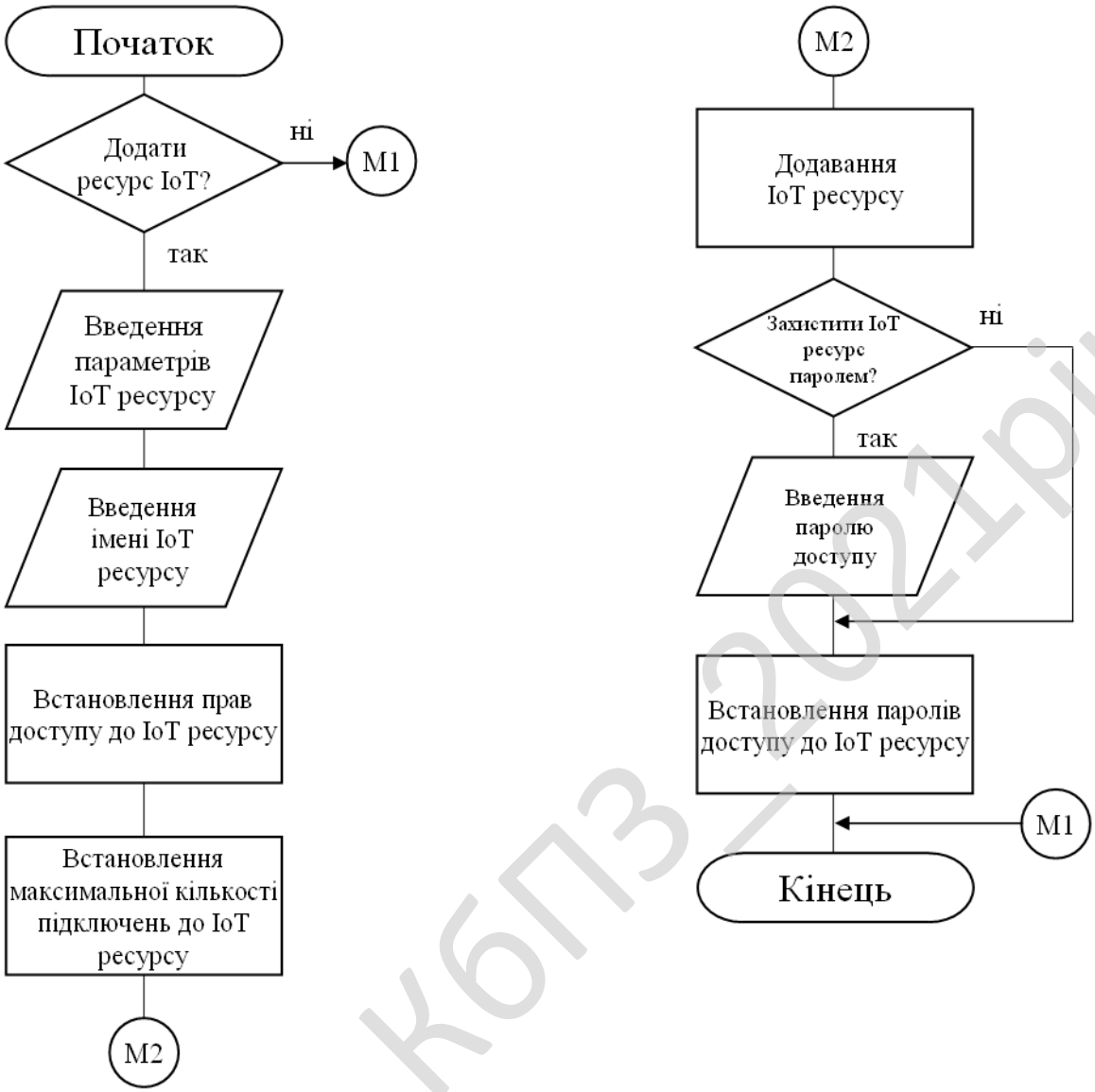

Рисунок 4.2 – Блок-схема роботи підпрограми

– Введення паролю доступу.

– Встановлення паролів доступу до IoT ресурсу.

Розглянемо модуль роботи з мережею. Як відомо IOT пристрої по суті являють собою мережний пристрій зі своїм набором параметрів при підклчені до цих пристроїв проводиться налаштування та отримання необхідної інформації. Вихідний код наступний:

```
unit uM;
// Юніт системи
```

```
interface
//Інтерфейс юніта

Uses
// підключення юніта
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, CommCtrl, Winsock;

Const
// константи юніта – опис ситуації у мережі
    RES_UNKNOWN = 'Unknown';
    RES_IP      = 'IP adress: ';
    RES_CMP     = 'Computer name: ';
    RES_USR     = 'User name: ';
    RES_DOM     = 'Domen: ';
    RES_SER     = 'Domen server: ';
    RES_COM     = 'Comment: ';
    RES_PROV    = 'Provider: ';
    RES_GRP     = 'Groups: ';
    RES_MAC     = 'MAC adress: ';
    RES_SHARES  = 'Available shares: ';
    RES_TIME    = 'Expended time: ';
    RES_COM_NO  = 'Absent';

// Для роботи з ARP (Address Resolution Protocol) таблицею
  IPHLPAPI = 'IPHLPAPI.DLL';
  MAX_ADAPTER_ADDRESS_LENGTH = 7;

type
  LMSTR = LPWSTR;
  NET_API_STATUS = DWORD;

// Наступні три типи використовуються для роботи з Iphlpapi.dll
  TMacAddress = array[0..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

// Це структура для одиничного запиту
  TMibIPNetRow = packed record
    dwIndex         : DWORD;
    dwPhysAddrLen   : DWORD;
    bPhysAddr       : TMACAddress;
    dwAddr          : DWORD;
    dwType          : DWORD;
```

```
end;
TMibIPNetRowArray = array [0..512] of TMibIPNetRow;

// запитувана структура
PTMibIPNetTable = ^TMibIPNetTable;
TMibIPNetTable = packed record
  dwNumEntries    : DWORD;
  Table: TMibIPNetRowArray;
end;

// Структура для перерахування користувачів
_WKSTA_USER_INFO_1 = record
  wkui1_username: LPWSTR;
  wkui1_logon_domain: LPWSTR;
  wkui1_oth_domains: LPWSTR;
  wkui1_logon_server: LPWSTR;
end;
WKSTA_USER_INFO_1 = _WKSTA_USER_INFO_1;
PWKSTA_USER_INFO_1 = ^_WKSTA_USER_INFO_1;
LPWKSTA_USER_INFO_1 = ^_WKSTA_USER_INFO_1;

// Структура для визначення приналежності користувача до груп
PGroupUsersInfo0 = ^_GROUP_USERS_INFO_0;
_GROUP_USERS_INFO_0 = packed record
  grui0_name: LPWSTR;
end;
TGroupUsersInfo0 = _GROUP_USERS_INFO_0;
GROUP_USERS_INFO_0 = _GROUP_USERS_INFO_0;

// Структура для отределенія доступних мережевих ресурсів
PSHARE_INFO_1 = ^SHARE_INFO_1;
_SHARE_INFO_1 = record
  shi1_netname: LMSTR;
  shi1_type: DWORD;
  shi1_remark: LMSTR;
end;
SHARE_INFO_1 = _SHARE_INFO_1;
TShareInfo1 = SHARE_INFO_1;
PShareInfo1 = PSHARE_INFO_1;

TMQW = class(TForm)
  btnGetInfo: TButton;
```

```
    memInfo: TMemo;
    Label1: TLabel;
    procedure btnGetInfoClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    IP, Font: Integer;

// Це змінні для роботи з WC_IPADDRESS классом
    edIP: HWND;
    function GetNameFromIP(const IP: String): String;
    function GetUsers(const CompName: String): String;
    function GetDomain(const CompName, Provider: String): String;
    function GetComment(CompName, Provider: String): String;
    function GetProvider(const CompName: String): String;
    function GetMacFromIP(const IP: String): String;
    function GetDomainServer(const DomainName: String): String;
    function GetGroups(DomainServer: String; UserName: String): String;
    function GetShares(const CompName: String): String;
  end;
  // Оголосимо функції, тут йде статична завантаження бібліотек
function WNetGetResourceInformation(lpNetResource: PNetResource;
lpBuffer: Pointer; var lpcbBuffer: DWORD;
lplpSystem: Pointer): DWORD; stdcall;

function GetIpNetTable(pIpNetTable: PTMibIPNetTable;
    pdwSize: PULONG; bOrder: Boolean): DWORD; stdcall;

function WNetGetResourceInformation; external mpr name
        'WNetGetResourceInformationA';
function GetIpNetTable; external IPHLPAPI name 'GetIpNetTable';
function NetGetAnyDCName(servername: LPCWSTR;  domainname: LPCWSTR;
                         bufptr: Pointer): Cardinal;
                         stdcall; external 'netapi32.dll';

function NetShareEnum(servername: LMSTR; level: DWORD; var bufptr: Pointer;
prefmaxlen: DWORD; entriesread, totalentries, resume_handle: LPDWORD):
NET_API_STATUS; stdcall; external 'Netapi32.dll';

function  NetApiBufferFree(buffer:  Pointer):  Cardinal;  stdcall;  external
          'netapi32.dll';
```

```
function NetWkstaUserEnum(ServerName: LPCWSTR;
                          Level: DWORD;
                          BufPtr: Pointer;
                          PrefMaxLen: DWORD;
                          EntriesRead: LPDWORD;
                          TotalEntries: LPDWORD;
                          ResumeHandle: LPDWORD): LongInt; stdcall;
                          external 'netapi32.dll';

function NetUserGetGroups(ServerName: LPCWSTR;
                          UserName: LPCWSTR;
                          level: DWORD;
                          bufptr: Pointer;
                          prefmaxlen: DWORD;
                          var entriesread: DWORD;
                          var totalentries: DWORD): LongInt; stdcall;
                          external 'netapi32.dll';
var
  MainForm: TMQW;

implementation
// РЕАЛІЗАЦІЯ МОДУЛЮ
{$R *.dfm}

// Для введення IP адреси будемо використовувати клас WC_IPADDRESS
procedure TMQW.FormCreate(Sender: TObject);
begin
  // Задамо початковий IP адреса (це адреса локальної машини)
  IP := MAKEIPADDRESS(192, 168, 2, 108);
  // Ініціалізіруем додаткові класи бібліотеки ComCtl32.dll.
  InitCommonControl(ICC_INTERNET_CLASSES);
  // Створюємо вікно
  edIP:= CreateWindow(WC_IPADDRESS, nil, WS_CHILD or WS_VISIBLE,
   6, 16, MainForm.Width-22, 21, MainForm.Handle, 0, hInstance, nil);
  // Зазначимо йому якийсь IP показувати
  SendMessage(edIP, IPM_SETADDRESS, 0, IP);
  Font := CreateFont(-11, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH or FF_DONTCARE, 'MS Sans Serif');
  SendMessage(edIP, WM_SETFONT, Font, 0);
end;
```

```
// Інфо
procedure TMQW.btnGetInfoClick(Sender: TObject);
var
  TmpCompName, TmpProvider, TmpGroup, TmpUser, TmpServer: String;
  Time: Cardinal;
  IPStr: String;
begin
  Time := GetTickCount;
// Початок відрізку часу

SendMessage(edIP, IPM_GETADDRESS, 0, Longint(PDWORD(@IP)));
IPStr := IntToStr(FIRST_IPADDRESS(IP));
IPStr := IPStr + '.' + IntToStr(SECOND_IPADDRESS(IP));
IPStr := IPStr + '.' + IntToStr(THIRD_IPADDRESS(IP));
IPStr := IPStr + '.' + IntToStr(FOURTH_IPADDRESS(IP));
  with memInfo, memInfo.Lines do
// виведення інформації
  begin
    Clear;
// очищаємо екран
    Refresh;
// Оновлюємо
    Add(RES_IP + IPStr);
// Виводимо IP адреса
    TmpCompName := GetNameFromIP(IPStr);
    if TmpCompName = RES_UNKNOWN then Exit;
    Add(RES_CMP + TmpCompName);
// Виводимо ім'я комп'ютера
    TmpUser := GetUsers(IPStr);
    Add(RES_USR + TmpUser);
// Виводимо ім'я користувача
    TmpProvider := GetProvider(TmpCompName);
    Add(RES_PROV + TmpProvider);
// виводимо провайдера
    Add(RES_COM + GetComment(TmpCompName,
      TmpProvider));
// Виводимо коментар до ресурсу
    TmpGroup := GetDomain(TmpCompName, TmpProvider);
    Add(RES_DOM + TmpGroup);
// виводимо групу
    TmpServer := GetDomainServer(TmpGroup);
    if TmpServer <> '' then
```

```
    begin
      Add(RES_SER + TmpServer);
// Виводимо ім'я сервера
      Add(RES_GRP + GetGroups(TmpServer, TmpUser));
// Виводимо групи домену в які входить користувач
    end;
    Add(RES_SHARES + GetShares(TmpCompName));
// Виводимо список доступних ресурсів
    Add(RES_MAC + GetMacFromIP(IPStr));
// Виводимо MAC адреса
    Add(RES_TIME + IntToStr(GetTickCount - Time));
// Скільки часу витрачено
  end;
end;

function TMQW.GetNameFromIP(const IP: String): String;
var
  WSA: TWSAData;
  Host: PHostEnt;
  Addr: Integer;
  Err: Integer;
begin
  Result := RES_UNKNOWN;
  Err := WSAStartup(WSA_TYPE, WSA);
  if Err <> 0 then
  begin
    ShowMessage(SysErrorMessage(GetLastError));
    Exit;
  end;
  try
    Addr := inet_addr(PChar(IP));
    if Addr = INADDR_NONE then
    begin
      ShowMessage(SysErrorMessage(GetLastError));
      WSACleanup;
      Exit;
    end;
    Host := gethostbyaddr(@Addr, SizeOf(Addr), PF_INET);
    if Assigned(Host) then
// обов'язкова перевірка
      Result := Host.h_name
    else
```

```
      ShowMessage(SysErrorMessage(GetLastError));
  finally
    WSACleanup;
  end;
end;

// Перераховуємо всіх користувачів
function TMQW.GetUsers(const CompName: String): String;
var
  Buffer, tmpBuffer: Pointer;
  PrefMaxLen       : DWORD;
  Resume_Handle    : DWORD;
  EntriesRead      : DWORD;
  TotalEntries     : DWORD;
  I, Size          : Integer;
  PSrvr            : PWideChar;
begin
  PSrvr := nil;
  try
    // Переводимо ім'я комп'ютера типу PWideChar
    Size := Length(CompName);
    GetMem(PSrvr, Size * SizeOf(WideChar) + 1);
    StringToWideChar(CompName, PSrvr, Size + 1);
    PrefMaxLen := DWORD(-1);
    EntriesRead := 0;
    TotalEntries := 0;
    Resume_Handle := 0;
    Buffer := nil;
    // Отримуємо список користувачів на комп'ютері з PSrvr
    if NetWkstaUserEnum( PSrvr, 1, @Buffer, PrefMaxLen, @EntriesRead,
      @TotalEntries, @Resume_Handle) = S_OK then
    begin
      tmpBuffer := Pointer(DWORD(Buffer) + SizeOf(WKSTA_USER_INFO_1));
      for I := 1 to TotalEntries - 1 do
      begin
        Result := Result + WKSTA_USER_INFO_1(tmpBuffer^).wkui1_username +
                  ', ';
        tmpBuffer := Pointer(DWORD(tmpBuffer) + SizeOf(WKSTA_USER_INFO_1));
      end;
      Result := Copy(Result, 1, Length(Result) - 2);
  end
  else
```

```
      ShowMessage(SysErrorMessage(GetLastError));
    finally
      NetApiBufferFree(Buffer);
      FreeMem(PSrvr);
    end;
  end;

  // рекурсивне сканування ресурсів
  function TMQW.GetComment(CompName, Provider: String): String;
  var
    StopScan: Boolean;
    TmpRes: TNetResource;

    // сканування
    procedure Scan(Res: TNetResource; Root: boolean);
    var
      Enum, I: Cardinal;
      ScanRes: array [0..512] of TNetResource;
      Size, Entries, Err: DWORD;
    begin

      if StopScan then Exit;
  // Використовуємо прапор для виходу з рекурсії
      if Root = True then
        Err := WNetOpenEnum(RESOURCE_GLOBALNET, RESOURCETYPE_DISK,
               0, nil, Enum)
      else
        Err := WNetOpenEnum(RESOURCE_GLOBALNET, RESOURCETYPE_DISK,
          0, @Res, Enum);
      if Err = NO_ERROR then
      begin
        Size := SizeOf(ScanRes);
        Entries := DWORD(-1);
        Err := WNetEnumResource(Enum, Entries, @ScanRes, Size);
        if Err = NO_ERROR then
        try
          for I := 0 to Entries - 1 do
          begin
            if StopScan then Exit;
  // Ще один прапор, так як вихід на верхній виклик
            with ScanRes[i] do
            begin
```

```
            if dwDisplayType = RESOURCEDISPLAYTYPE_SERVER then
              if lpRemoteName = CompName then
              begin
                Result :=  lpComment;
                StopScan := True;
// виставляємо прапор для виходу з рекуссіі
                Exit;
              end;
            if dwDisplayType <> RESOURCEDISPLAYTYPE_SERVER then
              Scan(ScanRes[i], False);
          end;
        end;
      finally
        WNetCloseEnum(Enum);
      end
      else
        if Err <> ERROR_NO_MORE_ITEMS then
// Немає елементів для відображення
          MessageDlg(SysErrorMessage(GetLastError), mtError, [mbOK], 0);
    end
    else
      ShowMessage(SysErrorMessage(GetLastError));
  end;

// Основна процедура
begin
  Result := RES_UNKNOWN;
  if CompName = RES_UNKNOWN then Exit;
// Якщо ім'я компа, не знайдено, зупиняємося
  CompName := '\\' + CompName;
StopScan := False;
// Знімемо прапор виходу з рекурсії.
// Тут обов'язково ініціалізіроваліе змінної
// типу Boolean
// Запускаємо сканування ...
  Scan(TmpRes, True);
  if Result = '' then Result := RES_COM_NO;
end;

function TMQW.GetDomain(const CompName, Provider: String): String;
var
  CurrRes: TNetResource;
```

```
  ParentName: array [0..1] of TNetResource;
  Enum: DWORD;
  Err: Integer;
begin
  with CurrRes do
  begin
    dwScope := RESOURCE_GLOBALNET;
    dwType := RESOURCETYPE_DISK;
    dwDisplayType := RESOURCEDISPLAYTYPE_SERVER;
    dwUsage := RESOURCEUSAGE_CONTAINER;
    lpLocalName := '';
    lpRemoteName := PChar('\\' + CompName);
    lpComment := '';
    lpProvider := PChar(Provider);
  end;
  Enum := SizeOf(ParentName);
  Err := WNetGetResourceParent(@CurrRes, @ParentName, Enum);
  if Err = NO_ERROR then
  begin
    Result := ParentName[0].lpRemoteName;
    if Result = '' then Result := RES_COM_NO;
  end
  else
    ShowMessage(SysErrorMessage(GetLastError));
end;

// дізнатися провайдера
function TMQW.GetProvider(const CompName: String): String;
var
  Buffer: array [0..255] of Char;
  Size: DWORD;
begin
  Size := SizeOf(Buffer);
  if WNetGetProviderName(WNNC_NET_LANMAN, @Buffer, Size) <> NO_ERROR then
    Result := RES_COM_NO
  else
    Result := String(Buffer);
end;

// будується повна ARP таблиця, на підставі якої
// можемо спокійно провести вибірку по потрібному IP адресою
function TMQW.GetMacFromIP(const IP: String): String;
```

```
        function GetMAC(Value: TMacAddress; Length: DWORD): String;
        var
          I: Integer;
        begin
          if Length = 0 then Result := '00-00-00-00-00-00' else
          begin
            Result := '';
            for i:= 0 to Length -2 do
              Result := Result + IntToHex(Value[i], 2) + '-';
            Result := Result + IntToHex(Value[Length-1], 2);
          end;
        end;
      // Отримуємо IP адреса  function GetDottedIPFromInAddr(const InAddr:
Integer): String;
        begin
          Result := '';
          Result := IntToStr(FOURTH_IPADDRESS(InAddr));
          Result := Result + '.' + IntToStr(THIRD_IPADDRESS(InAddr));
          Result := Result + '.' + IntToStr(SECOND_IPADDRESS(InAddr));
          Result := Result + '.' + IntToStr(FIRST_IPADDRESS(InAddr));
        end;
        // Основна функція
      var
        Table: TMibIPNetTable;
        Size: Integer;
        CatchIP: String;
        Err, I: Integer;
      begin
        Result := RES_UNKNOWN;
        Size := SizeOf(Table);
        Err := GetIpNetTable(@Table, @Size, False);
        if Err <> NO_ERROR then
      // Перевірка на помилку
        begin
          ShowMessage(SysErrorMessage(GetLastError));
          Exit;
        end;
        // Тепер ми маємо таблицю з IP адрес і відповідних їм MAC адрес
        for I := 0 to Table.dwNumEntries - 1 do
      // необхідний IP
        begin
          CatchIP := GetDottedIPFromInAddr(Table.Table[I].dwAddr);
```

```
    if CatchIP = IP then
// MAC ...
    begin
      Result := GetMAC(Table.Table[I].bPhysAddr,
              Table.Table[I].dwPhysAddrLen);
      Break;
    end;
  end;
end;

// Отримання доступних мережевих ресурсів на віддаленому комп'ютері
function TMQW.GetShares(const CompName: String): String;
type TShareInfo1Array = array of TShareInfo1;
var
  entriesread, totalentries: DWORD;
  Info: Pointer;
  I: Integer;
  CN: PWideChar;
begin
  CN := StringToOleStr(CompName);
  if NetShareEnum(CN, 1, Info, DWORD(-1), @entriesread,
    @totalentries, nil) = 0 then
    try
// список ресурсів
      if entriesread > 0 then
        for I := 0 to entriesread - 1 do
         Result := Result + TShareInfo1Array(@(Info^))[I].
                shi1_netname + ' ';
    finally
      NetApiBufferFree(Info);
    end;
end;

// ім'я сервера домена
function TMQW.GetDomainServer(const DomainName: String): String;
var
  Domain: PWideChar;
  Server: PWideChar;
begin
  GetMem(Domain, MAX_PATH);
  try
    StringToWideChar(DomainName, Domain, MAX_PATH);
```

```
    if NetGetAnyDCName(nil, Domain, @Server)= NO_ERROR then
    try
      Result := WideCharToString(Server);
    finally
      NetApiBufferFree(Server);
   end;
   finally
     FreeMem(Domain, MAX_PATH);
   end;
end;
end.
```

Була використана водоспадна (каскадна) модель життєвого циклу ПЗ (waterfall model) – послідовний метод розробки програмного забезпечення, названий так через діаграму схожу на водоспад.

Ця модель розробки запозичена з системної інженерії у виробництві та будівництві – областях, в яких зміни на пізніх етапах дуже дорогі, або неможливі. Наприклад, для створення складних інженерних конструкцій (споруд, літаків, мостів і т.п.). Зміни в проекті фундаменту будинку після того, як покладений дах коштують дуже дорого, тому перфекціонізм на початкових етапах проектування просто необхідний. Інженери, які починали займатись розробкою програмного забезпечення перейшовши з інших галузей, просто адаптували звичну модель, тому що на ранніх етапах розвитку комп'ютерної техніки не було методологій створених саме для програмування. Проте, схожі методології застосовуються для програмного забезпечення й далі, у випадках коли вимоги фіксовані, і вимагається висока якість та надійність, наприклад в системах для військових чи медичних потреб.

Перший формальний опис водоспадної моделі, після якої вона стала популярною був здійснений В. В. Ройсом у 1970. Попри те, що стаття містить переважно критику методу, на неї часто посилаються.

Переваги методу:

– Ніяких переробок.

– Гарна специфікація перетікає в гарну документацію.

– Зрозуміла модель.

– Розробники можуть мати низьку кваліфікацію.

Недоліки:

– Необхідний перфекціонізм на кожному етапі.

– Важко вносити зміни (якщо взагалі можливо).

– Надлишкове проектування.

– Поділ розробників на "perfect" та "code monkeys".

Модифікації. Через те що цей метод погано підходить для розробки саме ПЗ, частіше використовують його модифікації.

Найвідоміша модифікація – Sashimi. Названа так через японську страву сашимі (суші нарізане і сервіроване так, що складені рядочком шматочки накладаються один на одного). В моделі розробки Сашимі фази життєвого циклу йдуть одна за одною, але при цьому перекриваються одна з одною в часі.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-1, який заснований на перетвореннях алгоритму SHA-1. SHACAL-1 шифрує 160-бітний блок даних з використанням 512-бітного ключа шифрування. Допускається використання більше коротких ключів шифрування (не коротше 128 біт), які перед виконанням розширення ключа (процедура розширення ключа також успадкована від SHA-1 і буде описана нижче) повинні бути доповнені нульовими бітами для досягнення 512-бітного розміру.

В алгоритмі SHACAL-1 передбачено 80 раундів шифрування. Шифруєме повідомлення представляється у вигляді п'яти 32-бітних субблоків A, B, C, D і E, над якими в кожному раунді виконуються наступні дії:

$$A_{i+1} = K_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + M_i,$$

$$B_{i+1} = A_i,$$

$$C_{i+1} = B_i <<< 30,$$

$$D_{i+1} = C_i,$$

$$E_{i+1} = D_i,$$

де i – номер раунду (i = 0...79),

K_i – фрагмент розширеного ключа для і-го раунду,

f_i – функція для і-го раунду (див. нижче),

<<< – операція побітового циклічного зрушення вліво,

M_i – константи, що модифікують, певні в такий спосіб:

Раунди	Значення константи
0...19	5A827999
20...39	6ED9EBA1
40...59	8F1BBCDC
60...79	CA62C1D6

Використовувані в раундах функції f_i визначені так:

Раунди	Функція
0...19	$f(x,y,z)=(x\&y)\|(x'\&z)$
20...39,60…79	$f(x,y,z)=x \oplus y \oplus z$
40...59	$f(x,y,z)=(x \oplus y)\|(x \oplus z)\|(y \oplus z)$

У таблиці символами &, | і ⊕ позначені, відповідно, побітові логічні операції «і», «або» й «або, що виключає» (XOR); x' позначає побітовий комплемент до x.

Шифртекстом є конкатенація вмісту змінних A_{80}, B_{80}, C_{80}, D_{80} і E_{80}.

Процедура розширення ключа в алгоритмі SHACAL-1 також досить проста, вона виконується у два етапи:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта K_0…K_{15}...

Етап 2. Інші фрагменти розширеного ключа K_{16}…K_{79} обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = (K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}) <<< 1.$$

Раунди розшифрування виконуються у зворотній послідовності; кожний з них має на увазі виконання наступних операцій:

$$A_i = B_{i+1},$$

$$B_i = C_{i+1} <<< 2,$$

$$C_i = D_{i+1},$$

$$D_i = E_{i+1},$$

$$E_i = K'_i + (B_{i+1} <<< 5)' + f_i'(C_{i+1} <<< 2, D_{i+1}, E_{i+1}) + A_{i+1} + M'_i + 4.$$

Тут запис $f'(x)$ позначає побітовий комплемент результату виконання операції $f(x)$.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Функціональних кнопок роботи з ресурсом розробленого ПЗ.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Верхнього меню: Файл; Ресурси; Сесії; Трафік; Параметри; Довідка.

– Розділу обрання інтерфейсу.

– Розділу результату роботи системи.

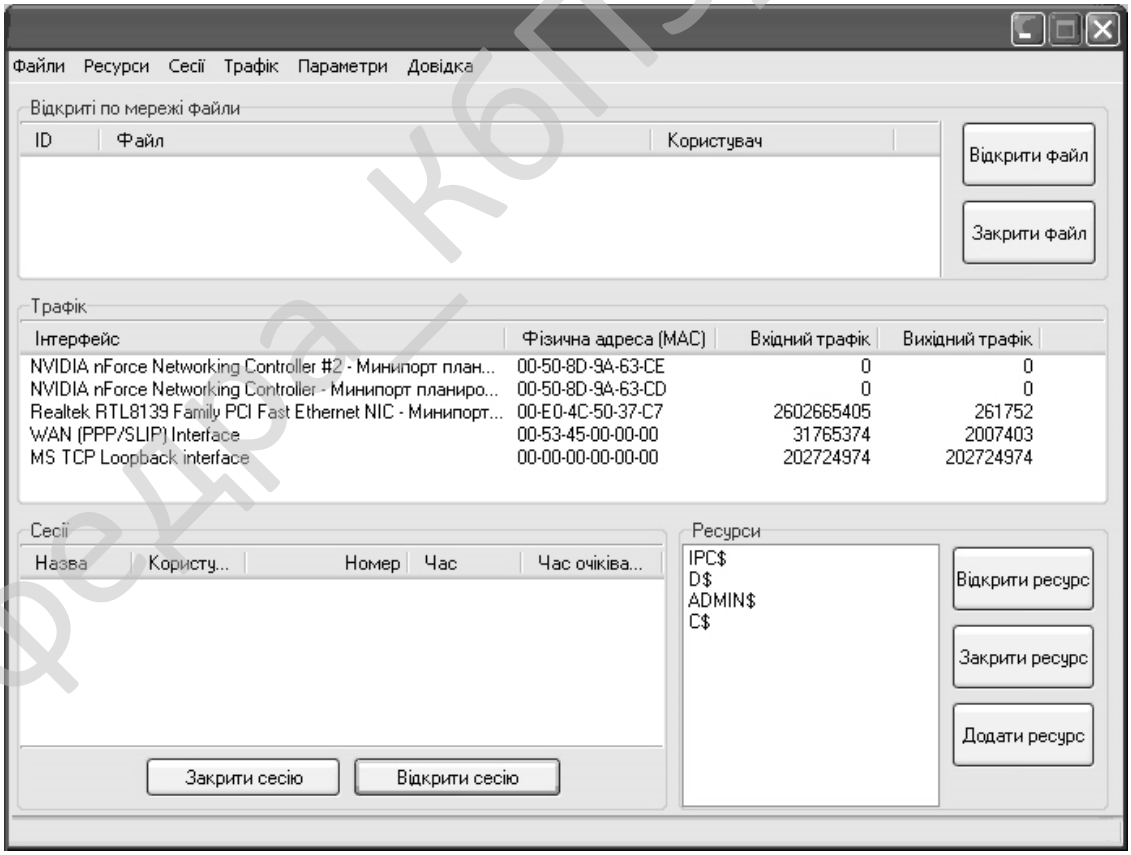

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

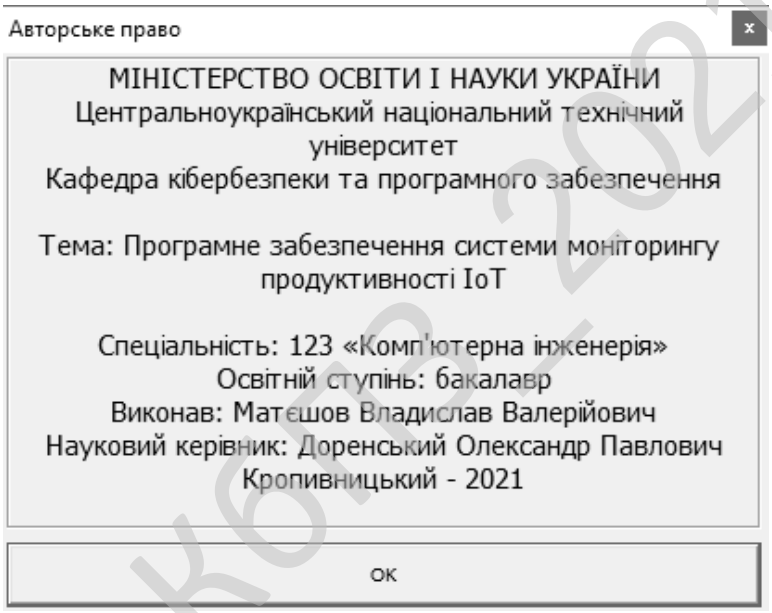

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи моніторингу продуктивності IoT.

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 продуктивності IoT.

– Досліджена система моніторингу продуктивності IoT.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу продуктивності IoT.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання моніторингу продуктивності IoT.

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 RAD Studio Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи моніторингу продуктивності IoT. 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 XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-1.

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. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. – 2-е изд. – СПб.: Питер, 2007. – 958 с.

2. Партыка С.А. Метод ускоренной коррекции SPT с использованием динамических алгоритмов [Електронний ресурс]. – Режим доступу до ресурсу: http://openarchive.nure.ua/bitstream/123456789/936/1/ASU_158_2012%20%2842-47%29.pdf

3. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

4. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

5. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

6. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

7. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. –

Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Рос сийский университет кооперации". – 2012. – Том.1. – С. 152-160.

8. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

9. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

10. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

11. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

12. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

13. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

14. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

15. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

16. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

17. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

18. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

19. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

20. Смирнов С. А. Комплекс GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. – практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

21. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

22. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

23. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

24. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

25. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

26. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та

захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

27. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

28. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

29. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

30. Смирнов С. А. Технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

31. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

32. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

33. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

34. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

35. Смирнов С. А. Разработка комплекса GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук. -практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

36. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

37. Смирнов С. А. GERT-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського

суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

38. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук. - практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

39. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

40. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

41. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук. -практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

42. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук. -

практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

43. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня – 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

44. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

45. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

46. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

47. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая – 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

48. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. – техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

49. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

50. Столлингс В. Современные компьютерные сети / Вильям Столлингс. – СПб.: Питер, 2003. – 778 с.

51. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

52. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

53. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

54. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

55. Elwalid, D. Mitra, I. Saniee, and I. Widjaja. Routing and Protection in GMPLS Networks: From Shortest Paths to Optimized Designs // Journal of lightwave technology. – 2003. – №21(11), P. 2828-28-38.

56. A.B. Bagula, M. Botha, and A.E Krzesinski. Online Traffic Engineering: The Least Interference Optimization Algorithm // IEEE Communications Society – 2004, P. 1232-1236.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					КБР-123.21.0035.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Матєшов В.В.			Програмне забезпечення системи моніторингу продуктивності IoT	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Доренський О.П.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-18-3СК	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи моніторингу продуктивності IoT.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 204-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи моніторингу продуктивності IoT.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи 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 продуктивності IoT;

– 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 архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 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 XP/Vista/7/8/10.

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 RAD Studio Delphi 10.4 Sydney.

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 – 93 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

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.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2020 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 3.06.2020 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Доренський О.П.

Програмне забезпечення системи моніторингу продуктивності IoT

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2021 року

Файл TCP_IP.pas- монітор TCP/IP з'єднань IoT

```
unit TCP_IP;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;

procedure TForm2.btRTTIClick(Sender: TObject);
var
```

```
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Відбулася помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( ' Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл Stat.pas- статистика мережі IoT

```
unit Stat;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
```

```
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Основна програма

Файл Monitor_IoT. dpr основної програми

```
program Monitor_IoT;

uses
  Forms,
  Main in ' Main.pas'  {MainForm},
  About in ' About.pas'  {Form1},
  TCP_IP in ' TCP_IP.pas'  {Form2},
  Stat in ' Stat.pas'  {Form3};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

Файл Main.pas основної програми

```
unit Main;

interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
```

```
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
         ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
  //  function OpenEnum(const NetContainerToOpen: PNetResource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_netname : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
   Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
```

```
    Sesi50_username    : PChar;
    sesi50_key         : Cardinal;
    sesi50_num_conns   : Word;
    sesi50_num_opens   : Word;
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;

type
  TFileInfo3 = packed record
    fi3_id          : DWORD;
    fi3_permissions : DWORD;
    fi3_num_locks   : DWORD;
    fi3_pathname    : PWChar;
    fi3_username    : PWChar;
  end;
  PFileInfo3 = ^TFileInfo3;
  TFileInfo3Array = array[0..512] of TFileInfo3;
  PFileInfo3Array = ^TFileInfo3Array;

type
  TFileInfo50 = packed record
    fi50_id          : Cardinal;
    fi50_permissions : WORD;
    fi50_num_locks   : WORD;
    fi50_pathname    : PChar;
    fi50_username    : PChar;
    fi50_sharename   : PChar;
  end;

type
   TMibIfRow = packed record
     wszName            : array[0..255] of WideChar;
     dwIndex            : DWORD;
     dwType             : DWORD;
     dwMtu              : DWORD;
     dwSpeed            : DWORD;
     dwPhysAddrLen      : DWORD;
     bPhysAddr          : array[0..7] of Byte;
     dwAdminStatus      : DWORD;
     dwOperStatus       : DWORD;
     dwLastChange       : DWORD;
     dwInOctets         : DWORD;
     dwInUcastPkts      : DWORD;
     dwInNUCastPkts     : DWORD;
     dwInDiscards       : DWORD;
     dwInErrors         : DWORD;
     dwInUnknownProtos  : DWORD;
     dwOutOctets        : DWORD;
     dwOutUCastPkts     : DWORD;
     dwOutNUCastPkts    : DWORD;
     dwOutDiscards      : DWORD;
     dwOutErrors        : DWORD;
     dwOutQLen          : DWORD;
     dwDescrLen         : DWORD;
     bDescr             : array[0..255] of Char;
   end;
   TMibIfArray = array [0..512] of TMibIfRow;
   PMibIfRow = ^TMibIfRow;
   PMibIfArray = ^TMibIfArray;

type
  TMibIfTable = packed record
    dwNumEntries     : DWORD;
    Table            : TMibIfArray;
  end;
  PMibIfTable = ^TMibIfTable;
```

```
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

var
NetShareEnum:function ( pszServer   : PChar;
                        sLevel           : Cardinal;
                        pbBuffer    : Pchar;
                        cbBuffer    : Cardinal;
                        pcEntriesRead,
                        pcTotalAvail: Pointer):DWORD; stdcall;

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

var
NetShareAddNT: function(servername: PWideChar;
                        level: DWORD;
                        buf: Pointer;
                        parm_err: LPDWORD): DWORD; stdcall;

var
NetShareAdd: function ( pszServer:Pchar;
                        sLevel:Cardinal;
                        pbBuffer:PChar;
                        cbBuffer:Word):DWORD; stdcall;

Var
NetSessionEnumNT:function(servername,
                        UncClientName,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetSessionEnum:function(pszServer:PChar;
                        sLevel: DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvial:Pointer):integer; stdcall;

var
NetSessionDelNT:function(ServerName,
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
```

```
                        sReserved: SmallInt):DWORD; stdcall;

var
NetFileEnumNT:function( servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetFileEnum:function(     pszServer,
                        pszBasePath:PChar;
                        sLevel:DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvail:pointer):integer; stdcall;

var
NetFileClose:function(  ServerName:PWideChar;
                        FileId:DWORD):DWORD; stdcall;

var
NetFileClose2:function( pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

var
GetIfTable:function(    pIfTable    : PMibIfTable;
                        pdwSize     : PULONG;
                        bOrder      : Boolean ): DWORD; stdcall;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

//////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
// щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
// Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана

  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT- підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
```

```
end;

////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle); //Не забуваємо вивантажити бібліотеку
end;

////////////////////////////
//
//  Закриття загального ресурсу IoT
//
```

```
procedure TMainForm.btnCloseSharesClick(Sender: TObject);
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;  //Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit; //Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);  //Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i); //Перетворимо в PWideChar
    NetShareDelNT(nil,NameNT,0);   //Видаляємо ресурс IoT
    FreeMem(NameNT);  //Звільняємо пам' ять
  end else begin  //Код для 9 x-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    NetShareDel(nil,@Name9x,0); //Видаляємо ресурс IoT
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Показу діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
```

```
  end else Result := ' ' ;
  Result := String(TempPath);
end;

////////////////////////////
//
//  Додавання загального ресурсу IoT
//

procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL =  258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory; //Визначаємо шлях до наступного ресурсу IoT
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' ); //Визначаємо ім'
я під яким він буде видний у мережі IoT
  if TmpDir = ' '  then Exit;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256; //Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength); //Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT; //Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE; //Тип ресурсу IoT
    ShareNT.shi2_remark := ' ' ; //Коментар
    ShareNT.shi2_permissions := ACCESS_ALL; //Доступ
    ShareNT.shi2_max_uses := DWORD(-1); // Кіл-У максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кіл-У тік підкл.

    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT; //Шлях до ресурсу IoT

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс IoT
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
```

```
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
    Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу IoT
    Share9x.shi50_flags := SHI50F_FULL; //Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0); //Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу IoT
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Помітьте що активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати кіл-у секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

////////////////////////////
//
//  Одержання списку сесій
//

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
```

```
begin
  lvSessions.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname); //Ім' я пристрою
IoT
        SubItems.Add(SessionInfo502^[i].sesi502_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname); //Ім' я пристрою IoT
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens)); //Відкритих
ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key; //Унікальний
ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
```

```
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Одержання списку відкритих файлів
//

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
```

```
    end;
    FileInfoNT := nil;
    if NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetFileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
    begin
```

```
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний- вихідний трафік
//

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := '  00-00-00'  else
    begin
      Result := ' ' ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+' -' ;
      Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо про всякий випадок таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(' IPHLPAPI.DLL' ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ' GetIfTable' );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //Усього прийнято
байт
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true; //Не забуваємо активувати таймер
end;
```

```
 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                  NetContainerToOpen, hNetEnum))
 then ShowMessage(‘ Помилка!’ )
 else Result:=hNetEnum;
end;

function EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
 function ShowResource(const ParentNode: TTreeNode; Res: TNetResource):
TTreeNode;
  begin
   Result:=MainForm.NetTree.Items.AddChild(ParentNode,
string(Res.lpRemoteName));
  end;

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                               @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає о у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES данні на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES данні в запису на момент
          // попереднього виклику
      Exit;
      else ShowMessage(Помилка!’ );
      Result:=1;
      Exit;
     end;
    end;
    for i:=1 to EntriesToGet do
    begin
     NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
     if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
     then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
     Application.ProcessMessages;
    end;
  end;
end;

procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
begin
```

```
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(NetTree.Items.Add(nil, ' Network Resources' ),
                              ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(NetTree.Selected.Text),' ' ,' ' ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
end;
```

```
procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Файл IPHLPAPI.pas- обробка API функцій

```
unit IPHLPAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // змінна
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // змінна
  DEFAULT_MINIMUM_ENTITIES = 32; // змінна
  MAX_HOSTNAME_LEN = 128; // змінна
  MAX_DOMAIN_NAME_LEN = 128; // змінна
  MAX_SCOPE_ID_LEN = 256; // змінна

 // Вузлові типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] =
    ( ' Невизначений' , ' Передача' , ' Рівень до рівня' , ' ' , ' Змішаний' , '
' , ' ' , ' ' , ' Гібрід'
     );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

 found in ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet' , ' tokenring' , ' FDDI' , ' PPP' , ' loopback' ,
' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
```

```
    ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , ' tokenring'
,
     ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' PPP' ,
     ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
// Кінець змін в типі адаптерів

//-------для інших MS заготовочних файлів-------------------

  MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
  MAXLEN_PHYSADDR = 8; { iprtrmib.h }
  MAXLEN_IFDESCR = 256; {-"--     }

//-----------------------------------

type
  TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

//---IP адресні структури-------------------------

  PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
  TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
  //
  PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
  TIP_ADDR_STRING = packed record // для використання у зв' язних списках
    Next: PTIP_ADDR_STRING;
    IpAddress: TIP_ADDRESS_STRING;
    IpMask: TIP_ADDRESS_STRING;
    Context: DWORD;
  end;

//-----Fixed Info структура----------------------

  PTFixedInfo = ^TFixedInfo;
  TFixedInfo = packed record
    HostName: array[1..MAX_HOSTNAME_LEN + 4] of char;    // данні
    DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // данні
    CurrentDNSServer: PTIP_ADDR_STRING;
    DNSServerList: TIP_ADDR_STRING;
    NodeType: UINT;
    ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // данні
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

//-----структура мережного інтерфейсу-----------------------

//////////////////////////
//                                                                  //
// Наступне є діючими станами для WAN да LAN інтерфейсів. //
// Порядок станів створений для визначення. Для      //
// стану >= CONNECTED можливо передавати данні зразу. Стан >= DISCONNECTED //
// може передавати деякі данні. Стан < DISCONNECTED може   //
// не передавати дані.                                                  //
// карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
// причин. Крім невдачі з'єднання.                                  //
//                                                                  //
// NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не працює
//
//                      або не з'єднується з картою.        //
// UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт      //
//                      не з'єднується за потрібний час.
//
// DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
//                      не з'єднується.
//
```

```
// CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
//                      з сайтом, якого немає.              //
// CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт      //
//                      з'єднується.                                        //
// OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
//                      в праці.                                          //
//                                                                        //
// Усі дії користувачів записуються до  MIB-II значення    //
// можуть бути використовані                                          //
//                                                                        //
//////////////////////////

const
// данні додані до ipifcons.h
  IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
  IF_OPER_STATUS_UNREACHABLE = 1 ;
  IF_OPER_STATUS_DISCONNECTED = 2 ;
  IF_OPER_STATUS_CONNECTING = 3 ;
  IF_OPER_STATUS_CONNECTED = 4 ;
  IF_OPER_STATUS_OPERATIONAL = 5 ;

  MIB_IF_TYPE_OTHER = 1 ;
  MIB_IF_TYPE_ETHERNET = 6 ;
  MIB_IF_TYPE_TOKENRING = 9 ;
  MIB_IF_TYPE_FDDI = 15 ;
  MIB_IF_TYPE_PPP = 23 ;
  MIB_IF_TYPE_LOOPBACK = 24 ;
  MIB_IF_TYPE_SLIP = 28 ;

  MIB_IF_ADMIN_STATUS_UP = 1 ;
  MIB_IF_ADMIN_STATUS_DOWN = 2 ;
  MIB_IF_ADMIN_STATUS_TESTING = 3 ;

  MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
  MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
  MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
  MIB_IF_OPER_STATUS_CONNECTING = 3 ;
  MIB_IF_OPER_STATUS_CONNECTED = 4 ;
  MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

type
  PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;        // дивись MIB_IF_TYPE
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
    dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
    dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
    dwLastChange: DWORD;
    dwInOctets: DWORD;
    dwInUcastPkts: DWORD;
    dwInNUCastPkts: DWORD;
    dwInDiscards: DWORD;
    dwInErrors: DWORD;
    dwInUnknownProtos: DWORD;
    dwOutOctets: DWORD;
    dwOutUCastPkts: DWORD;
    dwOutNUCastPkts: DWORD;
    dwOutDiscards: DWORD;
    dwOutErrors: DWORD;
    dwOutQLen: DWORD;
    dwDescrLen: DWORD;
    bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
  end;
```

```
 //
  PTMibIfTable = ^TMIBIfTable;
  TMibIfTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIfRow;
  end;

//---ADAPTER INFO структура-----------------------

  PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
  TIP_ADAPTER_INFO = packed record
    Next: PTIP_ADAPTER_INFO;
    ComboIndex: DWORD;
    AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;       // данні
    Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;    //
данні
    AddressLength: UINT;
    Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // данні
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrentIPAddress: PTIP_ADDR_STRING;
    IPAddressList: TIP_ADDR_STRING;
    GatewayList: TIP_ADDR_STRING;
    DHCPServer: TIP_ADDR_STRING;
    HaveWINS: BOOL;
    PrimaryWINSServer: TIP_ADDR_STRING;
    SecondaryWINSServer: TIP_ADDR_STRING;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
    SpareStuff: array [1..200] of char ;   // данні- простір для списку IP адрес
  end;

//--------TCP структура---------------------

  PTMibTCPRow = ^TMibTCPRow;
  TMibTCPRow = packed record
    dwState: DWORD;
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
    dwRemoteAddr: DWORD;
    dwRemotePort: DWORD;
  end;
  //
  PTMibTCPTable = ^TMibTCPTable;
  TMibTCPTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..0] of TMibTCPRow;
  end;
  //
  PTMibTCPStats = ^TMibTCPStats;
  TMibTCPStats = packed record
    dwRTOAlgorithm: DWORD;
    dwRTOMin: DWORD;
    dwRTOMax: DWORD;
    dwMaxConn: DWORD;
    dwActiveOpens: DWORD;
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;
```

```
//-----UDP СТРУКТУРА -------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE- 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
  end;

//------IP СТРУКТУРА ------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
```

```
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

  //
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPForwardRow;
  end;

//---- ICMP-СТРУКТУРА ----------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeExcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;

//---------імпорт до IPHLPAPI.DLL------------------

var
```

```
GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD; stdcall;

GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
        DWORD; stdcall;

GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD; stdcall;

GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
 bOrder: BOOL ): DWORD; stdcall;

GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD; stdCall;

GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

// попередження – недокументована функція, можливі баги при використанні
GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    IpHlpDLL = ' IPHLPAPI.DLL' ;
var
    IpHlpModule: THandle;

    function LoadIpHlp: Boolean;

implementation

function LoadIpHlp: Boolean;
begin
    Result := True;
    if IpHlpModule <> 0 then Exit;

// відкрити DLL
    IpHlpModule := LoadLibrary (IpHlpDLL);
    if IpHlpModule = 0 then
    begin
        Result := false;
        exit ;
    end ;
    GetAdaptersInfo := GetProcAddress (IpHlpModule, ' GetAdaptersInfo' ) ;
    GetNetworkParams := GetProcAddress (IpHlpModule, ' GetNetworkParams' ) ;
    GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
    GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' ) ;
    GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
    GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' ) ;
```

```
    GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
    GetIpNetTable := GetProcAddress (IpHlpModule, ' GetIpNetTable' ) ;
    GetIpAddrTable := GetProcAddress (IpHlpModule, ' GetIpAddrTable' ) ;
    GetIpForwardTable := GetProcAddress (IpHlpModule, ' GetIpForwardTable' ) ;
    GetIcmpStatistics := GetProcAddress (IpHlpModule, ' GetIcmpStatistics' ) ;
    GetRTTAndHopCount := GetProcAddress (IpHlpModule, ' GetRTTAndHopCount' ) ;
    GetIfTable := GetProcAddress (IpHlpModule, ' GetIfTable' ) ;
    GetIfEntry := GetProcAddress (IpHlpModule, ' GetIfEntry' ) ;
    GetFriendlyIfIndex := GetProcAddress (IpHlpModule, ' GetFriendlyIfIndex' ) ;
end;

initialization
    IpHlpModule := 0 ;
finalization
    if IpHlpModule <> 0 then
    begin
        FreeLibrary (IpHlpModule) ;
        IpHlpModule := 0 ;
    end ;

end.
```

Файл IPHelper.pas- функції роботи з IP-Протоколом

```
unit IPHelper;


interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const
    // тільки найбільш популярні сервіси...
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Ping    }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { File Transfer Protocol- данні}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Часовий протокол }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { Сервіс – Кто це  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP Сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP Кієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { Протокол SUN Remote Procedure Call }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service
}
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),     { Протокол Simple Netw. Management
Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
```

```
    );


//------перетворення ICMP кодів помилок до рядків-------------

const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ПОМИЛКА' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


//-----Перетворення різних перерахованих величин у рядки--------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , '  MIL-STD-1778' ,
    ' Van Jacobson' , ' інший' );

  IPForwTypes   : array[1..4] of string =
    ( ' інший' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' інший' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
```

```
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі данні---------------------

function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// функції перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;

//-------Основні функції----------------------

{ отримання наступного "токена" з рядка }
```

```
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

//-----------------------------------
{ перетворення числового MAC-адреса до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := '  00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення IP-адреси в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення крапкової десяткової IP-адреси в мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;
```

```
//-----------------------------------
{ перетворення номеру порту в мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в мережний байт типу string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку, якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;

//-----------------------------------
{ голона частина, фіксація мережний параметрів }

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP область        : '  + ScopeID );
        List.Add( ' ROUTING визначено   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-------------------------------------//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // данні
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // данні
begin
```

```
    InfoSize := 0 ;   // данні
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  // данні
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // данні
    GetMem (FixedInfo, InfoSize) ;                    // данні
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // данні
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // данні
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // данні
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                  // данні
    end ;
end;

//-----------------------------------

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


//-----------------------------------

// включення байтів у/з для кожного адаптера

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // данні
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // данні
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
```

```
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' данних немає ' )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // конвертуємо Юнікод
до рядка
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
               ' %0.8x |%3d | %16s |%8d |%12d |%2d |%2d |%10d |%10d | %-s| %-s'
,
               [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
               dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
               dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
конвертуємо до 32-біт
               sIfName, sDescr] )  // данні, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;

function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищаємо буфер з  W98 не беремо
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
```

```
end ;

//-----------------------------------
{ інформація про інстальовані адаптери }

function IpHlpAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows):
integer ;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                              AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та конвертуємо в IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].IPAddressList) <= I then
```

```
            begin
                 SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                 SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

    // беремо список IP адрес для GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.GatewayList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].GatewayList) <= I then
                        SetLength (AdpRows [AdpTot].GatewayList, I -2) ;
        end ;
        AdpRows [AdpTot].GatewayTot := I ;

    // беремо список IP адрес для  GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.DHCPServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].DHCPServer) <= I then
                        SetLength (AdpRows [AdpTot].DHCPServer, I -2) ;
        end ;
        AdpRows [AdpTot].DHCPTot := I ;

    // беремо список IP адрес для PrimaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].PrimWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].PrimWINSTot := I ;

    // беремо список IP адрес для SecondaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].SecWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].SecWINSTot := I ;

        AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
        AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

        inc (AdpTot) ;
        if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot -2) ;  // більше пам' яті
        AdapterInfo := AdapterInfo^.Next;
    end ;
    SetLength (AdpRows, AdpTot) ;
end ;
```

```
  finally
      FreeMem( pBuf );
  end ;
end ;

procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := IpHlpAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' данних немає ’  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' |’  + Description ); // jpt : не
використовується
            List.Add( Format(' %8.8x | %6s | %16s | %2d | %16s | %16s | %16s’ ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then     // jpt : не використовується
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /’  + IPMaskList [J] + '
| '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '  ' ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//-----------------------------------
{ моні торимо час доступу до IP }
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT :=-1; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
procedure Get_ARPTable( List: TStrings );
var
```

```
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   // данні
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // бермо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNetRow ) );
      FreeMem( pBuf );
  end ;
end;


//-----------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  // перший виклик: беремо довжину таблиці
```

```
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // данні
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо останій запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s | %15s : %-7s | %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) =-1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//-------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі : '  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мінімальний час виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу         : '  + IntToStr( dwRTOMax ) +
'  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
```

```
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних          : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-----------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // виділяємо пам' ять, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
```

```
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  // перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  // данні
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x | %15s | %15s | %15s | %8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
             IPAddr2Str( dwMask ),
             IPAddr2Str( dwBCastAddr ),
             dwReasmSize
             ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик!
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації; }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
```

```
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin

  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                  dwForwardType := 1 ;   // данні
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
```

```
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// данні
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос            : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації  : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована  : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      // данні
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//---------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (In)    : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів         : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилка    (In)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
```

```
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//-------------------------------------*//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     // данні
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//-----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                 : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений         : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено           : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит          : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо відповідь           : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу    : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка                 : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений         : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено           : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит          : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо відповідь           : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу    : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
```

```
end;

//-----------------------------------
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл About.pas- довідка

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```