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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Іванов А.А.
«____» _____ 2021 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Анна КОВАЛЕНКО
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Іванову Антону Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1*

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

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1*

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

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	
6. Наукова новизна	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Іванов А.А. Дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Метою розробки є дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Об'єктом дослідження є процес міжмашинних комунікацій з використанням стандарту 10Base-T1.

Предметом дослідження є методи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

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 Embarcadero RAD Studio Delphi 10.3.2.

Ключові слова: комп'ютерна інженерія, міжмашинні комунікації, 10Base-T1

ABSTRACT

Ivanov A.A. Research and software implementation of the machine communication system using the 10Base-T1 standard. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of h igher education the software which is intended for system of intermachine communications with use of the 10Base-T1 standard is developed.

The purpose of development is research and software implementation of the system of intermachine communications using the standard 10Base-T1.

The object of research is the process of inter-machine communications using the 10Base-T1 standard.

The subject of the research is the methods of inter-machine communications using the 10Base-T1 standard.

The research methods are based on the methods of computer network theory, methods of mathematical statistics, methods of software development.

The result is a software implementation of the machine -to-machine communication system using the 10Base-T1 standard.

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 is developed in the environment Embarcadero RAD Studio Delphi 10.3.2.

Keywords: computer engineering, inter-machine communications, 10Base-T1

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

10 Base-T1	–	Стандарт однопарного Ethernet
АСУД	–	Автоматизована система управління й диспетчеризації
АСУБ	–	Автоматизована система управління будинком
ЄДЦ	–	Єдиний диспетчерський центр
ІТ	–	Інформаційні технології
ПЗ	–	Програмне забезпечення
СКС	–	Структуровані кабельні системі
ЦДБ	–	Цифровий двійник будинку
ЦОД	–	Центр обробки даних
АСР	–	Automation Collaborative Platform – Єдина Платформа Автоматизації
AWG	–	American Wire Gauge – Американський стандарт калібру проводів
IEEE	–	Institute of Electrical and Electronics Engineers
IoT	–	Інтернет речей
ISO	–	International Organization for Standardization – Міжнародна організація по стандартизації
PoDL	–	Power over Data Lines
PoE	–	Power over Ethernet
ROI	–	Коефіцієнт повернення інвестицій
TIA	–	Telecommunications Industry Association – Асоціація телекомунікаційної індустрії

ВСТУП

Актуальність теми. Поява однопарного Ethernet було в значній мірі пов'язане з інтересами автомобілебудування. Тепер ця технологія здобуває все більше значення для різних застосунків автоматизації, включаючи інтелектуальні будинки й промислове виробництво. 10 Base-T1, новий стандарт однопарного Ethernet, буде підтримувати одночасну передачу даних і напругу живлення.

Традиційний «мідний» Ethernet з неекранованими й екранованими кабелями із крученими парами – UTP, ScTP або Sc/FTP – становить основу локальних обчислювальних мереж. А завдяки технології Power over Ethernet (PoE) з його допомогою можна досить просто забезпечити живленням відеокамери, точки доступу бездротових мереж і інші подібні пристроїв.

Завдяки широкому використанню Ethernet з'явилася можливість створювати на основі єдиного протоколу конвергентні рішення для передачі даних, підключення комплексів безпеки, з'єднання систем автоматизації будинків і виробничих процесів. Удосконалювання технологічних рішень, мікропроцесорів і елементної бази привело до появи комплексів автоматизації нового типу, що, у свою чергу, зажадало модернізації використовуваної кабельної інфраструктури й стало стимулом для подальшого розвитку комунікаційної технології Ethernet.

Так, у сучасних автомобілях широко застосовуються системи мікропроцесорного управління компонентами, а також розважальні комплекси з високо-якісним відео й звуковим супроводом. У новітніх поїздах бортові електронні пристрої й різні варіанти аудіо- і відеорозваг поєднуються в системи, у яких використовуються кілометри кабелів.

Транспортні застосунки вплинули на появу нового різновиду Ethernet з однією крученою парою. Вона відкриває шлях до стандартизації інфраструктури передачі даних для промислових застосунків і відмові від використовуваних раніше протоколів промислових польових шин.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

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

– Огляд існуючих систем міжмашинних комунікацій з використанням стандарту 10Base-T1.

– Дослідження системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

– Програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Об'єктом дослідження є процес міжмашинних комунікацій з використанням стандарту 10Base-T1.

Предметом дослідження є методи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод міжмашинних комунікацій з використанням стандарту 10Base-T1.

– Розроблено вітчизняний продукт міжмашинних комунікацій з використанням стандарту 10Base-T1, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі міжмашинних комунікацій з використанням стандарту 10Base-T1.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями

параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Розроблені інститутом інженерів електротехніки й електроніки (Institute of Electrical and Electronics Engineers, IEEE) протоколи Ethernet для передачі даних по однопарному кабелі UTP або STP (100 Base-T1 і 1000 Base-T1) розраховані на міжмашині комунікації, автомобільну автоматизацію й промислові застосунки.

Стандарт 802.3bp 1000 Base-T1 регламентує передачу трафіку Ethernet по збалансованій кручений парі зі швидкістю 1 Гбіт/с на відстань від 15 до 40 м. Стандарт 802.3bw 100 Base-T1 описує передачу даних по збалансованій кручений парі зі швидкодією 100 Мбіт/с на відстань до 15 м.

Системи на основі «однопарних» стандартів дозволяють, крім іншого, зменшити вагу пасивного мережного устаткування на транспортних об'єктах і, відповідно, знизити їхні енерговитрати й вплив на навколишнє середовище. За даними компанії Harting, вага однопарного кабелю, призначеного для передачі даних зі швидкістю 1 Гбіт/с, майже на 30% менше в порівнянні із традиційними чотирьохпарними кабельними структурами.

Однопарний Ethernet відповідає також тенденції мініатюризації, що не менш важливо в цей час, оскільки у всіх галузях промисловості устаткування стає усе більше компактним. Як відомо, раніше були потрібні дві кручені пари для 100-мегабітних комунікацій Fast Ethernet і чотири для Gigabit Ethernet.

1.2 Область застосування

Однопарний Ethernet для Інтернету речей

Зростаючий інтерес до всіх видів промислової автоматизації, що відповідно до сучасних підходів припускає, як правило, об'єднання в мережі

об'єктів, наділених певним інтелектом, стимулює створення кабельних систем нового покоління.

Чималу роль грає застосування технології Інтернету речей у промисловості. Для багатьох вона асоціюється винятково з мобільними комунікаціями. У значній мірі це відбувається під впливом численних конференцій, публікацій і обговорень, організованих операторами зв'язку, зацікавленими у використанні своєї інфраструктури для розгортання IoT.

Дійсно, мобільні технології досить перспективні для застосунків автоматизації, особливо якщо об'єкти розподілені по значній площі. Проте середовище й засоби передачі даних не є визначальними компонентами рішень для Інтернету речей.

Принципова відмінність систем IoT від традиційних комплексів автоматизації полягає в здатності систем, розгорнутих на базі Інтернету речей, надавати й обробляти значні обсяги даних, формувати на цій основі відомості для прийняття управлінських рішень і застосунків предиктивної аналітики.

Все це виявляється можливим завдяки двом основним факторам. По-перше, з'явилися недорогі мініатюрні датчики й виконавчі пристрої, які постачені мережними інтерфейсами, що володіють необхідної для систем IoT функціональністю. По-друге, для систем автоматизації сьогодні доступні високопродуктивні обчислювальні ресурси, які розміщаються в граничних міні-ЦОД, центральних центрах обробки даних і обчислювальних хмарах.

По оцінках аналітиків, до 2025 року інвестиції в Інтернет речей досягнуть 6,2 трлн доларів, з них 2,3 трлн доларів буде витрачено на промислову автоматизацію. У результаті обробки даних комплексами IoT очікується зниження тимчасових і операційних витрат, у тому числі за рахунок підвищення надійності й предиктивного технічного обслуговування устаткування.

Провідні компанії, що працюють в області промислової автоматизації, розглядають системи на базі IoT як еволюційний розвиток розроблених раніше технологій, продуктів і рішень. Для багатьох підтримуваних ними застосунків

потрібні традиційні провідні комунікації. І тут на допомогу приходить універсальна комунікаційна технологія Ethernet, нова реінкарнація якої враховує специфіку комплексів IoT.

Для комунікації таким системам потрібні мережі передачі даних з невеликою пропускною здатністю, тому що підключені до них прикінцеві пристрої формують короткі (по кілька байтів даних) повідомлення, яких до того ж небагато, – вони передаються по запитах контролерів або ініціюються зовнішніми подіями.

Однак в інфраструктурі Ethernet для систем IoT необхідно усунути певні обмеження, властиві традиційним структурованим кабельним системам. До них ставиться недостатнє для ряду застосунків автоматизації відстань, на яку передаються дані: у стандартних СКС цей показник не перевищує 100 м.

У комплексах управління комерційними будинками й фізичною безпекою, приміром, застосовуються рішення, у яких довжина з'єднань більше 100 м. У мережах промислової автоматизації довжина каналів зв'язку може досягати 1 км, що теж не підтримується діючими стандартами Ethernet. Разом з тим для живлення прикінцевих пристроїв найчастіше досить подачі по комунікаційних кабелях невеликої потужності.

Всі ці вимоги враховуються розроблювачами нового «однопарного» стандарту Ethernet, розрахованого на застосунки автоматизації й системи на базі Інтернету речей.

Таким чином, у даній роботі технологія 10 Base-T1 буде застосовуватися для інтелектуальних будинків.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Цифрова трансформація експлуатації комплексів будинків і споруджень

Інтерес до цифрової трансформації експлуатації будинків і споруджень обумовлений як світовою тенденцією, так і представленою концепцією «розумного міста». Поворот держави убік цифровізації й енергоефективності дав прекрасні можливості вітчизняним компаніям, що займаються інтеграцією, налаштуванням і обслуговуванням інженерних систем, створювати інтелектуальні рішення, спрямовані на підвищення безпеки й енергоефективності експлуатації комплексу інженерних систем. Однією з таких компаній є HMPS, уже 10 років працююча в сфері автоматизації будинків і накопичувша досвід і статистику результатів впровадження інтелектуальних систем.

Архітектура проекту ICONICS для кампусу Microsoft

Відповідно до статистики компанії Green Cities California, 41% всієї електроенергії в США споживають інженерні системи будинків (обігрів, охолодження, освітлення). На думку експертів компанії, цифровізація й впровадження систем моніторингу й аналітики для перекладу роботи на рівень «екологічний будинок» (green building) допомагає скоротити витрати по енерговитратах до 23%. Відповідно до цього тренда провідні західні компанії вже не перший рік проводять трансформацію в сфері експлуатації комплексів будинків.

Одним з яскравих прикладів такої трансформації є проект компанії ICONICS по моніторингу будинків кампусу Microsoft у м. Редмонд (США), що

обіцяє скоротити витрати по управлінню цими будинками на мільйони доларів. Впровадження пілотного проекту на трьох будинках показало відмінні результати – економію енергії на 10%, а також скорочення витрат на технічне обслуговування й комунальні послуги. Після підрахунку коефіцієнта повернення інвестицій (ROI) по пілотному проекті Microsoft з'ясував, що строк окупності проекту склав усього 9 місяців. При цьому замовник одержав можливість вести диспетчеризацію всього, що відбувається на території впровадженого проекту, і одержувати повну інформацію в режимі онлайн про те, як функціонують будинку. Було ухвалене рішення розгорнути систему на всі будинки кампусу. Сьогодні територія кампусу охоплює 125 будинків, футбольне поле й площадку для крикету, кілометри дерев'яних пішохідних доріжок і 1,38 кв. км офісного простору з лабораторіями. Всі інженерні системи цієї території тепер функціонують як єдина система.

До проекту компанія Microsoft використовувала розрізнені системи управління будинками для контролю 30 000 одиниць незв'язаного сенсорного устаткування. У будинках спостерігався повний дисонанс даних, і аналізувати їх у єдиному ключі було просто неможливо. Це проблема, з якої зіштовхуються багато компаній, особливо в державному секторі: застаріле обладнання вважається не здатним на енергоефективне функціонування. Виникло запитання: краще замінити все устаткування або спробувати зменшити енергоспоживання за допомогою технічного програмного рішення?

Вибір був зроблений на користь нового аналітичного рішення на базі ICONICS, що при інтеграції змогло надати всю необхідну інформацію про будинки. Лавини даних стікалися в центр управління кампусом і відкривали інженерам ока на всі, починаючи з неощадливого графіка включення освітлення до надзвичайно неефективних (однак раніше непомічених) воєн за підтримку температур між кондиціонерами й нагрівачами. Інженери більше не піднімаються на дахи, не перевіряють насосні й не дивляться під стельові плити – тепер вони витрачають 95% часу тільки на інженерію. У міру того як будинки заводилися в

мережу й з'являлися дані, формувалося так зване цільове середовище для рішення проблем. Раніше інженери переходили від будинку до будинку, витрачали на кожне по двох тижня підряд, щоб оглянути його досконально й настроїти, перш ніж переходити до наступного. У них би пішло п'ять років, щоб настроїти всі будинки кампусу, і потім вони робили б це знову й знову. Їхнього налаштування дозволяли будинкам функціонувати ефективніше, що щорічно заощаджувало компанії біля $250 тис. Однак нове інформаційне цільове середовище допомогло їм заощадити в шість разів більше.

Українська сучасна дійсність

Необхідність побудови комплексної автоматизованої системи складних об'єктів уводить у дію правила для інженерних систем будинків висотою більше 55 м і функціональних груп будинків, які необхідно оснащувати системами зв'язку, сигналізації, автоматизації й диспетчеризації. Архітектура, що вводиться, АСУД (автоматизована система управління й диспетчеризації) повинна забезпечувати централізований моніторинг, являючи собою гнучку, вільно програмувальну розподілену систему.

Крім наявності системи диспетчеризації, необхідно враховувати індекс інтелекту програмного рішення, що вводиться. Повинне рости число міст, управління якими здійснюється за допомогою інтеграції інформаційних і комунікаційних смарт-технологій.

Зросли й вимоги замовників – як до глибини автоматизації інженерних систем, так і до функціональних можливостей таких систем: кількості збираємих даних, звітам, аналітиці, прогнозуванню, мобільним версіям системи, інтеграції з корпоративними системами. Замовникові вже необхідна не просто система моніторингу, а платформа, що забезпечує можливість інтелектуальної обробки даних, контроль всіх подій і дій персоналу. Така система повинна інтегруватися не тільки з усім устаткуванням, але й з інформаційними системами замовника. При цьому вона повинна бути простій у використанні й сучасної: цьому

сприяють поліпшене юзабіліті, 3D-технології й повноцінні застосунки для мобільних пристроїв.

Київська компанія-інтегратор HMPS провела аналіз існуючих на ринку рішень для побудови «розумних» АСУД, проаналізувала приклади західних і вітчизняних впроваджень і дійшла висновку, що програмні інструменти GENESIS64 від ICONICS оптимально підходять для побудови розподілених систем управління комплексами будинків. Розглянемо переваги цих інструментів.

По-перше, сучасні будинки – це складні комплексні об'єкти, де кількість інженерних систем і їхню насиченість вимагають адекватного управління й супроводу, що неможливо реалізувати без розвитий і надійну інформаційну систему. ICONICS є одним зі світових лідерів ринку програмних рішень для побудови єдиних центрів автоматизації й диспетчеризації на базі сучасної 64-бітної платформи й сфальцьовано на надійній роботі розподілених архітектур.

По-друге, все частіше в проектах побудови централізованих АСУД потрібне об'єднання в одну систему раніше розрізнених інформаційних підсистем. Воно необхідно для підвищення ефективності й оперативності управління, поліпшення адміністрування системи і її розвитку. Інтегрована система єдиного управління повинна вміти обмінюватися даними об бізнес-процесах, що дає можливість виявляти закономірності й будувати складні прогностичні моделі. Інженерні системи в таких проектах потрібно інтегрувати з інформаційними продуктами класу ERP (SAP, OEBS, 1C), Service Desk, спеціалізованими обліковими системами управління експлуатацією, CRM. ICONICS GENESIS64 підтримує роботу моніторингу/читання або запису/сміни за відкритими стандартами IT-систем автоматизації будинків, у тому числі OPC UA/OPC Classic, BACnet/IP, Modbus/IP, TCP/IP, SNMP. Крім того, через модуль ICONICS BridgeWor доступні транзакції до більшості існуючих на ринку інформаційних систем класу ERP.

По-третє, розвиток технологій дає можливість вибрати рішення, що буде функціонувати й удосконалюватися разом з об'єктом автоматизації протягом

мінімум п'яти років, тому ICONICS намагається випливати актуальним технологічним трендам. На українському ринку автоматизації в технічному завданні все частіше стали з'являтися словосполучення: «інтеграція BIM-моделі», «готові застосунки для мобільних пристроїв», «прогнозний моніторинг устаткування», «функції доповненої реальності», «аналітичні форми для управління енергоефективністю». Всі ці функції підтримуються в програмних рішеннях ICONICS, що дає конкурентні переваги інтеграторам і замовникам.

Навіщо будинку потрібний цифровий двійник

Фактично при створенні системи автоматизації на програмних продуктах ICONICS мова йде про формування віртуального аналога реального об'єкта – цифрового двійника будинків (ЦДБ). Такі системи поєднують інформаційні й експлуатаційні технології й базуються на застосуванні «Інтернету речей» (IIo), хмарних технологій, аналітики більших даних.

Залежно від виду діяльності підприємства, призначення об'єктів, акцент може бути зроблений на тій або іншій функціональності системи. Наприклад, для аеропортів і банків це насамперед безаварійна й безвідмовна робота інженерних систем, а для агрохолдингів і адміністративних будинків, де успішно функціонують окремі системи пожежної безпеки, – це підвищення енергоефективності роботи систем.

Як правило, мотивами створення таких систем є:

– Підвищення часу безаварійної роботи. Сучасний інструментарій розробки й засобу моніторингу дозволяють на ранніх етапах побачити відхилення в роботі інженерних систем і попередити їхнє виникнення. По досвіду компаній, що вже впровадили систему предикативного аналізу, відбувається зниження кількості інцидентів до 70%, незапланований простій устаткування скорочується до 35%. Як наслідок, зменшуються витрати на усунення аварій і інцидентів.

– Безпека. Це те, що дуже важко виміряти в грошах, адже ціна помилки – життя й здоров'я людей. Для багатьох компаній, де штат команди експлуатації оптимізований, а інформаційна система для моніторингу інженерних систем і

попередження інцидентів відсутній, – це істотний аргумент. У першу чергу організується оперативний контроль роботи інженерних систем, далі розробляється механізм раннього виявлення й попередження інцидентів у роботі інженерних систем.

– Зниження вартості підтримки й розвитку системи. Розвиток функціонала інформаційної системи й повноцінна підтримка (наприклад, гаряче резервування) досить витратний захід, і нести такі витрати доцільно для дійсно важливих для компанії систем. До того ж багато компаній не в змозі розвивати існуючі системи моніторингу, написані, як правило, на різних платформах, і такі системи або функціонують самі по собі й не є затребуваними, або морально застаріли.

– Функціональність ЦДБ може створюватися й тиражуватися з меншими витратами відразу для всіх об'єктів, оптимізуючи загальну вартість впровадження. Як засоби розробки ЦДБ вибирають єдину програмну платформу, на основі якої розробляється додаток з універсальними формами, шаблонами, застосовними для всіх об'єктів. Надалі нові будинки, інфраструктурні об'єкти підключаються до вже працюючої системи. Ефект від централізованої розробки системи й супроводу об'єктів особливо відчуємо для групи об'єктів і дозволяє заощадити 20-40% бюджету в порівнянні з індивідуальною розробкою для кожного з об'єктів.

– Уніфікація устаткування. У процесі експлуатації єдина система робить збір даних про роботу інженерії, виявляє найбільш підходящі моделі/виробників устаткування для того, щоб на наступних об'єктах закладати в проекти добре зарекомендував себе устаткування й рішення. Це допомагає уникнути витрат на ремонти, а також оптимізувати складські запаси ЗИП і витрати на пошук і навчання персоналу для роботи з різним устаткуванням. Єдина система моніторингу й аналітики також виявляє «чорні списки» проблемного устаткування й формує автоматичні завдання на виключення застосування даних моделей на об'єктах замовника в майбутньому.

– Зниження витрат на адміністрування процесу експлуатації. Адміністрування підрядників. Процес адміністрування роботи підрядних організацій, як правило, вимагає заповнення чималої кількості паперових журналів, що часом дублюють один одного, звітів, ручного фіксування фактів усунення інцидентів, часу їхнього усунення, початих дій і відповідальних виконавців. Більшу частину процесів можна реалізувати в єдиній системі, у тому числі використовуючи засобу об'єктивного контролю, одержуючи дані прямо з об'єкта, а не по дзвінку від підрядник а. Як показує наш досвід, реальна економія від автоматизації цього процесу може досягати 80%.

– Підвищення якості управління. Достовірна, повна й оперативна інформація про роботу об'єктів, наявність аналітичної звітності й прогнозів дають керівництву більше повну базу для прийняття обґрунтованих рішень.

Користувачі системи

Єдина система моніторингу й аналітики**,** на відміну від класичних систем автоматизації й диспетчеризації, може інтегруватися по бізнес-процесам і даним з корпоративними інформаційними системами, наприклад ERP, CRM, Helpdesk. У підсумку користувачами системи стають:

– Керівники – їм надаються аналітичні звіти й дані для прийняття рішень, пов'язаних з підвищенням енергоефективності, стійкості роботи інженерного устаткування, статистика по усуненню інцидентів і аварій, виконання регламентів.

– Фінансисти – одержують дані енерговитрат по видах і центрам їхнього виникнення, контроль і прогнозування лімітів споживання, виявлення розбіжностей між виставленими рахунками й фактичним споживанням.

– Експлуатація – щоденний робочий інструмент, що дозволяє здійснювати моніторинг, локалізацію й запобігання інцидентів, формування потреби в ремонтах і обслуговуванні на підставі фактичних даних.

– Фахівці IT – відповідають за моніторинг працездатності ЦОД, серверних.

Цифровий двійник будинків: базовий функціонал

Процес впровадження ЦДБ відбувається, як правило, поетапно й багато в чому залежить від поточного ступеня автоматизації об'єктів, планів по модернізації устаткування, першочергових потреб організації як по економії ресурсів, так і по забезпеченню безперервності роботи. До базових і необхідних функцій ЦДБ можна віднести:

– Моніторинг інженерних систем. Об'єднання різнорідних інженерних систем об'єктів у єдину систему для зручності обслуговування й експлуатації. Одержання інформації про позаштатні ситуації в режимі реального часу для можливості їхнього попередження й усунення.

– Енергомоніторинг. Включає збір інформації з видів споживаних ресурсів, місцям споживання, здійснює моніторинг і аналіз сезонних коливань, оперативно виявляє розбіжності між споживанням і виставленими рахунками від постачальників ресурсів, сигналізує про перевищення лімітів споживання, виявляє несанкціоновані підключення.

– Комплексна діагностика. Розробка алгоритмів для тестування інженерних систем, наприклад включення резервних вузлів у випадку відмови основного устаткування.

– Звітність. Формування звітності для різних користувачів системи (керівництво, начальник служби, інженери, фінансові служби й ін.).

– Документування. Централізоване зберігання актуальної документації й можливість її відправлення відповідальним фахівцям (виконавча документація, паспорти устаткування, інструкції з ремонту й регламенти). Ведення й аналіз балок роботи систем.

Розширений функціонал доцільно впроваджувати після нагромадження показників по роботі устаткування, умов виникнення інцидентів, тобто створення великого масиву даних, якому можна аналізувати й формувати правила й логіку для попередження інцидентів.

До розширеного функціонала можна віднести:

– Діагностику інцидентів. Автоматичне визначення можливих причин виникнення інцидентів із процентною ймовірністю кожної із причини.

– Предикативний аналіз. Розробка функцій, технологій і алгоритмів, що дозволяють аналізувати поточну й історичну інформацію у відповідності із взаємозалежними симптомами/причинами, накопиченими в системі. Застосування алгоритмів з обчислення ймовірності виникнення несправностей і надання рекомендацій зі списком можливих причин відмов, з імовірністю їхнього виникнення.

– Аналітикові як інструмент підтримки прийняття рішень і виявлення проблемних ділянок.

– Візуалізацію об'єкта. Відображення на 3D-моделі роботи інженерних систем у реальному часі для більше наочного подання локалізації об'єкта й причин інцидентів.

– Засоби доповненої реальності. Можливість віртуального «огляду» схованих інженерних систем – їхнього розташування, зв'язку із суміжними системами.

Український досвід побудови ЦДБ

Єдина система моніторингу, управління й аналітики була впроваджена інтегратором HMPS в об'єкті одного із провідних київських девелоперів. У даний момент система перебуває в промисловій експлуатації, але як і раніше постійно розвивається: відбувається підключення нових житлових комплексів, розробляються нові модулі, пов'язані із предикативним аналізом роботи устаткування, формуються плани розвитку.

Об'єкт являє собою мережу житлових комплексів різного класу, розташованих по всій території Києва. Це більше двох десятків будинків, інженерні системи яких реалізовані на різноманітному устаткуванні, як вітчизняному, так і закордонному.

Єдиний диспетчерський центр (ЄДЦ) є верхнім рівнем системи управління будинками й поєднує в одну мережу локальні диспетчерські пункти всіх житлових комплексів. Система забезпечує контроль стану й параметрів життєво важливих інженерних систем, своєчасне одержання інформації про аварії/відхиленнях від заданих параметрів і прийняття автоматичних мір, що компенсують. Збір даних здійснюється із всіх житлових комплексів у режимі реального часу. У міру нагромадження інформації про роботу інженерії відбувається доналаштування аналітичних модулів.

До ЄДЦ, виконаному на базі платформи ICONICS, підключені всі основні інженерні системи: загальнобмінна вентиляція, холодопостачання й кондиціювання, електропостачання, електроосвітлення, технічний облік енергетичних ресурсів, пожежогасіння, противодимна вентиляція, водопостачання й каналізація, ліфтове устаткування.

Основними показниками результатів впровадження ЄДЦ на даному етапі є:

– Збільшення оперативності реагування на інциденти при роботі інженерних систем і, як наслідок, підвищення технічної безпеки житлових комплексів. Відображення порядку (інструкції) дій при критичних аваріях.

– Скорочення витрат на технічну експлуатацію об'єктів за рахунок оптимізації кількості персоналу, його пошуку й навчання роботі з різним устаткуванням, зниження паперового документообігу.

– Скорочення кількості інцидентів за рахунок своєчасного виявлення й усунення попереджень, а також прогнозування аварійних ситуацій; підвищення якості інформації для прийняття управлінських рішень.

– Поліпшення координації між обслуговуючим персоналом.

– Централізоване зберігання проектної й виконавчої документації по об'єктах і по основному інженерному устаткуванню. Доступність актуальної й достовірної документації для служби експлуатації на об'єктах.

Як результат, створена сучасна й передова інфраструктура для управління й експлуатації інженерних систем всієї мережі житлових комплексів девелопера.

Особливості проекту:

– Велика розмаїтість локального устаткування й рішень. Кожний з житлових комплексів проектувався й реалізовувався різними підрядниками. Житлові комплекси варіюються від варіанта «стандарт» до класу «клубний будинок». У підсумку система інтегрує устаткування різного класу й виробників, що працює на різних протоколах.

– Інтеграція з корпоративними IT-системами. У рамках проекту впроваджена система управління інцидентами: ведення електронних за вок від користувачів через телефон, електронну пошту, сайт. Система моніторингу, управління й аналітики інтегрована з нею й на підставі подій, що відбуваються, наприклад, автоматично формує завдання на усунення інцидентів і призначає відповідального за усунення.

– Захищені канали зв'язку. Передача даних між житловими комплексами й центральним офісом здійснюється по VPN-каналах.

– Інтеграція відеопотоку. У систему включена трансляція відеопотоку з реалізованої на житлових комплексах мережі відеокамер.

– Звітність. У системі формуються друковані форми для спрощення звітності й адміністрування підрядників на об'єкті.

– Керівництво до дії. Система надає відповідальним виконавцям на об'єктах доступ до центрального сховища з актуальною інформацією (модель, місце розташування, характеристики й робочі параметри) і документацією (виконавча документація, паспорти устаткування, інструкції з ремонту й регламенти) для локалізації проблеми і її усунення.

– Візуалізація 3D. Для одержання більше наочної інформації, прискорення локалізації несправних ситуацій інженерами й диспетчерами служби експлуатації – об'єкти представлені у вигляді повноцінної 3D-Графіки з відображенням критичних інженерних систем і окремих вузлів.

Smart BMS Bridge – універсальний інтелектуальний програмувальний відкритий шлюз для автоматизованих систем управління будинком

У розділі представлений інтелектуальний програмно-апаратний шлюз для автоматизованих систем управління будинком (АСУБ) Smart BMS Bridge, розроблений фахівцями компанії «ФІОРД» на основі продуктів світових брендів Rockwell Automation, Newron System і CompuLab.

Програмна складова Smart BMS Bridge (SBB) побудована шляхом інтеграції технології програмування контролерів ISaGRAF 6 і сервера даних DoMooV, що підтримує основні протоколи систем управління будинками BACnet, LonWorks, KNX, M-Bus, Modbus. Апаратна складова SBB – мініатюрні комп'ютери компанії CompuLab. Закінчене рішення SBB дає можливість системним інтеграторам за допомогою відкритого інструмента скористатися перевагами ведучих у своєму класі продуктів, значно скоротити час, працезатрати, вартість реалізації й впровадження проектів будь-якого ступеня складності.

Отже, у нашім конкретному контексті зміст терміна «універсальний» полягає в тому, що SBB підтримує весь спектр основних протоколів в області АСУБ (BACnet, LonWorks, KNX, M-Bus, Modbus). «Інтелектуальність» полягає в тому, що кінцевий користувач за допомогою базових програмних інструментів SBB може реалізувати рішення практично будь-якого ступеня складності, а також використовувати вже убудовані в SBB алгоритми для роботи в області АСУБ (водопостачання, освітлення, каналізація, електрика й ін.). У термін «програмувальний» вкладається простий зміст: наявність в SBB засобів підтримки різних мов програмування, причому відповідним міжнародним стандартам. І, нарешті, під терміном «відкритий» (розширюваний) ми розуміємо можливість розроблювачів, системних інтеграторів і OEM-виробників додавати свої функціональні можливості, орієнтовані на конкретний проект або предметну область. Саме сполучення цих чотирьох складових

(універсальність+інтелект+програмуємість+відкритість) виділяє SBB серед аналогічних продуктів на ринку.

З погляду користувача, програмне забезпечення (ПЗ) для конфігурування SBB являє собою надбудову над ISaGRAF 6 Workbench [1 –6] у складі Єдиної Платформи Автоматизації (Automation Collaborative Platform, ACP), у яку убудована підтримка DoMooV – інтегрованої програмної платформи для управління набором найпоширеніших відкритих протоколів BMS, а в перспективі також ZigBee, Z-Wave, Dali, C-Bus. Нагадаємо також, що ISaGRAF – це технологія розробки застосунків для програмувальних логічних контролерів у стандарті IEC 61131-3, у якому визначені п'ять мов прогр амування: IL (Instruction List – «Список інструкцій»), ST (Structured Text – «Структурований текст»), LD (Ladder Diagram – «Східчаста діаграма»), FBD (Function Block Diagram – «Діаграма функціональних блоків»), SFC (Sequential Function Chart – «Послідовна функціональна діаграма»). У якості апаратної складової BMS обрані мініатюрні комп'ютери компанії CompuLab – fit-PC2i, fit-PC3, Intense PC (мал.1) і uSVR, які досить докладно описані в недавньої розділі [9], тому тут ми не будемо докладно зупинятися на них. Відзначимо тільки, що вибір на користь цих пристроїв пояснюється їхньою надійністю, багатими функціональними можливостями, малим розміром і мінімальним енергоспоживанням, розширюваністю. Важливим для користувача аспектом є можливість вибору однієї із чотирьох пропонованих апаратних платформ SBB залежно від вимог конкретного проекту. Всі можливі конфігурації SBB можна подивитися на сайті http://shop.fiord.com.

Приведемо основні можливості, які буде забезпечувати SBB:

– підтримка протоколів BACnet, Modbus, M-Bus, LonWorks і KNX;

– прозорий доступ і функції шлюзу між різними мережами;

– створення прикладних програм на мовах стандарту IEC 61131-3 і 61499 для динамічної обробки даних і управління пристроями різнорідних мереж у рамках єдиного проекту за допомогою єдиного інструментарію;

– підтримка режиму гарячого резервування SBB;

– підтримка розподілених конфігурацій контролерів, датчиків і виконавчих пристроїв у рамках єдиного проекту, що включає трохи SBB із центральним диспетчерським пунктом або без нього;

– зв'язування (binding) входів і виходів декількох SBB;

– архівування даних в SBB на самому пристрої й періодичній передачі архівів у центральну базу даних MS SQL або PostgreeSQL;

– локальна (безпосередньо через відеопорт SBB) або віддалена (на комп'ютері) динамічна візуалізація протікання процесу (засіб ISaQT);

– доступ до даних (змінним) SBB через убудований BACnet Server або oBIX Server;

– доступ до даних проекту в режимі реального часу через FDA OPC-сервер, у тому числі при використанні декількох SBB у проекті;

– у перспективі – підтримка функцій планування (SBB Scheduler).

Зупинимося трохи докладніше на програмних компонентах Smart BMS Bridge.

Універсальний програмний компонент SBB: ACP і ISaGRAF 6

Концепція й технологія ACP [1–4] розроблена на основі ISaGRAF [5–6] і створена для обслуговування систем автоматизації (розроблювач – Rockwell Automation). ACP розроблена як середовище, керована за допомогою відкритих модулів, що підключаються – плагінів. ACP являє собою розширюваний шар абстракції із загальним інтерфейсом, що забезпечує уніфіковані функціональні можливості, обирані користувачем. Вона призначена для постачальників засобів автоматизації, OEM-виробників, системних інтеграторів, науково-дослідних інститутів і допомагає проектувальникам ПЗ, дозволяючи їм зосередитися на своїй основній предметній області, а не на системних програмних питаннях інфраструктури рішення. ACP підтримує кілька конкретних моделей автоматизації (Concrete Automation Model, CAM) одночасно, надаючи можливість інтеграції різнорідних продуктів у єдине середовище розробки. Дві з конкретних

моделей автоматизації, що входять у базову поставку АСР, забезпечують створення застосунків для виконавчих систем (віртуальних машин, таргетів) ISaGRAF 6&5&3, що працюють на різних апаратних платформах. Процес розробки полягає в створенні проекту, що складає із пристроїв з одним або декількома екземплярами ресурсів. Проекти можуть розроблятися, використовуючи різні мови програмування, включаючи мови стандарту IEC 61131-3 і IEC 61499. Після етапу розробки ресурси компілюються в TIC -код (Target Independent Code) або в програму мовою С.

АСР пропонує повністю готову до використання оболонку (Shell), спеціально розроблену для систем автоматизації, використовуючи інструментарій Microsoft Visual Studio і технологію .Net Framework. Платформа надає всі базові сервіси для взаємодії із продуктами третіх фірм і забезпечує налаштовуваність («кастомізацію») кінцевого рішення. Інакше кажучи, АСР – це середовище для створення рішень по комплексній автоматизації шляхом інтеграції технології ISaGRAF Workbench і компетенції OEM -виробника засобів автоматизації. Саме ця можливість і дозволила фахівцям компанії «ФІОРД» інтегрувати DoMooV у середовище АСР і розробити такий інноваційний продукт, як SBB.

Функціонально-орієнрований програмний компонент SBB: мультипротокольна платформа DoMooV для BMS

DoMooV – платформа, що уніфікує дані й поводження системи незалежно від протоколу, пристрої або виготовлювача. DoMooV (розробка французької компанії Newron System) базується на об’єктно-орієнрованій моделі, зменшує витрати й поєднує розробку рішень для BMS і для SCADA -систем. DoMooV – це framework, каркас програмної системи на основі .NET (мал. 3), що включає в себе засобу розробки (SDK) для створення користувачем власних централізованих або розподілених застосунків. Мультипротокольний сервер даних DoMooV Dataserver може підтримувати один або кілька движків збору даних для одного або декількох мережних інтерфейсів [7-8].

Мультипротокольний сервер даних DoMooV Dataserver також буде доступний для OS Linux і може бути убудований в апаратні пристрої на основі OS Linux (зараз версія для Linux проходить етап тестування й поки не поставляється користувачам). Багатопротокольні сервери DoMooV OPC і DoMooV BACnet працюють із одним або декількома централізованими або розподіленими серверами даних DoMooV і забезпечують подання даних у форматі OPC або BACnet. Сервери підтримують також функцію шлюзу для обміну даними між протоколами.

Рішення на основі DoMooV включає уніфіковану інформацію для обміну даними: будь-які застосунки (SCADA, корпоративні рішення, HMI і т.п.) можуть обмінюватися інформацією в єдиному уніфікованому форматі через один із серверів: сервер BACnet, сервер OPC або «native» сервер даних DoMooV. Платформа не тільки відкрита на рівні сервера, вона може також розширюватися для роботи з іншими польовими (fieldbus) протоколами. Її внутрішня структура забезпечує просту інтеграцію застарілих або частно-фірмових протоколів з наступним включенням їх у закінчене рішення.

DoMooV пропонує клієнт-серверну архітектуру й для SCADA -систем, і для інструментів управління мережею. SCADA-система може взаємодіяти через базовий (native) інтерфейс DoMooV, BACnet або OPC-сервер. Архітектура масштабується від найпростіших до дуже складних BMS-рішень.

Взаємодія ISaGRAF 6 Workbench с DoMooVv Dataserver

Для взаємодії з DoMooV у складі АСР був розроблений набір модулів, що підключаються, ISaDoMooV. При його допомозі можливе створення керуючої логіки в ISaGRAF з використанням складно організованих різнорідних мереж, низькорівневий доступ до яких забезпечується сервером даних DoMooVv. У цей час до складу ISaDoMooV входять два модулі: ISaDoMooV Driver і ISaDoMooV Configurator ISaDoMooV Driver забезпечує безперервне перенесення даних із внутрішніх змінних проекту ISaGRAF на сервер даних DoMooV і назад. ISaDoMooV Configurator виконує дві основні функції – автоматичне створення

змінних у проекті ISaGRAF, які будуть прив'язані до раніше зконфігурованих точок даних DoMooV, і створення конфігураційного файлу, що містить необхідну для роботи драйвера інформацію про відповідність цих змінних і точок даних DoMooV. Користувачеві надається можливість вибору необхідних змінних і вказівки, у якому з ресурсів або POU вони повинні бути створені. При запуску ISaDoMooV зчитує конфігураційні файли DoMooV змінних, вузлів і мереж LON, BACnet, KNX, Modbus і зберігає конфігураційну інформацію у внутрішній базі даних ISaGRAF (про це буде йти мова нижче).

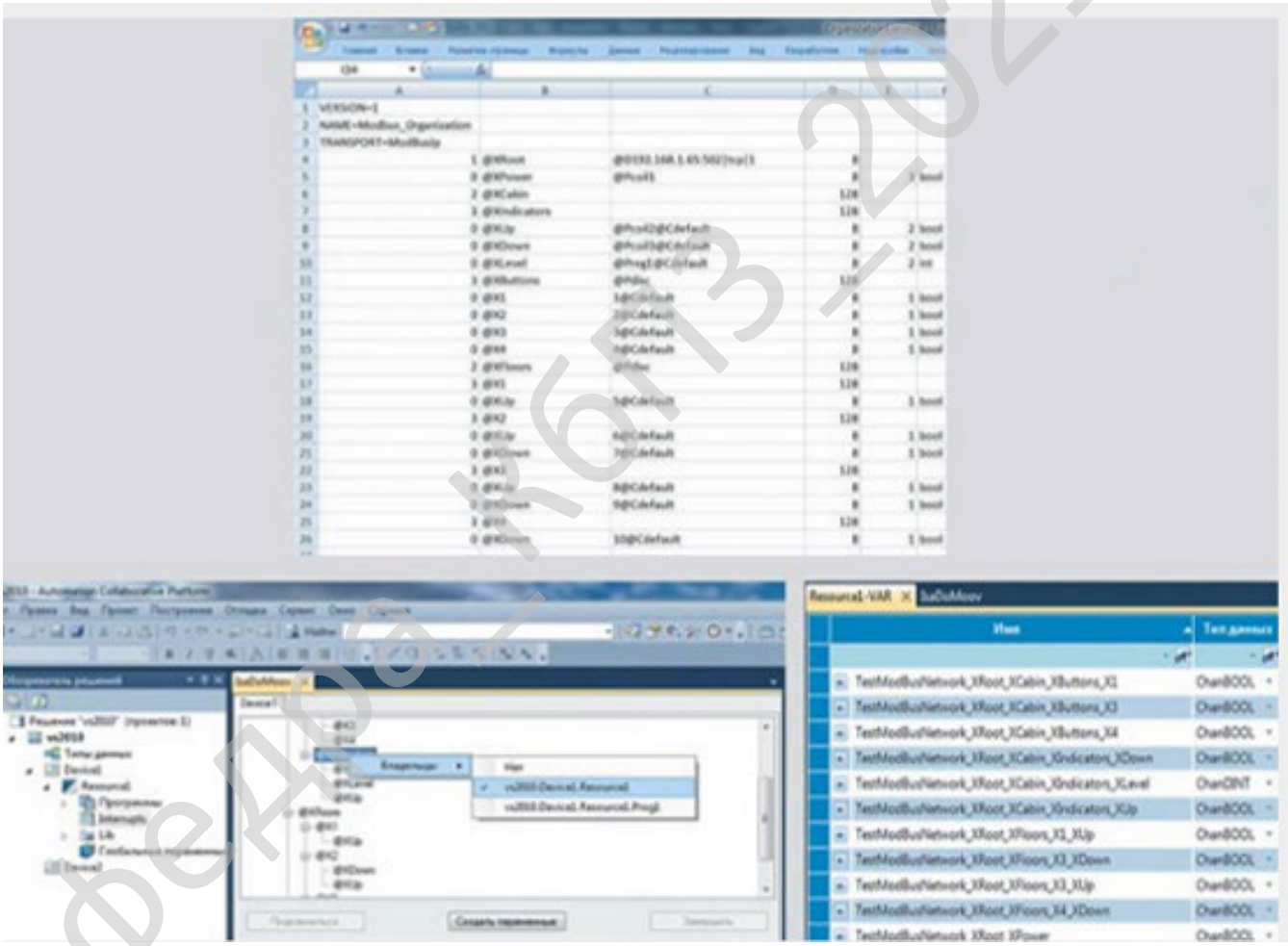

Рисунок 2.1 – Приклад подання даних Modbus у конфігураторі ISaDoMooV і в DoMooV

Відзначимо, що формування конфігураційних файлів DoMooV має на увазі використання спеціалізованих сервісних програм для конкретного протоколу: наприклад, LonMaker або NL220 для LonWorks. Після імпорту «мережних» змінних DoMooV у проекті ISaGRAF з ними можна робити всі припустимі операції. Розроблений проект виконується цільовою системою й дозволяє здійснювати автоматичне регулювання й обробку даних, візуалізувати і архівувати зміни значень змінних, передавати дані реального часу OPC-серверам або SCADA-пакетам. Таким чином, можна встановлювати зв'язку між змінними (об'єктами) вузлів різних мереж і з диспетчерськими пунктами в рамках розподілених АСУ різного призначення.

ISaQT: графічний інтерфейс для SBB

Модуль ISaQT для ISaGRAF дозволяє користувачеві мати графічний інтерфейс на контролері. ISaQT взаємодіє з виконавчою системою (таргетом) ISaGRAF по швидкому протоколі FDA, використовуючи бібліотеку віддаленого клієнта FDA. Тому ISaQT може виконувати візуалізацію змін даних у реальному часі як локально на самому контролері, так і віддалено, з іншого комп'ютера. ISaQT може взаємодіяти з декількома контролерами одночасно, тому можна робити візуалізацію не тільки автономної, але й розподіленої системи автоматизації. ISaQT уже реалізований під Linux і під Windows. Для створення мнемосхем використовується програма Qt Creator, що формує інтерфейс мовою QML 2, а мовою JavaScript формується динамічна складова. Програма Qt Creator – кросплатформна вільна IDE для розробки на C, C++ і QML у рамках фреймворка Qt. Сформований файл інтерпретується програмою ISaQT. У результаті користувач одержує графічний інтерфейс на контролері.

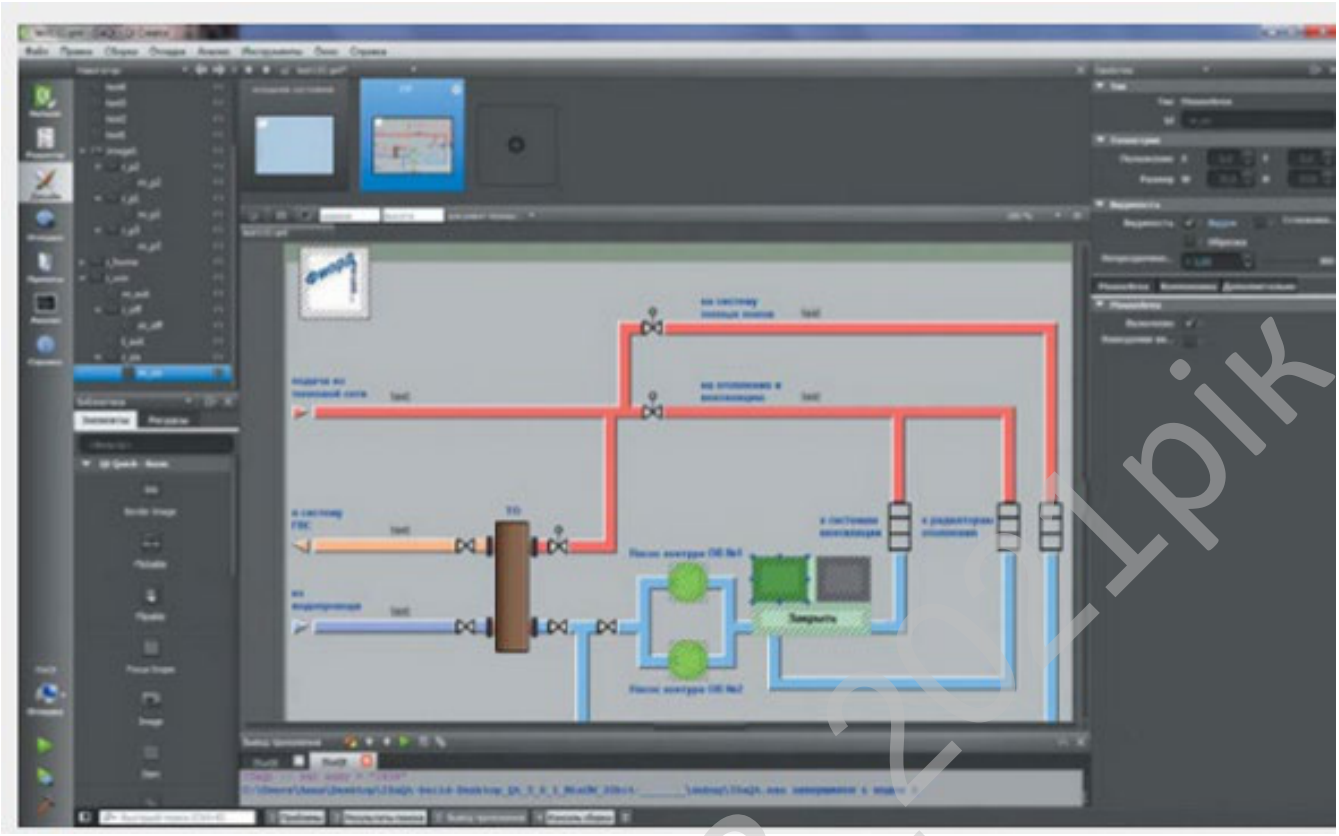

Рисунок 2.2 – Приклад графічного інтерфейсу на SBB

Режим резервування й відновлення (Failover) для SBB

В ISaGRAF 6 реалізована підтримка режиму резервування й відновлення після відмови (Failover), що потенційно може використовуватися в проектах з використанням SBB. Відновлення після відмови – це режим роботи, при якому функції системи управління приймає на себе вторинна система управління (у тому випадку, коли головна система стає недоступної через відмову устаткування або при запланованому простої). Використання цієї функції підвищує відказостійкість системи управління. Функція відновлення після відмови в SBB дає можливість користувачам модифікувати систему управління й міняти умови, при яких контролер одержує або втрачає управління. У режимі Failover середовище ISaGRAF реалізує наступні ключові можливості: ненаголошений перехід на резервну машину, робота з будь-яким типом POU (SFC, FBD, LD, ST, 61499), автоматичне завантаження проекту (одночасно на первинну й вторинну

станцію), автоматичне перемикання з Workbench на потрібну станцію в режимі налагодження, автоматичне перемикання OPC-серверів.

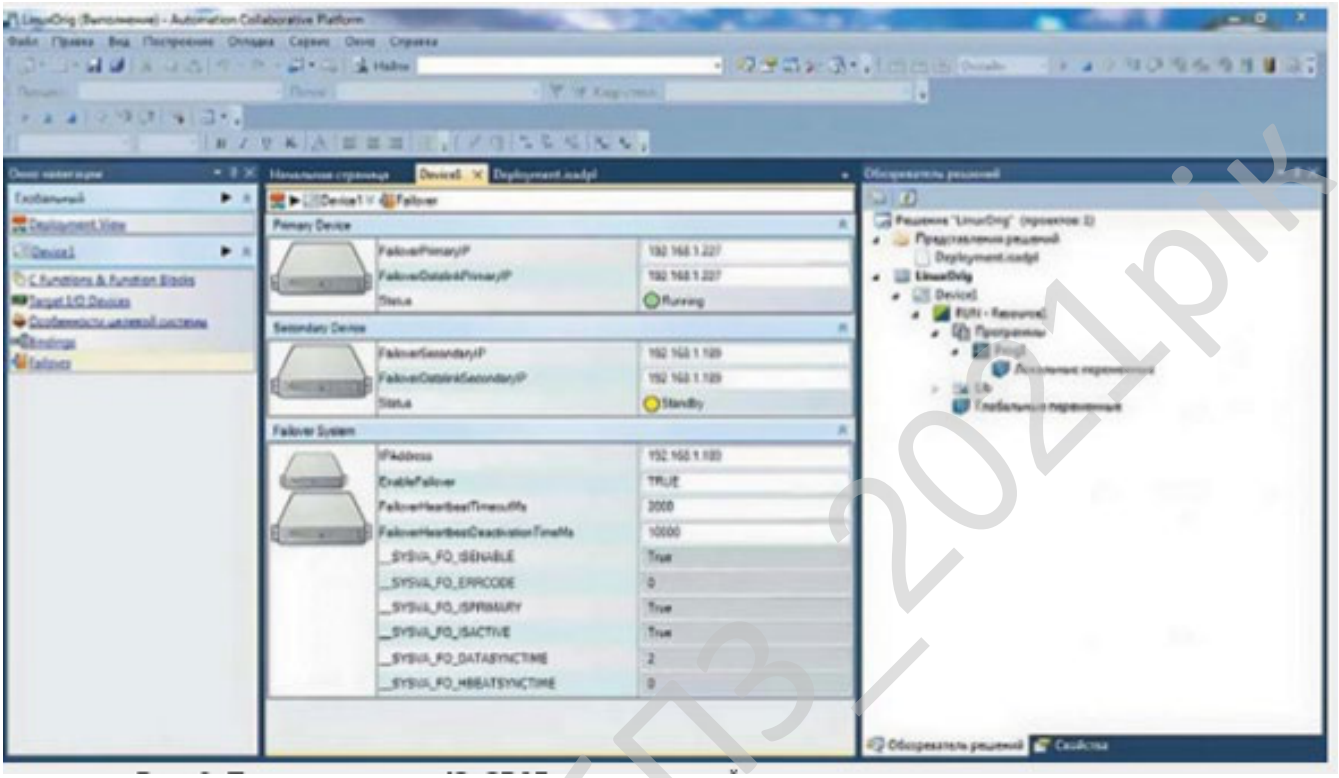

Рисунок 2.3 – Приклад проекту ISaGRAF з підтримкою режиму гарячого резервування

Розглянемо реалізацію режиму Failover у ядрі виконавчої системи ISaGRAF. Після завантаження додатка на активний контролер автоматично виконується його передача на резервний контролер. Обидва контролери починають паралельно виконувати те саме додаток. Вхідні зміні передаються з основного контролера в резервний перед кожним циклом виконання. Наприкінці кожного циклу виконується перевірка за обчисленою контрольною сумою, щоб гарантувати цілісність даних і результатів. У випадку розбіжності на резервний контролер передається вся область даних активного (основного) контролера. При збої на активному контролері резервний контролер стає активним (основним) і починає управляти процесом. Для зв'язку між основним і резервним контролером

за замовчуванням використовується мережа TCP/IP, але можуть бути використані й іншої мережі (послідовний канал, UDP, оптоволокно). Допускається налаштування умови, при якому відбувається зміна основного контролера. Конфігурування Failover складається з установки наступних параметрів: IP-адреса, номер порту й значення тайм-аутів.

Фахівці компанії «ФІОРД» провели тестування режиму гарячого резервування.

Умови тестування:

– 85 тис. змінних користувача;

– 4 тис. вхідних змінних;

– обсяг TIC-коду дорівнює 247 кбайт;

– розмір буферів зв'язку 64 кбайт;

– основний контролер працює під управлінням ОС Windows 7, резервний – під управлінням VMware ОС Windows XP.

Результати тестування: час циклу 1 мс із відключеним режимом Failover, час циклу 13 мс із включеним режимом Failover і повною синхронізацією дани х (88102 байта у двох фреймах), час циклу 6 мс із включеним режимом Failover і частковою синхронізацією даних (4127 байт в одному фреймі).

Розподілена система архівування даних SBB

Система архівування ISaGRAF Archive System (IAS) призначена для ведення архівів історичних даних на контролерах із цільовою системою ISaGRAF (у тому числі SBB), збору накопиченої інформації в єдину архівну базу й подальший аналіз архівних даних. Система IAS являє собою трирівневий комплекс програмних компонентів.

IAS Logger – нижній рівень розподіленої системи ведення історичних даних IAS. Він призначений для нагромадження архівних даних, що поставляються цільовою системою ISaGRAF, і збереження їх на диску контролера, а також для забезпечення доступу до збереженим даних локально або по мережі Ethernet по запитах верхнього рівня. Підсистема реалізована у вигляді

віртуального пристрою ISaGRAF і включає також сервіс обробки запитів на видачу даних і управління архівами, призначений для транспортування даних локальних архівів на сервер по запитах останнього. Налаштування підсистеми здійснюється за допомогою конфігураційних файлів, що містять перелік архівуємих змінних і задаваемих для кожного ресурсу ISaGRAF. Для кожної змінної може бути заданий поріг чутливості для запобігання запису в архів «дребезга» змінної. Налаштування IAS Logger дозволяють також обмежувати максимальний розмір дискової бази локального контролера, при цьому при заповненні відведеного простору нові дані записуються замість старих. Таким чином, система може зберігати актуальні архівні дані, накопичені протягом певного тимчасового відрізка.

IAS Configurator – конфігуратор системи архівування, призначений для імпорту проектів ISaGRAF у базу даних SQL (MS SQL або PostgreSQL), налаштування змінному, підлягаючому архівуванню, формування конфігураційних файлів і завантаження їх у відповідні контролери. Конфігуратор також створює скрипти, що дозволяють автоматизувати процес викачки архівних даних з контролерів і їхній імпорт у базу даних SQL. Основне призначення компонента – віддалене конфігурування системи ведення локальних архівів, зручний сервіс для конфігурування більших систем, заміна «ручного» конфігурування.

IAS Collector – система збору архівних даних з контролерів, запис даних у текстові файли або в базу даних MS SQL S erver, PostgreSQL. IAS Collector використовується для віддаленої доставки архівів у базу даних, централізованого збору архівів від різних джерел у єдину базу даних.

Інтелектуальний готель на базі ПЗ WebAccess

Завдання створення інтелектуального будинку звичайно ставиться на самому ранньому етапі планування в процесі проектування. Наділяючи будинку інтелектом, ми одержуємо цілий ряд переваг. Приклади тому – сучасний п'ятизірковий Pullman Beijing South Hotel і 13-поверховий будинок штаб-

квартири міжнародної комп'ютерної компанії, широко відомої своїми унікальними дослідженнями, розробками й виробничими процесами.

Інтелектуальний готель

Pullman Beijing South Hotel перебуває під управлінням французького оператора мережі готелів Accor Group і розташований у самому центрі нового економічно й технологічно розвиненого району Пекіна. Готелю була потрібна центральна система управління широким спектром спеціалізованих підсистем, включаючи систему охолодження, управління теплообмінниками, а також системи моніторингу кондиціювання й стани повітря в приміщеннях, системи подачі й витяжки повітря, системи водопостачання й водовідводу, трансформаторних і розподільних електричних установок, а також ліфтів. При цьому всі польові пристрої повною мірою розподілені по всьому обсязі готелю, включаючи самі віддалені ділянки. Інтелектуальна система моніторингу повинна мати можливість підключення до всіх необхідних вузлів за допомогою комунікаційної мережі, а також підтримувати функції розподіленого управління й віддаленого моніторингу стану систем. Творці готелю хотіли одержати вигідне сполучення системи адміністрування будинку й інтелектуальної системи управління для організації комфортної й теплої атмосфери в житлових приміщеннях і зниження витрат на матеріальні й енергетичні ресурси.

Опис системи

В готелі застосовується автоматизована система управління будинком серії BAS від компанії Advantech, що містить у собі програмне забезпечення моніторингу високого рівня WebAccess, BASPro – середовище логічного програмування DDC-контролерів серії BAS-3000, контролер BAS-3500, а також різні модулі розширення, призначені для ефективного контролю й управління електромеханічними пристроями в будинку готелю. Системи моніторингу готелю містять у собі систему охолодження, кондиціювання й освіження повітря, подачі й витяжки повітря, водопостачання й водовідводу.

Моніторинг системи охолодження

Система охолодження готелю містить у собі морозильну установку з насосом, що прохолоджує камеру з насосом, компенсаційний бак, а також клапан перепаду тиску. Централізована система моніторингу обмінюється даними з морозильним блоком у режимі реального часу за допомогою інтерфейсу Modbus і одержує інформацію про його робочий стан. При цьому управління різними польовими пристроями в рамках приміщення з устаткуванням виконує польовий контролер.

Моніторинг системи кондиціювання й подачі свіжого повітря

Вимоги до обслуговуючих систем п'ятизіркового готелю набагато вище, ніж до стандартних готелів з малою кількістю зірок. Робота Pullman Beijing South Hotel прямо залежить від центральної системи моніторингу, що контролює стан всіх підключених пристроїв. Значення температури й вологості в приміщеннях, отримані за допомогою різних прецизійних датчиків, дозволяють польовому DDC-контролеру регулювати роботу системи подачі й зволоження повітря, розташованої в пункті кондиціювання. Це гарантує підтримку необхідної температури, вологості й інших показників стану повітря.

Моніторинг системи подачі й відводу повітря

Від централізованої системи моніторингу залежить також процес подачі й відводу повітря в готелі. Вона дозволяє відслідковувати робочий стан витяжних блоків, одержувати інформацію про автоматичні або ручні зміни режиму роботи вентиляторів, а також включати/виключати вентилятори якщо буде потреба. Якщо система почне працювати в позаштатному режимі, то вона передасть стан польових пристроїв за допомогою комунікаційної мережі на екрани моніторів центрального пункту управління, а також забезпечить голосові сигнали тривог для залучення уваги операторів. Крім того, критично важливі дані можуть бути надруковані у вигляді звіту.

Моніторинг системи подачі й відводу води

Подача й відвід води також залежать від централізованої системи моніторингу, що дозволяє відслідковувати стан наступних компонентів: низький і високий рівні тривог водойми й бака запасу води, що тече стан і сигнали тривог насосів подачі води, низька й висока межі рівня води в накопичувальних баках, поточний стан і сигнали тривог водяного насоса відводу води.

«Розумна» штаб-квартира

Зовсім недавно міжнародній комп'ютерній компанії потрібна була інтелектуальна система автоматизації для штаб-квартири на Тайваневі, яка б дозволяла здійснювати моніторинг і автоматизоване управління 13-поверховим будинком. Цей проект дозволив об'єднати всіляке устаткування за допомогою потужних функціональних можливостей комунікаційної мережі. Конструкція цієї системи була спрямована на досягнення максимального заощадження енергоресурсів: кожний поверх будинку оснащений устаткуванням для автоматичного моніторингу й управління, системи електропостачання встановлюють спеціальний розклад, що визначає час включення й вимикання пристроїв, а ефективна система безпеки знижує кількість необхідних людських ресурсів. Крім того, замовникові було потрібно простої в застосуванні програмне забезпечення, що дозволяє управляти численними підсистемами одночасно.

Опис системи

Система автоматизації в цілому складається з наступних підсистем:

– система контролю температури з відображенням даних у вигляді графіків і їхньою реєстрацією;

– система захисту з контролем доступу, що генерує сигнали тривог у випадку виникнення порушень;

– цифрова система відеоспостереження, що працює 24 години на добу;

– енергосистема, що виконує функції енергозбереження;

– парковочна система, що забезпечує авторизований доступ за допомогою зчитування карт.

ПЗ WebAccess

Програмне забезпечення WebAccess на базі веб-технологій дозволяє операторам одержати можливість віддаленого моніторингу й управління системою автоматизації будинків за допомогою стандартного веб-браузера. Всі дані відображаються в режимі реального часу у вигляді графіків, що обновляються, або анімації, Крім того, існує можливість відображення відео на одному екрані із графічними даними, сигналами тривог і лініями графіків. Потужні інтеграційні можливості ПЗ дозволяють не тільки скоротити необхідні людські ресурси, але й знизити витрати на електроенергію, щоб максимально швидко одержати повернення своїх інвестицій.

WebAccess підтримує відкриті інтегрувальні компоненти, такі як Active X, HTML-світи, імпорт графічних зображень, DDE, OPC, ODBC I/F, а також скрипти JAVA/TCL/VB, які дозволяють системним інтеграторам легко розробляти інтерфейси для кожної підсистеми й формувати повноцінну інтелектуальну систему автоматизації будинку.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero RAD Studio Delphi 10.3.2 Rio Architect – це найшвидший спосіб створювати й обновляти інтенсивно працюючі з даними, сильно взаємодіючі застосунки з візуально насиченим користувальницьким інтерфейсом для Windows 10, Mac, мобільних пристроїв, IoT і інших платформ за допомогою Object Pascal і C++. Широкий вибір функцій підтримки Windows 10, у тому числі нові компоненти VCL для Windows 10, стилі для VCL і FMX, а також служби UWP (універсальної платформи Windows), наприклад повідомлення, дозволяють легко й швидко перенести застосунки в Windows 10, зберігши користувачів. Нова платформа дозволяє підтримувати великі проекти на більшому числі платформ із подвоєним обсягом пам'яті в середовищі розробки й удвічі більшим розміром

підтримуваних проектів. Крім того, підтримка декількох моніторів і десятки нових функцій середовища розробки, призначених для прискорення створення коду, зроблять роботу як ніколи ефективною. За допомогою RAD Studio 10 розроблювачі зможуть створювати застосунки в 5 разів швидше в порівнянні з іншими інструментами, а розробка застосунків для декількох настільних, мобільних, хмарних платформ і платформ баз даних, включаючи 32і 64 -розрядні версії Windows 10, Mac OS X, iOS і Android, стане ще швидше.

Зміни у версії 10.3 Rio:

– Створюйте міжплатформені застосунки. 80% інтернет користувачів мають смартфони й застосунки доступу, а також дані з мобільного пристрою й ноутбука / настільного комп'ютера, саме тому так важливо в цей час, щоб застосунки працювали в будь-якому пристрої.

– У всіх версіях Professional, Enterprise і Architect RAD Studio 10.3 надається підтримка процесу розробки застосунків для мобільних пристроїв. Розроблювачі RAD Studio кодують лише один раз, компілюють споконвічно для кожної платформи, що скорочує час і працезатрати на вивчення декількох мов і дозволяє паралельно управляти циклами розробки.

– Підтримка Android API26, відповідність вимогам Google Play Store відносно нових застосунків із серпня 2018 року й відновлення застосунків з листопада 2018 року.

– Власні елементи управління Android і стилізовані елементи управління FMX в одній і тій же формі Android, включаючи тему матеріального дизайну для Android 5.0 або вище.

– Підтримка iOS 12 (32і 64-біт) для створення App Store і корпоративних застосунків.

– Підтримка смайликів Юнікод.

– Програмуйте по-своєму. Завдяки двом новим темам самостійне налаштування інтегрованого середовища розробки для відповідності вашому стилю кодування ще ніколи не була настільки простій.

– Темне й світле оформлення Незалежно від того чи волієте ви кодувати вночі або у світлий час доби, завдяки темному й світлому оформленню RAD Studio ви можете вибрати потрібний вам стиль. Було доведено, що темне оформлення допомагає знизити зорову напругу в умовах низького освітлення, дозволяючи вам працювати більш продуктивно вночі. Немає нічого простіше, ніж перейти від темного до світлого оформлення й навпаки за допомогою меню панелі інструментів.

– Виконаєте користувальницьке налаштування свого середовища розробки Поліпшена програма установки інтерфейсу користувача й менеджера ліцензій інтерфейсу користувача дозволяє визначити ті можливості, які необхідні й опустити непотрібні, незалежно від того чи розробляєте ви застосунки для декількох платформ або всього однієї.

– Чистий, оновлений інтерфейс користувача інтегрованого середовища розробки Знайдіть потрібні можливості. Швидко. Головне вікно інтегрованого середовища розробки відцентровано й відрізняється високим ступенем читаності. Ви з легкістю визначите, де перебуває область фокусування клавіатури з оновленими змінами фонових квітів фокуса. Вкладки редактора більше, що полегшує читання шрифтів, тому ви можете швидко внести зміни й зберегти кодування.

– Чудові застосунки Windows з VCL. Бібліотека візуальних компонентів (Visual Component Library, VCL) пропонує просту й візуальну розробку користувальницького інтерфейсу застосунки, у версії 10.3 представлені нові відновлення, які дозволять вашим додаткам виглядати сучасними й свіжими.

– Розширена підтримка HighDPI. Завдяки новому елементу управління VCL High DPI ImageList у версії 10.3 розроблювачі, що створюють нові застосунки VCL для Windows або оновлюючі існуючі застосунки для High DPI дисплеїв, можуть повністю підтримувати зроблені до рівня пікселів зображення зі змінною розв’язною здатністю на всіх елементах управління, а також будь-яке

користувальницьке креслення, що вимагає масштабованих зображень для моніторів з різною розв'язною здатністю.

– Підтримка Per Monitor V2. Переконаєтеся, що ваш додаток масштабується правильно для всіх типів масштабування в Windows, реагуючи на зміни масштабування DPI на різних екранах під час виконання.

– Розширена підтримка Windows 10 і WinRT API. Сюди відноситься ряд ключових API-інтерфейсів WinRT і останні API-інтерфейси Windows 10, включаючи готові до використання компонента для убудованих у застосунки покупок і випробувань у магазині Windows 10 Store.

– Розгортання застосунків на основі служб за допомогою RAD Server. Продуктивність RAD Server була значно поліпшена завдяки десятикратному збільшенню потужності відносно простих операцій.

– Нові компоненти обробки JSON допоміжного засобу.

– Розширена підтримка RAD Server для клієнта Ext JS. Об'єднайте зовнішній інтерфейс javascript і веб-службу, підтримувану REST Server REST. (У версії Architect тепер включена ліцензія ExtJS Professional).

– Версії Enterprise включають ліцензію для одиничного розгортання RAD Server.

– Версії Architect включають ліцензію для розподіленого розгортання RAD Server.

– Що нового в C++? Підтримка C++17 Win32 збільшує продуктивність, поліпшує роботу компілятора й прискорює процес кодування. Були оновлені RTL і STL.

– Нова версія STL/Dinkumware 2018 для Win32 і Win64.

– Поліпшене автодоповнення коду Автодоповнення коду для даного компілятора тепер асинхронне, швидше й із кращими результатами, чим у попередньому автодоповненні коду C++. Уведення тексту не буде припинятися, поки виконується розрахунок.

– Тепер є підтримка налагодження для оптимізації компонувань.

– 2X швидкість математичної продуктивності для Win64.

– Нові додаткові лабораторії C++ в GetIt.

– Нові й поліпшені можливості роботи з базами даних. InterBase 2017 / IBToGo 2017 в RAD Studio. Версії Professional включають ліцензію розроблювача InterBase 2017, у той час як версії Enterprise і Architect містять у собі ліцензії InterBase ToGo. InterBase ToGo доповнена можливістю шифрування, функціями зміни подань, призначених для простої синхронізації даних застосунку по підписці без обмежень за розміром файлу бази даних.

– Поліпшена й оновлена підтримка для популярних баз даних, включаючи mySQL v8.0, MariaDB 10.3, SQL Server 2017, PostgreSQL v10, Firebird v3.0, MongoDB, InterBase, SQLite 3.23.1, SQL Anywhere і багатьох інших.

– Удосконалення DataSnap.

– Поліпшення REST. Підтримка додаткових родинних REST методів, типів і властивостей.

– Повністю оновлений модуль живлення версії Architect. Одержіть більше від версії Architect, включаючи ці ліцензії сімейства Idera.

– Ліцензія Sencha ExtJS Professional: Створіть свій ідеальний мережний вхідний інтерфейс за допомогою javascript і ExtJS.

– Ліцензія на розгортання InterBase ToGo. Додайте сховище даних у свої застосунки за допомогою цієї гнучкої, зашифрованої бази даних, що вбудовується.

– Ліцензія для розподіленого розгортання RAD Server. Ідеально підходить для серверного застосунку архітектури мікросервісів.

– Ліцензія AquaData Studio. Вражаючий аналіз бази даних.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне

забезпечення, яке призначено для системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Стандарт 10 Base-T1

В інституті IEEE комітетом 802.3 сформована робоча група 802.3cg Single Pair Ethernet для стандартизації комунікацій на основі однієї кручений пари з дальністю передачі понад 100 м, а при підготовці враховуються вимоги до мережі з боку рішень автоматизації на базі Інтернету речей.

Назва нового стандарту – 10 Base-T1. Розроблювальні специфікації розраховані на передачу даних на невеликі (short-reach) і значні (long-reach) відстані.

10 Base-T1S буде специфікувати тракт передачі даних до 15 м із чотирма з'єднувачами й діапазоном робочих частот від 0,3 до 200 МГц, а 10 Base-T1L – дальність до 1000 м, до 10 з'єднувачів у тракті, частоти від 0,1 до 20 Мгц.

Новий стандарт орієнтований на кабельні системи для комерційного застосування (у тому числі для використання в автомобілебудуванні), а також на комплекси промислової автоматизації, за допомогою яких численні датчики поєднуються з виконавчими пристроями, що передають незначні обсяги даних.

На прикінцеві пристрої, які підключені до однопарної кабельної інфраструктури, електроживлення, як правило, потрібно подавати дистанційно. Для цього розроблювачі мають намір забезпечити відповідність 10 Base-T1 вимогам специфікацій стандарту IEEE 802.3bu (Power over Data Lines, PoDL) – як для з'єднань точка-точка, так і для транкінгових схем подачі живлення (Powered Trunk).

Згодом стандарт IEEE 802.3cg 10 Mb/s Single Twisted-Pair може стати реальною альтернативою найбільш популярним польовим шинам, застосовуваним у різних системах автоматизації.

Стандартизація кабельної інфраструктури

Групи стандартизації кабельних систем Американського національного інституту стандартів (American National Standards Institute, ANSI) і Асоціації телекомунікаційної індустрії (Telecommunications Industry Association, TIA), а також Міжнародної організації по стандартизації (International Organization for Standardization, ISO) і Міжнародної електротехнічної комісії (International Electrotechnical Commission, IEC) приступилися до розробки специфікацій трактів передачі даних різної довжини для комерційних і індустріальних застосунків. Відповідні специфікації містять вимоги до кабелів, рознімань, комунікаційних шнурів і іншим компонентам каналів однопарних комунікацій.

Приміром, в TIA працюють над стандартом TIA-568.5, де визначаються однопарні з'єднання і їхні компоненти зі швидкодією 10 Мбіт/с для кабельних трактів довжиною до 100 м, 100 Мбіт/с – до 15 м, 1 Гбіт/с – до 15 і 40 м. Кабелі повинні підтримувати технологію Power over Data Lines, тобто забезпечувати подачу потужності до 50 Вт по одній кручений парі 24 AWG (American Wire Gauge, американський стандарт калібру проводів).

Розроблювальне доповнення до промислового стандарту TIA-1005 Industrial Cabling регламентує кабельну проводку для передачі даних на відстань до 1 км зі швидкістю 10 Мбіт/с. Щоб знизити втрати, у трактах такої довжини планується застосовувати кручену пару із провідниками калібру 18 AWG.

Подібні розробки здійснюються й в ISO. Так, стандарти ISO 11801-9906 Ed. 1 і 11801-6 Ed.1/Amd.1 будуть містити специфікації однопарної кабельної проводки для комерційних застосунків, у яких потрібне швидкодія 10, 100 і 1000 Мбіт/с. Стандарти ISO I11801-3 Ed.1/Amd.1 будуть присвячені кабельній інфраструктурі для промислових рішень.

Однопарні коннектори

До найважливіших компонентів кабельної інфраструктури ставляться з'єднувачі. За попередніми оцінками експертів, фронтальна площа з'єднувачів для однопарного Ethernet могла б становити від половини до третини відповідної

площі рознімань RJ45. Вони розробляються розраховуючи на проведення калібру 18-26 AWG, повинні бути здатні підтримувати струм до 1 А и можуть випускатися в екранованих і неекранованих модифікаціях. Провідні виробники комутаційних продуктів уже підготували свої рішення.

Пропозиція компанії CommScope засновано на розніманнях LC і враховує вимоги стандарту IEC 63171-1. У цих розніманнях оптичні компоненти замінені металевими контактами. Як за вляє розроблювач, такі екрановані й неекрановані компактні коннектори прості в установці, що дуже важливо при монтажі в польових умовах і у важкодоступних місцях.

З'єднувачі інсталюються без застосування складних інструментів, а їхні контактні поверхні не уступають по надійності компонентам рознімань RJ45. Вони цілком здатні замінити використовувані сьогодні однопарні рішення, не пов'язані з технологією Ethernet, уважають в CommScope. Коннектори, запропоновані CommScope, призначені для застосування в комерційних офісних додатках (Mechanical, Ingress, Climatic, Electromagnetic, MICE1).

На роботу в промислових умовах, включаючи найбільш складне навколишнє середовище (MICE2/MICE3), розраховані рознімання, запропоновані компанією Harting. Вони відповідають вимогам стандарту IEC 61076–3–125 і специфікаціям IP65/67 по запалювано– і вологозахищеності. Їхніми компонентами можуть служити в тому числі добре відомі в промислових додатках рознімання M8 і M12.

Екрановані модифікації коннекторів Harting забезпечують роботу на частотах до 600 Мгц. У запропонованих розніманнях передбачена підтримка технології Power over Data Lines і передача напруги живлення на відстань до 1 км.

Всеосяжний Ethernet

Отже, 10 Base-T1, нова технологія Ethernet, буде підтримувати однопарні комунікації й можливість одночасної передачі даних і напруги живлення (Power over Data Lines) по збалансованій кручений парі.

Поява однопарного Ethernet у значній мірі пов'язане з інтересами автомобілебудування. Однак у цей час ця технологія стає усе більше значимої для різних застосунків автоматизації, включаючи інтелектуальні будинки й промислове виробництво.

Очікувані комунікації 10 Base-T1 відкривають нові можливості для цифровізації систем автоматизації нижнього рівня. З їхньою появою взаємодія з датчиками й виконавчими пристроями в комплексах промислової автоматизації й інтелектуальних будинків може здійснюватися за допомогою цифрових, а не аналогових сигналів.

Спрощення кабельної інфраструктури, зменшення обсягів кабельної продукції й скорочення строків монтажу особливо важливі в рішеннях на основі технологій Інтернету речей, прикінцевих пристроїв у якому набагато більше, ніж у сучасних системах автоматизації.

Ethernet, таким чином, стає всеосяжною комунікаційною технологією, що охоплює все компоненти сучасних об'єктів нерухомості – від систем управління будинком (Building Management System, BMS) до розміщених у них центрів обробки даних. Крім того, вона уніфікує передачу даних для комплексів промислової автоматизації – від датчиків до IT-Інфраструктури, що підтримує ERP-Додатка. Завершення розробки стандартів 10 Base-T1 наближає появу й впровадження таких рішень.

3.2 Розробка структурної схеми

Темпи росту ринку автоматизації будинків перевищують темпи росту ринку будівництва будинків, оскільки, крім оснащення системами управління будинків-новобудов, при реконструкції й ремонті відбувається активне устаткування системами автоматизації більшої частини фонду будинків, що експлуатуються.

Рівень розвитку продуктів і систем автоматизації будинків є ключовим чинником, що забезпечує ефективне, безпечне, зручне й екологічно чисте функціонування будинків. Крім того, він впливає на всі елементи технічного оснащення будинку, особливо відносно систем опалення, вентиляції й кондиціювання повітря.

До продуктів автоматизації будинків ставляться спеціальні апаратні й програмні засоби й послуги з розробки й впровадження систем автоматизації й управління будинками.

Апаратні засоби включають датчики; виконавчі механізми й пристрої (керовані клапани, регулятори й т.д.); керуючі контролери, що здійснюють функції місцевого управління; комунікаційні контролери (маршрутизатори, шлюзи й т.п.), кабелі й кабельна арматури, призначені для побудови мереж необхідної топології, сумісності й продуктивності; а також комп'ютери для створення систем моніторингу й диспетчеризації систем управління будинку.

До програмних засобів ставиться:

– убудоване програмне забезпечення (ПЗ), що поставляється звичайно виготовлювачам устаткування й тому заставляється в ціну інтелектуальних апаратних засобів;

– ПЗ систем мережного зв'язку (для конфігурування, налаштування й тестування мереж);

– ПЗ для систем збору даних і диспетчерського управління (SCADA);

– спеціалізоване ПЗ для реалізації замовлених алгоритмів управління систем будинку.

Третю частину ринку становлять послуги з розробки й впровадження: проектування систем; розробка й прокладка кабельної системи будинку; послуги інжинірингу (реалізація алгоритмів управління системами, розробка зв'язку між підсистемами, розробка спеціалізованих апаратних засобів, написання замовленого ПЗ); монтаж устаткування; пусконалагоджувальні роботи, конфігурування, налаштування й тестування мережі й всієї системи управління.

Термін «інтелектуальний будинок» (intelligent building – англ.; intelligent – ‘розумний, тямущий’, у сполученні зі словом building – ‘гнучкий, що пристосовується’) у первісному змісті означає ‘будинок, готове до змін’ або ‘пристосовується здание, що,’, тобто будинок, здатне пристосовуватися до змін навколишнього середовища. Інакше кажучи, цей будинок, інженерні системи якого здатні забезпечити адаптацію до можливих змін у майбутньому.

Традиційні рішення інженерного устаткування будинки являють собою сукупність окремих, не взаємодіючих між собою систем. Будинок, у якому ці системи об'єднані в інтегрований комплекс і правильно організовані вже на етапі проектування (з обліком можливих майбутніх змін), має право називатися інтелектуальним.

У порівнянні з автономними системами комплексна система має наступні переваги:

– істотна економія на кабельних мережах і мережному устаткуванні;

– зниження енергоспоживання й підвищення надійності всієї системи;

– підвищення оперативності управління об'єктом;

– графічне подання інформації про стан систем і устаткування на різних рівнях (об'єктовому, зональному, адресному);

– зниження працезатрат експлуатаційних і диспетчерських служб;

– забезпечення необхідної взаємодії систем;

– зниження ймовірності виникнення страхових випадків;

– «відкритість» комплексу, що забезпечує можливість його нарощування й використання устаткування різних виробників.

Поняття «інтелектуальний будинок» ще не має точного тлумачення, але більшість людей, які ним користуються, сприймають його як автоматизовану технічну систему, що:

– «почуває», що відбувається усередині будинку й зовні;

– «реагує» таким чином, щоб найбільш ефективним способом забезпечити безпечне й комфортабельне перебування в ньому, звівши до мінімуму споживання енергії й енергоресурсів;

– «взаємодіє» з людьми за допомогою застосування простих і легко доступних засобів спілкування.

Інтелектуальний будинок є продуктом сучасного розвитку існуючих систем автоматики в будинках у напрямку:

– комплексної оптимізації використання ресурсів;

– підвищення гнучкості конфігурування й зниження загальної вартості володіння;

– інтеграції із широким спектром технологічного й телекомунікаційного устаткування;

– спрощення взаємодії з користувачем.

Характерні риси інтелектуальних будинків

До основних особливостей інтелектуальних будинків можна віднести:

– здатність оптимально реагувати на зміни в процесах, що відбуваються в будинку;

– сполучення децентралізованих (розподілених) принципів побудови систем із централізацією функції моніторингу;

– структурований підхід до побудови інженерних систем будинку;

– можливість внесення змін з мінімальними витратами;

– надання певного набору послуг мешканцям будинку.

Інтелектуальний будинок створюється для людини, тому основним критерієм ефективності проекту інтелектуального будинку є якість його взаємодії з мешканцями.

«Інтелект» житлового середовища сучасного будинку забезпечується взаємозалежною роботою автоматизованих будинкових і квартирних систем. Всі системи з'єднані високотехнологічними керуючими й інформаційною мережами, які прокладені у всіх житлових і суспільних приміщеннях будинку.

Основні системи інтелектуальних будинків

При інтеграції в структуру інтелектуальні будинки стають учасниками єдиної системи, тому особливе значення надається можливості гнучкої взаємодії з іншими підсистемами:

1. Комплекс систем життєзабезпечення. До складу входять:

– система управління вентиляцією й кондиціюванням повітря;

– система управління тепло- і водопостачанням;

– система управління електропостачанням;

– система управління освітленням;

– система управління поновлюваними джерелами енергії.

2. Комплекс систем безпеки. Забезпечують моніторинг стану інтелектуального будинку, запобігання й ліквідацію аварійних і небезпечних ситуацій, частково є надбудовою над технологічними підсистемами й можуть використовувати ті самі датчики, інтерфейси й виконавчі механізми, якщо це не заважає їхній роботі. До складу входять:

– контроль і управління електричними споживачами;

– контроль і управління внутрішнім кліматом;

– контроль протікань води;

– система пожежної безпеки;

– система охоронної сигналізація;

– контроль стану зовнішнього середовища;

– контроль і управління доступом до ресурсів будинку;

– контроль за дітьми;

– контроль і обслуговування свійських тварина.

3. Комплекс систем інформатизації. Є базисом, на якому будуються все компоненти інформаційно-обчислювальних мереж інтелектуального будинку. Правильна організація системи визначає надійність функціонування системи інтелектуального будинку як інтегрованого комплексу:

– мережа (ЛОМ);

– система телефонної мережі;

– система прийому ефірного й супутникового телебачення;

– телекомунікаційна підсистема;

– система радіофікації;

– засоби оперативного радіозв'язку персоналу й інші системи.

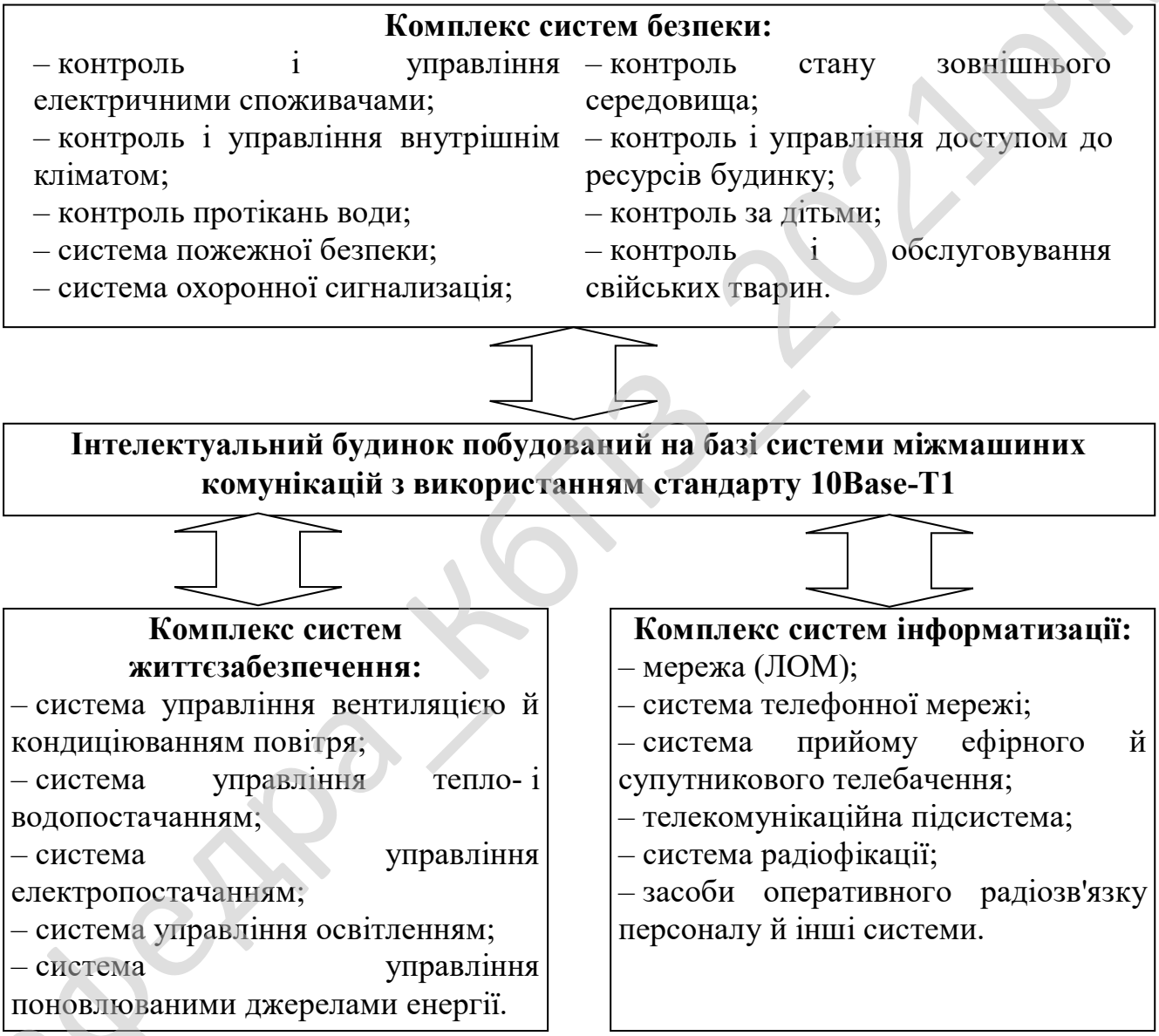

Рисунок 3.1 – Структурна схема системи

Управління інтелектуальним будинком виконується по сценаріях. Сценарії розділяються на дві основні групи:

– технологічні, які визначають роботу інженерних систем з метою створення безпечних (з погляду техніки, санітарних норм, екології) умов проживання;

– користувальницькі, які визначають комфортні умови проживання, максимально адаптовані до індивідуальних характеристик.

Визначення вимог до інтелектуальних будинків

Визначення вимог до інтелектуальних будинків простіше встановити виходячи із сукупності процесів життєдіяльності будинку, розглядаючи функціонування інтелектуальних будинків нерозривно від взаємодії з людиною. Абстрагуючись, інтелектуальні будинки можна представити як набір сервісів і способів їхньої реалізації. Ступінь автоматизації залежить від бажання людини перекласти на системи інтелектуальних будинків ту частину процесів, що повинна виконуватися автоматично, напівавтоматично або якимось зручним способом. Тому не може існувати єдиного рецепта по автоматизації будинку.

Пророблення вихідних вимог до інтелектуальних будинків повинна відбуватися в тісній взаємодії з тими, хто буде брати участь в експлуатації будинку, забезпечувати реалізацію сервісів і слуг, і тими, хто буде ними користуватися, або їхніми представниками. Нерідко в практиці будівництва сучасних будинків на комерційній основі немає можливості здійснити пророблення вимог з тими, хто буде експлуатувати будинок і користуватися його сервісами. У такому випадку організація пророблення вимог лягає на ті, хто їх представляє – комерційного забудовника або ріелтора. Їхнім завданням стає визначення співвідношення вартості пропонованого комерційного об'єкта до состава систем і сервісів, видам і рівню послуг проектованого об'єкта.

Успіх комерційного проекту визначається правильно обраним співвідношенням споживчих якостей і вартості. У цьому випадку під якістю розуміється вся сукупність параметрів об'єкта. І серед них немаловажним стає інтелектуальність будинку. Визначення системи якісних показників інтелектуального будинку, пропонованих споживачеві, стає однієї з основних

завдань при дослідженні ринку перед початком реалізації комерційного будівництва.

У розробці системи показників якості інтелектуального будинку зацікавлені всі учасники ринку. Організацію даної роботи можна виконати силами фахівців у цій області разом з комерційними забудовниками й ріелторами. Рішення даної проблеми дозволить припинити спекуляції на тему ступеня інтелектуальності будинків і допоможе формуванню цивілізованого ринку в будівництві.

Принципи побудови інтелектуального будинку

До основних технічних принципів побудови інтелектуального будинку ставляться:

– Стандартизація архітектури комплексу систем (відкритість систем). Під відкритістю розуміється наявність єдиного протоколу взаємодії устаткування різних виробників. В основі побудови інтелектуального будинку саме й лежать принципи «відкритої архітектури». При оснащенні будинку системами й устаткуванням від різних виробників важливо, щоб технічні пристрої не конфліктували між собою, а були б сумісні й представляли єдине ціле. Для того щоб системи розуміли один одного, вони повинні використовувати ті самі правила – стандарти – при обміні даними. В області телекомунікацій такі правила називають протоколами.

У цей час широке поширення в області систем управління будинками одержали стандарти 10 Base-T1,BACnet, LonWorks, EIB і ін.

Стандарт 10 Base-T1 був описаний вище. Саме він використовується в даному проекті.

Розглянемо інші стандарти, які можуть застосовуватися в області систем управління будинками.

Стандарт BACnet (Building Automation Control Network – мережний протокол для автоматизації будинків) був розроблений Американським

суспільством інженерів по опаленню, охолодженню й кондиціюванню повітря (ASHRAE).

Стандарт EIB (European Installation Bus – європейська інсталяційна шина) призначений для управління енергоспоживанням, освітленням, жалюзі, мікрокліматом і для контролю доступу; визначає вимоги до:

– каналів зв'язку (провідні, інфрачервоної, телефонні, радіо, мережі 220 В 50 Гц, оптоволокно, локальні комп'ютерні мережі Ethernet);

– формату інформації, що курсує;

– принципам взаємодії з користувачем (спеціалізовані інформаційні панелі й програмне забезпечення для персонального комп'ютера).

Технологія EIB дозволяє організувати передачу повідомлень від пристроїв фіксації подій до виконавчих механізмів по наступних інтерфейсах:

– провідні канали зв'язку;

– зв'язок по силовим електричним проводам;

– телефонні й радіоканали;

– інфрачервоне випромінювання;

– інтерфейси комп'ютерних мереж.

У європейських країнах все більше поширення в якості основного мережного стандарту одержує LonWorks, розроблений у компанії Echelon Corporation. Спочатку цей стандарт був розроблений для HVAC (систем опалення, вентиляції й кондиціювання повітря), однак у цей час він уже використовується при побудові комплексних систем (включаючи системи безпеки, обліку енергоносіїв, освітлення й ін.). З метою пропаганди й поширення стандарту LonWorks у травні 1994 року була створена асоціація LonMark, що поєднує виробників і інсталяторів Lon-Продуктів. Мережа управління LonWorks підтримує різні середовища для передачі інформації: кабель «кручена пари», коаксіальний кабель, волоконно-оптичний кабель, радіоканал і ін. Стандарт LonWorks дозволяє будувати системи управління будинками по вільній топології,

що щонайкраще відповідає структурі комплексних систем інтелектуального будинку:

– типізація устаткування й процесів;

– єдине фізичне середовище передачі інформації;

– централізація (функцій моніторингу й управління) і інтеграція систем;

– децентралізація (розподілені системи управління);

– сегментація (модульний принцип побудови систем);

– адаптація (готовність до змін);

– наращуємість і надмірність (наявність резерву).

Реалізація проекту інтелектуального будинку істотно відрізняється від традиційної схеми побудови будинку.

При проектуванні інтелектуального будинку визначальним принципом є формування єдиного підходу при побудові всіх систем різних комплексів.

3.3 Розробка функціональної схеми

Функціональна схема системи зображена на рисунку 3.2. З рисунку видно, що розроблена система складається з наступних блоків:

– Моніторинг трафіку міжмашиних комунікацій з використанням стандарту 10Base-T1.

– Робота з файлами.

– Монітор з'єднань.

– Статистика подій.

– Робота з сесіями.

– Функції для роботи з мережею міжмашиних комунікацій з використанням стандарту 10Base-T1.

– Робота з ресурсами мережі міжмашиних комунікацій з використанням стандарту 10Base-T1.

Розглянемо детальніше кожний з блоків.

Моніторинг трафіку міжмашиних комунікацій з використанням стандарту 10Base-T1 включає в себе:

– Визначення підключених інтерфейсів.

– Визначення вхідного та вихідного трафіку міжмашиних комунікацій з використанням стандарту 10Base-T1.

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

Робота з файлами включає в себе:

– Одержання списку відкритих файлів.

– Закриття відкритого файлу.

Можна переглянути, які з Ваших файлів, що Ви відкрили для загального доступу, переглядають по мережі міжмашиних комунікацій з використанням стандарту 10Base-T1. Програма відобразить список файлів та користувачів, які їх переглядають. Також можна відкрити чи закрити файл.

Монітор з'єднань включає в себе:

– Відстеження TCP- з'єднань.

– Відстеження UDP- з'єднань.

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

Статистика подій включає в себе відстеження подій в наступних протоколах:

– IP-протокол.

– ICMP-протокол.

– TCP-протокол.

– UDP-протокол.

Статистика ведеться по цілому ряду параметрів. Наприклад для TCP-протоколу фіксується: Тип алгоритму повторної передачі, мінімальний тайм-аут, максимальний тайм-аут, максимальна кількість помилок з'єднання, активні

з'єднання, пасивні з'єднання, невдалі спроби відкриття, скидання встановлених з'єднань, отримані сегменти, надіслані сегменти, повторно передані сегменти, помилки тощо.

Робота з 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.

Робота з ресурсами мережі міжмашиних комунікацій з використанням стандарту 10Base-T1 включає в себе:

– Визначення доступних ресурсів міжмашиних комунікацій з використанням стандарту 10Base-T1.

– Закриття локального ресурсу міжмашиних комунікацій з використанням стандарту 10Base-T1.

– Відкриття локального ресурсу міжмашиних комунікацій з використанням стандарту 10Base-T1.

– Приховання й показ ресурсів міжмашиних комунікацій з використанням стандарту 10Base-T1.

Система відображає на вні у мережі міжмашиних комунікацій з використанням стандарту 10Base-T1 ресурси у вигляді дерева. Пошук ресурсів міжмашиних комунікацій з використанням стандарту 10Base-T1 можна здійснювати по заданим умовам: локальні чи глобальні ресурси; всі ресурси, тільки файли, чи тільки принтери, тощо.

Можна додавати до загальних ресурсів мережі міжмашиних комунікацій з використанням стандарту 10Base-T1 свої власні, а також закривати їх потім.

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

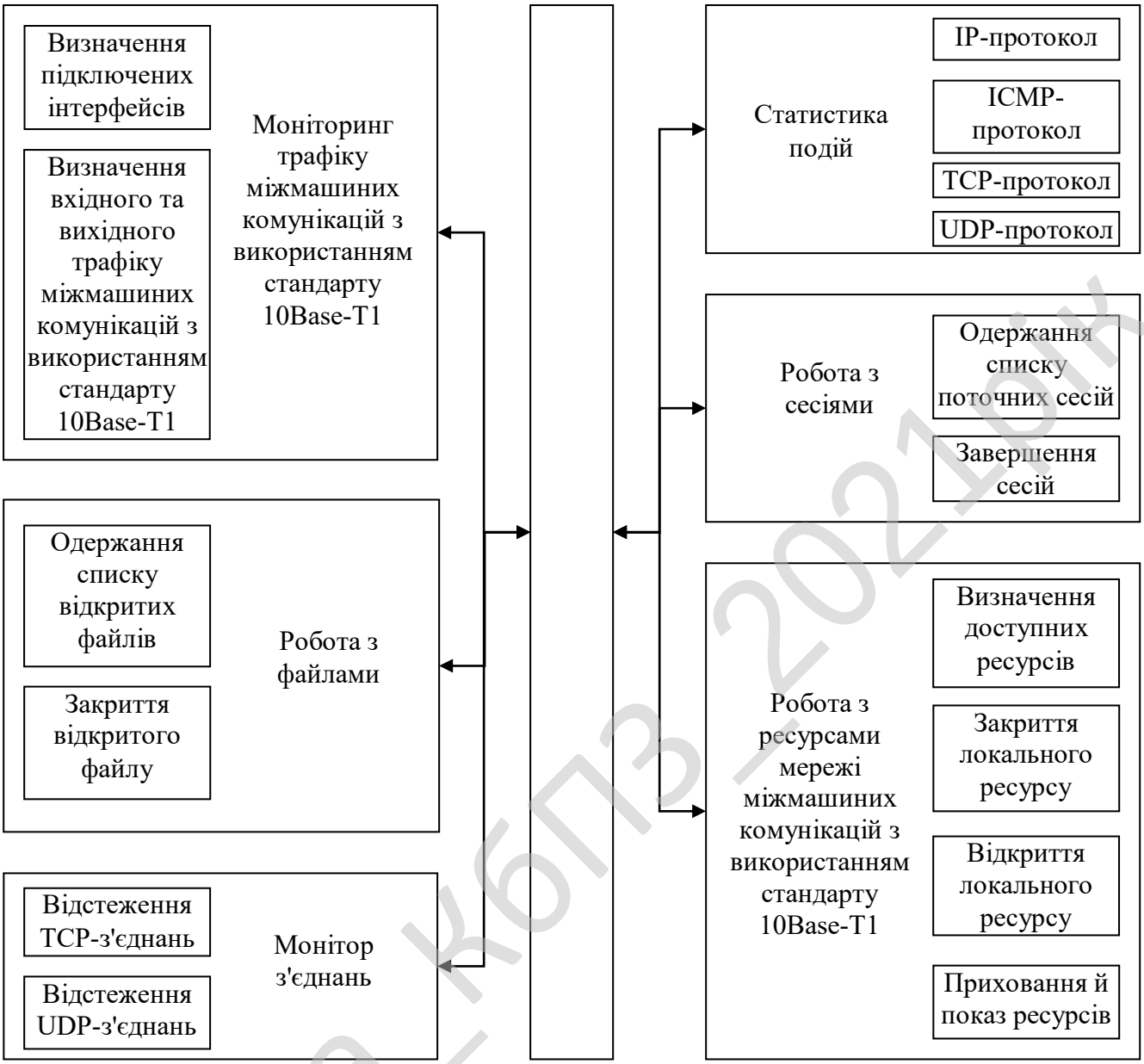

Рисунок 3.2 – Функціональна схема системи

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. 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.

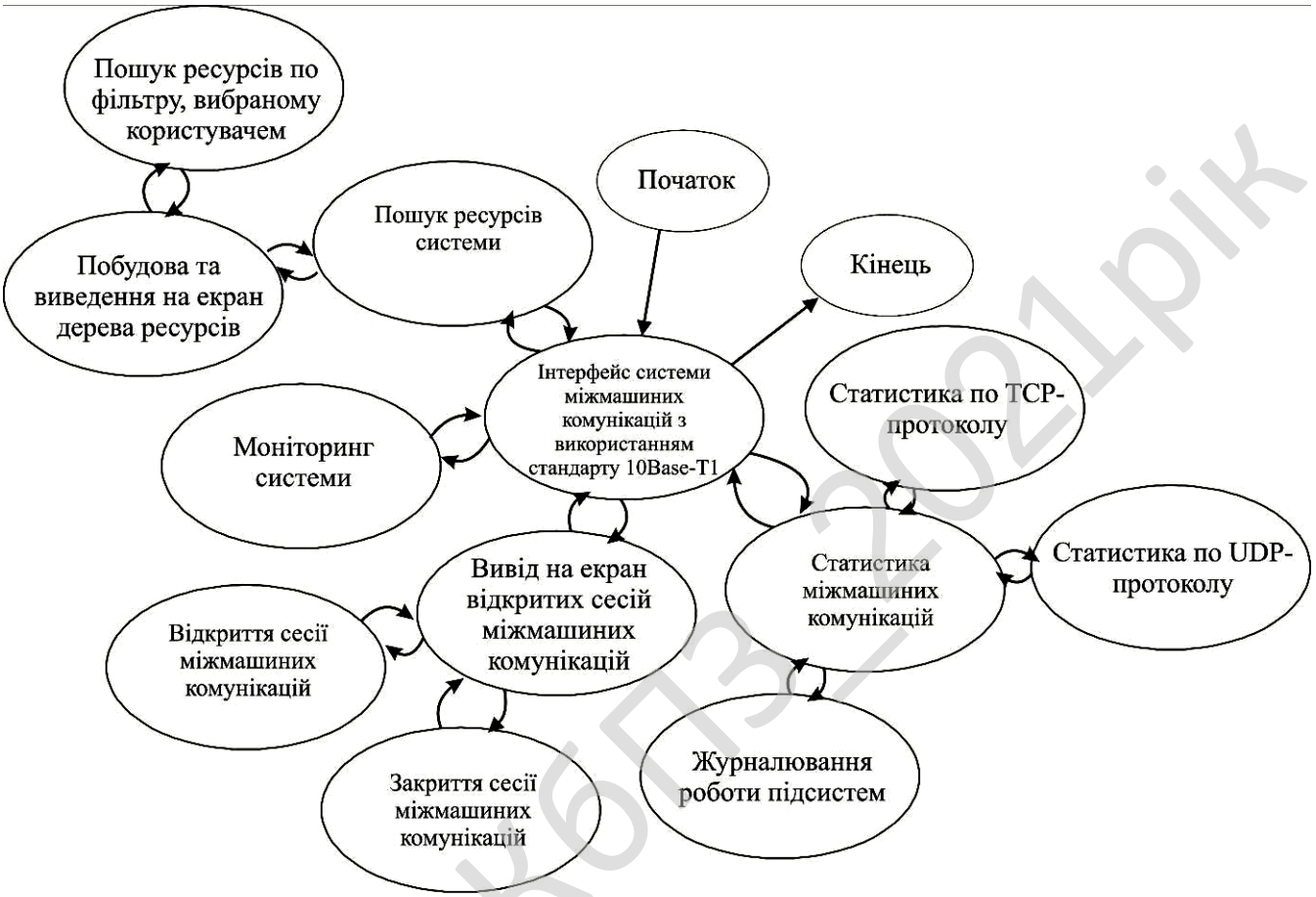

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

– Сховища даних (репозиторії).

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію магістерської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень. Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз'яснення який саме тип блок-схем використовується.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

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

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

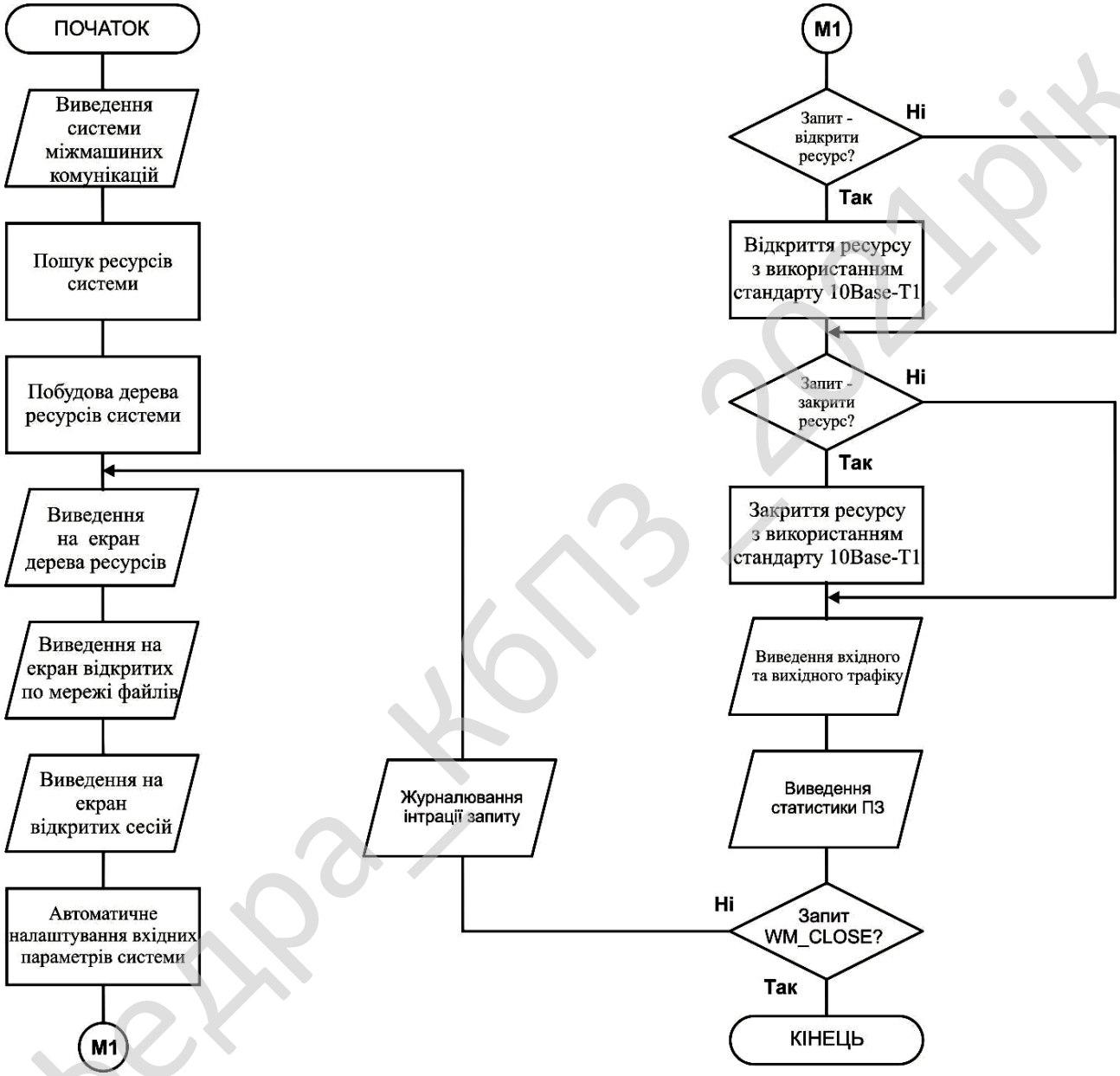

Рисунок 4.1 – Блок-схема основної програми

Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що

використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

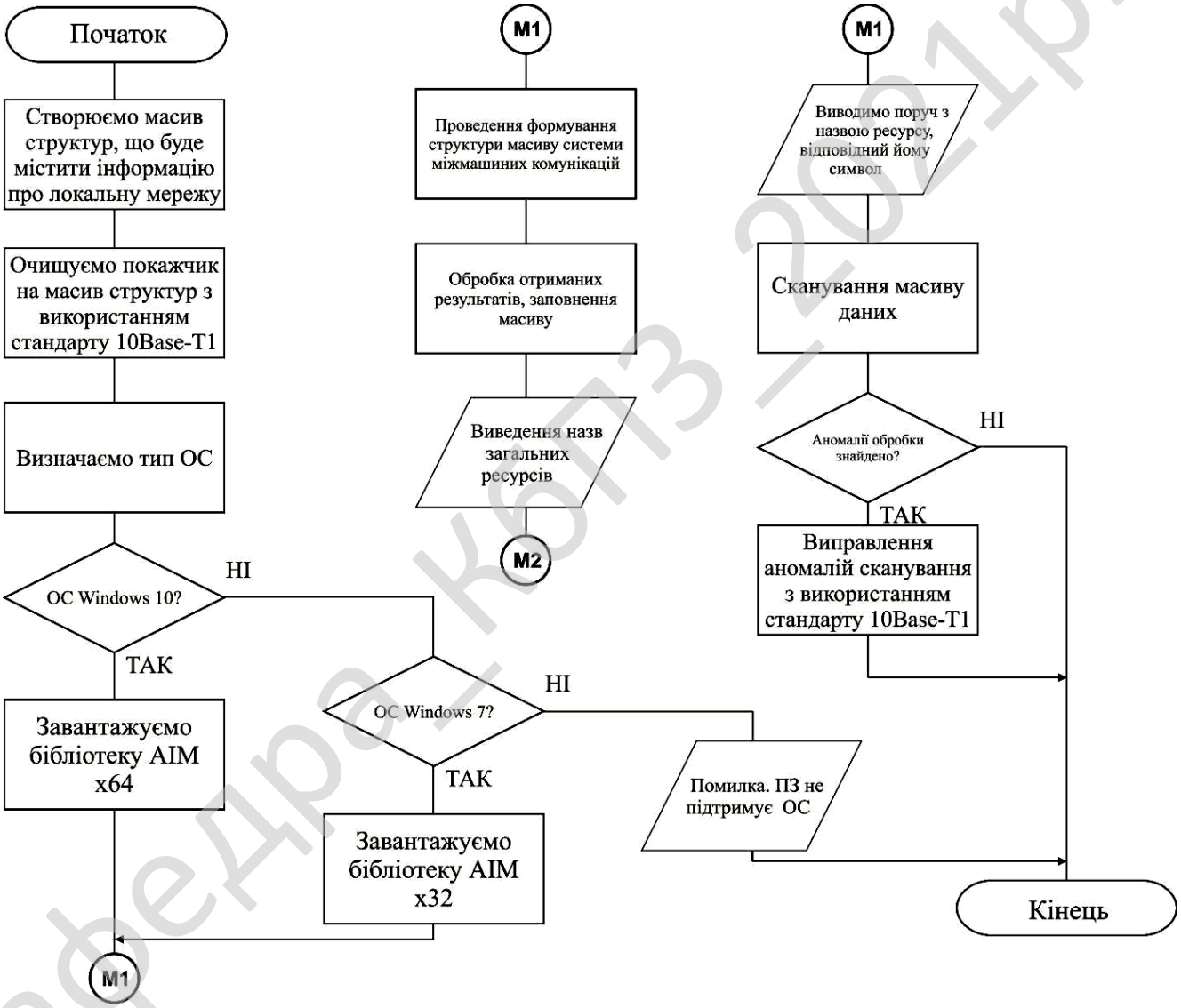

Рисунок 4.2 – Блок-схема роботи підпрограми

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка

робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

При розробці ПЗ було використано підходи ри́зик-ме́неджменту – це система управління ризиками, яка включає в себе стратегію та тактику управління, направлені на досягнення основних цілей. Ефективний ризик-менеджмент включає:

– систему управління;

– систему ідентифікації і вимірювання;

– систему супроводження (моніторингу та контролю).

Сучасна наука представляє ризик як вірогідну подію, в результаті настання якої можуть відбутися позитивні, нейтральні або негативні наслідки. Якщо ризик припускає наявність як позитивних, так і негативних результатів, він відноситься до спекулятивних ризиків. Якщо ж наслідки негативні, або відсутні взагалі, такий ризик іменується чистим.

Мета ризик-менеджменту – підвищення конкурентоспроможності господарюючих суб'єктів за допомогою захисту від реалізації чистих ризиків.

Теорія ризик-менеджменту ґрунтується на трьох базових поняттях: корисності, регресії і диверсифікації.

У 1738 швейцарський математик Даніель Бернуллі доповнив теорію вірогідності методом корисності або привабливості того або іншого результату подій. Ідея Бернуллі полягала в тому, що в процесі ухвалення рішення люди приділяють більше уваги розміру наслідків різних результатів, ніж їх вірогідність.

В кінці XIX століття англі йський дослідник Ф. Гальтон запропонував вважати регресію або повернення до середнього значення універсальною статистичною закономірністю. Суть регресії трактувалася ним як повернення явищ до норми з часом. Згодом було доведено, що правило регресії діє в найрізноманітніших ситуаціях, починаючи з азартних ігор та розрахунку вірогідності виникнення нещасних випадків, і закінчуючи прогнозуванням коливань економічних циклів.

У 1952 аспірант Університету Чикаго Гарі Марковіц в статті «Диверсифікація вкладень» («Portfolio Selection») математично обґрунтував стратегію диверсифікації інвестиційного портфеля, зокрема, він показав, як шляхом продуманого розподілу вкладень мінімізувати відхилення прибутковості від очікуваного показника. У 1990 Г. Марковіцу присуджена Нобелівська премія за розробку теорії і практики оптимізації портфеля фондових активів.

Етапи ризик-менеджменту

У ризик-менеджменті прийнято виділяти декілька ключових етапів:

– на першому етапі відбувається виявлення ризику з супутньою оцінкою вірогідності його реалізації і масштабу наслідків;

– на другому етапі здійснюється розробка ризик-стратегії з метою зниження вірогідності реалізації ризику і мінімізації можливих негативних наслідків;

– на третьому етапі вибираються методи і інструменти управління виявленим ризиком;

– на четвертому етапі проводиться безпосереднє управління ризиком;

– на завершальному етапі оцінюються досягнуті результати і коректується ризик-стратегія.

За ключовий етап ризик-менеджменту вважається етап вибору методів і інструментів управління ризиком.

Методи і інструментарій ризик-менеджменту

Базовими методами ризик-менеджменту є відмова від ризиків, зниження, передача і ухвалення.

Ризик-інструментарій значно ширший. Він включає політичні, організаційні, правові, економічні, соціальні інструменти, причому ризик-менеджмент як система допускає можливість одночасного застосування декількох методів і інструментів ризик-управління.

Найбільш часто вживаним інструментом ризик-менеджменту є страхування. Страхування припускає передачу відповідальності за відшкодування передбачуваного збитку сторонній організації (страхової компанії).

Прикладами інших інструментів можуть бути відмова від надмірно ризикової діяльності (метод відмови), профілактика або диверсифікація (метод зниження), аутсорсинг витратних ризикових функцій (метод передачі), формування резервів або запасів (метод ухвалення).

Розглянемо розробку проекта в контрольованому середовищі (PRojects IN Controlled Environments – PRINCE) – це метод управління проектами. Метод включає в себе управління, контроль і організацію проекту. «PRINCE2» – це другий реліз зазначеного методу, що є зареєстрованим торговим знаком, державної торгово–промислової палати (Office of Government Commerce – OGC), незалежного підрозділу державного казначейства Сполученого Королівства.

PRINCE2 походить від більш раннього методу PROMPT та методу проектного управління PRINCE, який був розроблений у 1989 р. Центральним телекомунікаційним та комп'ютерним агентством (Central Computer and Telecommunications Agency – CCTA), як державний стандарт Великобританії з управління проектами у сфері інформаційних технологій (IT). Незабаром зазначений стандарт почали використовувати за межами IT сфери. У 1996 р. PRINCE2 був визнаний універсальним методом управління проектами. PRINCE2 став надзвичайно популярним і зараз де–факто є стандартом проектного управління в Великобританії.

Остання версія була випущена у 2009 р., як результат проекту оновлення Prince2:2009 державної торгово–промислової палати Великобританії.

PRINCE2:2009 Оновлення: З 2006 р. метод переглядався, а 16 Червня, 2009 р. був опублікований під назвою «Оновлення PRINCE2:2009». Назва «PRINCE2» (замість «PRINCE3» чи схожої) використано задля демонстрації збереження головних принципів методу.

Все ж таки, це найбільш суттєвий перегляд методу з 1996 р. з метою його адаптації до мінливого бізнес середовища, спрощення і «полегшення», а також кращої інтеграції з іншими методами державної торгово–промислової палати Великобританії (ITIL, P3O, P3M3, MSP, etc.). Головна різниця між релізом 2009 р. та більш ранніми релізами полягає в появі двох довідників замість одного.

«Управління проектами, використовуючи PRINCE2» (Managing Projects Using PRINCE2) – оновлена методика управління, що призначена для менеджерів проектів, для яких управління проектами є щоденною діяльністю. «Стратегічне

управління проектами, використовуючи PRINCE2» (Directing Projects Using PRINCE2) – новий документ для Керівника проектів і Ради управління проектам, тобто спонсорів/замовників проектів.

Екзамен як з Теоретичної так і з Практичної частини буде базуватися на новому довіднику «Управління проектами» та не буде включати матеріали з довідника «Стратегічне управління проектами». Прохідний бал для Теоретичного екзамену залишиться не змінним. Прохідний бал для Практичного екзамену збільшиться з 50 % до 55 %. Тривалість екзамену з Практичної частини буде зменшено з 3 годин до 2,5 годин.

PRINCE2 – це структурований підхід до управління проектами, який спрямований на управління проектами в рамках чітко визначеної організаційної структури. PRINCE2 описує процедури координації людей та завдань в проекті, я к створювати/планувати проект та що робити, якщо проект потребує внесення змін через невідповідність фактичного стану виконання проекту плану його виконання. Кожний процес зазначено з ключовими вхідними та вихідними даними, а також з специфічними цілями та завданнями, які необхідно виконати, що загалом дає можливість контролю відхилень від плану.

Поділ методу на частини, що підлягають управлінню, забезпечує ефективний контроль ресурсів, завдяки чому можливо здійснювати контрольований та організований моніторинг проекту. PRINCE2 забезпечує єдину термінологію для всіх учасників проекту. Різноманітні ролі управлінців та зони відповідальності повністю описані та можуть бути адаптовані відповідно до складності проекту та компетенцій організації.

Помилки використання

Іноді PRINCE2 вважають не доцільним для дуже маленьких проектів через обсяг роботи, необхідної для створення та оновлення документів, протоколів та реєстрів. Досить часто це трапляється через нерозуміння, які частини PRINCE2 використовувати, адже PRINCE2 можливо повністю масштабувати.

Звичайно все це чудово, але сприйняття PRINCE2 погіршене тим, що для кожного ефективного проекту, який використовує методологію PRINCE2 існує певна кількість не ефективних проектів, які використовують, так звану, методологію PINO (PRINCE In Name Only – дослівно, PRINCE лише в назві).

Загальними проблемами є:

– Стадія початку проекту опускається/проштовхується і організація переходить відразу до створення Документу ініціювання проекту (Project initiation document – PID).

– Рада проекту не є ефективною – ескалації уходять в «чорну діру» тощо.

– Відсутній Опис продукту (Product description – PD), без якого управління якістю в PRINCE2 не працює.

– Припустимі відхилення не встановлюються, або встановлюються лише для часу виконання та вартості – чітко не визначено, в якій мірі повноваження передані виконавцям або коли необхідно використовувати ескалацію (піднімати рівень прийняття рішень).

– Проект фактично складається з однієї стадії (або «фази» використовуються, щоб уникнути Огляду виконання стадії завершення (End Stage Review)).

– Документи ініціювання проекту фактично копі–паст (copy–paste – копіювати – вставити) Документу ініціювання проекту з останнього проекту – їм не слідують і вони створюються тому що «потрібно».

– Низький рівень залучення «бізнесу», адже PRINCE2 вважається «технічним» методом – виходи (продукти) проекту можуть не призвести до бажаних результатів.

– Успіхи та невдачі використання PRINCE2 були однією з причин створення Моделі зрілості PRINCE2 (PRINCE2 Maturity Model – P2MM).

Модель зрілості PRINCE2. Модель зрілості PRINCE2 описує набір Сфер Ключових Процесів (Key Process Areas – KPAs), які необхідні для ефективного впровадження та використання PRINCE2 в організації. Довідник PRINCE2 описує

управління проектом і не включає опис процесів впровадження PRINCE2 в організації, в той час як P2MM описує саме процеси впровадження PRINCE2.

P2MM описує ключові практики, які додаються до процесів та складових PRINCE2 з метою забезпечення повторюваного використання методу (Рівень 2 KPAs), а також ключові практики необхідні для впровадження методу (Рівень 3 KPAs), як стандартного бізнес–процесу управління проектами, що включає: призначення власника (3.1), адаптацію методу (3.2), навчання (3.3), інтеграцію з іншими системами управління (3.4), механізм забезпечення якості (3.5).

Огляд методу

Блок–схема процесів PRINCE2. Стрілки означають потоки інформації. PRINCE2 – це процесуальний метод управління проектами на відміну від адаптивних методів, таких як Scrum. PRINCE2 2009 визначає 40 окремих активностей (завдань) і об'єднує їх у сім процесів.

Початок проекту.

На цьому етапі визначаються учасники проектної команди та готується короткий опис проекту (опис цілей проекту та комерційне обґрунтування проекту). Відповідно до загальних засад методу на цьому ж етапі планується наступний етап проекту. Після завершення зазначених робіт Рада проекту має погодити наступний етап проекту – ініціювання проекту.

Головні завдання включають: призначення керівника проекту та менеджера проекту, визначення та призначення команди управління проектом, створення короткого опису проекту, визначення методу управління проектом, планування наступного етапу проекту (ініціювання).

Ініціювання проекту

Цей етап базується на результатах виконання завдань етапу початку проекту. Короткий опис проекту розширюється до бізнес плану (Business case). Точки контролю якості проекту узгоджуються з точками контролю стану виконання проекту. Створюється план виконання наступного етапу проекту.

Результати етапу виносяться на погодження Ради проекту з метою погодження продовження роботи над проектом.

Головні завдання включають: планування показників якості, створення плану проекту, деталізація бізнес плану та ризиків, визначення точок контролю проекту, створення Документу ініціювання проекту (Project Initiation Document).

Стратегічне управління проектом

Цей процес визначає яким чином Рада проекту (що уособлює роль спонсору проекту) буде загалом контролювати проект. Рада проекту дозволяє/санкціонує етап ініціювання проекту, а також сам проект. Стратегічне управління також визначає яким чином Рада проекту має погоджувати план етапів, включаючи погодження будь-якого плану етапу, що може змінити існуючий план проекту у зв'язку з будь-якими змінами строків або ключових показників проекту.

Також визначається, яким чином Рада проекту може змінити план проекту, способи закриття проекту.

Головні завдання включають: засоби погодження ініціювання проекту, погодження проекту, погодження етапу або вилучення етапу проекту, підтвердження завершення проекту.

Контроль

PRINCE2 передбачає поділ проекту на етапи, які підлягають контролю. Загалом визначено яким чином пакети завдань призначаються та погоджуються. Також визначається яким чином відслідковується статус виконання завдань і яким чином такий статус доповідається Раді проекту. Засоби для відслідковування та оцінювання виконання проекту пропонуються разом з способами застосування коригуючих дій. Також закріплюються методи ескалації визначеного кола проблем Раді проекту.

Головні завдання включають: призначення та погодження пакетів завдань, оцінювання статусу виконання завдань, звітування статусу виконання завдань,

застосування коригуючи дій, ескалація проблем проекту, підтвердження завершення пакету завдань.

Управління межами етапів

Контроль виконання етапів проекту – це визначення переліку завдань, які мають бути виконані в межах етапу. Управління межами етапу визначає, що має бути виконано при завершені етапу. При завершені етапу необхідно створити план виконання наступного етапу. Загальний план проекту, перелік ризиків проекту та бізнес план за необхідності переглядається та оновлюється. Також визначається перелік дій на випадок, якщо етап не відповідає показникам виконання. Загалом визначається, яким чином звітується про завершення етапу.

Головні завдання включають: планування етапу, оновлення плану проекту, оновлення бізнес плану проекту, оновлення ризиків проекту, звітування про завершення етапу, створення плану дій на випадок не виконання певних завдань проекту.

Завершення проекту

Визначаються завдання, які повинні бути виконані при завершені проекту. Проект має бути формально завершений (ресурси мають бути вивільнені для виконання інших завдань), результат виконання завдань має бути визначений, а сам проект має отримати формальну оцінку.

Головні завдання включають: завершення проекту, визначення результату виконання завдань, оцінку виконання проекту.

Управління вимогами це процес запису, аналізу, трасування, пріоритезації і узгодження вимог та контролю змін і доведення до їх зацікавлених сторін. Це безперервний процес протягом всього життя проекту. Вимога – якість, якій мають відповідати результати проекту (продукту або послуги).

Мета управління вимогами полягає в тому, щоб переконатися, що організація відповідає потребам і очікуванням своїх клієнтів, внутрішніх або зовнішніх зацікавлених сторін. Управління вимогами починається з аналізу і виявлення цілей і обмежень організації. Управління вимогами додатково включає

в себе підтримку планування вимог, інтеграції вимог і організації роботи з ними (атрибути для вимог).

Управління вимогами передбачає спілкування між членами проектної групи і зацікавленими сторонами, і адаптацію до змін у вимогах протягом всього проекту. Щоб запобігти перетину поля одного класу вимог з іншим, постійні зв'язки між членами команди розробників є критичними. Наприклад, при розробці програмного забезпечення для внутрішнього використання у бізнесу можуть бути настільки сильні потреби, що він може проігнорувати вимоги користувачів, або вважати, що створені сценарії використання покриють також і користувальницькі вимоги.

Відслідковування вимоги фактично означає документування всього життєвого циклу вимоги. Часто необхідно дізнатися першоджерело кожної вимоги. Для цього всі зміни вимог повинні бути задокументовані, щоб досягти стану повного відстеження. Відстежувати треба бути навіть використання реалізованих вимог.

Вимоги мають різні джерела, такі як ділова людина, що замовляє продукт, менеджер зі збуту і фактичний користувач. У всіх цих людей є різні вимоги до продукту. Використовуючи відслідковування вимог, реалізована в системі функція може бути простежена назад до людини або групі, яка замовляла її під час збору вимог. Ця особливість може, наприклад, використовуватися в процесі розробки для пріоритезації вимог, визначаючи, наскільки цінною є дана вимога для певного користувача.

Відслідковування може також використовуватися після розгортання продукту. Наприклад, коли вивчення використання системи показує, що якась функція не використовується, можна визначити навіщо вона була потрібна спочатку.

Завдання управління вимогами

На кожному етапі процесу розробки існують ключові методи і задачі пов'язані з управлінням вимогами. Для ілюстрації, розглянемо наприклад

стандартний процес розробки з п'ятьма фазами: дослідженням, аналізом здійсненності, дизайном, розробкою та тестуванням і випуском.

Дослідження. Під час фази дослідження збираються перші три класи вимог від користувачів, бізнесу і команди розробників. У кожній області задають однакові питання: які цілі, які обмеження, які використовуються процеси та інструменти і так далі. Тільки коли ці вимоги добре зрозумілі, можна приступати до розробки функціональних вимог.

Тут необхідне застереження: незалежно від того, як сильно група намагається це зробити, вимоги не можуть бути повністю визначені на початку проекту. Деякі вимоги змінюються, або тому що вони просто не були знайдені спочатку, або тому що внутрішні чи зовнішні сили торкаються проекту в середині циклу. Таким чином, учасники групи повинні спочатку погодитися, що головна умова успіху – гнучкість у мисленні та діях.

Результатом стадії дослідження є документ – специфікація вимог, схвалений усіма членами проекту. Пізніше, в процесі розробки, цей документ буде важливий для запобігання розповзання меж проекту або непотрібних змін. Оскільки система розвивається, кожна нова функція відкриває світ нових можливостей, таким чином специфікація вимог прив'язує команду до оригінального бачення системи і дозволяє контрольоване обговорення змін.

У той час як багато організацій все ще використовують звичайні документи для керування вимогами, інші управляють своїми базовими вимогами, використовуючи програмні інструменти.

Ці інструменти керують вимогами використовуючи базу даних, і зазвичай мають функції автоматизації відстеження (наприклад, дозволяючи створювати зв'язки між батьківськими і дочірніми вимогами, або між тестами і вимогами), управління версіями, і управління змінами. Зазвичай такі інструментальні засоби містять функцію експорту, яка дозволяє створювати звичайний документ, експортуючи дані вимог.

Аналіз здійсненності

На стадії аналізу здійсненності визначається вартість вимог. Для користувальницьких вимог поточна вартість роботи порівнюється з майбутньою вартістю встановленої системи. Задаються питання такі як: «Скільки нам зараз варті помилки введення даних?» Або, «Яка вартість втрати даних через помилки оператора пов'язаної з використовуваним інтерфейсом?». Фактично, потреба в новому інструменті часто розпізнається, коли подібні питання потрапляють до уваги людей, що займаються в організації фінансами.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою національного стандарту захисту інформації на основі алгоритму шифрування/дешифрування ДСТУ 4145-2002 з використанням еліптичних кривих над двійковим розширеним полем Галуа. У системі шифрування/дешифрування як параметри розглядається еліптична крива E_p (a,b) і точка G на ній. Учасник В вибирає закритий ключ n і обчислює відкритий ключ $P_B = n \times G$. Щоб зашифрувати повідомлення P_m використовується відкритий ключ одержувача В P_B. Учасник А вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій.

$$C_m = \{k \times G,\ P_m + k \times P_B\}. \quad (4.1)$$

Щоб дешифрувати повідомлення, учасник В множить першу координату точки на свій закритий ключ і віднімає результат від другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m. \quad (4.2)$$

Учасник А зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Супротивнику для відновлення повідомлення доведеться обчислити k. Зробити це буде нелегко. Одержувач також не знає k, але йому як підказку посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач одержить значення, що було

додано відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи *k*, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

Нехай задано просте число $p > 4$. Тоді еліптичною кривою *E*, визначеною над розширеним двійковим полем полем F_{2^m}, називається безліч пар чисел $(x, y), x, y \in F$, що задовольняють тотожності:

$$y^2 \equiv x^3 + a \cdot x + b (\mathrm{mod}\, 2^m), \tag{4.3}$$

де $4 \cdot a^3 + 27 \cdot b^2$ не рівно з нулю по модулю 2^m.

Інваріантом еліптичної кривої називається величина *J*(*E*), що задовольняє тотожності:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} (\mathrm{mod}\, 2^m). \tag{4.4}$$

Коефіцієнти *a*, *b* еліптичної кривої *E*, по відомому інваріанту *J*(*E*), визначаються таким чином:

$$\begin{cases} a \equiv 3k (\mathrm{mod}\, 2^m) \\ b \equiv 2k (\mathrm{mod}\, 2^m) \end{cases} \text{де } k \equiv \frac{J(E)}{1728 - J(E)} (\mathrm{mod}\, 2^m), J(E) \neq 0 \text{ або } 1728. \tag{4.5}$$

Пари (*x*,*y*), що задовольняють тотожності (4.1), називаються точками еліптичної кривої *E*, *x* та *y* – відповідно *x*– та *y*-координатами точки.

Точки еліптичної кривої позначатимемо *Q*(*x*,*y*) або просто *Q*. Дві точки еліптичної кривої рівні, якщо рівні їх відповідні *x*– і *y*-координати.

На безлічі всіх точок еліптичною кривою E введемо операцію додавання, яку позначатимемо знаком "+". Для двох довільних точок $Q_1(x_1, y_1)$ та $Q_2(x_2, y_2)$ еліптичної кривої E, розглянемо декілька варіантів.

Нехай координати точок Q_1 та Q_2 задовольняють умові $x_1 \neq x_2$. В цьому випадку їх сумою називатимемо точку $Q_3(x_3, y_3)$ координати якої визначаються порівняннями:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, 2^m). \tag{4.6}$$

Якщо виконана рівність $x_1 = x_2$ та $y_1 = y_2 \neq 0$, то визначимо координати точки Q_3 таким чином:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{3x_1^2 + a}{2y_1} (\mathrm{mod}\, 2^m). \quad (4.7)$$

У разі, коли виконана умова $x_1 = x_2$ та $y_1 = -y_2 (\mathrm{mod}\, p)$, суму точок Q_1 та Q_2 називатимемо нульовою точкою О, не визначаючи її х– і у-координати. В цьому випадку, точка Q_2 називається запереченням точки Q_1. Для нульової точки О виконана рівність:

$$Q + 0 = 0 + Q = Q, \quad (4.8)$$

де Q – довільна точка еліптичної кривої Е.

Щодо введеної операції складання безліч всіх точок еліптичною кривою Е, разом з нульовою точкою, утворюють кінцеву абельову (комутативну) групу порядку т, для якого виконана нерівність:

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p} \quad (4.9)$$

Точка Q називається точкою кратності k, або просто – кратною точкою еліптичної кривої E, якщо для деякої точки P виконана рівність:

$$Q = P + ... + P = kP \quad (4.10)$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у магістерської дипломної роботі система міжмашиних комунікацій з використанням стандарту 10Base-T1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Розділу обрання групи; Розділу виведення результату роботи системи.

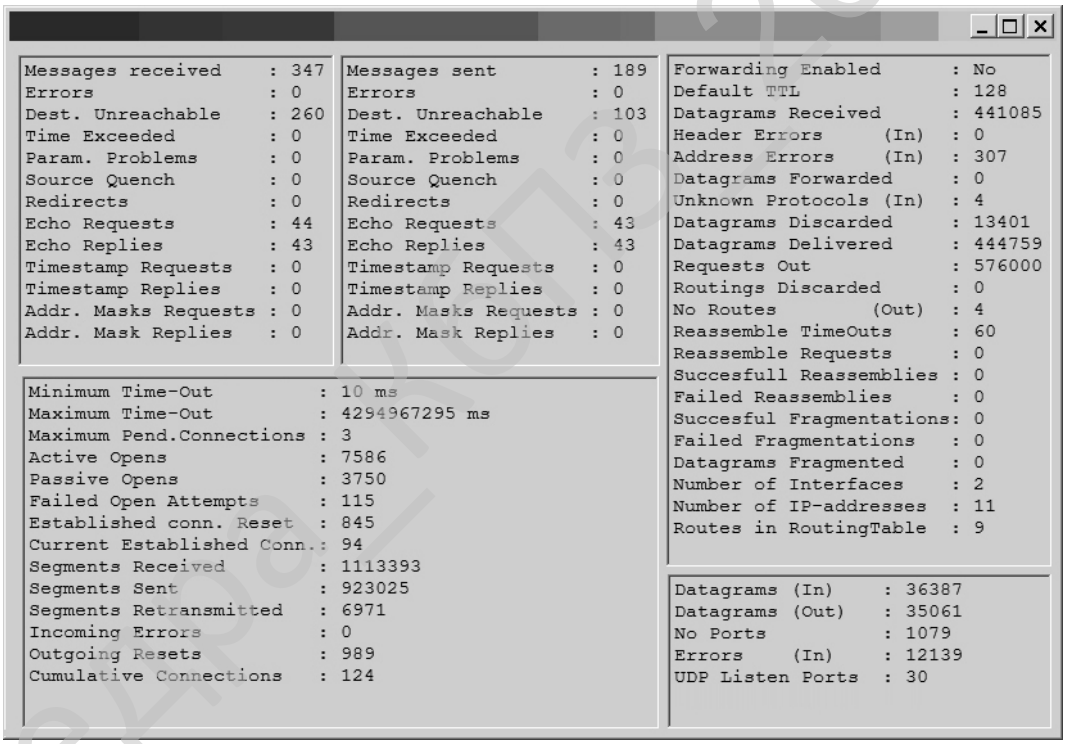

Рисунок 5.1 – Головне вікно ПЗ

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.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

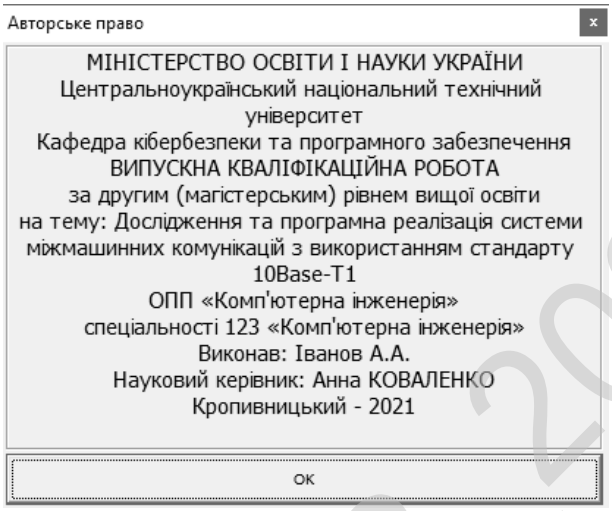

Рисунок 5.2 – Авторське право

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.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc.

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії

(тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Метою розробки є дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Об'єктом дослідження є процес міжмашинних комунікацій з використанням стандарту 10Base-T1.

Предметом дослідження є методи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод міжмашинних комунікацій з використанням стандарту 10Base-T1.

– Розроблено вітчизняний продукт міжмашинних комунікацій з використанням стандарту 10Base-T1, який має більш широ кі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір та системні потреби;

б) незалежність від встановлених на комп'ютері баз даних;

в) зручність у користуванні та надійність

Таблиця 7.1 – Початкові данні

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	60 (2 ост. цифри № зал *10)
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	60000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B \qquad (7.1)$$

де А – коефіцієнт Боема, А=2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПВ_j, \qquad (7.3)$$

де $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$

$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \qquad (7.4)$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot3{,}23\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot150=306 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	306	Ф 7.1-7.4
Впровадження	13	Д13
Всього	347	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів,

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{347 \cdot 1}{48\text{-}5} = 8 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість обладнання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 2}{1,2} = 378 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=378/(48·8)=0,98 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів–електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	1	15000	30000
Продакт-менеджер	0,5	12000	12000
Інженер-програміст	8	14000	224000
Інженер-електронщик	1	9000	18000
Інженер-системотехнік	0,2	9000	3600
Адміністратор мережі	0,2	11000	4400
Системний програміст	0,2	9000	3600
Дизайнер WEB	0,2	9000	3600
Інженер-верстальник	0,2	9000	3600
Бухгалтер-економіст	0,2	10000	4000
Всього за період розробки	$R_{сп}$=11,7	-	$Ф_{роб}$=306800

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{306800}{11,7 \cdot 48} = 546 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць.

$S_у$ – питома площа на одне робоче місце, $м^2$,

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$I_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за комерційною пропозицією фірми Brain за 15.10.21 – джерело http://brain.com.ua/

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	Intel Core i7-2700K (S1155/4x3.5GH /5GT/s/8MB/95 Вт)	2500
Системна плата	Huanan B75 (s1155, Intel B75, PCI-Ex16 DVI/VGA/HDMI	1100
Жорсткий диск	HDD 500 Gb SAMSUNG Barracud HD502HJ (3.5", 500ГБ, 16МБ, SATA II-300)	1290
Оперативна пам’ять	DIMM 4096Mb DDR3 PC3-12800 Kingstor 1600MHz, 512M x 64, CL9-9-9-27, 1.65V w/heatsink, HyperX	834
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	416
Корпус	Logicpower 8702 – 550w 12cm	1411
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	220
інше	Клавіатура, мишка	Подаруно
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість ,шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	8	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	143000	-	
6. Господарський інвентар	28000	-	-
Всього по групі	176190	20	35238
7. Нематеріальні активи	60000	10	6000
Разом	$К_р$= 1769406		$А_р$= 174246

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де $N_е$ – Кількість екземплярів програм, шт.

$$З_о = 546 \cdot 347/60 = 3158 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 3158 \cdot 10 \cdot 0{,}01 = 316 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(3158+316) = 764 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом *Нг*=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картріджей, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=80 грн, визначаємо вартість паперу за період розробки $N_м$=2 міс:

$$З_{М1} = Ц_п \cdot N. \quad (7.16)$$

$$З_{М1} = 80 \cdot 1 = 80 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д., \quad (7.17)$$

де $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 2 грн/шт, DVD-R LG 4,7Gb, 16x speed Cake box – 2 грн/шт.

$$З_{М2} = 60 \cdot 8 + 10 = 490 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \tag{7.18}$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (80 + 490 + 1702)/60 = 38 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \tag{7.19}$$

де $Н_п$ – норматив витрат на освоєння нових мов програмування, %

$$О_п = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 60 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \tag{7.20}$$

де $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 174246 \cdot 2/(60 \cdot 12) = 484 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \tag{7.21}$$

$$С_п = 3158 + 316 + 764 + 474 + 38 + 474 + 484 = 5708 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де $Р_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 5708 = 2854 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	3158
2. Додаткова зарплата виконавців	$З_д$	316
3. Відрахування на соціальні потреби	$С_{оц}$	764
4. Загальногосподарські витрати	$Г_{осп}$	474
5. Витрати на матеріали	$З_М$	38
6. Освоєння нових операційних систем, мов програмування	$О_п$	474
7. Амортизація основних фондів	$А_м$	484
8. Повна собівартість програмного забезпечення	$С_п$	5708
9. Плановий прибуток	$П_р$	2854
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	8562
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1712,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	10274,4

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	10274
Всього капітальних витрат	–	10274

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позна-чення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	29524	12078
2. Витрати на електроенергію	$З_{ел}$	2736	1368
3. Витрати на амортизацію	$З_{ам}$	0	5137
Всього витрат за рік	$І$	32260	18583

Витрати на технічне обслуговування:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*,

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 220 годин на рік до 90 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 220 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 29524 \text{ грн.}$$

до

$$З_{р\,нов} = 90 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 12078 \text{ грн.}$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$).

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Зел баз=0,2·7200·1,9=2736 грн

Зел нов=0,2·3600·1,9=1368 грн

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	10274	–	5137
Всього відрахувань	-	–	10274	–	5137

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (8562 - 5708) \cdot 60 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 + \\ +0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 142199 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{361406}{(8562 - 5708) \cdot 60 \cdot 12 / 2} = 0,35 \text{ років}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (32260\text{-}18583)\text{-}0,5 \cdot 10274 = 8540 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{10274}{32260 - 18583} = 0,75 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	60
2. Повна собівартість розробленої програми	Грн.	5708
3. Ціна розробленої програми	Грн.	8562
4. Плановий прибуток від реалізації розробленої програми	Грн.	2854
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1769406
7. Загальний прибуток від реалізації програмної продукції	Грн.	171240
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	142199
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,35
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	10274
11.Величина економічного ефекту у користувача програмної продукції	Грн.	8540
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,75

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя. Охорона праці є складовою частиною безпеки життя.

Законом України “Про охорону праці” регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж ic тотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та

«Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	4
Довжина	5
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,6
Обсяг, V	$м^3$	не менше 20.0	20

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працє 3 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у

розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23,5	41-58	0,11
Тепла	23-25	50-70	0,1	24-25	50-60	0,12

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер *Xerox Phaser 3020BI*, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп’ютером, згідно державних будівельних норм ДБН В.2.5 – 28 – 2006 р. (які замінили СНиП 11-4-79), можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд

зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою

наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору розтіканню електричного струму на землю).

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Початкові данні для розрахунку штучного захисного заземлення:

Тип заземлення: повторне заземлення нульового дроту на вводі в об'єкт. Напруга – 220/380 В. Розташування заземлюючих електродів – по контуру.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача методом коефіцієнта використання заземлювачів.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – суглинок полутвердий. Умовна товщина верхнього шару грунта: $H=0,7$ *м*. Для захисного заземлення: застосовуються вертикальні електроди – пруток довжиною $L=2$ *м.* Відстань між вертикальними заземлювачами (електродами) $A=4$ *м*. Діаметр вертикального електрода (прутка) $D=12$ *мм*, Тип гори зонтального заземлювача: металевий пруток. Діаметр горизонтального електрода (прутка) *12 мм* ($b=12$ *мм*.). Опір заземлювача, який нормується: $R_{ЗН} = 10$ *Ом*. Глибина закладення горизонтального контура заземлення $t= 0,7$ *м*.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+2/2=1,7 \text{ м.}$$

Еквівалентний питомий опір грунта:

$$\rho_{екв}= \psi \rho_1 \rho_2 L /[\rho_1 \psi(L-H+t)+ \rho_2 \psi(H-t)]=100 \text{ Ом}$$

де

$\psi = 1,2$ – табличне значення коефіцієнта сезонності для відповідної климатичної зони у багатошаровому грунті (джерело: http://ftemk.mpei.ac.ru/bgd/_private/Grunt_/t_goda.htm);

$\rho_1 = 50$ *Ом*м.* – табличне значення питомого опору верхнього шару грунта (джерело: https://el-line.ru/udelnoe_soprotivlenie_grunta.shtml);

$\rho_2 = 100$ *Ом*м.* – табличне значення питомого опору нижнього шару грунта (джерело: https://zandz.com/ru/udelnoe_soprotivlenie_grunta).

Опір розтіканню електричного струму одного електрода вертикального заземлювача:

$$R_O = [\rho_1/(2\pi*L)]*[ln(2L/D)+0,5ln(4T+L)/(4T-L)] = 48,66 \text{ Ом}.$$

Опір розтіканню електричного струму заземлювача, який нормується, $R=1$ якщо $\rho_{екв} \leq 100$ *(у нас* $\rho_{екв} = 100 \leq 100$*),* *та* $R = R_{ЗН} * \rho_{екв}/100$, *якщо* $\rho_{екв} > 100$.

Попередня (орієнтовна) кількість вертикальних заземлювачів:

$$N_{попередн.} = R_O \psi / R_{ЗН} = 48,66 * 1,2/10 = 5,7 \text{ шт.}$$

Опір розтіканню електричного струму горизонтального заземлювача (полоси):

$$R_П = 0,366(\rho_{екв} \psi / L_П \cdot \eta_П) lg(2 / L_П * L_П / bt) = 28,5 \text{ Ом.}$$

де $\eta_П = 0,48$ – табличне значення коефіцієнта використання горизонтального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення *A/L*; довжина горизонтального заземлювача (полоси) при розташуванні заземлювачів по контуру:

$$L_П = A*N = 4*4 = 16 \text{ м.};$$

де n – ітераційна кількість вертикальних заземлювачів.

Загальний опір штучного заземлюючого пристрою розтіканню електричного струму на землю:

$$R_B = R_П R / (R_П - R) = 15,41 \text{ Ом}.$$

Враховуючи, що значення ρ_1 *та* ρ_2 по факту можуть бути меншими, розрахункове значення R_B цілком прийнятне.

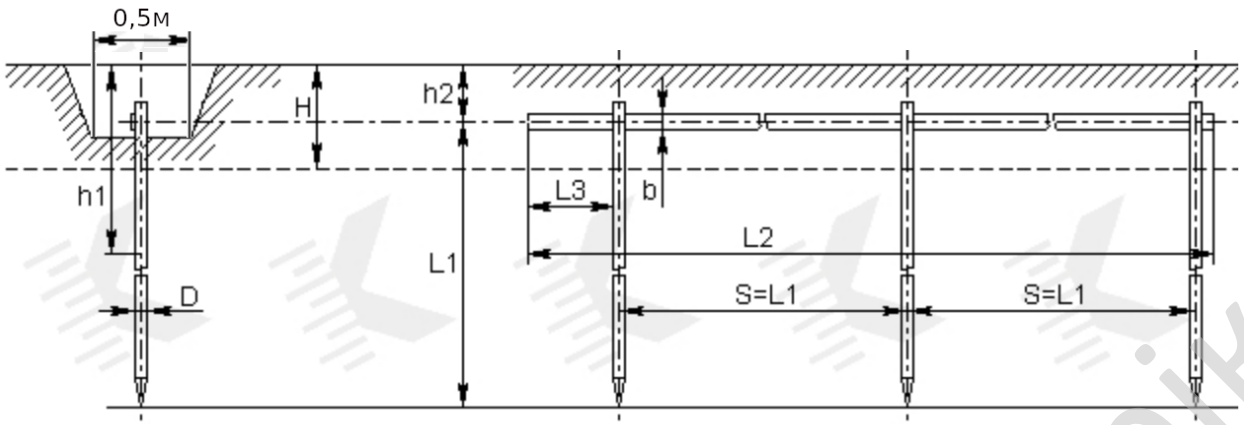

Рисунок 8.1 – Штучне повторне заземлення нульового дроту на вводі в об’єкт

Остаточна кількість вертикальних заземлювачів:

$$N = R_O \ / (R_B * \eta_B) = 4 \text{ шт.}$$

де $\eta_B = 0,73$ – табличне значення коефіцієнта використання вертикального заземлювача, залежить від розташування (в ряд або по контуру) та співвідношення A/L.

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр.13-16 [4], або аналогічної.

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров’я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів міжмашинних комунікацій з використанням стандарту 10Base-T1.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем міжмашинних комунікацій з використанням стандарту 10Base-T1.

– Досліджена система міжмашинних комунікацій з використанням стандарту 10Base-T1.

– На основі отриманих результатів досліджень створена програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання міжмашинних комунікацій з використанням стандарту 10Base-T1.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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 Embarcadero RAD Studio Delphi 10.3.2. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 ДСТУ 4145-2002.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 8540 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,75 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іванов А.А. Дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1 // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Абрамчук В.Е. Технические средства диагностирования: справочник / В.Е. Абрамчук, В.В. Клюев, П.П. Пархоменко; под общ. ред. В.В. Клюева. – М.: Машиностроение, 1989. – 672 с.

3. Автоматизовані системи. Терміни та визначення: ДСТУ 2226-93 – [Чинний від 1993–09–09]. – Київ: Держстандарт України, 1994. – 91 с. – (Національний стандарт України).

4. Аде Ф.Г. Искусственный интеллект / Ф.Г. Аде., В.Н. Бондарев. – Севастополь: Изд-во СевНТУ, 2002. – 615 с.

5. Алексєєва Т. В. Технічна діагностика гідравлічних приводів / В.Д. Бабанська, Т.М. Башта та ін. – М.: Машинобудування, 1989. – 263 с.

6. Андрощук О.С. Методологічні аспекти побудови інтелектуальних систем підтримки прийняття рішень в особливих ситуаціях / О.С. Андрощук, В.В. Огурцов, О.І. Демідова // Восточно-Европейский журнал передовых технологий. – Х.: Технологический цент, 2009. – 5/2 (41). – С. 18-21.

7. Ашеров А. Т. Эргономика информационных технологий: учеб. издание / А.Т. Ашеров, С.А. Капленко, В.В. Чубук. – Х.: ХГЭУ, 2000. – 224 с.

8. Байлов В. В. Эксплуатация и сервис радиоэлектронных систем: учеб. Пособие / В.В. Байлов, В.С. Плаксиенко. – Таганрог: Изд-во ТРТУ, 2002. – 90 с.

9. Барзилович Е.Ю. Модели технического обслуживания сложных систем: учеб. пособие / Барзилович Е.Ю. – М.: Высш. школа, 1982. – 231 с.

10. Бартіш М.Я. Дослідження операцій. Лінійні моделі: підручник / М.Я. Бартіш, І.М. Дудзяни. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. – Ч.1., 168с.

11. Бейхельт Ф. Надежность и техническое обслуживание. Математический подход / Ф. Бейхельт, П. Франкен. Пер. с нем. – М.: Радио и связь, 1988. – 392 с.

12. Беневоленский С.Б. Алгоритм идентификации процессов деградации физических свойств технических объектов / С.Б Беневоленский, А.А. Лисов // Измерительная техника. – М: Стандартинформ, 2005. – № 2. – С. 16-18.

13. Бет Э.В. Метод семантических таблиц / Э.В. Бет; перев. под ред. А.В. Идельсона и Г.Е. Минца // Математическая теория логического вывода. – М.: Наука, 1967. – С. 191-199.

14. Бланк И. А. Инвестиционный менеджмент / И.А. Бланк. – К.: Ника – Центр, 2006. – 448 с.

15. Борисюк А. О. Теоретичні основи автоматизації процесів вироблення рішень в системах управління Повітряних Сил: навч. посіб. для слухачів, курс. та студ. вищ. навч. закл. 2-ге вид. переробл. та доп. / А.О. Борисюк, М.А.Павленко, О.І.Тимочко; Мін-во освіти та науки України, ХУПС. – Х.: ХУПС, 2011. – 184 с.

16. Брюшинкин В. Н. Логика и процедуры поиска вывода / В.Н. Брюшинки; отв. за ред. Карпенко А.С // Логические исследования; Ин-т философии РАН. – М.-СПб: ЦГИ, 2010. – Вып. 16. – С. 85-106.

17. Василевський О. М. Нормування показників надійності технічних засобів: навчальний посібник / О. М. Василевський, В. О. Поджаренко. – Вінниця: ВНТУ, 2010. – 129 с.

18. Веклич В.Ф. Діагностування технічного стану тролейбусів./ В.Ф. Веклич – М.: Транспорт, 1990. – 295 с.

19. Вентцель Е.С. Теория вероятностей и ее инженерные приложения / Е.С. Вентцель, Л.А. Овчаров. – М.: Высш. шк., 2000. – 480 с.

20. Военный энциклопедический словарь / [ред. А.Э. Сердюков]. – М: Воениздат, 2007. – 831 с.

21. Волков И.К. Исследование операций: Учебное пособие для вузов. 2-е изд. / И.К. Волков , Е.А. Загоруйко; под ред. В.С. Зарубина, А.П. Крищенко. – М.: МГТУ им. Н.Э. Баумана, 2002. – 436 с.

22. Воронин М.Я. Технико-экономическая эффективность сложных оптико-радиоэлектронных систем СВЧ: монография / М.Я. Воронин, И.Н. Карманов, М.Г. Карманова, И.В. Лесных, М.Ф. Носков, А.В. Синельников ; под общ. ред. М.Я. Воронина. – Новосибирск : СГГА, 2012. – 156 с.

23. Герасимов Б.М. Системы поддержки принятия решений: проектирование, применение, оценка эффективности / Б.М. Герасимов, М.М. Дивизинюк, И.Ю. Субач. – Севастополь: Издательский центр, 2004. – 318 с.

24. Герасимов Б.М. Человеко-машинные системы принятия решений с элементами искусственного интеллекта / Б.М. Герасимов, В.А. Тарасов, И.А. Токарев. – К.: Наукова думка, 1993. – 184 с.

25. Гладков Л.А. Генетические алгоритмы: учебн. пос. / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик. – М.: Физматлит, 2006. – 320 с.

26. Головина Е.Ю. Интеллектуальные методы для создания информационных систем: учебн. пос. / Е.Ю. Головина. – М.: Издательский дом МЭИ, 2011. – 102 с.

27. Горбонос Ф.В. Економіка підприємств: підруч. / Ф.В. Горбонос, Г.В. Черевно, Н.Ф. Павленчик, А.О. Павленчик. – К.: Знання, 2010. – 463 с.

28. Горфинкель В.Я. Экономика предприятия / В.Я. Горфинкель, В.А. Швандар. – М.: ЮНИТИ-ДАНА, 2007. – 70 с.

29. Грицунов О.В. Інформаційні системи та технології. Навч. посібн. / О.В. Грицунов – Х.: ХНАМГ, 2010. – 222 с.

30. Данилевич С.Б. Влияние погрешности измерения на достоверность результатов выборочного измерительного контроля / С.Б. Данилевич, В.В. Княжевский // Измерительная техника. – М: Стандартинформ, 2004. – № 12. – С. 8-11.

31. Данилевич С.Б. Специфика измерений и допускового измерительного контроля / С.Б. Данилевич // Измерительная техника. – М: Стандартинформ, 2003. – № 8. – С. 16-18.

32. Данилов А.А. Метрологическое обеспечение измерительных систем: учеб. пособие / А.А. Данилов. – Пенза: Профессионал, 2008. – 63 с.

33. Демидов Б.А. Системная методология планирования развития, предпроектных исследований и внешнего проектирования вооружения и военной техники: монография / Б.А. Демидов, М.И. Луханин, А.Ф. Величко, М.В. Науменко. – К.: Стилос, 2011. – 464 с.

34. Демидов Б.А. Системно-концептуальные основы деятельности в военно-технической области: в 3 кн., кн. 2. О ганизационно-методические основы деятельности в военно-технической области / Б.А. Демидов. – К.: Технол. парк, 2006. – 1152 с.

35. Демидович Б. П. Краткий курс высшей математики: учеб. пособие для ВУЗов / Б.П. Демидович, В.А. Кудрявцев. – М.: Астрель, 2001. – 655 с.

36. Державна цільова науково-технічна програма створення державної інтегрованої інформаційної системи забезпечення управління рухомими об'єктами (зв'язок, навігація, спостереження): Постанова Кабінету Міністрів України від 17 вересня 2008 р. № 834 // Офіційний вісник України. – 2008. – №29. – С.1145.

37. Джарратано Д. Экспертные системы: принципы разработки и программирование / Д. Джарратано, Г. Райли.; пер. с англ. – М.: Вильямс, 2006. – 1152 с.

38. Евграфов В. Г. Особенности эргономического проектирования и экспертизы тренажерно-обучающих систем / В.Г. Евграфов. – Питер: СПб., 2007. – 224 с.

39. Заболотній С.В. Застосування розкладу в просторі з порідним елементом для вирішення задач ймовірнісної діагностики / С.В. Заболотній //

Восточно-Европейский журнал передовых технологий. – Харьков: Технологический цент, 2014. – Вып. 4/4 (70). – С. 28-35.

40. Закон України “Про інфорацію” № 2658-XII: станом на 09 квітня 2015 р./ Верховна Рада України. – Офіц. вид. – Київ: Парлам. Ви-во, 2015. – № 26. – С. 219.

41. Зайцев Д. В. Многопозиционные радиолокационные системы. Методы и алгоритмы обработки информации в условиях помех / Д.В. Зайцев. – М: Радиотехника, 2007. – 114 с.

42. Зайцев Д. В. Багатопозиційні радіолокаційні системи / Д.В. Зайцев – М.: Радіотехніка, 2007. – 96 с.

43. Золотов С. И. Интеллектуальные информационные системы: учебн. пособ. / С.И. Золотов. – Воронеж: Научная книга, 2007. – 140 с.

44. Искусственный интеллект. Справочник / Под ред. Д.А. Поспелова. – М.: Радио и связь, 1990. – 348 с.

45. Інформаційні технології. Взаємозв'язок відкритих систем. Базова еталонна модель. Частина 1. Базова модель (ISO/IEC 7498-1:1994, IDT):ДСТУ ISO/IEC 7498-1:2004 – [Чинний від 2006–01–01]. – Київ: Держспоживстандарт України, 2007. – 67 с. – (Національний стандарт України).

46. В.Г. Олифер, Н.А. Олифер. Компьютерные сети: Принципы, технологии, протоколы.

47. Cisco Systems, Inc. Internetworking Technology Handbook, 4-th Edition. /Indianapolis: CiscoPress, September 2003 ISBN: 1587051192

48. Терентьев А.М. Информационная безопасность в крупных локальных сетях. – «Концепции», N1(9), 2002, с. 25-30.

49. Терентьев А.М. Построение и развитие системы сетевого мониторинга. / Развитие и использование средств сетевого мониторинга и аудита. Вып. 1. Сборник статей под ред. А.М.Терентьева – М.: ЦЭМИ РАН, 2004, с. 5-23.

50. Сторожук Д. О. Увеличение безопасности работы в локальной сети при использовании систем мониторинга / Гусева А. И., Сторожук Д. О. // Безопасность информационных технологий. – 2007, № 1. – с. 46-50.

51. Сторожук Д. О. Оптимизация по времени багатопоточной модели опроса компьютерной сети / Гусева А. И., Сторожук Д. О // Информационные технологии. – 2007, № 8. – с. 30-33.

52. Сторожук Д. О. Оптимизация по времени багатопоточной модели опроса сети / Гусева А. И., Сторожук Д. // XIY международный научно - технический семинар современные технологии в задачах управления, автоматики и обработки информации: сб. научных трудов – Алушта, 2005. – с. 49.

53. Сторожук Д. О. Оптимальное количество потоков в сети при мониторинге. сети / Сторожук Д. О. // XY международный научно-технический семинар современные технологии в задачах управления, автоматики и обработки информации: сб. научных трудов – Алушта, 2006. – с. 52.

54. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

55. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

56. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

57. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.21.0006.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Іванов А.А.			Дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Коваленко А.С.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи міжмашинних комунікацій з використанням стандарту 10Base-T1.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи міжмашинних комунікацій з використанням стандарту 10Base-T1;

– 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 забезпеченням.

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 Embarcadero RAD Studio Delphi 10.3.2.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 124 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Коваленко А.С.

Дослідження та програмна реалізація

системи міжмашинних комунікацій з використанням стандарту 10Base-T1

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2021 року

Основна програма

Файл Control_10Base_T1.dpr основної програми

```
program Control_10Base_T1;

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
//
```

```
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
    FreeMem(NameNT);  //Звільняємо пам' ять
  end else begin  //Код для 9 х-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
  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
//

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
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' ); //Визначаємо ім'
я під яким він буде видний у мережі
  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
    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

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    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
    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
    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 комп' ютера
        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 комп' ютера
        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;
    dwTimeEcxcds: 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 TCP_IP.pas- монітор TCP/IP з'єднань

```
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

```
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.
```

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.
```