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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти

на тему

“Дослідження та програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-2
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Тимчук М.М.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2023 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 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

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

Тимчуку Мирославу Мирославовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи збору, обробки та відображення інформації об'єкту моніторингу у мережі*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 33-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи збору, обробки та відображення інформації об'єкту моніторингу у мережі*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої 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. 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.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 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	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Тимчук М.М. Дослідження та програмна реалізація системи збору, обробки та відображення інформації об'єкту моніторингу у мережі. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи збору, обробки та відображення інформації об'єкту моніторингу у мережі.

Метою розробки є дослідження та програмна реалізація системи збору, обробки та відображення інформації об'єкту моніторингу у мережі.

Об'єктом дослідження є процес збору, обробки та відображення інформації об'єкту моніторингу у мережі.

Предметом дослідження є методи збору, обробки та відображення інформації об'єкту моніторингу у мережі.

Методи дослідження базуються на методах теорії телекомунікацій, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи збору, обробки та відображення інформації об'єкту моніторингу у мережі.

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 10/11.

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерні науки, моніторинг у мережі

ABSTRACT

Tymchuk M.M. Research and software implementation of the system for collecting, processing and displaying information of the monitoring object in the network. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the system of collecting, processing and displaying information of the monitoring object in the network.

The purpose of the development is research and software implementation of a system for collecting, processing and displaying information of the monitoring object in the network.

The object of the study is the process of collecting, processing and displaying information of the monitoring object in the network.

The subject of the study is methods of collecting, processing and displaying information of the monitoring object in the network.

Research methods are based on telecommunications theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system for collecting, processing and displaying information of the monitoring object in the network.

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

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Builder C++ environment.

Keywords: computer science, network monitoring

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи..... 6
1.2 Область застосування..... 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 21
2.3 Розгорнута постановка завдання 23
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 25
3.1 Опис функціонування системи 25
3.2 Розробка структурної схеми..... 41
3.3 Розробка функціональної схеми 43
3.4 Розробка діаграми процесів..... 53
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 55
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 55
4.2 Захист розробленого програмного забезпечення..... 73
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 78
6 НАУКОВА НОВИЗНА 80

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ..... 81
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 81
7.2 Розрахунок трудомісткості розробки програмної продукції..... 83
7.3 Визначення чисельності виконавців і планового фонду зарплати..... 85
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника..... 90
7.5 Визначення собівартості розробки та ціни програмної продукції..... 94
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 97
7.7 Визначення експлуатаційних витрат..... 97
7.8 Визначення економічної ефективності програмної продукції..... 99
7.9 Висновок..... 101
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ..... 102
8.1 Вступ..... 102
8.2 Аналіз умов праці на робочому місці програміста..... 103
8.3 Розробка заходів з умов поліпшення охорони праці..... 106
8.4 Дослідження інформаційного навантаження на програміста..... 107
8.5 Висновки до розділу..... 110
9 ОСНОВНІ ВИСНОВКИ..... 112
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 114

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ	–	автоматизована система управління
АЦП	–	аналогово-цифровий перетворювач
КПК	–	кишеньковий персональний комп’ютер
ПЗ	–	програмне забезпечення
ПЗО	–	пристрій зв'язку з об'єктом
ПК	–	персональний комп’ютер
СУБД	–	системи управління базами даних
ТП	–	технологічний процес
HMI	–	людино-машиний інтерфейс
SCADA	–	диспетчерське управління й збір даних
WAP	–	Wireless Application Protocol

ВСТУП

Актуальність теми. Чимала кількість промислових виробництв має розподілений характер. Під розподіленістю розуміється територіальний поділ функціональних виробничих ділянок-вузлів, що реалізують технологічні функції, і їхня територіальна далекість від центрального вузла – ядра системи, у якому приймається те або інше рішення про порядок реалізації технологічного циклу. З обліком істотної географічної розподіленості вузлів технологічного виробництва, їхнього великого числа, складності реалізованих ними функцій, завдання приймаче-передачі, обробки первинних даних і вироблення відповідного керуючого рішення може бути досить складним. Тому дистанційний моніторинг і управління віддаленими об'єктами – досить важливе й актуальне завдання, рішення якого складається із двох аспектів:

– принципового рішення про метод, спосіб збору первинних даних з об'єктів контролю й управління, і передачі їх у центральний вузол системи (диспетчерський пункт);

– технічного рішення про систему дистанційного моніторингу й управління, що реалізує вище позначені принципи.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

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

– Огляд існуючих систем збору, обробки та відображення інформації об’єкту моніторингу у мережі.

– Дослідження системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

– Програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

Об'єктом дослідження є процес збору, обробки та відображення інформації об'єкту моніторингу у мережі.

Предметом дослідження є методи збору, обробки та відображення інформації об'єкту моніторингу у мережі.

Методи дослідження базуються на методах теорії телекомунікацій, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод збору, обробки та відображення інформації об'єкту моніторингу у мережі.

– Розроблено вітчизняний продукт збору, обробки та відображення інформації об'єкту моніторингу у мережі, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі збору, обробки та відображення інформації об'єкту моніторингу у мережі.

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 LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи збору, обробки та відображення інформації об'єкту моніторингу у мережі, 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

Існують різні способи телемеханічного контролю й управління віддаленими об'єктами [1-3]. Відомі способи включають виконання стандартної послідовності операцій, що полягає в тому, що при використанні технічних засобів:

– збирають дані про стан віддалених об'єктів;

– здійснюють перетворення даних у необхідну (найбільш зручну для передачі) форму;

– передають дані по каналах зв'язки на пункти більше високої ієрархії;

– здійснюють прикінцеве управління віддаленими об'єктами по командах з диспетчерського пункту;

– прийом даних, їхню обробку й візуалізацію організують при використанні спеціального програмного забезпечення (у тому числі SCADA-технологій).

1.2 Область застосування

Як програмно-апаратне рішення для диспетчерського пункту в цей час типовим є застосування багатосерверної архітектури із установленим на ній відповідним програмним забезпеченням на базі SCADA-технологій. З одного боку, це дозволяє візуалізувати дані про параметри об'єктів у режимі реального часу й, у підсумку, підвищити ефективність взаємодії диспетчера й сервера в складі людино-машинної системи контролю й управління, з іншого боку, вимагає істотних програмно-апаратних витрат, властивих для таких рішень, які, у підсумку:

– характеризуються складністю й надмірністю програмно-апаратної реалізації, у зв'язку із класичним поділом функцій SCADA і WebSCADA між декількома комп'ютерними станціями – вузлами архітектури;

– обмежують набір установлюваного встаткування, висувають додаткові вимоги до нього, ускладнюють процес настроювання й налагодження систем телемеханіки;

– обмежують можливості доступу віддалених кінцевих користувачів, пов'язаних з додатковим ліцензуванням точок доступу, що є принциповою обмежувальною рисою систем телемеханіки подібного типу.

Телемеханічні системи контролю й управління розподіленими об'єктами існують і застосовуються [4-6].

Типова телемеханічна система має у своєму составі:

– масив віддалених терміналів, що реалізують функції збору даних з об'єктів контролю й управління, прикінцеве управління цими об'єктами по вступниках командам, а також приймаче-передачу даних по існуючих каналах зв'язку на пункти більше високої ієрархії. Термінали можуть являти собою вимірювальні й керуючі приймаче-передающі пристрої;

– диспетчерський пункт, що здійснює прийом, обробку й подання даних від масиву віддалених терміналів, а також реалізує функції управління в автоматичному або автоматизованому режимах. Диспетчерський пункт може являти собою виділену одиночну комп'ютерну станцію (сервер), або комп'ютерну мережу;

– комунікаційна система, що включає у свій состав канал зв'язку. Комунікаційна система може бути орієнтована на застосування безлічі провідних стандартних інтерфейсів (RS-485, Industrial Ethernet, оптоволокно) з використанням відповідних промислових протоколів (MODBUS, PROFIBUS, CAN, TCP/IP), а також може бути орієнтована на застосування бездротових каналів зв'язку: виділений радіоканал FM, GSM GPRS з використанням відповідних протоколів. З метою забезпечення надійності й відказостійкості

комунікаційна система може використовувати різні канали зв'язку. Призначення комунікаційної системи – приймаче-передача даних від масиву віддалених терміналів до диспетчерського пункту.

Особливий інтерес представляють телемеханічні системи, у яких як канал зв'язку використовується GSM GPRS. Відмітними рисами цього каналу зв'язку є широка зона покриття в межах населених пунктів і доріг, доступність, відносна дешевина використання.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі, 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

SCADA-системи PROMOTIC

PROMOTIC це комплекс інструментів для розробки додатків для моніторингу, управління й візуалізації технологічних процесів у всіляких галузях промисловості.

PROMOTIC:

– Призначений для ОС Windows 8/7/Vista/XP/XPmb/ 2003-8 Server і вище.

– Дозволяє ефективно створювати додатки в самих різних галузях промисловості.

– Призначений для всіх розроблювачів і проектантів.

– Дозволяє створення додатків у точності відповідно до вимог.

– Надає приємний користувальницький інтерфейс для створення додатків.

– Починаючи з версії 8 систему PROMOTIC можна використовувати й у режимі freeware – PmFree – Free development environment and a runtime licence of the PROMOTIC system.

У систему PROMOTIC убудовані всі необхідні компоненти для створення простих і складних систем візуалізації й управління:

– Редактор додатків з ієрархічним деревом об'єктів.

– Широкий вибір об'єктів PROMOTIC.

– Мова Microsoft Basic (VBScript) для запису алгоритмів.

– Редактор зображень.

– Велика палітра технологічних зображень створених у векторній SVG графіці.

– Графічні об'єкти – елементарні й комплексні універсально конфігуруємі елементи.

– Автоматичне перетворення зображень у формат HTML і XML.

– Система трендів (тобто збереження даних з тимчасовою міткою).

– Система тривог і операторських подій (івентів).

– Web технології Інтернет/Інтранет.

– SQL і ODBC інтерфейси для баз даних.

– Убудовані інтерфейси: XML, OPC, Active, DDE.

– Комунікаційні драйвери для доступу до ПЛК.

– Управління користувачами, дозволами, реєстрація.

– Захист запущених додатків.

– Язикові варіанти PROMOTIC.

– INFO – інформаційна й діагностична система.

– Електронна й друкована документація.

Редактор додатків – це основний інструмент створення додатку в системі PROMOTIC, служить для визначення деревоподібної структури PROMOTIC об'єктів, їхнього налагодження, визначення алгоритмів, і т.д.

Убудована мова VBScript із синтаксисом Visual Basic служить для запису користувальницьких алгоритмів у подійному програмуванні, для доступу до методів і властивостей об'єктів системи PROMOTIC або інших програм. Розроблювачеві це дає необмежену можливість для розвитку проекту.

Для налагодження додатка є інформаційна й діагностична INFO система. INFO система дає можливість переглядати важливу інформацію при ході роботи додатка. Пропонується можливість дистанційного настроювання запущених додатків:

– у мережах Інтернет і Інтранет через PROMOTIC Web;

– або програми TeamViewer, PCAnyWhere, LapLink, CarbonCopy, і т.д.

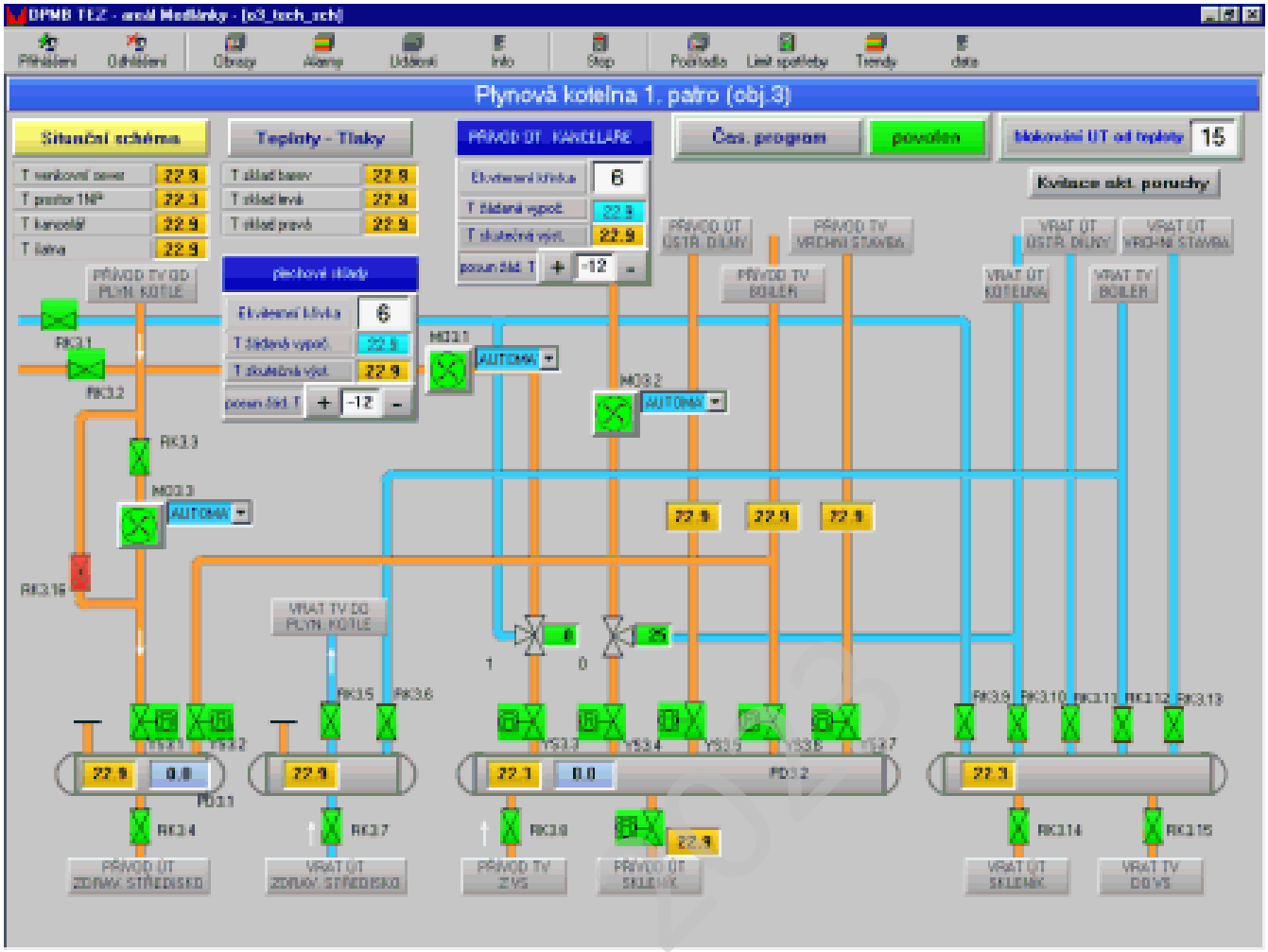

Рисунок 2.1 – Інтерфейс користувача PROMOTIC

Редактор зображень служить для створення груп графічних елементів з палітри визначених елементів. У цих елементах можна задавати їхні статичні властивості, або ж ці властивості з'єднати зв'язком даних і пожвавити таким способом створювану візуалізацію. З'єднати зв'язком даних можна всі важливі властивості графічних елементів: Колір, Позицію, Розмір, Значениеня й багато чого іншого.

Підтримка створення власних створених графічних елементів і імпорт/експорт в XML (у текстові дані) приносить економію роботи розроблювача й швидке створення додатка.

Користувач може створювати зображення, які є стандартними вікнами Windows, але може визначати вікна зі стабільною позицією, стабільним розміром,

вікна додатка на весь екран, без віконних рамок або визначити максимальну кількість одночасно відкритих вікон.

Мнемосхеми доступні у вигляді автоматично генеруємих динамічних HTML сторінок, включаючи зворотні втручання й управління. Динамічні сторінки створюються автоматично в інтегрованому середовищі розробки й зберігаються у файлах даних додатка. Таким чином, розроблювач може зробити з локального додатка мережний за кілька хвилин.

На віддаленому ПК можна розглядати в реальному часі процеси, тренди, тривоги й події.

PROMOTIC Web-сервер забезпечує перенесення даних за допомогою конфігуратора користувачів і дозволів.

Всі передані дані засновані на HTTP або HTTPS протоколах (легко прохідні через firewall).

Система PROMOTIC є системою з повністю відкритою архітектурою. Убудовані стандартні програмні інтерфейси: XML, Active, ODBC, DAO, OLE, OPC, DDE, TCP/IP, WEB дозволяють провести повну інтеграцію системи PROMOTIC з іншими програмними продуктами. Завдяки цій концепції можливо, наприклад, пряме з'єднання системи PROMOTIC із внутрішньозаводською базою даних (MSSQL, MySql, dBase, Access, Oracle, SAP, і т.д.), з'єднання з комунікаційними серверами, або ж з іншими програмними системами.

Складовою частиною системи PROMOTIC є система трендів, що зберігає значення обраних змінних з необхідною дискретністю на диску комп'ютера для подальшого відображення в графічному або табличному виді:

– Обрані змінні зберігаються у форматі DBase, Access, бінарний PROMOTIC, MS SQL Server, MySQL, Oracle, FireBird.

– Перегляд трендів можливий у мережах Інтернет/Інтранет через браузер.

– Можна налаштувати, щоб змінні зберігалися регулярно, за бажанням і зі змінною дискретністю.

– В одному браузері можна переглядати тренди різних додатків, розташованих на різних комп'ютерах.

– Можна налаштовувати конфігурацію й колір окремих процесів.

Тривога – сигналізація стану, що не повинен наступити при нормальних умовах (напр. подолання технологічних границь певної величини, обрив комунікації, і т.д.). PROMOTIC має убудовану систему тривог, що:

– реєструє стан тривоги в певних тривожних групах;

– архівує стан;

– пропонує зручний перегляд актуальних тривог з фільтрацією, квітируванням, звуковим супроводом;

– дозволяє переглядати історію тривог.

Івент (операторська подія) це повідомлення про виконання певних дій (напр. початку, кінця або успішного виконання певного процесу, втручання персоналу, і т.д.). Це повідомлення записується у файл для наступного перегляду. У такий спосіб створюється "хроніка" про хід даної технології.

Додаток, спрямований на спостереження й управління технологічними процесами одержує або зберігає дані з/на зовнішніх джерелах. Джерелом таких значень може бути напр.: база даних, ПЛК, файл на диску, смарт-карта для входу/виходу в ПК, інший сервер у локальному або віддаленому ПК, і т.д.

PROMOTIC розташовує:

– більшою кількістю власних, високопараметризованих комунікаційних драйверів для ПЛК:

Siemens Simatic, SAIA, Mitsubishi, Allen-Bradley DF1, DF1Koyo, Omron, Telemecanique, Modicon, ADAM, комунікаційний протокол Modbus, M-BUS, IEC 60870-5, і інші.

У розташуванні перебувають драйвери для модемного зв'язку, передачі в радіомережах і мережах GSM.

– Через убудовані стандартні інтерфейси OPC, DDE, Active можна підключитися до комунікаційних серверів інших фірм, напр. Simatic, Landis&Gyr,

Honeywell, Bernecker&Reiner, Moeller, ABB, Allen-Bradley, Sauter, Unitronics, GE-Fanuc, Festo, Yokogawa, Lonworks, Lenze, і інші.

– Для створення децентралізованих додатків у мережах Інтернет і Інтранет є в наявності комунікаційні інтерфейси із протоколами TCP/IP, HTTP, XML, DCOM.

Можливості комунікації з навколишніми програмними системами практично необмежені.

PROMOTIC пропонує для безпеки додатків:

– Систему реєстрації операторів: Ім'я, Пароль, дозволи, локальні й мережні допуски.

– Можливість блокування всіх критичних клавіш в Windows.

– Контроль ходу виконання додатка: програмний WatchDog.

– Захист від переповнення диска: циклічна структура трендів, тривог і івентів.

– Установка пароля для проекту в середовищі розробки – захист користувальницького "ноу-хау".

Система PROMOTIC використовує символи Unicode, і завдяки цьому, можливо створювати додатка на будь-якій мові. Основні тексти системи переведені на наступні мови:

– В інтегрованому середовищі розробки можна вибрати англійська, чеська й польська мова.

– У рантайм режимі можна вибрати для системних текстів: чеський, словенський, англійський, польський, німецький, російський, французький, угорський, і т.д.

PROMOTIC підготовлений до швидкого розширення інших перекладів.

PROMOTIC – підходяще рішення для всіх областей промисловості:

– енергетика (теплоелектростанції, гідроелектростанції, підстанції, когенерація...);

– моніторинг руху транспортних засобів;

– металургійні заводи (сталеплавильні печі, випалювальні печі, коксові заводи, лінії прокатки, агломерації...);

– екологія (емісійний моніторинг, управління віддільників, станція очищення стічних вод, очищення від пилу...);

– телеметричні й керуючі системи (водопровідні станції, газові заводи, шахти, теплові мережі...);

– вимір і регуляція споживання енергії (електроенергія, тепло, газ, вода...);

– організація харчових технологій (пивоварні заводи, молочні комбінати, цукрові заводи, маслозаводи, млини...);

– теплове господарство (теплообмінні установки, котельні установки...);

– хімічна промисловість;

– обробна промисловість;

– навчання й дослідження;

– і т.д.

SCADA-системи IGSS

IGSS (англ. Interactive Graphical SCADA System) – Інтерактивна Графічна Система SCADA – система диспетчерського управління й збору даних (SCADA – Supervisory Control And Data Acquisition).

SCADA-система IGSS заснована на технології "клієнт-сервер" і може масштабуватися від одного автоматизованого робочого місця "Stand Alone" (від 50 об'єктів ~150 тегів) до 50 АРМ (до 400 000 об'єктів, більше мільйона тегів) з резервуванням серверів.

Підтримка різноманітних стандартів і інтерфейсів, включаючи DDE, ODBC, OPC, SQL, VBA/Automation, OLE і Active, дозволяє обмінюватися даними з додатками інших розроблювачів.

За рахунок широкого списку драйверів для програмувальних логічних контролерів, що входять у комплект стандартної поставки, забезпечується простий і надійний обмін із ПЛК провідних виробників: Siemens, ABB, Schneider, Omron...

Інтерфейси користувача/диспетчера (Supervice) і розроблювача (Definition) переведені на багато мов, у тому числі й на українську. При проектуванні є можливість створення багатомовного інтерфейсу.

Інтернет-портал (WEB-Portal) дозволяє віддалено відслідковувати виробничий процес за допомогою наладонників, стандартних браузерів і СМС-повідомлень на стільниковому телефоні.

GSS є "клієнт-серверною" системою, що дозволяє використовувати її як на локальних автоматизованих робочих місцях, так і для мережних рішень, з великою кількістю операторських станцій, серверів баз даних, резервуємих серверів і т.д.

П'ять основних переваг IGSS становлять наступні критерії:

– Масштабованість. Можливість поступового розширення автоматизованої системи на базі IGSS, починаючи від окремого АРМ оператора (Stand Alone) до загальнозаводської мережі з декількома серверами без внесення глобальних змін у ППО існуючих станцій, а лише підключаючи їх до ланцюга, що розширюється, як окремі ланки.

– Обмін даними з використанням широко розповсюджених технологій дозволяє максимально збільшувати ефективність взаємодії системи IGSS із системами й додатками сторонніх виробників.

– Широкий спектр підтримуваних драйверів різних програмувальних логічних контролерів і протоколів обміну даними дозволяє без додаткових витрат підключати до створюваної заводської мережі всі нові й нові технологічні лінії.

– Для обміну даними з іншим прикладним і/або інструментальним програмним забезпеченням, IGSS підтримує повністю відкриту архітектуру. Використання існуючих стандартів дозволяє істотно заощаджувати час розробки й інтеграції, а також уникнути серйозних помилок у подальшій роботі системи в цілому.

IGSS є об’єктно-орієнтованою системою. Тобто при розробці ППО робота виробляється не з окремими тегами (наприклад, один тег – один канал

дискретного уведення ПЛК), а з об'єктами, які можуть з’єднати у собі до 10 тегів. Такий підхід значно полегшує розуміння технологічного процесу (звичайно, один об'єкт – це один фізичний компонент у процесі, наприклад, двигун, вентилятор або клапан, що може мати трохи аналогових і кілька дискретних входів/виходів).

Рано або пізно на будь-якому об'єкті автоматизації доводиться зіштовхуватися із проблемою розширення виробництва, а отже, і з необхідністю модернізації існуючих АСУ ТП.

Масштабованість і зворотна сумісність – два основних критерії на захисті інвестицій.

Масштабованість і захист інвестицій – ці концепції пов'язані з IGSS споконвічно. Система IGSS побудована на основі архітектури клієнт-сервер. IGSS масштабується від одного додатка для однієї операторської станції з охватом декількох точок уведення/виводу до комплексної системи, що включає 50 операторських станцій і до 250 тис. точок уведення/виводу с резервуванням серверів. Використовуючи IGSS, система диспетчеризації може початися із простою, і продовжувати рости з розширенням вимог виробництва. Витрати на розробку не будуть загублені – розширення системи IGSS просто додає будівельних блоків до вже закладеного фундаменту.

Програмне забезпечення IGSS назад сумісне. Незалежно від того, як багато нових характеристик додано до IGSS системними розроблювачами, нова версія завжди буде назад сумісною з більше ранніми версіями.

Правила ліцензування IGSS також відповідають стратегії покрокового розширення. Це дозволяє здобувати додаткові ліцензії поступово, відповідно до росту бізнесу.

WEB-орієнтована SCADA система IntegraXor

SCADA-система IntegraXor це якісний WEB сервер, розроблений на базі W3C-сумісної технології. Для роботи не потрібно додаткового WEB сервера, що робить його установку й експлуатацію більше простою і економічно привабливою.

IntegraXor успішно працює під управлінням операційних систем Windows XP, Vista, Windows 7.

Широко відомо, що стандартні (не WEB-орієнтовані) HMI/SCADA системи мають проблеми з пересиланням власних величезних файлів із мнемосхемами через мережу Internet. Мнемосхеми в IntegraXor використовують стандарт SVG (Scalable Vector Graphics – масштабована векторна графіка) для відображення графічних елементів. Стандарт SVG базується на XML, що забезпечує мінімізацію обсягів переданої інформації.

Фірмою ECAVA розроблений і запатентований унікальний метод відображення анімації файлів SVG у реальному часі з використанням стандартного браузера (у цей час підтримуються Internet Explorer 8 і Firefox 3.5. Ця технологія інтегрована в IntegraXor для відображення Ваших даних у реальному часі.

Для роботи необхідно встановити SCADA систему IntegraXor для розробки проекту і його запуску, а також Adobe SVG Viewer для відображення мнемосхем у браузері.

Для створення проекту необхідно скачати середовище розробки й диспетчеризації Ecava IntegraXor, графічний редактор Inkscape SAGE, а також установити Adobe SVG Viewer.

Демонстраційна версія IntegraXor працює 48 годин у режимі Диспетчеризації, що дозволить повністю відчути всі достоїнства продукту.

Після установки пакета Ecava IntegraXor, буде потрібно запустити IntegraXor Editor (редактор проектів) і вибрати пункт "New Project" з меню "File".

Інтерфейс користувача системи IntegraXor Editor ретельно проробленний і є інтуїтивно зрозумілим кожному, хто в тому або іншому ступені зіштовхувався з розробкою HMI/SCADA.

Після створення нового (або відкриття існуючого проекту), екран системи буде виглядати (з урахуванням особливостей відкритого проекту) у такий спосіб:

Опис SCADA системи Winlog Pro

Простий, гнучкий і економічний продукт Winlog Pro – пакет програм SCADA/HMI для використання як на промислових, так і на побутових об'єктах автоматизації.

Інтегроване середовище розробки надає різні інструменти (Gate Builder, Template Builder, Code Builder) для швидкого й інтуїтивного створення проектів.

Велика бібліотека протоколів, включаючи протокол OPC клієнта забезпечують комунікацію з великим спектром електронних пристроїв, таких як PLCs, контролери, електроприводи, модулі віддаленого вводу-виводу.

Підтримка стандартного формату файлів історії (DBF, CSV) і ODBC (SQL) гарантує сумісність із більшістю додатків Windows (Excel, Access, і т.д.).

SCADA система Winlog Pro дозволяє створити розподілену архітектуру Клієнт-сервер на основі протоколу TCP/IP у мережах Інтранет/Інтернет і з легкістю створювати веб-додатки, доступні для стандартних браузерів.

Для зв'язку з віддаленими пристроями можуть використовуватися як стаціонарні, так і стільникові (GSM\GPRS\3G) телефонні лінії й SMS.

Засоби розробки містять у собі Symbol Factory 2.0, саму популярну бібліотеку для автоматизації, що містить більше 4000 графічних виробничих і індустріальних об'єктів, таких як насоси, клапани, двигуни, резервуари, PLCs, трубопроводи, і т.д.

Інтегрований редактор дозволяє змінювати розміри й змінювати колір, схему й орієнтацію об'єктів (бітмап (bmp) або metafile). На додаток до бібліотеки в Winlog Pro представлені такі інтерактивні графічні об'єкти як: круглі (Dial, GearDial) і лінійні (Vslider, Hslider) потенціометри, що вказують, кутові (120 градусів, 180 градусів, 270 градусів) і лінійні (Vmeter, Hmeter) індикатори, термометри, вимикачі (RockerSwitch, ToggleSwitch) і багато чого іншого.

Будь-який проект може бути запущений на виконання, використовуючи одну із двох самих доступних і економічних ліцензій run-time (до 128 змінних і більше 128).

У випадку багато язикових проектів (на двох мовах або більше) перемикання між мовами здійснюється натисканням однієї кнопки.

Звіти й історичні дані можуть бути зареєстровані в стандартному форматі (DBF, CSV), до якого можуть легко одержати доступ сторонні додатки Windows (Excel, Access і т.д.).

SCADA система Winlog Pro побудована із застосуванням Клієнт-сервер архітектури й може працювати по протоколі TCP/IP через мережу Інтранет/Інтернет. Можливо створити мульти-мастерні структури, у яких кожний термінал (комп'ютер з Winlog Pro) може спілкуватися з іншими й надавати можливість як читання так і запису тегів.

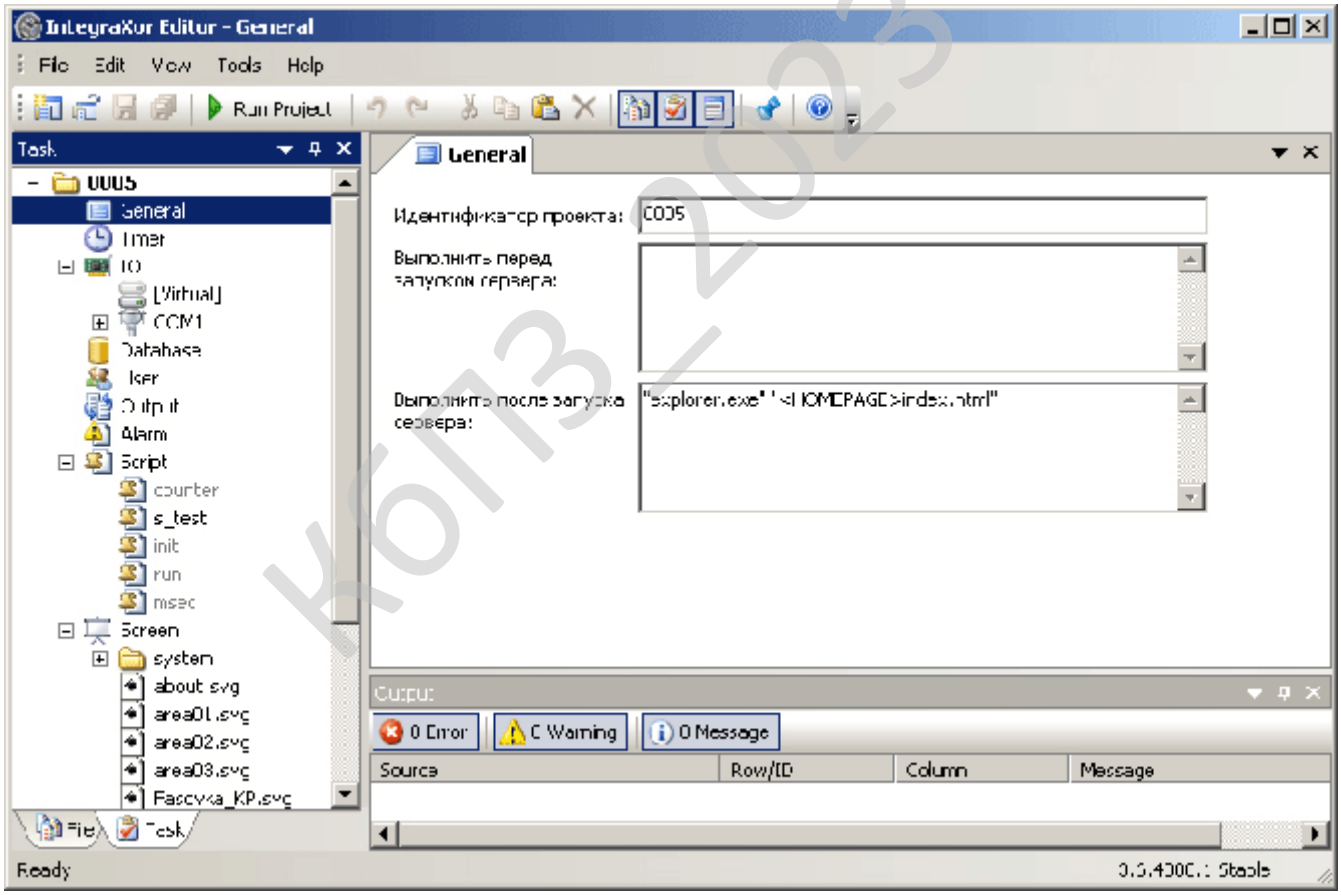

Рисунок 2.2 – Інтерфейс користувача Winlog Pro

Для зв'язку з віддаленими пристроями можуть використовуватися стаціонарні й стільникові телефонні лінії. Існує можливість відправляти SMS віддаленим операторам, у такий спосіб центр збору інформації й спостереження може надавати віддалену підтримку мережі об'єктів, розташованих по всій країні.

У лівій частині вікна розташований навігатор, у якому відображаються завдання й состав проекту. У правій частині, на окремих вкладках відкриваються файли проекту.

Варто звернути увагу, що система IntegraXor використовує для створення й редагування мнемосхем програми сторонніх виробників (у цей час – безкоштовний графічний редактор Inkscape + SAGE). Для того, щоб відкривати файли мнемосхем (*.SVG) безпосередньо з навігатора IntegraXor Editor, потрібно налаштувати прив'язку програми редактори до файлів по розширенню. Для цього необхідно виконати наступні кроки.

1. Скачати й установити графічний редактор Graphic Editor Inscape +SAGE.

2. Відкрити "Провідник" або будь-який файловий менеджер, знайти файл із розширенням *.SVG і клацнути по ньому правою кнопкою миші. У меню, що відкрився, вибрати пункт "Відкрити за допомогою" і далі – "Вибрати програму":

3. У діалоговому вікні, що з'явилося, необхідно вибрати редактор Inkscape і встановити прапор "Використовувати неї для всіх файлів такого типу".

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об’єму в

порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими

функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції застосунку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

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 збору, обробки та відображення інформації об’єкту моніторингу у мережі.

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

Через широке поширення телемеханічних систем з використанням SCADA (WebSCADA), слід зазначити наступні особливості застосування цих програмно-апаратних систем:

– програмно-апаратне рішення припускає, як мінімум, двосерверну конфігурацію. При цьому один із серверів (SCADA), будучи пов'язаним із пристроями (контролерами) нижнього рівня по виділених каналах зв'язку (в основному, електричні сигнали промислових стандартів: струм, напруга, послідовні імпульсні/цифрові інтерфейси), виконує функції приймаче-передачі, обробки, візуалізації, зберігання даних. Інший сервер (WebSCADA), реалізований на базі виділеного Internet-сервера зі статичною IP-адресою й пов'язаний з першим згаданим сервером по каналах Intranet/Internet, реалізує «експозицію» даних в Web, тобто приймаче-передачу їхнім віддаленим користувачам. Дана двосерверна конфігурація не є оптимальною з погляду компактності, функціональності й вартості;

– тип устаткування, установлюваного на нижньому рівні системи, строго застережений у відповідному переліку SCADA, і пов'язаний з набором установлюваних на нижньому рівні пристроїв і їхніх драйверів. Це, у свою чергу, уводить обмеження по номенклатурі застосовуваної апаратної частини;

– кількість точок уведення/виводу даних на нижньому рівні системи, так само як і кількість точок доступу до даних системи, строго регламентується придбаними програмними засобами й ліцензіями. Розширення системи у всіх відносинах може бути пов'язане з істотною переконфігурацією, придбанням додаткового програмного забезпечення, драйверів, ліцензій, що істотно стримує подальший розвиток телемеханічної системи.

У силу відзначених особливостей і обставин застосування SCADA (WebSCADA) у телемеханічних системах характеризується громіздкістю, надмірністю архітектури й програмно-апаратного забезпечення, що є принциповим недоліком.

Відомий спосіб телемеханічного контролю й управління, відповідно до якого збір первинних даних і прикінцеве управління об'єктом здійснюють за допомогою мікропроцесорного блоку управління, зв'язаного із засобом приймаче-передачі сигналів по каналі стільникового зв'язка, приймаче-передачу даних здійснюють із мобільних телефонів осіб, що приймають рішення [1].

Недоліки зазначеного способу наступні:

– застосування мікропроцесорного блоку з обмеженою функціональністю не дозволяє приймати з об’єкту масив його сигналів, що характеризують;

– застосування загального поняття «канал стільникового зв'язка» не дозволяє оцінити якість і ефективність приймаче-передачі даних;

– застосування як засіб приймаче-передачі даних на стороні пункту управління мобільного телефону істотно обмежує функціональність системи й позбавляє спосіб і його систему, що реалізує, необхідної обробки даних, що не задовольняє вимогам більшості промислових систем.

Відомий спосіб телемеханічного контролю й управління, застосовуваний для контролю й регулювання режиму роботи трубопроводу, у якому збір інформації реалізують за допомогою вузла засобів вимірів і мікропроцесорного контролера, передачу первинних даних від контрольованого пункту до пункту управління здійснюють за допомогою застосування радіомодемів, як пристрій збору й обробки даних у пункті управління використовують комп'ютерну станцію [2].

Спосіб можна охарактеризувати наступними недоліками:

– у способі не простежується уніфікація системи збору даних і управління, застосування первинної апаратури орієнтовано на збір даних про режим роботи трубопроводу;

– використання виділеного радіоканалу не є оптимальним рішенням в умовах існування покриття територій стандартним стільниковим зв'язком, наприклад GSM 900/1800;

– застосування як пристрій збору даних і управління відособленої локальної комп'ютерної станції не дозволяє організувати перегляд і обробку даних на інших комп'ютерних станціях, що мають вихід у локальні або глобальні комп'ютерні мережі.

Відомий спосіб телемеханічного контролю й управління, застосовуваний для моніторингу й управління станом станцій катодного захисту Тверца-900 «Тверца монітор», у якому збір первинних даних про стан об'єктів роблять із використанням мікропроцесорної вимірювальної системи, а передачу даних у диспетчерський пункт – за допомогою приймаче-передачі. SMS по каналі стільникового радіозв'язку GSM 900/1800 [3].

Згаданому способу властиві наступні недоліки:

– застосування як інформаційні посилки SMS по радіоканалу GSM 900/1800 обмежує функціональні можливості системи;

– застосування як пристрій збору й обробки даних відособленої локальної комп'ютерної станції не дозволяє масштабувати систему збору, обробки даних і дистанційного управління об'єктами на інші комп'ютерні станції, включені в локальні або глобальні комп'ютерні мережі.

Відома система телемеханічного контролю й управління, що містить спеціалізований блок сполучення для збору інформації про стан віддаленого об’єкту, а також прийому даних від блоку контролю й управління, передачі даних про об'єкт по каналі стільникового зв'язка й стабілізації контрольованого параметра [4].

Недоліки згаданої системи телемеханічного контролю й управління:

– мала розмірність кількості входів блоку сполучення не дозволяє вводити масив сигналів, що характеризують об'єкт;

– обмежена функціональність блоку сполучення, що дозволяє реалізувати тільки режим стабілізації одного з параметрів об’єкту;

– відсутність функцій обробки й перетворення даних у блоці сполучення й блоці контролю й управління;

– застосування загального поняття «канал стільникового зв'язка» не дозволяє оцінити якість і ефективність приймаче-передачі даних;

– обмежена функціональність блоку контролю й управління, при цьому не ясні функції по візуалізації даних, їх приймаче-передачі на інші вузли прийняття рішень, використанню для цього комп'ютерних мереж як локальних, так і глобальних.

У підсумку, наведені відомості по системі не дозволяють оцінити ефективність застосування (якщо це має місце) SCADA (WebSCADA) у цьому випадку.

Відома система телемеханічного контролю й управління, що містить контролери для збору даних і прикінцевого управління об'єктом, радіоканал для приймаче-передачі даних з пункту управління, модеми, установлені на прикінцевих сторонах радіотракту й підключені в пункті управління до комп'ютерних станцій [5].

Однак описаній системі властиві наступні недоліки:

– збір даних з контрольованих пунктів виробляється тільки при використанні аналогових і дискретних входів і не припускає роботу зі стандартними цифровими інтерфейсами (RS-232, RS-485, CAN);

– складна архітектура реалізації вказує на застосування необґрунтовано складних способів приймаче-передачі даних;

– використання виділеного радіоканалу не є оптимальним рішенням в умовах існування покриття територій стандартним стільниковим зв'язком, наприклад GSM 900/1800;

– використання на обох сторонах приймаче-передачі даних спеціалізованих модемів ускладнює архітектуру системи й підвищує витрати як при проектуванні, так і при експлуатації;

– обмеженість функцій приймаче-передачі, візуалізації, обробки й зберігання даних у пункті управління, не розкриті ці функції при використанні комп'ютерних локальних і глобальних мереж.

У підсумку, наведені відомості по системі не дозволяють оцінити ефективність застосування (якщо це має місце) SCADA (WebSCADA) у цьому випадку.

Відома система телемеханічного контролю й управління, реалізована програмно-технічним комплексом «СКАТ-4» і застосовувана для моніторингу й управління розподіленими об'єктами, що містить вимірювальний комплекс як засіб збору первинних даних, виділені фізичні лінії, які можуть бути використані як засоби приймаче-передачі даних з нижнього на верхній рівень, виділені/ телефонні лінії, щокомутируються, виділені радіоканали, GSM GPRS, локальні комп'ютерні станції для збору й обробки даних, підключені за допомогою локальних мереж до комутаційного, комунікаційному серверам і серверу бази даних [6].

Недоліки згаданої системи:

– використання єдиного протоколу інформаційного обміну для різних комунікаційних середовищ приймаче-передачі даних з нижнього рівня на верхній є надлишковим стосовно різних типів інтерфейсів;

– архітектура системи містить велику кількість серверів, що реалізують різні функції, пов'язаних з розподіленої SCADA. Загалом, система характеризується надмірністю, громіздкістю, низькими функціональністю й надійністю;

– відсутність масштабування автоматизованих робочих місць диспетчерів у глобальні комп'ютерні мережі, тим більше, без придбання додаткових ліцензій.

Найбільш близьким технічним рішенням для способу, прийнятим за прототип, є спосіб [7], використовуваний при телеметрії, телесигналізації й телеуправлінні в області об'єктів електричних мереж, і полягає в тому, що:

– установлюють на віддалених об'єктах контролери телемеханіки;

– використовують спеціальні комунікатори, що формують масив аналогових, дискретних і цифрових первинних даних;

– використовують комунікатори масиву вихідних команд для управління станом об'єктів;

– використовують термінали GSM, які реєструють в GSM мережі й, використовуючи сервіс GPRS, підключають до Internet через шлюз оператора стільникового зв'язка;

– приймаче-передачу даних здійснюють із використанням протоколу на основі стека TCP/IP;

– використовують Internet-сервер з виділеною статичною IP-адресою;

– використовують сервери збору й обробки даних;

– програмне забезпечення сервера формують із сервера каналів, конфігуратору сервера каналів, сервера опитування, конфігуратору сервера опитування, модуля телемеханіки й управління, конфігуратору контролерів, бази даних, сервера звітів, WEB сервера;

– автоматизоване робоче місце оператора оснащують спеціальним програмним продуктом, що забезпечує обмін з Internet-сервером, візуалізацію даних та інші сервісні функції;

– забезпечують доступ до перегляду даних про віддалені об'єкти.

Прототипу властиві наступні недоліки:

– велика кількість програмних продуктів, що становлять програмно-апаратний комплекс, не є оптимальним;

– Internet-сервер і інші програмні засоби вимагають істотних апаратних ресурсів (не виключене їхнє розміщення на декількох серверних станціях), що

ускладнює архітектуру системи, зменшує швидкодію, знижує надійність, збільшує її вартість;

– для відображення інформації про стан віддалених об'єктів необхідно використовувати спеціальну програму автоматизованого робочого місця, що може бути встановлена на обмеженій кількості комп'ютерних станцій, за умови придбання відповідних ліцензій.

Загальні ознаки опису [7] дозволяють судити про те, що в цьому випадку як ядро системи використовується програмно-апаратний комплекс із використанням компонентно розподіленої SCADA (WebSCADA). Тому всі перераховані недоліки способу можна віднести до принципового стосовно SCADA (WebSCADA).

Як прототип системи обраний телемеханічна система контролю й управління, застосовувана в промисловості [8].

Ієрархія описуваної системи трьохрівнева. На першому рівні системи телемеханіки – верстатах, насосних установках, пунктах обліку теплової й електричної енергії – установлюються вимірювально-обчислювальні комплекси «МЕГА». Вони роблять збір даних з об’єкту контролю й управління, передають дані по каналі GSM GPRS на контролер зв'язку, установлений на другому ієрархічному рівні системи й підключений до сервера збору даних і управління. Інформація про об'єкти доступна диспетчерам, фахівцям і керівникам на автоматизованих робочих місцях (АРМ), що представляють третій рівень ієрархії. Масив комп'ютерних станцій АРМ підключається до сервера збору даних і управління з використанням провідних каналів зв'язку. Загальний потік даних від об’єкту до диспетчера (фахівцеві, керівникові), таким чином, проходить наступні програмно-апаратні вузли: дані від об’єкту (перший рівень) – комплекс «МЕГА» (перший рівень) – канал зв'язку GSM GPRS – контролер зв'язку (другий рівень) – сервер збору даних і управління (другий рівень) – масив АРМ (третій рівень).

Програмне забезпечення сервера збору даних і управління, а також автоматизованих робочих місць (АРМ) являє собою компонентно-розподілену SCADA (WebSCADA).

Однак принципи побудови й функціонування розглянутої телемеханічної системи не вільні від недоліків:

– недостатня в описі системи інформація про тип використовуваного програмного протоколу пакетної передачі даних не дозволяє однозначно судити про ефективність застосованих принципів маршрутизації, адресації й захисти даних, уніфікації цього рішення стосовно існуючих стандартних мережних протоколів;

– використання SCADA припускає необхідність добірки, установки й настроювання на комп'ютерах АРМ відповідної операційної системи, драйверів і іншого спеціалізованого програмного забезпечення;

– інформація, що надходить із об'єктів контролю й управління, доступна тільки для обмеженої групи користувачів, на комп'ютерах яких зроблені всі відповідні настроювання.

Застосовувані SCADA (WebSCADA) можуть бути власною розробкою, або розробкою відомих фірм. При цьому потрібно мати на увазі, що процес розробки SCADA досить складний і вимагає істотних часових і фінансових витрат, що не може не відбитися на вартості готового програмного продукту. Тобто локальність розглянутої системи спричиняє необхідність придбання відповідних ліцензій.

Таким чином, при використанні SCADA (WebSCADA) формується система, розрахована на роботу з обмеженою кількістю пристроїв нижнього й верхнього рівнів. Використання SCADA (WebSCADA) має на увазі розміщення її на декількох програмно-апаратних серверах, вимагає істотних обчислювальних ресурсів, компонується з безлічі програмних модулів, що, у підсумку, спричиняється її громіздкість, схильність помилкам і збоям, відносно повільне виконання функцій.

Також подібні системи характеризуються обмеженням кількості точок доступу до даних (станцій оператора, диспетчерських пунктів), що не завжди є зручним – диспетчер може переглянути всю необхідну інформацію про об'єкти контролю й управління тільки зі свого АРМ.

Завданням справжнього винаходу є створення компактної, функціональної й надійної телемеханічної системи з необмеженою кількістю точок регламентованого доступу до інформації про стан віддалених об'єктів шляхом виключення SCADA (WebSCADA) з реалізації проекту за рахунок:

– введення в архітектуру системи єдиного сервера телемеханіки, реалізованого на базі виділеного Internet-сервера зі статичним IP-адресою, що одночасно забезпечує приймаче-передачу й обробку даних як на віддалені об'єкти контролю й управління, так і на автоматизовані робочі місця віддалених користувачів (функціонально сполученого в одному блоці);

– організації необмеженого регламентованого віддаленого доступу до інформації про стан об'єктів контролю й управління, при використанні стандартних Internet-браузерів, з можливістю контролю доступу й захисту інформації в порядку класифікації імен користувачів і застосування системи паролів.

Технічний результат полягає в забезпеченні можливості реалізації ефективного дистанційного моніторингу й управління станом віддалених об'єктів при використанні недорогих каналів зв'язку GSM GPRS, стека протоколів TCP/IP, програмно-апаратного забезпечення мережі Internet, єдиного сервера телемеханіки, виконаного шляхом функціонального сполучення сервера збору даних і управління й виділеного Internet-сервера зі статичним IP-адресою, на одній програмно-апаратній платформі, комплексу серверних програмних засобів, установленого на єдиному сервері телемеханіки й виконуючої функції візуалізації, обробки й зберігання даних, що забезпечує архітектурну й програмно-апаратну компактність ядра телемеханічної системи, підвищує її швидкодія, розширює функціональність, знижує вартість розробки й

експлуатації, підвищує практична зручність при використанні, що, у підсумку, вигідно відрізняє дану систему від аналогів.

Технічний результат досягається застосуванням способу телемеханічного контролю й управління, що полягає в тому, що:

– на кожний віддалений об'єкт установлюють контролери телемеханіки, що представляють собою в справжній системі контролери телеуправління;

– за допомогою контролерів телеуправління збирають масиви даних по аналогових, дискретних і цифрових інтерфейсах (RS-232, RS-485, CAN), а також реалізують віддалене управління;

– використовують автоматизовані робочі місця;

– для приймаче-передачі даних між сервером телемеханіки й контролерами телеуправління використовують протокол на основі стека TCP/IP;

– у центрі збору й обробки даних установлюють єдиний сервер телемеханіки, що реалізують на базі виділеного Internet-сервера зі статичним IP-адресою, шляхом сполучення функцій приймаче-передачі даних з нижнього рівня системи на верхній, на одній програмно-апаратній платформі;

– на єдиному сервері телемеханіки встановлюють комплекс серверних програмних засобів для одночасного виконання функцій як по приймально-передачі, так і по візуалізації, обробці й зберіганню даних;

– перегляд даних про віддалені об'єкти й видачу команд управління організують при використанні стандартних Web-браузерів із завданням IP-адреси єдиного сервера телемеханіки;

– доступ до перегляду даних і видачі команд управління на віддалені об'єкти при використанні стандартних Web-браузерів регламентують системою імен користувачів (login) і паролів (password);

– як канали зв'язку єдиного сервера телемеханіки, віддалених контролерів телеуправління й автоматизованих робочих місць використовують лінії Internet.

Описаний спосіб може бути реалізований телемеханічною системою контролю й управління віддаленими об'єктами, що містить властиво віддалені

об'єкти, контролери телеуправління з інтегрованими модемами GSM GPRS, канали периферійного зв'язку, антенно-фідерні пристрої контролерів телеуправління, GSM GPRS-канали приймаче-передачі даних з використанням програмного протоколу на основі стека TCP/IP, антенно-фідерні пристрої мобільного оператора, GSM GPRS-сервер мобільного оператора зв'язку, сервер збору даних і управління, виділений Internet-сервер зі статичним IP-адресою, автоматизовані робочі місця, а також комплекс серверних програмних засобів, Internet-канал зв'язку, що забезпечує обмін даними між GSM GPRS сервером мобільного оператора зв'язку, сервером збору даних і управління й автоматизованих робітників місцями, при цьому:

– сервер збору даних і управління функціонально сполучений з виділеним Internet-сервером зі статичною IP-адресою, з одержанням єдиного сервера телемеханіки;

– комплекс серверних програмних засобів установлений на єдиному сервері телемеханіки й реалізований засобами Web-програмування;

– комплекс серверних програмних засобів складається з підпрограми приймаче-передачі й підпрограми візуалізації, обробки, зберігання даних;

– дана система забезпечує необмежену кількість точок регламентованого доступу до інформації про віддалені об'єкти, при використанні стандартних Web-браузерів із завданням IP-адреси єдиного сервера телемеханіки, застосування системи імен користувачів (login) і паролів (password).

Технічний результат забезпечується тим, що:

– збір даних і управління віддаленими об'єктами здійснюється контролерами телеуправління з використанням периферійних каналів зв'язку. Приймаче-передача даних здійснюється з використанням інтегрованих у контролери телеуправління модемів GSM GPRS і відповідних антенно-фідерних пристроїв. Контролери телеуправління використовують доступні недорогі канали зв'язку GSM GPRS, стек протоколів TCP/IP, що дозволяє застосовувати для

приймаче-передачі даних програмно-апаратні засоби каналів зв'язку мережі Internet;

– GPRS-сервер мобільного оператора зв'язку здійснює приймаче-передачу, адресацію й маршрутизацію пакетів даних у системі протоколу TCP/IP каналів зв'язку Internet, що дозволяє організувати наскрізний обмін даними між віддаленими контролерами телеуправління і єдиним сервером телемеханіки, із забезпеченням високої швидкості й вірогідності передачі даних;

– сполучення на єдиній програмно-апаратній платформі серверних апаратних засобів (єдиного сервера телемеханіки) і комплексу серверних програмних засобів, що складає з підпрограм приймаче-передачі, візуалізації, обробки й зберігання даних і виконаного при використанні Web-програмування, дозволяє сформувати оптимальне по обсязі, функціональності, надійності й швидкодії ядро телемеханічної системи, що принципово вигідно відрізняє розглянуту систему від архітектур з використанням SCADA;

– компонування ядра системи на єдиному сервері телемеханіки дозволяють масштабувати систему (підключати додаткові пристрої нижнього й верхнього рівнів приймаче-передачі даних) телемеханіки за умови мінімальних додаткових програмно-апаратних, матеріальних і трудових витрат;

– дана система забезпечує необмежену кількість точок регламентованого доступу до інформації про віддалені об'єкти, при використанні стандартних Web-браузерів із завданням IP-адреси єдиного сервера телемеханіки, застосування системи імен користувачів (login) і паролів (password).

У підсумку, застосування заявляються способу й телемеханічної системи дозволяє здійснити ефективний дистанційний моніторинг і управління віддаленими об'єктами.

Архітектура телемеханічної системи контролю й управління віддаленими об'єктами, містить:

– властиво віддалені об'єкти;

– контролери телеуправління з інтегрованими модемами GSM GPRS;

– канали периферійного зв'язку;

– антенно-фідерні пристрої контролерів телеуправління;

– GSM GPRS канали приймаче-передачі даних;

– антенно-фідерні пристрої мобільного оператора;

– GSM GPRS сервер мобільного оператора зв'язку;

– Internet-канал зв'язку;

– єдиний сервер телемеханіки із установленим на ньому комплексом серверних програмних засобів, що складають із підпрограми приймаче-передачі даних і підпрограми візуалізації, обробки й зберігання даних;

– канали зв'язку Intranet/Internet;

– автоматизовані робочі місця.

Контролери телеуправління через канал периферійного зв'язку зчитують дані й здійснюють прикінцеве управління віддаленими об'єктами (по аналогових, дискретних входах і цифрових інтерфейсах).

На початку своєї роботи контролери телеуправління при використанні антенно-фідерних пристроїв виявляють мережа стандарту GSM 900/1800, формують запити реєстрації через антенно-фідерні пристрої й GPRS сервер мобільного оператора на єдиний сервер телемеханіки, використовуючи протокол TCP/IP Internet-каналу зв'язку. Єдиний сервер телемеханіки має статична адреса й порт у системі адресації TCP/IP. При реєстрації на єдиному сервері телемеханіки кожний контролер телеуправління одержує динамічну адресу й порт у системі адресації TCP/IP, ця адреса єдиний сервер телемеханіки зберігає як ідентифікатор контролера телеуправління протягом усього сеансу зв'язку. Після реєстрації на єдиному сервері телемеханіки контролер телеуправління готовий до передачі даних про режим і параметри роботи віддаленого об’єкту, а також до прийому команд управління від станцій автоматизованих робочих місць через єдиний сервер телемеханіки.

Для втримання каналу зв'язку, через певні інтервали часу, контролер телеуправління формує тестові посилки KEEP ALIVE на єдиний сервер телемеханіки.

У процесі роботи контролер телеуправління регулярно обновляє дані про режим і параметри роботи віддаленого об’єкту. У контролері телеуправління можуть бути реалізовані спеціалізовані алгоритми обробки даних: аналогова/цифрова фільтрація, функція прийому даних при нестабільній відповіді віддаленого об’єкту через цифрові інтерфейси, а також функції «ковзного середнього» для основних параметрів її роботи. Отримані дані передаються через певні інтервали часу, які формує єдиний сервер телемеханіки, направляючи запити в контролер телеуправління. Крім того, у системі телемеханіки може бути організований позачерговий запит даних, по команді з автоматизованого робочого місця, що має право такого запиту. Режим роботи віддалених об'єктів також може бути змінений по команді з автоматизованого робочого місця, що має право реалізації команд управління. Зв'язок автоматизованих робочих місць із єдиним сервером телемеханіки здійснюється по каналі зв'язку Intranet/Internet.

У випадку виникнення певних подій (відкриття/закриття двері об’єкту, включення/вимикання живлення ~ 220 В, 50 Гц, виникнення помилки віддаленого об’єкту) контролер телеуправління самостійно ініціює й формує інформаційні посилки на єдиний сервер телемеханіки. Робота й приймаче-передача даних від контролера телеуправління у випадку вимикання живлення ~ 220 В, 50 Гц здійснюється при використанні резервного джерела живлення (акумулятора).

Дані, що приходять від масиву «віддалені об'єкти – контролери телеуправління», обробляються в комплексі серверних програмних засобів єдиного сервера телемеханіки. Цей програмний комплекс, що складається з підпрограми приймаче-передачі даних і підпрограми їхньої візуалізації, дозволяє приймати, обробляти, візуалізувати, зберігати дані по кожній обліковій точці віддалених об'єктів.

Єдиний сервер телемеханіки має статична адреса й порт у системі адресації TCP/IP, що дозволяє реалізувати реєстрацію й доступ до підпрограми приймаче-передачі даних, що організує зв'язок з масивом контролерів телеуправління, і організувати доступ до підпрограми візуалізації даних, що забезпечує зв'язок з масивом станцій автоматизованих робочих місць. У результаті диспетчер (фахівець, керівник) може переглянути дані про кожний віддалений об'єкт, задавши IP-адресу в рядку власного браузера станції автоматизованого робочого місця, увівши ім'я (login) і пароль (password). При цьому відкривається екранна форма WEB-інтерфейсу, що надає функції перегляду, обробки даних по кожному віддаленому об'єкті, а також функції управління (залежить від прав користувача).

Таким чином, що заявляються спосіб телемеханічного контролю й управління віддаленими об'єктами й телемеханічною системою припускають проходження даних через три наступні рівні перетворення: об'єкти (перший рівень) – контролер телеуправління (перший рівень) – канал зв'язку GSM GPRS – сервер GPRS мобільного оператора (другий рівень) – єдиний сервер телемеханіки, підпрограма приймаче-передачі даних комплексу серверних програмних засобів (другий рівень) – єдиний сервер телемеханіки, підпрограма візуалізації, обробки й зберігання даних комплексу серверних програмних засобів (третій рівень) – автоматизоване робоче місце (третій рівень).

Єдиний сервер телемеханіки дозволяє сполучити в одному функціональному блоці другий і третій рівні приймаче-передачі даних в архітектурі системи й, таким чином, на одній програмно-апаратній платформі одночасно реалізувати як функції приймаче-передачі даних від контролерів телеуправління віддалених об'єктів, так і функції обробки, передачі цих даних на автоматизовані робочі місця віддалених користувачів. При цьому уведення й вивід даних з єдиного сервера телемеханіки на різні рівні ієрархії телемеханічної системи здійснюється з використанням каналів зв'язку Internet, що також є принциповою відмінністю способу і його телемеханічної системи, що реалізує.

Використання контролерів телеуправління, що перетворять дані з контрольованих об'єктів в IP-пакети, з наступною передачею по GSM GPRS-каналі на єдиний сервер телемеханіки зі статичною IP-адресою, на якій встановлений комплекс серверних програмних засобів з підпрограмою приймаче-передачі й підпрограмою їхньої візуалізації, дозволяє здійснити прямий доступ інформації від віддалених об'єктів контролю й управління на загальнодоступний виділений Internet-сервер. При цьому сполучення програмно-апаратних функцій приймаче-передачі, обробки, візуалізації й зберігання даних на єдиному виділеному Internet-сервері телемеханіки забезпечує компактність і функціональність архітектури. У той же час використання SCADA має на увазі розміщення її на декількох програмно-апаратних серверах, вимагає істотних обчислювальних ресурсів, компонується з безлічі програмних модулів, що, у підсумку, спричиняється її громіздкість, схильність помилкам і збоям, відносно повільне виконання функцій, при відносно високій вартості відповідних компонентів і ліцензій.

Обмін даними по GSM GPRS-каналах відповідає вимогам, пропонованим до автоматизованих систем, таким як вірогідність, надійність, якість передачі даних. Це дозволяє робити моніторинг розподілених (віддалених) об'єктів і вчасно формувати керуючі сигнали.

Комплекс серверних програмних засобів, що складає з підпрограми приймаче-передачі й підпрограми візуалізації, обробки й зберігання даних, виконує функції громіздк і дорогої SCADA і дозволяє виключити її із проекту. При цьому він реалізований засобами Web-програмування й володіє набагато більше оптимальними показниками за комплексним критерієм «функціональність – програмно-апаратні витрати – продуктивність».

Віддалений доступ до даних і формування керуючих команд на віддалені об'єкти контролю й управління в телемеханічній системі реалізований на основі класифікації імен користувачів (login) і застосування системи паролів (password).

Техніко-економічна перевага пропонованого підходу полягає в тому, що реалізація системи як готового програмно-апаратного рішення для збільшення кількості автоматизованих робочих місць віддалених користувачів і розширення кількості логінів і паролів доступу до системи не вимагає додаткового придбання ліцензій. Таким чином, доступ до даних стає необмеженим.

Справжній спосіб і його система, що реалізує, дозволяють одержати ефективний засіб моніторингу й управління віддаленими об'єктами, що працюють у режимі реального часу, при скороченні часу й витрат на розробку телемеханічної системи, оптимізації її роботи.

У підсумку, використання пропонованого способу й телемеханічної системи для його здійснення дозволяють створювати надійні системи з необмеженою кількістю точок регламентованого доступу до інформації.

3.2 Розробка структурної схеми

Розроблювальне в даному магістерському проекті програмне забезпечення відноситься до категорії продуктів WebSCADA.

Під цим терміном розуміється відображення інформації на екрані монітора в зрозумілій для людини формі стосовно до систем диспетчерського контролю й збору даних на основі web-технологій. Це дозволяє здійснювати функції диспетчеризації через стандартний браузер (наприклад Firefox, Opera).

Підключення клієнтів до розробленого програмного забезпечення через інтернет дозволяє їм взаємодіяти із прикладним завданням автоматизації як із простою web-сторінкою:

– Облік відпускання (споживання) теплової енергії й витрати енергоносіїв (води, пари, природного газу, кисню, стисненого повітря й ін.)

– Облік відпускання (споживання) електроенергії.

– Ваговий облік (устаткування ВНК).

– Телеметричний контроль режимів роботи електричних, теплових і газових мереж.

– Візуалізація оперативних і архівних параметрів

– Диспетчеризація

– Розрахунок параметрів (похідні від вимірів, наприклад, питомих витрат енергоносіїв).

– Формування звітних документів.

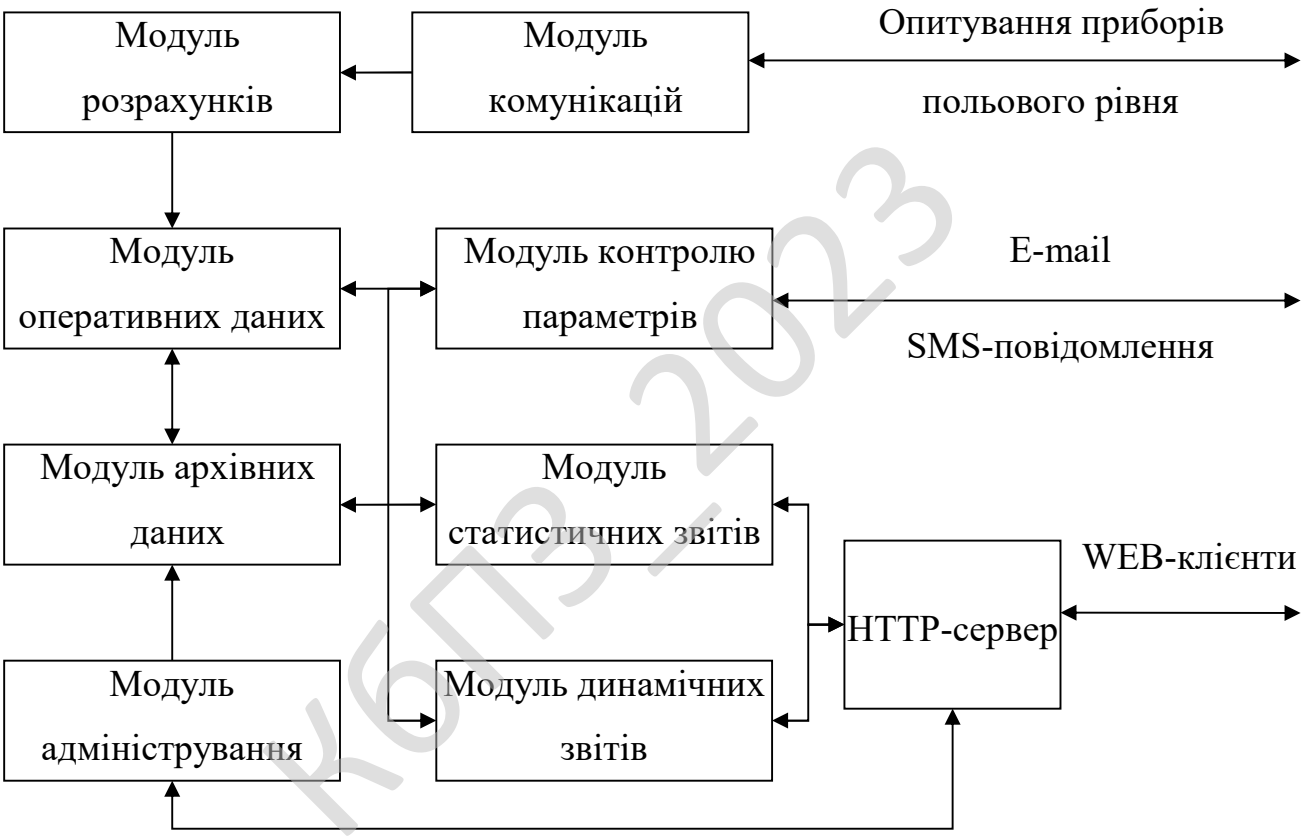

Рисунок 3.1 – Структурна схема системи

Функціональні можливості:

– Побудова територіально розподілених систем диспетчеризації – ЦТП, ІТП, ваговий облік, об'єкти інженерної інфраструктури, АСУ будинків.

– Об'єднання інформації від різних локальних АСУ й забезпечення до неї доступу з будь-якого ПК у мережі Інтернет.

– WEB-Інтерфейс у всіх режимах роботи.

– Робота з УСПД, лічильниками й витратомірами.

– Інтеграція до складу автоматизованих систем підприємства.

– СУБД (Oracle, PostgreSQL).

Структурний состав комплексу:

– Модуль комунікацій (поточні показання приладів обліку, архівні, автовідновлення даних).

– Модуль оперативних даних (оперативні значення, глибина регулюється).

– Модуль архівних даних (довгострокове зберігання, аналіз по періодах).

– Модуль обчислень (обробка даних, розрахунок додаткових параметрів).

– Модуль контролю параметрів (запис аварійних подій, розсилання повідомлень).

– Модуль адміністрування (настроювання комплексу, управління).

– Модуль статичних звітів (html, xls).

– Модуль динамічних звітів (графіки, тренди).

Підтримувані пристрої:

– комплекс енергопідрахунку ТЕКОН– 10-17 (TF1.1, TF1.2);

– теплорозраховувач МАГІКА (MODBUS RTU);

– теплорозраховувач ВКТ-7;

– теплорозраховувач Multical-MC601;

– ваговий комплекс ТЕНЗО-М (на базі вагового термінала ТВ011).

3.3 Розробка функціональної схеми

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

Система WebScada призначена для рішення завдань диспетчеризації й моніторингу об'єктів, автоматизованих за допомогою програмно-технічних засобів. Система WebScada виконана в клієнт-серверній архітектурі. Серверна

частина системи може бути встановлена на глобальному Інтернет-сервері або локальному сервері. Ніяких спеціальних програм на комп'ютері клієнта (користувача) установлювати не потрібно, досить мати лише Інтернет-браузер і вихід у мережу Інтернет або локальна мережа, залежно від побудови загальної системи моніторингу.

Короткий опис основних функціональних блоків системи.

Розробка мнемосхем у середовищі розробленого програмного забезпечення системи збору, обробки та відображення інформації об'єкту моніторингу у мережі

Розробка мнемосхем може бути здійснена безпосередньо в редакторі WebScada. Дана можливість опирається на бібліотеку примітивів, які мають ряд параметрів. Параметри «за замовчуванням» можуть мінятися користувачем. Середовище дозволяє створювати мнемосхеми, які будуть виконувати закладений у них функціонал. Параметри (характеристики, значення) одного примітива можуть впливати на параметри іншого, у результаті утвориться повнофункціональна схема із залежними елементами. Дана можливість обумовлена динамічними властивостями примітивів. У результаті створюється деякий сценарій, відповідно до якого функціонує вся система, або її блоки. У загальному бібліотеку примітивів можна розділити на кілька груп: фігури (або об'єкти) використовувані в технологічному процесі (баки, казани, труби та ін.), датчики, логіка, арифметика й т.п. У цілому є каталог примітивів, розбитий у групи. Є можливість створення нової групи примітивів, редагування наявної й т.п.

Розробка примітивів

Можливість створення власних примітивів, з додаванням їх у базу наявних. Дана можливість дозволить створювати примітиви із графічних зображень, у тому числі анімованих; з найпростіших фігур: прямокутників, ліній і т.п. Такий примітив буде наділений властивостями (параметрами) і може бути використаний як повноцінний примітив, що має «розум». При створенні він

міститься в зазначену групу примітивів. Таким чином, здійснюється нарощування бібліотеки. Бібліотека примітивів виконується як окремий функціональний модуль, що може бути оновлений, не залежно від системи в цілому. Можливе створення спеціалізованих бібліотек примітивів для різних технологічних процесів.

Спостереження в реальному часі мнемосхем об'єктів моніторингу з динамічно, що змінюються параметрами

Дана можливість розроблена з використанням динамічного завантаження (підзавантаження) даних. Достоїнством даної технології є те, що немає необхідності багаторазово обновляти дані на сторінці. На завантаженої один раз мнемосхемі відбувається відновлення лише необхідної інформації: параметрів (значень), завантаження потрібних зображень і т.п. При цьому вся інша сторінка вже не завантажується. Система дозволяє міняти не тільки значення у вигляді цифр, рядків, але й змінювати форму, колір, положення примітива. Оскільки дана система використовується в мережі Інтернет, то дозволяє раціонально використовувати трафік, виконуючи лише необхідні відновлення на сторінці.

Управління встаткуванням

Оскільки система являє собою комплекс пристроїв і програмного забезпечення, тобто можливість не тільки спостерігати за параметрами мнемосхеми, але й вносити нові значення. Це дає можливість управляти Технологічним Процесом. Перш ніж значення будуть внесені (змінені), вони пройдуть перевірку на валідність (правильність) і адекватність. Це забезпечується заданими обмеженнями на значення.

Графічний моніторинг об’єкту моніторингу

Для наочного подання процесу, що відбувається, використовується графічний моніторинг. На графіку більш наочно можна бачити зміни й статистику роботи різних пристроїв, блоків, реєстрації датчиків і т.п.

Засоби оповіщення

Для найбільш швидкого реагування на різні ситуації, що відбуваються в технологічному процесі, використовуються засоби оповіщення: SMS-повідомлень і відправлення електронної пошти. Це може бути корисним при відмовах системи, помилках і інших позаштатних ситуаціях. Також можна одержувати статистику від системи поштою за певний період її роботи, якщо така необхідність є.

Планувальник

У систему закладена можливість задавати параметри системи на певний час (години : хвилини день/місяць/рік). Може задаватися інтервал, на якому необхідно мати певні параметри. У результаті система працює згідно закладеного в неї плану (програми).

Система логістики

Система веде логі (логістику) по всьому технологічному процесі. У користувачів є можливість у будь-який момент відобразити ці параметри на екрані (або вивести їх в xls-файл). По певних характеристиках можна побудувати графік для наочного моніторингу процесу. Система логістики фіксує всі зміни, які вносить кожний користувач, тобто всі дії фіксуються.

Права доступу до системи

Передбачений 3-х рівневий (SuperAdmin, Admin і User) багатокористувальницький доступ користувачів у систему. Права на групи Admin і User визначаються при реєстрації користувачів. Кожна група має певні можливості й доступ до розділів системи.

Відображення структури Об’єкту моніторингу, максимально наближеної до реальності

Створюється мнемосхема, яка копіює технологічний процес, з усіма можливими елементами які в ньому присутні. Іншими словами це просто модель процесу. Число мнемосхем у проекті не обмежено.

Тобто існує можливість продублювати деякі Об’єкти моніторингу в одному проекті, або ж попросту розбити один процес на більше дрібні підпроцеси.

Число елементів мнемосхем не обмежено. Тут варто керуватися тільки наочністю процесу. Тобто якщо у вас багато елементів на схемі то попросту можна не зрозуміти де що – плутанина. Як елемент мнемосхеми може бути використаний будь-який елемент управління ActiveX, з можливістю динамізації будь-якої його властивості, а також одного зі стандартних властивостей (положення, розміру, відрисовки, миготіння й т.п.).

В основі створення мнемосхем лежить використання стандартних елементів, типових. Бібліотеки типових елементів нараховують біля тридцяти стандартних елементів, включаючи об'ємні елементи з убудованим індикатором заповнення, елементи для створення користувальницьких діалогів, елементи, що відтворюють повний комплект приладів щитового контролю й управління. Є убудований редактор для створення мультфільмів (з регульованою прозорістю зображення) з різними законами трансформації вихідних графічних файлів (покадровий показ, прокручування в будь-якому напрямку, зміна різкості або розміру й т.п. із завданням часу й кількості кадрів). Об'ємні трубопроводи довільної конфігурації створюються в кілька клацань миші.

Існує можливість використання в мнемосхемі таких елементів як gif– анімація, різних відео файлів.

Підтримуються всі стандартні графічні формати: bmp, gif, jpg, avi. Всі імпортовані зображення й відеокліпи можуть бути відображені в режимі з прозорістю, що налаштовується, і одночасної динамізації будь-яких інших властивостей.

Так само використовується векторна графіка, що дає широке коло операцій над обраним об'єктом.

Аналіз архіву параметрів Об’єкту моніторингу в графічному виді

Для аналізу й подання інформації використовуються тренди. Тренди призначені для перегляду даних у графічному й табличному виді.

Можливе відображення будь-якого числа графіків будь-яких архівуємих і неархівуємих змінних. Вставка змінних у тренд виробляється їхнім перетаскуванням із проекту з автоматичним спадкуванням діапазону й одиниці виміру. Тренд реального часу й історичний об'єднаний в одному вікні. Кожне перо може мати свій діапазон осі значень. Масштаб часу й значень може бути змінений у процесі перегляду.

Існує можливість зберегти тренд у вигляді графічного файлу з розширенням jpg у папку "Тренди" даного об’єкту.

Створення звітів і їх друк в призначений час

Для створення рапортів (кількість рапортів не обмежене) використовується Microsoft Excel (надалі буде також можливо використовувати убудований редактор рапортів). Excel відкривається безпосередньо у вікні редагування пакета. Змінні в таблицю рапорту перетаскуються з дерева проекту. Вставлені змінні можна використовувати у формулах графіках і діаграмах стандартним образом. Друк або збереження рапортів виробляється за розкладом або подією в зручний час.

Рапорти використовуються для перегляду й друку даних у певний момент часу. Об'єкт може мати кілька Рапортів. Існує можливість за своїм розсудом зконфігурувати таблицю й помістити в неї необхідні змінні відповідні осередки. У час роботи проекту туди будуть міститися поточні дані, тобто значення на даний момент часу.

У режимі виконання ви зможете переглядати й друкувати Рапорти. Ці дії так само зручно автоматично здійснювати в заданий час за допомогою розкладу об’єкту.

Реакція на виникнення аварійної ситуації, створення журналу аварій

У пакеті підтримується необмежена кількість повідомлень. Повідомлення підрозділяються на чотири види:

– системні (про недостачу місця на диску, відсутності зв'язку й т.п. – формуються самим пакетом);

– функціональні (формуються функціональними блоками за результатами обробки вхідних даних відповідно до логіки, закладеної розроблювачами блоку);

– контролю границь (за результатами контролю границь і швидкості зміни змінних);

– подійні (формуються при настанні передбачених проектом подій, що обчислюються по заданій формулі, мають певний у проекті текст).

Повідомлення мають категорію, пріоритет, джерело. Для кожної категорії вказуються дозволені канали виводу. Є п'ять основних каналів виводу: принтер, вікно повідомлень, рядок статусу, журнал повідомлень, архів повідомлень; (їхнє число може розширюватися за рахунок, наприклад, мультимедійних каналів):

– спливаюче вікно повідомлень;

– рядок статусу;

– журнал повідомлень;

– принтер;

– архів.

Повідомлення завжди відноситься до тієї або іншої Категорії. Є ряд визначених категорій. Є можливість створення своїх категорій.

Категорія – атрибут, що дозволяє сортувати й фільтрувати повідомлення залежно від їхнього призначення.

Категорія має ім'я й настроювання:

– колір тексту й тла повідомлень;

– звуковий файл, що програється при виникненні повідомлення;

– пріоритет;

– список каналів виводу (для каналу виводу Журнал задається перелік журналів, у які будуть надходити повідомлення даної категорії);

Основний спосіб перегляду повідомлень, включаючи архівні, це журнал. Журнал повідомлень є документом, що належить об'єкту. У ньому передбачені засоби фільтрації й сортування по кожному з полів і/або джерел повідомлень. Журнал може бути роздрукований, експортований у вигляді XML-файлу або опублікований в Інтранеті/Інтернеті. Журнал є клієнтом повідомлень у стандарті OPC.

Журнал – засіб для перегляду повідомлень, у якому реалізована можливість фільтрувати повідомлення по тим або інших ознаках. Журнал відкривається зі сторінок властивостей об’єкту (а також за розкладом об’єкту або по дії, настроєній для події цього об’єкту). Можна створити кілька форматів Журналів, що мають різний зовнішній вигляд, заголовки, фільтри для повідомлень і т.д. Приналежність Журналу до того або іншого Об’єкту визначається лише тим, що в ньому видні повідомлення тільки від джерел, що належать даному Об'єкту. Журнал не є в точному значенні документом об’єкту, тому що не зберігає повідомлення, а тільки виводить їх на екран.

Журнали об'єктів можуть відкриватися як з режиму розробки, так і з режиму виконання. При відкритті Журналу в режимі розробки переглядати можна тільки архівні повідомлення (кнопка включення фільтра активних подій у цьому випадку недоступна). Для перегляду вмісту Журналу Можливо використовувати додаткові фільтри, за допомогою яких можна внести заборони на вивід повідомлень по тих або інших ознаках. Таким чином, відкриваючи який-небудь Журнал об’єкту, можливо бачити вибірку повідомлень, зроблену виходячи з наступних позицій:

– від елементів тільки цього об’єкту й, можливо, дочірніх;

– відповідно до настроєного для даного типу Журналу фільтрами;

– відповідно до настроювань додаткових фільтрів.

Фільтрація повідомлень по категоріях відбувається зворотним образом: у настроюваннях кожної категорії вказується, у яких Журналах відображати повідомлення даної категорії. За замовчуванням для кожної категорії вказується тільки "Основний журнал", інші типи Журналів варто призначити спеціально.

Зв'язок із зовнішнім програмним забезпеченням

Можливість експорту архіву параметрів. Підтримуються архіви даних, повідомлень і рапортів. Обсяги архівів обмежуються тільки самим користувачем.

Архіви розподілені по об'єктах. Інформація в архів даних направляється з використанням індивідуально обраного для кожної змінної фільтра. Можна задати обмеження тривалості або обсягу зберігання індивідуально для кожного об’єкту й типу архіву. Для сумісності із зовнішніми базами даних забезпечується експорт архівів (у тому числі в режимі off-line) з можливістю об'єднання архівів декількох об'єктів в один загальний зовнішній архів.

Для перегляду Архіву даних його можна експортувати у файл *mdb (після чого за допомогою Microsoft Access його можна переглянути або конвертувати в зручний для Вас формат). Швидко переглянути дані ви можна так само на закладці "Дані" в змінній, яку необхідно переглянути, (перегляд можливий, якщо в змінній встановлений прапор "Архівувати"). Експорт Архівів даних або повідомлень може здійснюватися автоматично, за розкладом Об’єкту, або по дії Події.

Існує можливість у проектах використовувати MS SQL, що у свою чергу дозволяє створювати незалежні програми для аналізу інформації, використовувати вже наявне програмне забезпечення, орієнтоване на обробку даних.

Можливість перегляду даних через Internet. Доступ з будь-якого комп'ютера підприємства до інформації, що надходить від виробничого об’єкту моніторингу, від будь-якої підсистеми стає необхідністю. А різного типу клієнтські додатки можуть надавати відповідному виробничому процесу у величезному обсязі дані в прийнятному для користувача виді.

Тому існує можливість перегляду всіх документів і властиво мнемосхем через Internet.

Рисунок 3.2 – Функціональна схема системи

Зв'язок з нижнім рівнем – Контролером. Зв'язок з нижнім рівнем здійснюється по засобах ОПС сервера, що здійснює буферизацію даних які йдуть від фізичного пристрою, щоб потім розподілити їх між різними клієнтськими додатками або управляє передачею даних на різні фізичні пристрої, по запитах відповідних клієнтських додатків.

ОПС дає можливість у користувальницькі додатки інтегрувати нові частини програмного забезпечення.

Постачальники ПЗ промислового призначення можуть розробляти продукти, сумісні з розповсюдженими стандартами ПЗ.

Таки образом, підвищується продуктивність і якість продукту.

Можливість контролю дій оператора й обмеження прав доступу. Є розвинені засоби багаторівневого обмеження прав доступу, що включають формування переліку посад з настроюванням прав доступу для кожної посади, адміністрування призначення операторів на посаді й у зміни, реєстрацію в зашифрованому журналі всіх дій операторів.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3.

Де після початку роботи ПЗ ми потрапляємо до головного вікна програми з можливістю перегляду налаштування ПЗ, довідкової системи, провести аналіз роботи ПЗ з переглядом журналу подій та звітів роботи ПЗ.

Крім цього через моніторинг роботи пристроїв ми проводимо сигналізацію виключних ситуацій, автоматичне керування, обробку змін параметрів та обробку даних та обчислення.

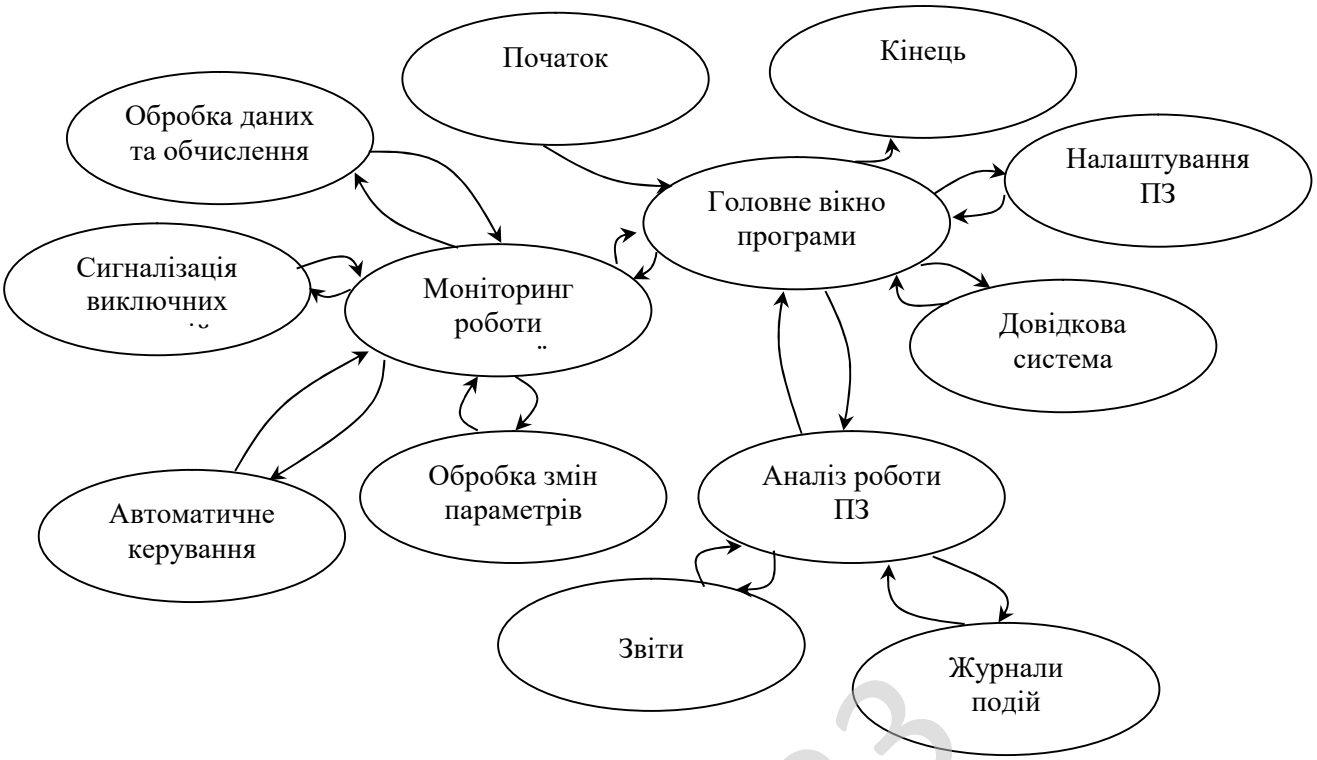

Рисунок 3.3 – Діаграма взаємодії процесів

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

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків:

– Виведення головного вікна ПЗ.

– Сканування нових пристроїв.

– Додавання в базу даних нових пристроїв.

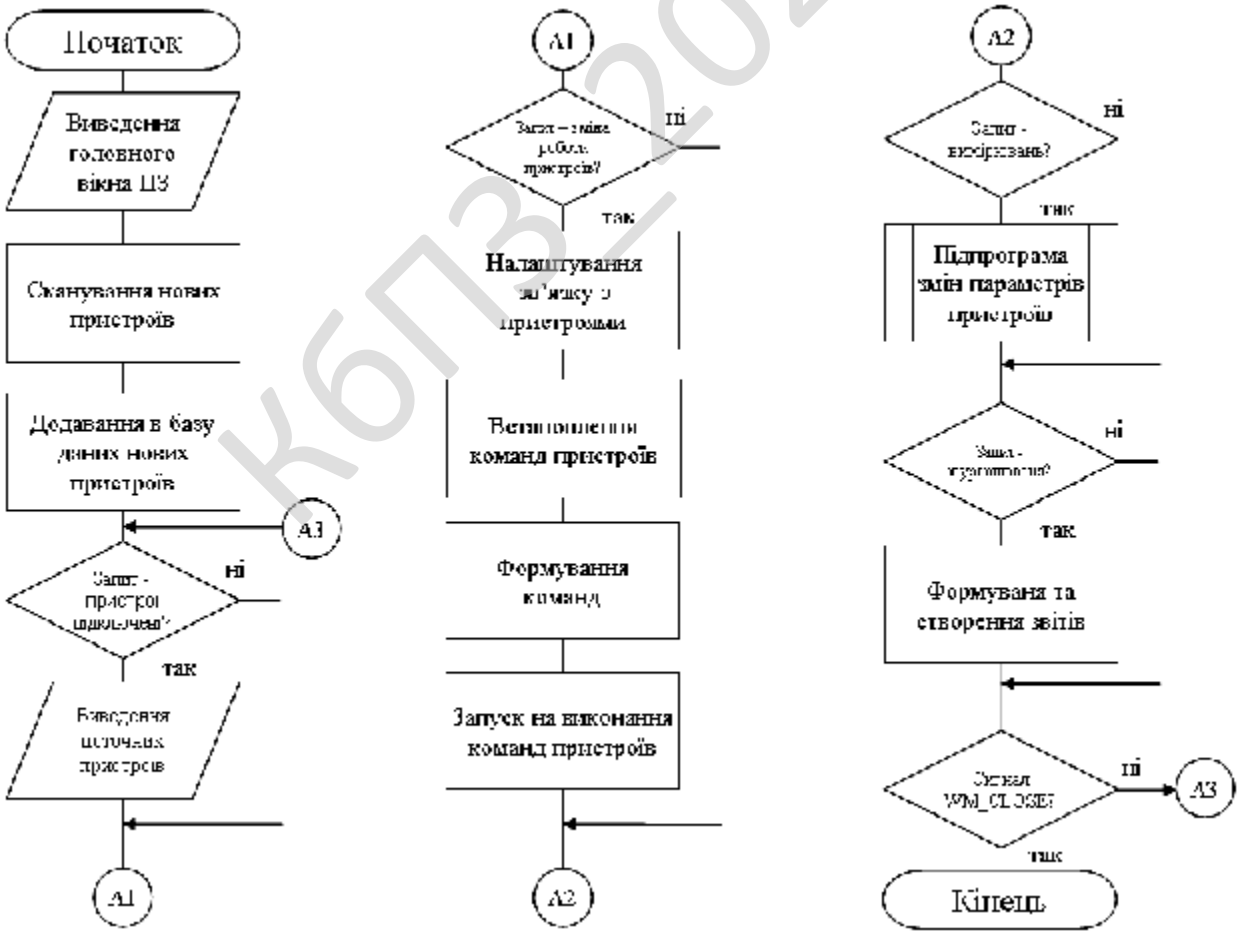

Рисунок 4.1 – Блок-схема основної програми

– Запит – пристрої підключені?

– Виведення поточних пристроїв.

– Запит – зміна роботи пристроїв?

– Налаштування зв’язку з пристроями.

– Встановлення команд пристроїв.

– Формування команд.

– Запуск на виконання команд пристроїв.

– Запит – вимірювань?

– Підпрограма змін параметрів пристроїв.

– Запит – журналюваня?

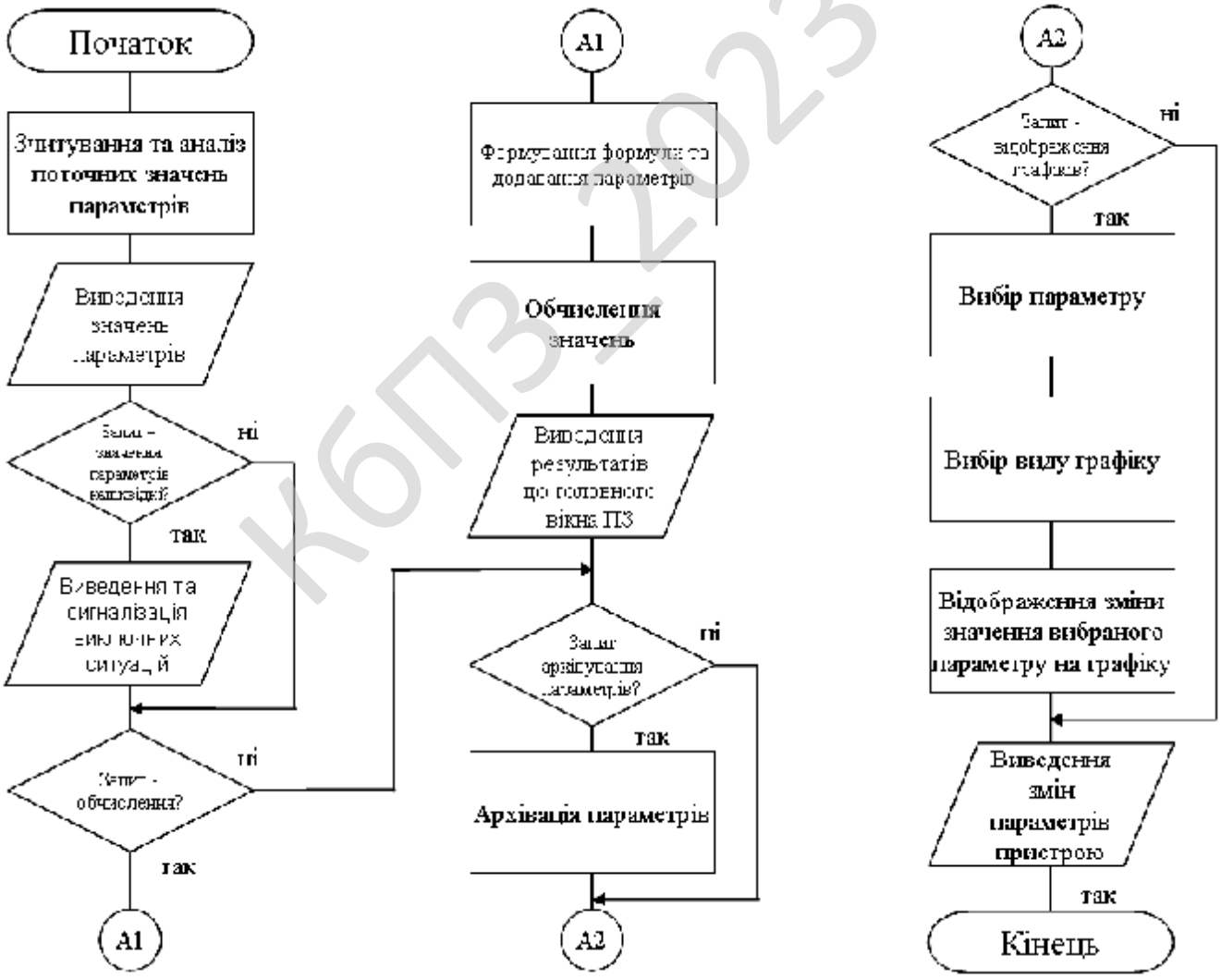

Рисунок 4.2 – Блок-схема підпрограми

– Формування та створення звітів.

– Сигнал WM_CLOSE?

На рисунку 4.2 наведено блок-схему підпрограми змін параметрів пристроїв. Її робота складається з виконання наступних кроків:

– Зчитування та аналіз поточних значень параметрів.

– Виведення значень параметрів.

– Запит – значення параметрів неліквідні?

– Виведення та сигналізація виключних ситуацій.

– Запит – обчислення?

– Формування формули та додавання параметрів.

– Обчислення значень.

– Виведення результатів до головного вікна ПЗ.

– Запит архівування параметрів?

– Архівація параметрів.

– Запит – відображення графіків?

– Вибір параметру.

– Вибір виду графіку.

– Відображення зміни значення вибраного параметру на графіку.

– Виведення змін параметрів пристрою.

Розглянемо опис алгоритмів функціонування системи.

Основна частина системи це – модулі. Тісна інтеграція модулів з ядром поліпшує стабільність системи в цілому, завдяки повторному використанню налагодженого коду.

Загальносистемні користувальницькі об'єкти. Абстрактний об'єкт являє собою асоціативний контейнер властивостей і функцій. Властивості можуть містити як дані чотирьох базових типів, так і інші об'єкти.

Доступ до властивостей об'єкта звичайно здійснюється за допомогою запису імен властивостей через крапку до об'єкта <obj.prop>, а також за допомогою висновку ім'я властивості у квадратні дужки <obj["prop"]>.

Очевидно, що перший механізм статичний, а другий дозволяє вказувати ім'я властивості через змінну. Базове визначення об'єкта не містить функцій.

Операції копіювання об'єкта насправді роблять посилання на вихідний об'єкт. При видаленні об'єкта здійснюється зменшення лічильника посилань, а при досягненні лічильника посилань нуля об'єкт віддаляється фізично.

Різні компоненти можуть довизначати базовий об'єкт особливими властивостями й функціями. Стандартним розширенням об'єкта є масив "Array".

Об'єкт Array. Особливістю масиву є те, що він працює із властивостями, як з індексами, і повне їхнє іменування безглуздо, а значить доступний механізм обігу тільки висновком індексу у квадратні дужки <arr[1]>. Масив зберігає властивості у власному контейнері одномірного масиву. Цифрові властивості масиву використовуються для доступу безпосередньо до масиву, а символьні працюють як властивості об'єкта.

Масив надає спеціальну властивість "length" для одержання розміру масиву <var = arr.length;>. Також масив надає наступні функції:

– string join(string sep = ","), string toString(string sep = ","), string valueOf(string sep = ",") – Повертає рядок з елементами масиву розділеними <sep> або символом

– Array concat(Array arr); – Додає до вихідного масиву елементи масиву <arr>. Повертає вихідний масив зі змінами.

– int push(ElTp var, ...); – Поміщає елемент(ы) <var> у кінець масиву, як у стек. Повертає новий розмір масиву.

– ElTp pop(); – Видалення останнього елемента масиву й повернення його значення, як зі стека.

– Array reverse(); – Зміна порядку розташування елементів масиву. Вертається вихідний масив зі змінами.

– ElTp shift(); – Зрушення масиву у верх. При цьому перший елемент масиву віддаляється, а його значення вертається.

– int unshift(ElTp var, ...); – Засовує елемент(и) <var> у масив. Перший елемент в 0, другий в 1 і т.д.

– Array slice(int beg, int end); – Повертає фрагмент масиву від <beg> до <end>. Якщо значення початку або кінця негативно, то відлік ведеться з кінця масиву. Якщо кінець не зазначений, то кінцем є кінець масиву.

– Array splice(int beg, int rem, ElTp vall, ElTp val2, ...); – Вставляє, видаляє або заміняє елементи масиву. Повертає вихідний масив зі змінами. У першу чергу здійснюється видалення елементів з позиції <beg> і кількістю <rem>, а потім вставляються значення <vall> і т.д., починаючи з позиції <beg>.

– Array sort(); – Сортування елементів масиву в лексикографічному порядку.

Об'єкт RegExp. Об'єкт роботи з регулярними вираженнями, заснований на бібліотеці PCRE. При глобальному пошуку встановлюється атрибут об'єкта "lastIndex", що дозволяє продовжити пошук при наступному виклику функції. У випадку невдалого пошуку атрибут "lastIndex" скидається в нуль.

Як аргументи створення об'єкта передається рядок з текстом регулярного вираження й прапори у вигляді рядка символів:

– 'g' – режим глобального пошуку;

– 'i' – режим регістронезалежного пошуку;

– 'm' – режим багаторядкового пошуку;

– 'p' – тестування вираження по звичайному шаблоні із ключовими символами: '?', '*' і '\'.

Властивості об'єкта:

– source – Вихідний шаблон регулярного вираження, тільки читання.

– global – Ознака глобального пошуку, тільки читання.

– ignoreCase – Ознака ігнорувати регістр символів при пошуку, тільки читання.

– multiline – Ознака багаторядкового пошуку, тільки читання.

– lastIndex – Індекс символу за підрядком останнього пошуку. Використовується в глобальному режимі для продовження сканування, при наступному виклику.

Функції об'єкта:

– Array exec(string val); – Виклик пошуку по рядку <val>. Повертає знайдену підрядок (0) і підвиразу (>0) у масиві. Установлює атрибут масиву "index" у позицію знайденого підрядка. Установлює атрибут масиву "input" у значення вихідного рядка.

```
var re = new RegExp("(\\d\\d)[-/](\\d\\d)[-/](\\d\\d(?:\\d\\d)?)","");
var rez = re.exec("12/30/1969"); var month = rez[1]; var day = rez[2]; var year = rez[3];
- bool test(string val); - Повертає "true" якщо подрядок знайдений в <val>.
var re = new RegExp("(\\d\\d)[-/](\\d\\d)[-/](\\d\\d(?:\\d\\d)?)","");
var OK = re.test("12/30/1969");
```

Об'єкт XMLNodeObj. Функції:

– string name() – Ім'я вузла, XML-тегу.

– string text() – Текст вузла, уміст XML-тегу.

– string attr(string id) – Значення атрибута вузла <id>.

– XMLNodeObj setName(string vl) – Установка ім'я вузла в <vl>. Повертає поточний вузол.

– XMLNodeObj setText(string vl) – Установка тексту вузла в <vl>. Повертає поточний вузол.

– XMLNodeObj setAttr(string id, string vl) – Установка атрибута <id> у значення <vl>. Повертає поточний вузол.

– int childSize() – Кількість вкладених вузлів.

– XMLNodeObj childAdd(ElTp no = XMLNodeObj); XMLNodeObj childAdd(string no) – Додавання об'єкта <no> як вкладеного. <no> може бути як безпосередньо об'єктом-результатом функції SYS.XMLNode(), так і рядком з ім'ям нового тегу. Вертається вкладений вузол.

– XMLNodeObj childIns(int id, ElTp no = XMLNodeObj); XMLNodeObj childIns(int id, string no) – Вставка об'єкта <no> як вкладеного в позицію <id>. <no> може бути як безпосередньо об'єктом-результатом функції SYS.XMLNode(), так і рядком з ім'ям нового тегу. Вертається вкладений вузол.

– XMLNodeObj childDel(int id) – Видалення вкладеного вузла в позиції <id>. Повертає поточний вузол.

– XMLNodeObj childGet(int id) – Одержання вкладеного вузла в позиції <id>.

– XMLNodeObj parent() – Одержати батьківський вузол.

– string load(string str, bool file = false, bool full = false, string cp = " UTF-8") – Завантаження XML з рядка <str> або з файлу зі шляхом в <str> якщо <file> "true", з кодуванням <cp>. <full> – для повного завантаження, із блоками тексту й коментарями в спеціальних вузлах.

– string save(int opt = 0, string path = "", string cp = " UTF-8") – Збереження дерева XML у рядок або у файл <path> з параметрами форматування <opt> і кодуванням <cp>. Повертає текст XML або код помилки. Передбачено наступні опції форматування <opt>:

– 0x01 – переривати рядок перед відкриваючим тегом;

– 0x02 – переривати рядок після відкриваючого тегу;

– 0x04 – переривати рядок після закриваючого тегу;

– 0x08 – переривати рядок після тексту;

– 0x10 – переривати рядок після інструкції;

– 0x1E – переривати рядок після всіх;

– 0x20 – вставляти стандартний XML-заголовок;

– 0x40 – вставляти стандартний XHTML-заголовок.

– XMLNodeObj getElementBy(string val, string attr = "id") – одержати елемент із дерева по атрибуті <attr> зі значенням <val>.

Система (SYS). Функції об'єкта:

– string system(string cmd, bool noPipe = false); – здійснює виклик консольних команд <cmd> ОС із поверненням результату по каналі. Якщо <noPipe> установлений то вертається код повернення виклику й можливий запуск програм у тлі ("sleep 5 &"). Функція відкриває широкі можливості користувачеві WebSCADA шляхом виклику будь-яких системних програм, утиліт і скриптів, а

також одержання за допомогою їх доступу до величезного обсягу системних даних. Наприклад команда "ls -l" поверне деталізований уміст робочої директорії.

– string fileRead(string file); – Повертає вміст файлу <fle> у рядку.

– int fleWrite(string file, string str, bool append = false); – Записує рядок <str> у файл <file>, видаляючи присутній файл або додаючи в нього <append>. Повертає кількість записаних байт.

– int message(string cat, int level, string mess); – формування системного повідомлення <mess> з категорією <cat>, рівнем <level>. Негативне значення рівня формує порушення (Alarm).

– int messDebug(string cat, string mess); int messlnfo(string cat, string mess); int messNote(string cat, string mess); int messWarning(string cat, string mess); int messErr(string cat, string mess); int messCrit(string cat, string mess); int messAlert(string cat, string mess); int messEmerg(string cat, string mess); – формування системного повідомлення <mess> з категорією <cat> і відповідним рівнем.

– XMLNodeObj XMLNode(string name = ""); – створення об'єкта вузла XML с ім'ям <name>.

– string cntrReq(XMLNodeObj req, string stat = ""); – запит інтерфейсу керування до системи за допомогою XML. Звичайний запит записується у вигляді <get path="/OPath/ %2felem"/>. При вказівці станції здійснюється запит до зовнішньої станції.

```
req = SYS.XMLNode("get").setAttr("path","/%2fgen%2fid");
SYS.cntrReq(req);
idSt = req.text();
```

– string sleep(int tm, int ntm = 0); – приспати потік виконання на <tm> секунд і <ntm> наносекунд. Функція додана тільки для повноти й украй не рекомендована до використання, особливо в процедурах користувальницького інтерфейсу оскільки це приведе до блокування інтерфейсу.

– int time(int usec); – повертає абсолютний час у секундах від епохи 1.1.1970 і мікросекундах, якщо <usec> зазначений.

– int localtime(int fullsec, int sec, int min, int hour, int mday, int month, int year, int wday, int yday, int isdst); – повертає повну дату в секундах (sec), хвилинах (min), годинниках (hour), днях місяця (mday), місяці (month), році (year), днях тижня (wday), днях у році (yday) і ознака літнього часу (isdst), виходячи з абсолютного часу в секундах <fullsec> від епохи 1.1.1970.

– string strftime(int sec, string form = "% Y-Y-%m-%d %H:%M:%S"); – Перетворить абсолютний час <sec> у рядок потрібного формату <form>. Запис формату відповідає POSIX-функції strftime.

– int strptime(string sttr, string form = "% Y-Y-%m-%d %H:%M:%S"); – Повертає час у секундах від епохи 1.1.1970, виходячи зі строкового запису часу <str>, відповідно до зазначеного шаблона <form>. Наприклад, шаблону "% Y-Y-%m-%d %H:%M:%S" відповідає час " 2006-08-08 11:21:55". Опис формату шаблона можна одержати з документації на POSIX-функцію "strptime".

– int cron(string cronreq, int base = 0); – Повертає час, спланований у форматі стандарту Cron <cronreq>, починаючи від базового часу <base> або від поточного, якщо базове не зазначено.

– string strFromCharCode(int charl, int char2, int char3, ...); – Створення рядка з кодів символів char1, char2 ... char.

– string strCodeConv(string src, stringfromCP, string toCP); – Кодування тексту <src> з кодування <fromCP> в <toCP>. Якщо кодування опущене, то використовується внутрішня.

Будь-який об'єкт (TCntrNode) дерева WebSCADA (SYS.*). Функції об'єкта:

– TArrayObj nodeList(string grp = "", string path = ""); – Одержання списку дочірніх вузлів для групи <grp> і вузла по шляху <path>. Якщо <grp> порожня то вертаються вузли всіх груп.

– TCntrNodeObj nodeAt(string path, string sep=""); – Підключення до вузла <path> у дереві об'єктів WebSCADA. Якщо вказується роздільник в <sep> то шлях обробляється як рядок з роздільником.

– TCntrNodeObj nodePrev(); – Одержати попередній, батьківський, вузол.

– string nodePath(string sep = "", bool fromroot = true); – Одержання шляху до поточного вузла, у дереві об'єктів WebSCADA. Один символ роздільника вказується в <sep> для одержання шляху через роздільник, наприклад, "DAQ.ModBus.PLC1.P1.var", інакше "/DAQ/ModBus/PLC1/P1/var". <from_root> указує на необхідність формувати шлях від кореня й без вказівки ідентифікатора станції.

Підсистема "Безпека" (SYS.Security). Функції об'єкта підсистеми (SYS.Security):

– int access(string user, int mode, string owner, string group, int access) – Перевірка для користувача <user> доступу до ресурсу який належить <owner> і <group> з доступом <access> для режим <mode>:

– user – користувач для перевірки доступу; mode – режим доступу (4-R, 2-W, 1-X); owner – власник ресурсу; group – група ресурсу;

– access – режим доступу до ресурсу (RWXRWXRWX – 0777).

Функції об'єкта користувача (SYS.Security["usr_User"]) і групи (SYS.Security["grp_Group"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

Підсистема "БД" (SYS.BD).

Функції об'єкта БД (SYS.BD["TypeDB"]["DB"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

– Array SQLReq(string req); – Формування SQL-запиту до БД.

```
DBTbl=SYS.BD.MySQL.GenDB.SQLReq("SELECT *  from DB;");
for(var i rw = 0;  i rw < DBTbl.length;  i rw++)
{
var rec = "";
```

```
        for(var i_fld = 0;  i_fld < DBTbl[i_rw].length;  i_fld++ )
        rec  +=  DBTbl[i_rw][i_fld]+"\t";  SYS.messDebug("TEST  DB","Row  "+i_rw+":
"+rec);
        if(i_rw)   SYS.messDebug("TEST DB:   ","Row "+i_rw+":   'NAME'"+DBTbl[i_rw]
["NAME"]);
        }
```

Функції об'єкта Таблиці (SYS.BD["TypeDB"]["DB"]["Table"]):

– XMLNodeObj fieldStruct(); – Одержання структури таблиці у вигляді XML вузла "field" з дочірніми вузлами-колонками <RowId type="real" len="10.2" key="1" def="Default value">{Value}</RowId>**,** де:

– {RowId} – ідентифікатор колонки;

– {Value} – значення колонки;

– type – тип значення колонки: str – рядок, int – ціле, real – речовинне й bool – логічне;

– len – розмір значення колонки, у знаках;

– key – ознака того, що колонка є ключем, і пошук здійснюється по його значенню;

– def – значення колонки за замовчуванням.

– string fieldSeek(int row, XMLNodeObj fld); – Запит поля <row> таблиці. Якщо поле отримане то вертається "1" інакше "0". У випадку помилки вертається "0:Error".

– string fieldGet(XMLNodeObj fld); – Запит значень поля. У випадку помилки вертається "0:Error".

```
req = SYS.XMLNode("field");
req.childAdd("user").setAttr("type","str").setAttr("key","1").setText("root");
req.childAdd("id").setAttr("type","str").setAttr("key","1").
setText("/Lang2CodeBase");
req.childAdd("val").setAttr("type","str");
SYS.BD.MySQL.GenDB.SYS.fieldGet(req);
SYS.messDebug("TEST DB","Value:   "+req.childGet(2).text());
```

– string fieldSet(XMLNodeObj fld); – Установка поля. У випадку помилки вертається "0:Error".

– string fieldDel(XMLNodeObj fld); – Видалення поля. У випадку помилки вертається "0:Error".

Підсистема "Збір даних" (SYS.DAQ). Функції об'єкта підсистеми (SYS.DAQ):

– bool funcCall(string progLang, TVarObj args, string prog); – виклик тексту функції <prog> з аргументами в об'єкті <args> для мови програмування <progLang>. Повертає "true" при коректному виклику.

```
var args = new Object(); args.y = 0; args.x = 123;
SYS.DAQ.funcCall("BuilderCLikeCalc.BuilderCScript",args,"y=2*x;");
SYS.messDebug("TEST Calc","TEST Calc rezult:   "+args.y);
```

Функції об'єкта контролера (SYS.DAQ["Modul"]["Controller"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

– string name() – ім'я контролера.

– string descr() – опис контролера.

– string status() – статус контролера.

– bool alarmSet(string mess, int lev = -5, string prm = "") – установка/зняття порушення <mess> з рівнем <lev> (негативний для установки інакше зняття), для параметра <prm>.

– bool enable(bool newSt = EVAL) – одержання стану "Включена" або зміна його призначенням атрибута <newSt>.

– bool start(bool newSt = EVAL) – одержання стану "Запущена" або зміна його призначенням атрибута <newSt>.

Функції об'єкта параметра контролера (SYS.DAQ["Modul"]["Controller"]["Parameter"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

Функції об'єкта атрибута параметра контролера (SYS.DAQ["Modul"]["Controller"] ["Parameter"]["Attribute"]):

– ElTp get(int tm = 0, int utm = 0, bool sys = false) – запит значення атрибута на час <tm:utm> і ознакою системного доступу <sys>.

– bool set(ElTp val, int tm = 0, int utm = 0, bool sys = false) – запис значення <val> в атрибут з міткою часу <tm:utm> і ознакою системного доступу <sys>.

– TCntrNodeObj arch() – одержання об'єкта архіву, пов'язаного із цим атрибутом. У випадку відсутності зв'язаного архіву вертається "false".

– string descr() – опис атрибута.

– int time(int utm) – час останнього значення в секундах і мікросекундах в <utm>.

– int len() – довжина поля.

– int dec() – дозвіл для дійсного числа.

– int flg() – прапори поля.

– string def() – значення за замовчуванням.

– string values() – список припустимих значень або діапазон.

– string selNames() – список імен припустимих значень.

– string reserve() – резервна властивість значення.

Функції об'єкта бібліотеки шаблона (SYS.DAQ[tmplb_Lib"]) і шаблона (SYS.DAQ[tmplb_Lib"]["Tmpl"]) параметра контролера:

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

Модуль DAQ.BuilderCLikeCalc. Об'єкт "Бібліотека функцій".

(SYS.DAQ.BuilderCLikeCalc["lib_Lfunc"])

ElTp funcID}(ElTp prml, ...) – виклик функції бібліотеки funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція".

```
(SYS.DAQ.BuilderCLikeCalc["lib_Lfunc"]["func"])
```

ElTp call(ElTpprml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

Модуль DAQ.ModBus.

Об'єкт "Контролер" (this.nodePrev()).

string messIO(string pdu) – відправлення PDU <pdu> через транспорт об'єкта контролера за допомогою ModBus протоколу. PDU результату запиту міститься замість запиту в <pdu> , а помилка вертається в результаті функції.

Підсистема "Архіви" (SYS.Archive).

Функції об'єкта підсистеми:

– Array messGet(int btm, int etm, string cat = "",int lev = 0,string arch = ""); – запит системних повідомлень за час від <btm> до <etm> для категорії <cat>, рівня <lev> і архіватора <arch>. Вертається масив об'єктів повідомлень із властивостями:

– tm – час повідомлення, секунди;

– utm – час повідомлення, мікросекунди;

– categ – категорія повідомлення;

– level – рівень повідомлення;

– mess – текст повідомлення.

Функції об'єкта архіватора повідомлень (SYS.Archive["mod_Modul"] ["mess_Archivator"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

– bool status() – одержання статусу архіватора.

– int end() – повчання часу закінчення даних архіватора.

– int begin() – повчання часу початку даних архіватора.

Функції об'єкта архіву (SYS.Archive["va_Archive"]) і архіватора значень (SYS.Archive["val_Modul"]["val_Archivator"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

Підсистема "Транспорти" (SYS.Transport).

Функції об'єкта вхідного транспорту (SYS.Transport ["Modul"]["in_Transp"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

Функції об'єкта вихідного транспорту (SYS.Transport["Modul"]["out_Transp"]):

– ElTp cfg(string nm) – одержання значення конфігураційного поля <nm> об'єкта.

– ElTp cfgSet(string nm, ElTp val) – установка конфігураційного поля <nm> об'єкта в значення <val>.

– string messIO(string mess, real timeOut = 0); – відправлення повідомлення <mess> через транспорт із таймаутом очікування <timeOut> (у секундах). У випадку нульового таймаута цей час береться з настроювань вихідного транспорту.

```
rez=SYS.Transport.Serial.out_ttyUSB0.messIO(SYS.strFromCharCode(0x4B,0x0  0,
0x37,0x40),0.2);
while(true)
{
trez = SYS.Transport.Serial.out_ttyUSB0.messIO("");
if(  !trez.length )   break;
rez+=trez;
}
```

– int messIO(XMLNodeObj req, string prt); – відправлення запиту <req> до протоколу <prt> для здійснення сеансу зв'язку через транспорт за допомогою протоколу.

```
req = SYS.XMLNode("TCP");
req.setAttr("id","test").setAttr("reqTm",1).setAttr("node",1).setAttr("req
Try",2).setText(SYS.strFromCharCode(0x03,0x0 0,0x0 0,0x0 0,0x05));
 SYS.Transport.Sockets.out_testModBus.messIO(req,"ModBus");
test = Special.FLibSYS.strDec4Bin(req.text());
```

Підсистема "Користувальницькі інтерфейси" (SYS.UI). Модуль UI.VCAEngine. Об'єкт "Сеанс" (this.ownerSess()).

– string user() – поточний користувач сеансу.

– string alrmSndPlay() – шлях віджета для якого на даний момент відтворюється повідомлення про порушення.

– int alrmQuittance(int quittmpl, string wpath = "") – квітирування порушень <wpath> із шаблоном <quit_tmpl>. Якщо <wpath> порожній рядок то виробляється глобальне квітирування.

Об'єкт "Віджет" (this)

– TCntrNodeObj ownerSess() – одержати об'єкт сеансу даного віджета.

– TCntrNodeObj ownerPage() – одержати об'єкт батьківської сторінки даного віджета.

– TCntrNodeObj ownerWdg(bool base = false) – одержати батьківський віджет даного віджета. При вказівці <base> буде повернуті й об'єкти сторінок.

– TCntrNodeObj wdgAdd(string wid, string wname, string parent) – додавання віджета <wid> з ім'ям <wname> на основі бібліотечного віджета <parent>.

```
//Додати новий віджет на основі віджета текстового примітива
nw = this.wdgAdd("nw",   "Новий віджет",   "/wlb originals/wdg Text");
nw.attrSet("geom",   50).attrSet("geom",   50)7                ~
```

– bool wdgDel(string wid) – видалення віджета <wid>.

– bool attrPresent(string attr) – перевірка атрибута віджета <attr> на факт присутності.

– ElTp attr(string attr) – одержання значення атрибута віджета <attr>. Для відсутніх атрибутів вертається порожній рядок.

– TCntrNodeObj attrSet(string attr, ElTp vl) – установка атрибута віджета <attr> у значення <vl> . Вертається поточний об'єкт для конкатенації функцій установки.

– string link(string attr, bool prm = false) – одержання посилання для атрибута віджета <attr>. При установці <prm> запитується посилання для групи атрибутів (параметр), представлених зазначеним атрибутом.

– string linkSet(string attr, string vl, bool prm) – установка посилання для атрибута віджета <attr> . При установці <prm> здійснюється установка посилання для групи атрибутів (параметр), представлених зазначеним атрибутом.

```
//Установити посилання восьмого тренда параметром
this.linkSet("el8.name",   "prm:/LogicLev/experiment/Pi",   true);
```

Об'єкт "Віджет", примітива "Документ" (this).

string getArhDoc(integer nDoc) – одержати текст документа архіву на глибині <nDoc> (0-0-{aSize-1}).

Підсистема "Спеціальні" (SYS.Special). Модуль Special.FLibSYS. Об'єкт "Бібліотека функцій" (SYS.Special.FLibSYS).

ElTp {funcID}(ElTp prml, ...) – виклик функції бібліотеки {funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція". (SYS.Special.FLibSYS["funcID"]).

ElTp call(ElTp prml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

Модуль Special.FLibMath. Об'єкт "Бібліотека функцій" (SYS.Special.FLibMath).

ElTp {funcID}(ElTp prml, ...) – виклик функції бібліотеки {funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція". (SYS.Special.FLibMath["funcID"]).

ElTp call(ElTp prml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

Модуль Special.FLibComplex1. Об'єкт "Бібліотека функцій" (SYS.Special.FLibComplex1).

ElTp {funcID}(ElTp prml, ...) – виклик функції бібліотеки {funcID}. Повертає результат викликуваної функції.

Об'єкт "Користувальницька функція". (SYS.Special. FLibComplex1 ["funcID"]).

ElTp call(ElTp prml, ...) – виклик даної функції з параметрами <prm{N}>. Повертає результат викликуваної функції.

На основі модулів реалізовані наступні функціональні частини системи WebSCADA:

– бази даних;

– архіви (повідомлень і значень);

– протоколи комунікаційних інтерфейсів;

– комунікаційні інтерфейси, транспорти;

– джерела даних і збір даних;

– інтерфейси користувача (GUI, TUI, WebGUI, speach, signal ... .);

– додаткові модулі, спеціальні.

Керування модулями здійснюється підсистемою «Керування модулями». Функціями підсистеми є: підключення, відключення, відновлення модулів, а також інші операції, пов'язані з модулями й бібліотеками модулів.

Підсистеми. Архітектурно система WebSCADA ділиться на підсистеми. Підсистеми можуть бути двох типів: звичайні й модульні. Модульні підсистеми мають властивість розширення за допомогою модулів. Кожна модульна підсистема може містити безліч модульних об'єктів. Наприклад, модульна підсистема «Бази даних» містить модульні об'єкти типів баз даних. Модульний об'єкт є коренем усередині модуля.

Усього система WebSCADA містить 9 підсистем з них 7 підсистем є модульними. 9 підсистем системи WebSCADA є базовими й присутні в будь-якій конфігурації. До списку 9 підсистем можуть додаватися нові підсистеми за допомогою модулів. Підсистеми системи WebSCADA:

– Архіви (модульна).

– Бази даних (модульна).

– Безпека.

– Інтерфейси користувача (модульна).

– Керування модулями.

– Збір даних (модульна).

– Транспортні протоколи (модульна).

– Спеціальні (модульна).

– Транспорти (модульна).

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою NTRU. NTRUEncrypt (абревіатура Nth-degree TRUncated polynomial ring або Number Theorists aRe Us) – це криптографічна система з відкритим ключем, що раніше називалася NTRU.

Криптосистема NTRUEncrypt, заснована на ґратчастій криптосистемі, створена як альтернатива RSA і криптосистемам на еліптичних кривих (ECC). Стійкість алгоритму забезпечується труднощами пошуку найкоротшого вектора ґрати, що більше стійка до атак, здійснюваним на квантових комп'ютерах. На відміну від своїх конкурентів RSA,ECC, Elgamal, алгоритм використовує операції над кільцем:

$$\mathbb{Z}[X]/(X^N - 1),$$

усічених багаточленів ступеня, що не перевершує $N - 1$:

$$\mathbf{a}(X) = \mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1}.$$

Такий багаточлен можна також представити вектором:

$$\vec{a}(X) = \vec{a} = \sum_{i=0}^{N-1} a_i X^i = [a_0, a_1, a_2, \cdots, a_{N-2}, a_{N-1}].$$

Як і будь-який молодий алгоритм, NTRUEncrypt погано вивчений, хоча й був офіційно затверджений для використання в сфері фінансів комітетом Accredited Standards Committee X9.[1]

Існує реалізації NTRUEncrypt з відкритим вихідним кодом.[2]

NTRUEncrypt, що споконвічно називався NTRU, був винайдений в 1996 році й представлений на конференціях CRYPTO, Конференція RSA, Eurocrypt. Причиною, що послужила початком розробки алгоритму в 1994 році, стала стаття [3], у якій говорилося про легкість злому існуючих алгоритмів на квантових комп'ютерах, які, як показало час, не за горами[4]. У цьому ж році, математики Jeffrey Hoffstein, Jill Pipher і Joseph H. Silverman, що розробили систему разом із засновником компанії NTRU Cryptosystems, Inc. (пізніше перейменованої в SecurityInnovation), Даніелєм Лієманом (Daniel Lieman) запатентували свій винахід.[5]

Кільця усічених багаточленів

NTRU оперує над багаточленами ступеня не переважаючої $N-1$:

$$\mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1},$$

де коефіцієнти $a_0, \cdots, a_{N-1}$ – цілі числа. Щодо операцій додавання й множення за модулем багаточлена $X^N - 1$.Такі багаточлени утворять кільце *R*, називане кільцем усічених багаточленів, що ізоморфно кільцю відносин:

$$\mathbb{Z}[X]/(X^N - 1).$$

NTRU використовує кільце усічених багаточленів *R* разом з діленням за модулем на взаємно прості числа *p* і *q* для зменшення коефіцієнтів багаточленів.

У роботі алгоритму також використовуються зворотні багаточлени в кільці усічених багаточленів. Слід зазначити, що не всякий багаточлен має зворотний, але якщо зворотний поліном існує, то його легко обчислити.[6][7]

Генерація відкритого ключа

Для передачі повідомлення від Аліси до Боба необхідні відкритий і закритий ключі. Відкритий знають як Боб, так і Аліса, закритий ключ знає тільки Боб, що він використовує для генерації відкритого ключа. Для цього Боб вибирає

два «маленьких» поліноми *f g* з *R*. «Малість» поліномів мається на увазі в тому розумінні, що він маленький щодо довільного полінома за модулем *q*: у довільному поліномі коефіцієнти повинні бути приблизно рівномірно розподілені за модулем *q*, а в малому поліномі вони багато менше *q*. Малість поліномів визначається за допомогою чисел *df* і *dg*:

Поліном *f* має *df* коефіцієнтів рівних «1» і *df*-1коефіцієнтів рівних «-1», а інші – «0». У цьому випадку говорять, що:

$$\mathbf{f} \in \mathcal{L}_f$$

Поліном *g* має *dg* коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0» У цьому випадку говорять, що:

$$\mathbf{g} \in \mathcal{L}_g$$

Причина, по якій поліноми вибираються саме таким чином, полягає в тому, що *f* , можливо, буде мати зворотний, а *g* – однозначно немає (*g* (1) = 0, а нульовий елемент не має зворотного).

Боб повинен зберігати ці полінома в секреті. Далі Боб обчислює зворотні поліноми $\mathbf{f}_p$ й $\mathbf{f}_q$, тобто такі, що:

$$\mathbf{f} \cdot \mathbf{f}_p \equiv 1 \pmod{p} \text{ й } \mathbf{f} \cdot \mathbf{f}_q \equiv 1 \pmod{q}$$

Якщо *f* не має зворотного полінома, то Боб вибирає інший поліном *f*.

Секретний ключ – це пари $(\mathbf{f}, \mathbf{f}_p)$, а відкритий ключ *h* обчислюється за формулою:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod q.$$

Приклад:

Для приклада візьмемо *df*=4, а *dg*=3. Тоді як поліноми можна вибрати

$$\mathbf{f} = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$

$$\mathbf{g} = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$$

Далі для полінома *f* шукаються зворотні поліноми за модулем *p*=3 иq=32:

$$\mathbf{f}_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9$$

$$\mathbf{f}_q = 5+9X+6X^2+16X^3+4X^4+15X^5+16X^6+22X^7+20X^8+18X^9+30X^{10}$$

Заключним етапом є обчислення самого відкритого ключа *h*:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod 32 = 8+25X+22X^2+20X^3+12X^4+24X^5+15X^6+19X^7+12X^8+19X^9+16X^{10}.$$

Шифрування

Тепер, коли в Аліси є відкритий ключ, вона може відправити зашифроване повідомлення Бобові. Для цього потрібно повідомлення представити у вигляді полінома m з коефіцієнтами за модулем *p*, обраними з діапазону (-*p*/2, *p*/2]. Тобто *m* є «малим» поліномом за модулем *q*. Далі Алісі необхідно вибрати інший «малий» поліном r, що називається «сліпучої», обумовлений за допомогою числа *dr*:

Поліном r має *dr* коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0». У цьому випадку говорять, що:

$$\mathbf{r} \in \mathcal{L}_r.$$

Використовуючи ці поліноми, зашифроване повідомлення виходить за формулою:

$$\mathbf{e} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod q.$$

При цьому кожній, хто знає (або може обчислити) осліплюючий поліном r, зможе прочитати повідомлення *m*.

Приклад:

Припустимо, що Аліса хоче послати повідомлення, представлене у вигляді полінома:

$$\mathbf{m} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10},$$

і вибрала «осліплюючий» поліном, для якого *dr*=3:

$$\mathbf{r} = -1 + X^2 + X^3 + X^4 - X^5 - X^7.$$

Тоді шифротекст *e*, готовий для передачі Бобові буде:

$$\mathbf{e} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod 32 = 14+11X+26X^2+24X^3+14X^4+16X^5+30X^6+7X^7+25X^8+6X^9+19X^{10}$$

Розшифрування

Тепер, одержавши зашифроване повідомлення *e*, Боб може його розшифрувати, використовуючи свій секретний ключ. Спочатку він одержує новий проміжний поліном:

$$\mathbf{a} = (\mathbf{f} \cdot \mathbf{e}) \bmod q.$$

Якщо розписати шифротекст, то одержимо ланцюжок:

$$\mathbf{a} = (\mathbf{f} \cdot \mathbf{e}) \bmod q = (\mathbf{f} \cdot (\mathbf{r} \cdot \mathbf{h} + \mathbf{m})) \bmod q = (\mathbf{f} \cdot (\mathbf{r} \cdot p\mathbf{f}_q \cdot \mathbf{g} + \mathbf{m})) \bmod q$$

і остаточно:

$$\mathbf{a} = (p\mathbf{r} \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{m}) \bmod q.$$

Після того, як Боб обчислив поліном a за модулем *q*, він повинен вибрати його коефіцієнти з діапазону (-*q*/2, *q*/2] і далі обчислити поліном b, одержуваний з полінома a приведенням за модулем *p*:

$$\mathbf{b} = \mathbf{a} \bmod p = (\mathbf{f} \cdot \mathbf{m}) \bmod p,$$

так як:

$$(p\mathbf{r} \cdot \mathbf{g}) \bmod p = 0.$$

Тепер, використовуючи другу половину секретного ключа й отриманий поліном *b*, Боб може розшифрувати повідомлення:

$$\mathbf{c} = (\mathbf{f}_p \cdot \mathbf{b}) \bmod p.$$

Неважко бачити, що:

$$\mathbf{c} \equiv \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \equiv \mathbf{m} \pmod p.$$

У такий спосіб отриманий поліном *c* дійсно є вихідним повідомленням *m*.

Приклад:

Боб одержав від Аліси шифроване повідомлення *e*:

$$\mathbf{e} = 14+11X+26X^2+24X^3+14X^4+16X^5+30X^6+7X^7+25X^8+6X^9+19X^{10}$$

Використовуючи секретний ключ *f* Боб одержує поліном *a*:

$$\mathbf{a} = \mathbf{f} \cdot \mathbf{e} \pmod{32} = 3-7X-10X^2-11X^3+10X^4+7X^5+6X^6+7X^7+5X^8-3X^9-7X^{10} \pmod{32},$$

з коефіцієнтами, що належать проміжку (-*q*/2, *q*/2]. Далі перетворить поліном *a* у поліном *b*, зменшуючи коефіцієнти за модулем *p*.

$$\mathbf{b} = \mathbf{a} \pmod 3 = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10} \pmod 3$$

Заключний крок – перемножування полінома *b* із другою половиною закритого ключа $\mathbf{f}_p$:

$$\mathbf{c} = \mathbf{f}_p \cdot \mathbf{b} = \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \pmod 3 = \mathbf{m} \pmod 3$$

$$\mathbf{c} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

Який є вихідним повідомленням, що передавала Аліса.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню ПЗ: Файл; Налаштування; Довідка.

– Лівого блоку деревовидних параметрів: Контролер; Параметри; Налаштування; журнал дій.

– Правих функціональних клавіш: Сканування; Формування звіту; Корегування шаблоні пристроїв; Видалення; Збір даних; Налаштування; Керування модулями.

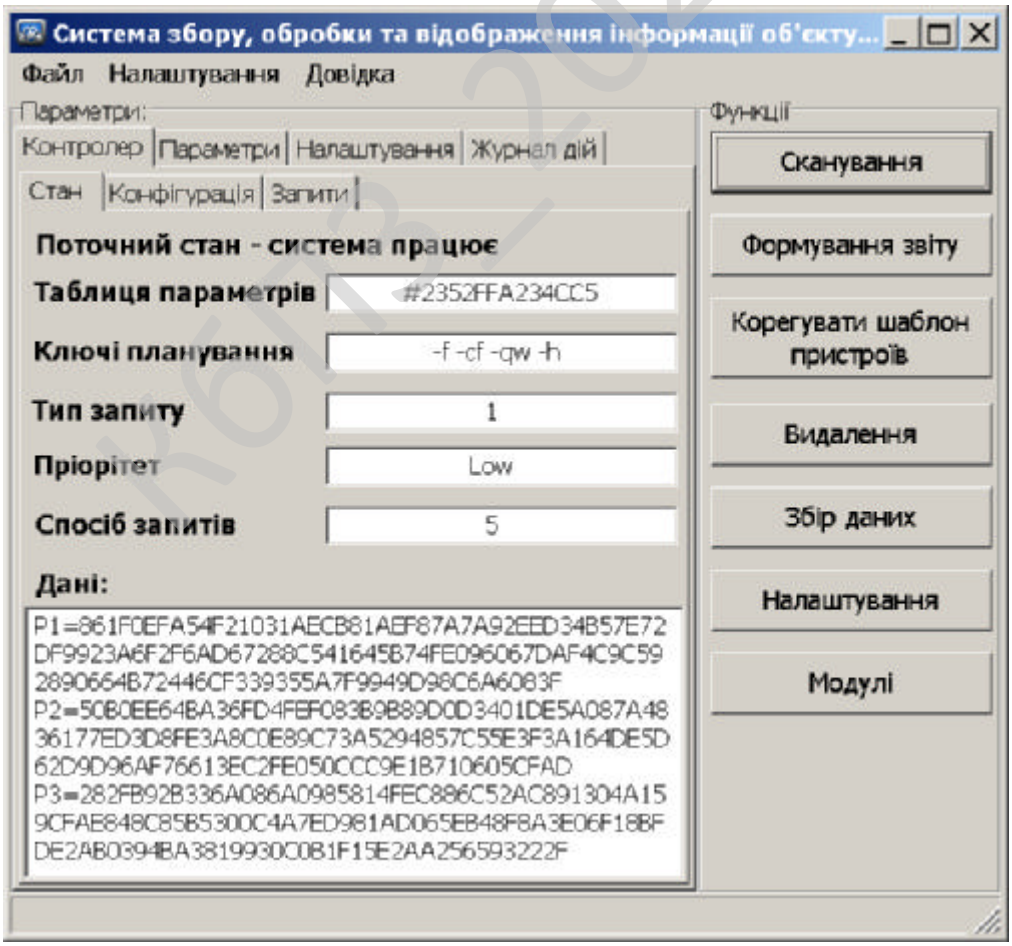

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права. 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 (неповнофункціональна 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 (дискету чи CD-ROM). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

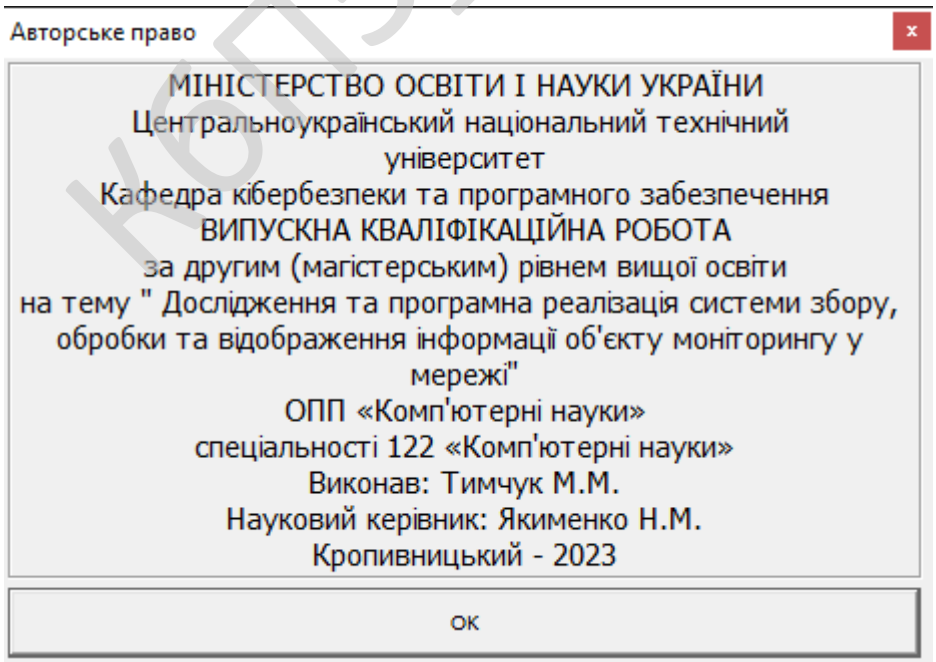

Рисунок 5.2 – Вікно авторського права

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

Метою розробки є дослідження та програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

Об’єктом дослідження є процес збору, обробки та відображення інформації об’єкту моніторингу у мережі.

Предметом дослідження є методи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

Методи дослідження базуються на методах теорії телекомунікацій, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод збору, обробки та відображення інформації об’єкту моніторингу у мережі.

– Розроблено вітчизняний продукт збору, обробки та відображення інформації об’єкту моніторингу у мережі, який має більш широкі 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 збору, обробки та відображення інформації об'єкту моніторингу у мережі. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	58
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

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

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.:

$$T_{уточн}=T_{ном}\, ПВ_j, \tag{7.3}$$

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:

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

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)%

$$T_{РП}=0{,}3\cdot3{,}23\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot82=168 \text{ люд/день}$$

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	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

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,

$$Ч = \frac{209 \cdot 1}{48 - 5} = 4{,}8 \text{ ставки}$$

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 облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	12	1080	18
Монітор	60	12	720	12
Клавіатура	30	12	360	6
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	3	360	6
Принтер струминний	60	2	120	2
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	6	180	3
Кабельні господарства ЛОМ на 1 м.п.	2,5	400	1000	16,67
Копіювальний апарат	140	2	280	4,67
Усього за рік:			$З_ч$	74,67

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф^{с}_{др} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф^{с}_{др} = \frac{75 \cdot 3}{1,2} = 187,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=187,5/(60·8)=0,4 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	12098,5	24197
Продакт-менеджер	0,5	12000	12000
Інженер-програміст	4,8	14000	134400
Інженер-електронщик	0,4	9000	7200
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	$R_{сп}$=7,9	-	$Ф_{роб}$=200597

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{200597}{7,9 \cdot 48} = 529 \; грн.$$

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 «Пектораль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 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

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

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

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

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 10.11.23 – 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™ i3 12100 (BX8071512100) 1700 4 ядра, 8 потоків, 3.3 GHz, 4.3 GHz, TDP 60 Вт, 10nm, Alder Lake, BOX	-
Системна плата	MSI PRO H610M-E DDR4 сокет – 1700 чипсет – Intel H610, DDR4, M.2 2280	-
Жорсткий диск	SSD M.2 2280 240GB Apacer 240 GB, TLC M.2, SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2400 MHz Patriot	-
Відеокарта	Вбудована, Intel UHD Graphics 730	-
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	-
Корпус	ProLogix E105 Minitower, Micro – ATX	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	-
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
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608

Продовження таблиці 7.8

1	2	3	4
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.

$$З_о = 529 \cdot 209/58 = 1906 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних т 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, %

$$З_д = 1906 \cdot 10 \cdot 0{,}01 = 191 \text{ грн}$$

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

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

де H_c – відрахування на соціальні потреби, %

$$C_{оц} = 0{,}01 \cdot 22(1906+191) = 461 \text{ грн}$$

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

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

де $H_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1906 \cdot 15 \cdot 0{,}01 = 286 \text{ грн}$$

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, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм приймаємо 1/3 пачку паперу на період розробки. Тод враховуючи, що вартість пачки паперу складає $Ц_п$=240 грн., визначаємо вартіст паперу за період розробки $N_м$=2 міс:

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

$$З_{М1} = 240 \cdot 1/3 = 80 \text{ грн.}$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартіст CD/DVD дисків в кількості, що дорівнює кількості 10 CD + 1 DVD:

$$З_{М2} = \sum Ц_д., \quad (7.17)$$

де $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box 40 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box – 74 грн/шт.

$$З_{М2} = 40 \cdot 10 + 74 = 474 \text{ грн.}$$

Згідно виданих норм одноразовій заправці підлягають усі друкуючі пристрої становить:

$$З_{МЗ} = \sum Ц_з., \quad (7.18)$$

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.

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

$$З_М = (80 + 474 + 1702)/58=39 \text{ грн.}$$

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

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

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

$$О_п = 1906 \cdot 15 \cdot 0{,}01=286 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програ ($N_е$ = 58 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

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

$$А_м = 174246 \cdot 2/(58 \cdot 12)=501 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтям калькуляції

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

$$С_п = 1906+191+461+286+39+286+501=3670 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмно 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, %

$$П_р = 0{,}01 \cdot 50 \cdot 3670 = 1835 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмног забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	1906
2. Додаткова зарплата виконавців	$З_д$	191
3. Відрахування на соціальні потреби	$С_{оц}$	461
4. Загальногосподарські витрати	$Г_{осп}$	286
5. Витрати на матеріали	$З_М$	39
6. Освоєння нових операційних систем, мов програмування	$О_п$	286
7. Амортизація основних фондів	$А_м$	501
8. Повна собівартість програмного забезпечення	$С_п$	3670
9. Плановий прибуток	$П_р$	1835
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	5505
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1101
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	6606

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	–	6606
Всього капітальних витрат	–	6006

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	$З_р$	40260	21472
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	3303
Всього витрат за рік	$І$	40260	24775

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 300 годин на рік до 160 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 300 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 40260\ грн.$$

до

$$З_{р\ нов} = 160 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 21472\ грн.$$

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 ($Ц_{ел}$).

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

Витрати на електроенергію при впровадженні нової системи не змінюються.

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	–	6606	–	3303
Всього відрахувань	-	–	6606	–	3303

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.

$$Ев = (5505 - 3670) \cdot 58 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 77389 \text{ грн.}$$

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.

$$T_в = \frac{1769406}{(5505 - 3670) \cdot 58 \cdot 12 / 2} = 2,77 \text{ років}$$

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

$$E_{сп} = (40260\text{-}24775)\text{-}0,5 \cdot 6606 = 12182 \text{ грн.}$$

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

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

$$T_{сп} = \frac{6606}{40260 - 24775} = 0,4 \ року$$

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.	58
2. Повна собівартість розробленої програми	Грн.	3670
3. Ціна розробленої програми	Грн.	5505
4. Плановий прибуток від реалізації розробленої програми	Грн.	1835
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.	106430
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	77389
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,77
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	6606
11.Величина економічного ефекту у користувача програмної продукції	Грн.	12182
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,4

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

Наявний в даний час в нашій країні комплекс розроблених організаційних заходів та технічних засобів захисту, накопичений передовий досвід роботи ряду обчислювальних центрів показує, що є можливість домогтися значно більших успіхів у справі усунення впливу на працюючих небезпечних і шкідливих виробничих факторів. Проте стан умов праці та його безпеки в ряді обчислювальних центрів (ОЦ) та підприємств ще не задовольняють сучасним вимогам. Оператори ЕОМ, оператори підготовці даних, програмісти та інші працівники ОЦ та підприємств ще стикаються з впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика і інші.

Багато працівників ОЦ та підприємств пов'язані з впливом таких психофізичних факторів, як розумова перенапруга, перенапруження зорових і слухових аналізаторів, монотонність праці, емоційні перевантаження. Вплив зазначених несприятливих факторів призводить до зниження працездатності, викликане розвиваються втомою. Поява і розвиток втоми пов'язане зі змінами, які виникають під час роботи в центральній нервовій системі, з гальмівними процесами в корі головного мозку. Наприклад сильний шум викликає труднощі з розпізнаванням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5 – 12% продуктивність праці. Тривала дія шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30 – 60%.

Медичні обстеження працівників ОЦ та підприємств показали, що крім зниження продуктивності праці високі рівні шуму призводять до погіршення

слуху. Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Аналіз травматизму серед працівників ВЦ показує, що в основному нещасні випадки відбуваються від впливу фізично небезпечних виробничих факторів при заправці носія інформації на обертовий барабан при знятому кожусі, під час співробітниками невластивих їм робіт. На другому місці випадки, пов'язані з дією електричного струму.

8.2 Аналіз 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	6
Довжина	7
Висота	2,9

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	7
Об'єм, V	$м^3$	не менше 20.0	20,3

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працюють 6 людей. 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 не відповідають нормативним

вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], але 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» [3] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таким чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

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, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в 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.

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.

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	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,1

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 PCanon Pixma G540, 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.

З 2019 року діють Державні будівельні норми України "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], 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. [1], 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.

8.3 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці програміста.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

З точки зору забезпечення пожежної безпеки до цих заходів можна віднести наявність схеми евакуації з приміщення вузла, у випадку пожежі, повішену на вхідні двері.

Аналіз умов праці на робочому місці інженера-програміста показав, що на робочому місці не виконуються вимоги ергономіки. Для виконання їх можна запропонувати заміну не регульованого сидіння на крісло з регульованими

ергономічними параметрами, а також заміну використовуваного столу на робоче місце оператора ЕОМ. Різними фірмами в сукупності розроблено понад 11 схем регулювань параметрів робочого крісла, які забезпечують: плавне переміщення сидіння по висоті за допомогою газової пружини; плавна зміна нахилу спинки і сидіння; регулювання пружинного противотиску спинки крісла на спину оператора; перестановку спинки по висоті; зміна глибини сидіння шляхом зміни вигину краю сидіння; синхронне повторення рухів оператора сидінням і спинкою в правильному кутовому співвідношенні; синхронне повторення спинкою крісла рухів верхньої частини тулуба того, що сидить; амортизацію сидіння.

Найбільш популярними моделями комп'ютерних крісел, які володіють чотирма основними регулюваннями (висоти сидіння, висоти, глибини і нахилу спинки), є італійські, фінські моделі «Senior», «Volos», «Ergo», «Toronto», «Bini», «Metro», «NewStar», «Capris», «Fenix», «Xenus», «Quintus» і т.д. Подібні крісла, відрегульовані відповідно до зростання і ваги оператора, а також характеру виконуваної роботи, дозволяють понизити навантаження на опорно-руховий апарат людини, що працює за комп'ютером.

8.4 Дослідження інформаційного навантаження на програміста

Програміст, у залежності від підготовки і досвіду, вирішує задачі різної складності, але в загальному випадку робота програміста будується по наступному алгоритму.

Таблиця 8.4 – Алгоритм

Етап	Зміст	Витрата часу, %
III	Постановка задачі Вивчення матеріалу за поставленою задачею	6.25
III	Визначення методу рішення задачі	6.25
IV	Складання алгоритму рішення задачі	12.5
V	Програмування	25
VI	Налагодження програми, складання звіту	50

Даний алгоритм відображає загальні дії програміста при рішенні поставленої задачі незалежно від її складності.

Таблиця 8.5 – Алгоритм

Етап	Член алгоритму	Зміст роботи	Літерне позначення
I	1	Одержання першого варіанта технічного	A
	7	Складання і уточнення технічного завдання	Ri
		Одержання остаточного варіанта технічного завдання	Сіл
	4	Складання переліку матеріалів, що існують	HII2
	5	Вивчення матеріалів по тематиці задачі	Аг
	6	Вибір метолу рішення	C2J3
	7	Уточнення й узгодження обраного методу	B2
	8	Остаточний вибір методу рішення	СчT4
	9	Аналіз вхідної і вихідної інформації	B2
	10	Вибір мови програмування	C4TS
	11	Визначення структури програми	H3C5q
	12	Складання блок-схеми програми	C6q2
	13	Логічний аналіз програми і коректування її	F1H4W2
	14	Компіляція програми	F2
	15	Виправлення помилок	D1W2
	16	Редагування програми	F2H5B3W
	17	Виконання програми	F3
	18	Аналіз результатів виконання	H6W5
	19	Тестування	C7W6
	20	Підготовка звіту про роботу	F4

Підрахуємо кількість членів алгоритму і їхню частоту (імовірність) щодо загального числа, прийнятого за одиницю. Імовірність повторення і-ої ситуації визначається по формулі:

$$Pi = к/п, \quad (8.1)$$

де к – кількість повторень кожного елемента одного типу, п – сумарна кількість повторень від джерела інформації, одного типу.

Результати розрахунку зведемо в таблицю.:

Таблиця 8.6 – Результати розрахунку

Джерело	Члени алгоритму	Символ	Кількість	Частота повторень
1	Аферентні – усього		6	1,00
	Вивчення технічної	А	2	0,33
	Спостереження	Р	4	0,67
2	Еферентні – усього		18	1,00
	Уточнення	В	3	0,17
	Вибір найкращого	С	8	0,44
	Виправлення	0	1	0.06
	Аналіз отриманих	н	6	0,33
	Виконання	к	0	0
3	Логічні умови		13	1,00
	Прийняття рішень на осно вивчення	І	5	0,39
	Грабічні матеріали	Ч	2	0.15
	Отриманого тексту	V	6	0.46
	Усього:		37	

Кількісні характеристики алгоритму (Табл.7.6) дозволяють розрахувати інформаційне навантаження програміста. Ентропія інформації елементів кожного джерела інформації розраховується по формулі, біт/сигн:

$$H_j = \sum_{i=1}^{m} p_i \log_2 p_i, \qquad (8.2)$$

де т – число однотипних членів алгоритму розглянутого джерела інформації.

$$H1=2 \times 2+2 \times 4 = 12$$

$$H2 = 3 \times 1,585 +8 \times 3+0+6 \times 2,585 = 44,265$$

$$H3 = 5 \times 2,323 + 2 \times 1+6 + 2,585 = 29,125$$

Потім визначається загальна ентропія інформації, біт/сигн:

$$Hs=Hl+H2+H3, \quad (8.3)$$

де HI, H2, H3 – ентропія аферентних, еферентних елементів і логічних умов відповідно.

Hs = 10 + 44,265 + 29,125 = 83,39. Далі визначається потік інформаційного навантаження біт/хв,

$$F = \frac{H_{\sum} \cdot N}{t}, \quad (8.4)$$

де N – сумарне число всіх членів алгоритму; I – тривалість виконання всієї роботи, хв.

Від кожного джерела в інформації (члена алгоритму) у середньому надходить 3 інформаційних сигнали в годину, час роботи – 225 годин,

$$\Phi = \frac{83,39 \cdot 37 \cdot 3 \cdot 225}{13500} = 2,6 \text{ біт/с}$$

Розраховане інформаційне навантаження порівнюється з припустимою. При необхідності приймається рішення про зміни в трудовому процесі.

Умови нормальної роботи виконуються при дотриманні співвідношення:

$$\Phi_{доп.мин} < \Phi_{расч} < \Phi_{доп.макс}, \quad (8.5)$$

де $\Phi_{доп.мін}$, і $\Phi_{доп.макс}$ мінімальний і максимальний припустимі рівні інформаційних навантажень (0,8 і 3,2 біт/з відповідно); $\Phi_{расч}$ – розрахункове інформаційне навантаження 0,8 < 2,6 <3,2

8.5 Висновки до розділу

У цій частині дипломної проекту були розглянуті вплив факторів трудового і виробничого середовища програмістів, дослідження інформаційного навантаження на програміста. Дотримання умов, що визначають оптимальну організацію робочого місця програміста і навантаження, отримані ним в процесі роботи, дозволить зберегти гарну працездатність протягом усього робочого дня,

підвищить, як у кількісному, так і в якісному відносинах продуктивність праці програміста.

Список використаних джерел інформації

1. Державні будівельні норми України: ДБН В.2.5-28:2018. – *Режим доступу до ресурсу: https://goo.su/9AkQ (дата звернення: 16.10.2023).*

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – *Режим доступу до ресурсу:* https://zakon.rada.gov.ua/rada/show/v0007282-98 *(дата звернення: 16.10.2023).*

3. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – *Режим доступу до ресурсу:* https://zakon.rada.gov.ua/laws/show/z0508 *(дата звернення: 16.10.2023).*

4. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. — 19 с. [Електронний ресурс]. – Режим доступу : http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 *(дата звернення: 16.10.2023).*

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

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 збору, обробки та відображення інформації об’єкту моніторингу у мережі.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем збору, обробки та відображення інформації об’єкту моніторингу у мережі.

– Досліджена система збору, обробки та відображення інформації об’єкту моніторингу у мережі.

– На основі отриманих результатів досліджень створена програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання збору, обробки та відображення інформації об’єкту моніторингу у мережі.

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 Builder C++. 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 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм NTRU.

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 12182 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,4 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тимчук М.М. Дослідження та програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Peter Hoddie, Lizzie Prader «IoT Development for ESP32 and ESP8266 with JavaScript: A Practical Guide to XS and the Moddable SDK» ISBN-13 (pbk): 978-1-4842-5069-3 ISBN-13 (electronic): 978-1-4842-5070-9
3. STM32CubeMX for STM32 configuration and initialization C code generation. User manual. June 2022. 397 p.
4. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.
5. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.
6. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.
7. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.
8. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.
9. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.
10. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises».

Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

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

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

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

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

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

16. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T.**, «**Representation of Cascade Codes in the Frequency Domain**».** In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

31. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-

телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

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

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

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

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

36. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

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

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

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

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

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

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

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

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

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

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

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

48. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

49. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

50. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

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. № 33-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі.

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 збору, обробки та відображення інформації об’єкту моніторингу у мережі;

– 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 10/11 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 10/11.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Builder C++.

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 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинне бути розглянуте дослідження інформаційного навантаження на програміста.

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 – 120 аркушів.

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 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 12.12.2023 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

_____ Якименко Н.М.

Дослідження та програмна реалізація системи збору, обробки та відображення інформації об’єкту моніторингу у мережі

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 68

Літера: РП

Кропивницький – 2023 року

resalloc.cpp – Розподіл ресурсів у системі

```
// системний файл: resalloc.cpp


#include "errno.h"

#include "tsys.h"
#include "resalloc.h"

using namespace WSCADA;

//*****************************************
//* Об'єкт ресурсу                        *
//*****************************************
Res::Res( )
{
#if !__GLIBC_PREREQ(2,4)
    wThr = 0;
#endif
    if(pthread_rwlock_init(&rwc,NULL))
      throw TError("ResAlloc",_("Помилка відкриття семафору!"));
}

Res::~Res( )
{
    pthread_rwlock_wrlock(&rwc);
    pthread_rwlock_destroy(&rwc);
}

void Res::resRequestW( unsigned short tm )
{
    int rez = 0;
#if !__GLIBC_PREREQ(2,4)
    //EDEADLK імітація
    if(wThr && wThr == pthread_self()) rez == EDEADLK;
    else
#endif
    if(!tm) rez = pthread_rwlock_wrlock(&rwc);
    else
    {
      timespec wtm;
      clock_gettime(CLOCK_REALTIME,&wtm);
      wtm.tv_nsec += 1000000*(tm%1000);
      wtm.tv_sec += tm/1000 + wtm.tv_nsec/1000000000; wtm.tv_nsec =
wtm.tv_nsec%1000000000;
      rez = pthread_rwlock_timedwrlock(&rwc,&wtm);
    }
    if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує заблокувати
потік!"));
    else if(tm && rez == ETIMEDOUT) throw TError("ResAlloc",_("Ресурс не
відповідає!"));
#if !__GLIBC_PREREQ(2,4)
    wThr = pthread_self();
#endif
}

bool Res::resTryW( )
{
    int rez = pthread_rwlock_trywrlock(&rwc);
    if(rez == EBUSY) return false;
    else if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує
заблокувати потік!"));
    return true;
}
```

```
void Res::resRequestR( unsigned short tm )
{
    int rez = 0;
#if !__GLIBC_PREREQ(2,4)
    //EDEADLK імітація
    if(wThr && wThr == pthread_self()) rez == EDEADLK;
    else
#endif
    if(!tm) rez = pthread_rwlock_rdlock(&rwc);
    else
    {
      timespec wtm;
      clock_gettime(CLOCK_REALTIME,&wtm);
      wtm.tv_nsec += 1000000*(tm%1000);
      wtm.tv_sec += tm/1000 + wtm.tv_nsec/1000000000; wtm.tv_nsec =
wtm.tv_nsec%1000000000;
      rez = pthread_rwlock_timedrdlock(&rwc,&wtm);
    }
    if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує заблокувати
потік!"));
    else if(tm && rez == ETIMEDOUT) throw TError("ResAlloc",_("Ресурс не
відповідає!"));
}

bool Res::resTryR( )
{
    int rez = pthread_rwlock_tryrdlock(&rwc);
    if(rez == EBUSY) return false;
    else if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує
заблокувати потік!"));
    return true;
}

void Res::resRelease( )
{
    pthread_rwlock_unlock(&rwc);
#if !__GLIBC_PREREQ(2,4)
    if(wThr == pthread_self()) wThr = 0;
#endif
}

//****************************************
//* Автоматичний розподіл ресурсів *
//****************************************
ResAlloc::ResAlloc( Res &rid ) : mId(rid), mAlloc(false)
{

}

ResAlloc::ResAlloc( Res &rid, bool write, unsigned short tm ) : mId(rid),
mAlloc(false)
{
    request(write, tm);
}

ResAlloc::~ResAlloc( )
{
    if(mAlloc) release();
}

void ResAlloc::request( bool write, unsigned short tm )
{
    if(mAlloc) release();
    mAlloc = false;
    try
    {
      if(write) mId.resRequestW(tm);
      else mId.resRequestR(tm);
      mAlloc = true;
```

```
    }catch(TError err) { if(err.cod!=10) throw; }
}

void ResAlloc::release()
{
    if(!mAlloc) return;
    mId.resRelease( );
    mAlloc = false;
}

//*****************************************
//* Рядок + ресурс для                    *
//*****************************************
ResString::ResString( const string &vl )
{
    pthread_mutex_init(&mRes, NULL);
    setVal(vl);
}

ResString::~ResString( )
{
    pthread_mutex_lock(&mRes);
    pthread_mutex_destroy(&mRes);
}

size_t ResString::size( )     { return getVal().size(); }

bool   ResString::empty( )    { return getVal().empty(); }

void ResString::setVal( const string &vl )
{
    pthread_mutex_lock(&mRes);
    str = vl;
    pthread_mutex_unlock(&mRes);
}

string ResString::getVal( )
{
    string rez;
    pthread_mutex_lock(&mRes);
    rez = str;
    pthread_mutex_unlock(&mRes);
    return rez;
}

ResString &ResString::operator=( const string &val )
{
    setVal(val);
    return *this;
}
```

tarchives.cpp – Робота з архівом

```
//WebSCADA системний файл: tarchives.cpp

#include <unistd.h>
#include <getopt.h>
#include <signal.h>
#include <sys/time.h>
#include <string.h>
#include <algorithm>

#include "tsys.h"
#include "tarchives.h"

#define BUF_SIZE_DEF 500
#define BUF_SIZE_MAX 100000

using namespace WSCADA;

//************************************************
//* Підсистема архівування                       *
//************************************************

//************************************************
//* TArchiveS                                    *
//************************************************
TArchiveS::TArchiveS( ) :
    TSubSYS(SARH_ID,"Archives",true), elMess(""), elVal(""), elAval(""),
bufErr(0), mMessPer(10), prcStMess(false),
    headBuf(0), headLstread(0), mValPer(1000), mValPrior(10), prcStVal(false),
endrunReqVal(false)
{
    mAval = grpAdd("va_");

    //> архіватор повідомлення у структурі БД
    elMess.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
    elMess.fldAdd( new TFld("MODUL",_(" Ім’я модуля
(плагіна)"),TFld::String,TCfg::Key,"20") );
    elMess.fldAdd( new
TFld("NAME",_("Ім’я"),TFld::String,TCfg::TransltText,"50") );
    elMess.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elMess.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1") );
    elMess.fldAdd( new TFld("CATEG",_("Категорії
повідомлень"),TFld::String,0,"100") );
    elMess.fldAdd( new TFld("LEVEL",_("Рівні
повідомлень"),TFld::Integer,0,"1","","0;7") );
    elMess.fldAdd( new TFld("ADDR",_("Адреса"),TFld::String,0,"100") );

    //> Значення архіватора у структурі БД
    elVal.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
    elVal.fldAdd( new TFld("MODUL",_("Ім’я модуля
(плагіна)"),TFld::String,TCfg::Key,"20") );
    elVal.fldAdd( new TFld("NAME",_(" Ім’я
"),TFld::String,TCfg::TransltText,"50") );
    elVal.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elVal.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1","0")
);
    elVal.fldAdd( new TFld("ADDR",_("Адреса"),TFld::String,0,"50") );
    elVal.fldAdd( new TFld("V_PER",_("Value period
(sec)"),TFld::Real,0,"12.6","1","0;1000000") );
    elVal.fldAdd( new TFld("A_PER",_("Період архівації
(sec)"),TFld::Integer,0,"4","60","0;1000") );

    //> Значення архіву у структурі БД
    elAval.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
```

```
    elAval.fldAdd( new TFld("NAME",_(" Ім'я
"),TFld::String,TCfg::TransltText,"50") );
    elAval.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elAval.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1","0")
);
    elAval.fldAdd( new TFld("SrcMode",_("Режим джерела"),TFld::Integer,0,"1") );
    elAval.fldAdd( new TFld("Source",_("Джерело"),TFld::String,0,"100") );
    elAval.fldAdd( new TFld("VTYPE",_("Тип значення"),TFld::Integer,0,"1") );
    elAval.fldAdd( new TFld("BPER",_("Період буферізації
(sec)"),TFld::Real,0,"9.6","1","0;10000") );
    elAval.fldAdd( new TFld("BSIZE",_("Розмір
буферу()"),TFld::Integer,0,"6","100","0;1000000") );
    elAval.fldAdd( new TFld("BHGRD",_("Буфер у режимі грід-
системи"),TFld::Boolean,0,"1","1") );
    elAval.fldAdd( new TFld("BHRES",_("Значення буфера останім
часом"),TFld::Boolean,0,"1","0") );
    elAval.fldAdd( new TFld("ArchS",_("Процес
архівування"),TFld::String,0,"500") );

    setMessBufLen( BUF_SIZE_DEF );

    //> Створення повідомлення часу архівування
    struct sigevent sigev;
    memset(&sigev,0,sizeof(sigev));
    sigev.sigev_notify = SIGEV_THREAD;
    sigev.sigev_value.sival_ptr = this;
    sigev.sigev_notify_function = ArhMessTask;
    sigev.sigev_notify_attributes = NULL;
    timer_create(CLOCK_REALTIME,&sigev,&tmIdMess);
}

TArchiveS::~TArchiveS(  )
{
    //> Поввідомлення про закінчення архівування
    timer_delete(tmIdMess);

    //> Перерривання архівування
    if(prcStVal) SYS->taskDestroy(nodePath('.',true)+".vals", &endrunReqVal);

    //> Визволення усіх ресурсів
    nodeDelAll();
}

int TArchiveS::valPeriod( )         { return vmax(1,mValPer); }

void TArchiveS::setValPrior( int ivl )    { mValPrior = vmax(-1,vmin(99,ivl));
modif(); }

void TArchiveS::load_( )
{
    //> Завантажуємо параметри з командного рядка
    int next_opt;
    const char *short_opt="h";
    struct option long_opt[] =
    {
      {"help"    ,0,NULL,'h'},
      {NULL      ,0,NULL,0  }
    };

    optind=0,opterr=0;
    do
    {
      next_opt=getopt_long(SYS->argc,(char * const *)SYS-
>argv,short_opt,long_opt,NULL);
      switch(next_opt)
      {
          case 'h': fprintf(stdout,"%s",optDescr().c_str()); break;
          case -1 : break;
```

```
      }
    } while(next_opt != -1);

    //> Завантажуємо параметри
    setMessBufLen(
atoi(TBDS::genDBGet(nodePath()+"MessBufSize",TSYS::int2str(messBufLen())).c_str(
)) );
    setMessPeriod(
atoi(TBDS::genDBGet(nodePath()+"MessPeriod",TSYS::int2str(mMessPer)).c_str()) );
    setValPeriod(
atoi(TBDS::genDBGet(nodePath()+"ValPeriod",TSYS::int2str(mValPer)).c_str()) );
    setValPrior(
atoi(TBDS::genDBGet(nodePath()+"ValPriority",TSYS::int2str(mValPrior)).c_str())
);

    //> LidDB
    //>> Повідомлення завантаження архіватора
    string id,type;
    map<string, bool>   itReg;
    try
    {
      TConfig c_el(&elMess);
      c_el.cfgViewAll(false);
      vector<string> db_ls;

      //>> Шукаємо у БД і створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_mess_proc",nodePath()+subId()+"_me
ss_proc",fld_cnt++,c_el); )
          {
            id = c_el.cfg("ID").getS();
            type = c_el.cfg("MODUL").getS();
            if(modPresent(type) && !at(type).at().messPresent(id))
                at(type).at().messAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
            itReg[type+"."+id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            vector<string> m_ls;
            modList(m_ls);
            for(unsigned i_m = 0; i_m < m_ls.size(); i_m++)
            {
                at(m_ls[i_m]).at().messList(db_ls);
                for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                    if(itReg.find(m_ls[i_m]+"."+db_ls[i_it]) == itReg.end() &&
SYS->chkSelDB(at(m_ls[i_m]).at().messAt(db_ls[i_it]).at().DB()))
                        at(m_ls[i_m]).at().messDel(db_ls[i_it]);
            }
        }
    }catch( TError err )
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Повідомлення вомилки завантаження
архіватора."));
    }

    //>> Завантажуємо значення архіватора
    try
    {
      TConfig c_el(&elVal);
      c_el.cfgViewAll(false);
      vector<string> db_ls;
      itReg.clear();
```

```
      //>> Шукаємо у БД та створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_val_proc",nodePath()+subId()+"_val
_proc",fld_cnt++,c_el); )
          {
            id = c_el.cfg("ID").getS();
            type = c_el.cfg("MODUL").getS();
            if(modPresent(type) && !at(type).at().valPresent(id))
                at(type).at().valAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
            itReg[type+"."+id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            vector<string> m_ls;
            modList(m_ls);
            for(unsigned i_m = 0; i_m < m_ls.size(); i_m++)
            {
                at(m_ls[i_m]).at().valList(db_ls);
                for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                    if(itReg.find(m_ls[i_m]+"."+db_ls[i_it]) == itReg.end() &&
SYS->chkSelDB(at(m_ls[i_m]).at().valAt(db_ls[i_it]).at().DB()))
                        at(m_ls[i_m]).at().valDel(db_ls[i_it]);
            }
        }
    }catch( TError err )
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Помилка завантаження значення
архіватора."));
    }

    //>> Завантажуємо значення архіватора
    try
    {
      TConfig c_el(&elAval);
      c_el.cfgViewAll(false);
      vector<string> db_ls;
      itReg.clear();

      //>> Шукаємо у БД та створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_val",nodePath()+subId()+"_val",fld
_cnt++,c_el); )
          {
              id = c_el.cfg("ID").getS();
              if(!valPresent(id)) valAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
              itReg[id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            valList(db_ls);
            for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                if(itReg.find(db_ls[i_it]) == itReg.end() && SYS-
>chkSelDB(valAt(db_ls[i_it]).at().DB()))
                    valDel(db_ls[i_it]);
        }
```

```
    }catch(TError err)
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Помилка завантаження значення
архіватора."));
    }
}

void TArchiveS::save_( )
{
    vector<string> t_lst, o_lst;

    //> Зберігаємо параметри
    TBDS::genDBSet(nodePath()+"MessBufSize",TSYS::int2str(messBufLen()));
    TBDS::genDBSet(nodePath()+"MessPeriod",TSYS::int2str(messPeriod()));
    TBDS::genDBSet(nodePath()+"ValPeriod",TSYS::int2str(valPeriod()));
    TBDS::genDBSet(nodePath()+"ValPriority",TSYS::int2str(valPrior()));
}

void TArchiveS::valAdd( const string &iid, const string &idb )
{
    if( valPresent(iid) ) return;
    chldAdd(mAval,new TVArchive(iid,idb,&aValE()));
}

string TArchiveS::optDescr(  )
{
    char buf[STR_BUF_LEN];
    snprintf(buf,sizeof(buf),_(
      "======================== Підсистема \"Архів\" Опції
==================\n"
      "------------ Параметри частини '%s' у кофігураційний файл -----------\n"
      "MessBufSize   <items>       Повідомлення розміру буферу.\n"
      "MessPeriod    <sec>         Повідомлення періоду архівування.\n"
      "ValPeriod     <msec>        Значення періоду архівування.\n"
      "ValPriority   <level>      Значення завдання пріорітетного рівня.\n"
      "MaxReqMess    <items>       Повідомлення максимального запиту.\n"
      "MaxReqVals    <items>       Значення  максимального запиту.\n\n"
      ),nodePath().c_str());

    return buf;
}

void TArchiveS::subStart( )
{
    mess_info(nodePath().c_str(),_("Запускаємо підсистеми."));

    SubStarting = true;

    vector<string> t_lst, o_lst;

    modList(t_lst);
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      AutoHD<TTipArchivator> mod = modAt(t_lst[i_t]);

      //> Повідомлення про початок роботи архіватора
      mod.at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TMArchivator> mess = mod.at().messAt(o_lst[i_o]);
          if( /*!mess.at().startStat() &&*/ mess.at().toStart() )
            try{ mess.at().start(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Повідомлення про помилку роботи
архіватора."),o_lst[i_o].c_str());
            }
```

```
      }
      //> Значення початку роботи архіватора
      mod.at().valList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TVArchivator> val = mod.at().valAt(o_lst[i_o]);
          if( /*!val.at().startStat() &&*/ val.at().toStart() )
            try{ val.at().start(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Помилка початку роботи
архіватора."),val.at().workId().c_str());
            }
      }
    }

    //> Значення початку роботи архіватора
    valList(o_lst);
    for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
    {
      AutoHD<TVArchive> aval = valAt(o_lst[i_o]);
      if( /*!aval.at().startStat() &&*/ aval.at().toStart() )
          try{ aval.at().start(); }
          catch(TError err)
          {
            mess_err(err.cat.c_str(),"%s",err.mess.c_str());
            mess_err(nodePath().c_str(),_("Помилка початку роботи
архіватора"),o_lst[i_o].c_str());
          }
    }

    //> Повідомлення про початок роботи інтервального таймера
    struct itimerspec itval;
    itval.it_interval.tv_sec = itval.it_value.tv_sec = messPeriod();
    itval.it_interval.tv_nsec = itval.it_value.tv_nsec = 0;
    timer_settime(tmIdMess, 0, &itval, NULL);

    //> Значення початку роботи завдань
    if(!prcStVal) SYS->taskCreate(nodePath('.',true)+".vals", valPrior(),
TArchiveS::ArhValTask, this);

    TSubSYS::subStart( );

    SubStarting = false;
}

void TArchiveS::subStop( )
{
    mess_info(nodePath().c_str(),_("Зупинка підсистеми."));

    TSubSYS::subStop( );

    vector<string> t_lst, o_lst;

    //> Зупинка інтервального таймера для періодичних потоків, для створення
повідомлення архівації структур;
    itval.it_interval.tv_sec = itval.it_interval.tv_nsec =
      itval.it_value.tv_sec = itval.it_value.tv_nsec = 0;
    timer_settime(tmIdMess, 0, &itval, NULL);
    if(TSYS::eventWait( prcStMess, false, nodePath()+"mess_stop",10))
      throw TError(nodePath().c_str(),_("Архівація повідомлень потоків не може
бути зупинена!"));

    //> Значення зупинки роботи завдань
    if(prcStVal) SYS->taskDestroy(nodePath('.',true)+".vals", &endrunReqVal);

    //> Виклик останього повідомлення архіватора
    sigval obj; obj.sival_ptr = this;
```

```
    ArhMessTask(obj);

    //> Зупинка архіватора
    modList(t_lst);
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      AutoHD<TTipArchivator> mod = modAt(t_lst[i_t]);
      //Значення зупинки архіватора
      mod.at().valList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TVArchivator> val = mod.at().valAt(o_lst[i_o]);
          if( val.at().startStat() )
            try{ val.at().stop(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Значення архіватора '%s' stop
error."),o_lst[i_o].c_str());
            }
      }
      // Повідомлення про зупинку архіватора
      mod.at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TMArchivator> mess = mod.at().messAt(o_lst[i_o]);
          if( mess.at().startStat() )
            try{ mess.at().stop(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Повідомлення про помилку зупинки
архіватора"),o_lst[i_o].c_str());
            }
      }
    }

    //> Значення зупинки архіватора
    valList(o_lst);
    for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
    {
      AutoHD<TVArchive> aval = valAt(o_lst[i_o]);
      if( aval.at().startStat() )
          try{ aval.at().stop(); }
          catch(TError err)
          {
            mess_err(err.cat.c_str(),"%s",err.mess.c_str());
            mess_err(nodePath().c_str(),_("Значення помилки зупинки
архіватора"),o_lst[i_o].c_str());
          }
    }
}

void TArchiveS::messPut( time_t tm, int utm, const string &categ, int8_t level,
const string &mess )
{
    //> Відправляємо повідомлення у буфер
    ResAlloc res(mRes,true);
    mBuf[headBuf].time  = tm;
    mBuf[headBuf].utime = utm;
    mBuf[headBuf].categ = categ;
    mBuf[headBuf].level = (TMess::Type)abs(level);
    mBuf[headBuf].mess  = mess;
    if( ++headBuf >= mBuf.size() ) headBuf = 0;
    //> Перервіряємо, чи це не повідомлення архіватора
    if( headBuf == headLstread )
    {
      if( !(bufErr&0x01) )
      {
```

```
            bufErr |= 0x01;
            res.release();
            mess_err(nodePath().c_str(),_("Буфер заповнений. Останнє
повідомлення!"));
            res.request(true);
        }
        if( ++headLstread >= mBuf.size() ) headLstread = 0;
    }
    //> Перевіряємо швидкість заповнення буфера.
    else if( headBuf-headLstread > messBufLen( )/2 )
    {
        if( !(bufErr&0x02) )
        {
            bufErr |= 0x02;
            res.release();
            mess_warning(nodePath().c_str(),_("Повідомлення про т, що швидкість
заповнення буфера дуже висока!"));
            res.request(true);
        }
    }
    else bufErr = 0;

    //> Обробка тривог. Для рівня менше 0 тривоги встановлено
    map<string,TMess::SRec>::iterator p;
    if( level < 0 ) mAlarms[categ] =
TMess::SRec(tm,utm,categ,(TMess::Type)abs(level),mess);
    else if( (p=mAlarms.find(categ)) != mAlarms.end() ) mAlarms.erase(p);
}

void TArchiveS::messPut( const vector<TMess::SRec> &recs )
{
    for(unsigned i_r = 0; i_r < recs.size(); i_r++)
      messPut(recs[i_r].time,recs[i_r].utime,recs[i_r].categ,recs[i_r].level,rec
s[i_r].mess);
}

void TArchiveS::messGet( time_t b_tm, time_t e_tm, vector<TMess::SRec> & recs,
const string &category, int8_t level, const string &arch, time_t upTo )
{
    recs.clear();

    ResAlloc res(mRes,false);
    if(!upTo) upTo = time(NULL)+STD_INTERF_TM;
    TRegExp re(category, "p");

    //> Отримуємо записи з буфера
    unsigned i_buf = headBuf;
    while(level >= 0 && (!arch.size() || arch==BUF_ARCH_NM) && time(NULL) <
upTo)
    {
      if(mBuf[i_buf].time >= b_tm && mBuf[i_buf].time != 0 && mBuf[i_buf].time
<= e_tm &&
           mBuf[i_buf].level >= level && re.test(mBuf[i_buf].categ))
          recs.push_back(mBuf[i_buf]);
      if(++i_buf >= mBuf.size()) i_buf = 0;
      if(i_buf == headBuf) break;
    }

    //> Отримуємо записи з архівів
    vector<string> t_lst, o_lst;
    modList(t_lst);
    for(unsigned i_t = 0; level >= 0 && i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size() && time(NULL) < upTo; i_o++)
      {
          AutoHD<TMArchivator> archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
```

```
            archtor.at().get(b_tm,e_tm,recs,category,level);
      }
    }

    //> Запит процесу тривоги
    if(level < 0)
    {
      vector< pair<int64_t,TMess::SRec* > > mb;
      for(map<string,TMess::SRec>::iterator p = mAlarms.begin(); p !=
mAlarms.end() && time(NULL) < upTo; p++)
          if((p->second.time >= b_tm || b_tm == e_tm) && p->second.time <= e_tm
&&
              p->second.level >= abs(level) && re.test(p->second.categ))
            mb.push_back(pair<int64_t,TMess::SRec* >(FTM(p->second),&p-
>second));
      sort(mb.begin(),mb.end());
      for(unsigned i_b = 0; i_b < mb.size(); i_b++)
recs.push_back(*mb[i_b].second);
    }
}

time_t TArchiveS::messBeg( const string &arch )
{
    time_t rez = 0;
    ResAlloc res(mRes,false);
    if(arch.empty() || arch == BUF_ARCH_NM)
    {
      unsigned i_buf = headBuf;
      while(!arch.size() || arch == BUF_ARCH_NM)
      {
          rez = rez ? vmin(rez,mBuf[i_buf].time) : mBuf[i_buf].time;
          if(++i_buf >= mBuf.size()) i_buf = 0;
          if(i_buf == headBuf) break;
      }
      if( !arch.empty() ) return rez;
    }

    //- Отримуємо записи з архівів -
    vector<string> t_lst, o_lst;
    modList(t_lst);
    AutoHD<TMArchivator> archtor;
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
            rez = rez ? vmin(rez,archtor.at().begin()) : archtor.at().begin();
      }
    }

    return rez;
}

time_t TArchiveS::messEnd( const string &arch )
{
    time_t rez = 0;
    ResAlloc res(mRes,false);
    if(arch.empty() || arch == BUF_ARCH_NM)
    {
      unsigned i_buf = headBuf;
      while(!arch.size() || arch == BUF_ARCH_NM)
      {
          rez = rez ? vmax(rez,mBuf[i_buf].time) : mBuf[i_buf].time;
          if(++i_buf >= mBuf.size()) i_buf = 0;
          if(i_buf == headBuf) break;
      }
```

```
      if(!arch.empty()) return rez;
    }

    //> Отримуємо записи з архівів
    vector<string> t_lst, o_lst;
    modList(t_lst);
    AutoHD<TMArchivator> archtor;
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
            rez = rez ? vmax(rez,archtor.at().end()) : archtor.at().end();
      }
    }

    return rez;
}

void TArchiveS::setMessBufLen(unsigned len)
{
    ResAlloc res(mRes,true);
    len = vmin(BUF_SIZE_MAX,vmax(BUF_SIZE_DEF,len));
    while(mBuf.size() > len)
    {
      mBuf.erase(mBuf.begin() + headBuf);
      if(headBuf >= mBuf.size()) headBuf = 0;
      if(headLstread >= mBuf.size())     headLstread = mBuf.size()-1;
    }
    while(mBuf.size() < len) mBuf.insert(mBuf.begin() + headBuf, TMess::SRec());
    modif();
}

void TArchiveS::setActValArch( const string &id, bool val )
{
    unsigned i_arch;

    ResAlloc res(vRes,true);
    for( i_arch = 0; i_arch < actUpSrc.size(); i_arch++ )
      if( actUpSrc[i_arch].at().id() == id ) break;

    if( val && i_arch >= actUpSrc.size() )
      actUpSrc.push_back(valAt(id));
    if( !val && i_arch < actUpSrc.size() )
      actUpSrc.erase(actUpSrc.begin()+i_arch);
}

void TArchiveS::setMessPeriod( int ivl )
{
    mMessPer = ivl;
    modif();

    if( subStartStat( ) )
    {
      struct itimerspec itval;
      itval.it_interval.tv_sec = itval.it_value.tv_sec = mMessPer;
      itval.it_interval.tv_nsec = itval.it_value.tv_nsec = 0;
      timer_settime(tmIdMess, 0, &itval, NULL);
    }
}

void TArchiveS::ArhMessTask( union sigval obj )
{
    TArchiveS &arh = *(TArchiveS *)obj.sival_ptr;
    if( arh.prcStMess )  return;
    arh.prcStMess = true;
```

```
    //> Читаємо повідомлення з буфера
    if( arh.headLstread != arh.headBuf )
      try
      {
          ResAlloc res(arh.mRes,false);

          //>> Беремо нове повідомлення
          unsigned new_headLstread = arh.headBuf;
          unsigned i_m = arh.headLstread;
          vector<TMess::SRec> o_mess;
          while( i_m != new_headLstread )
          {
            o_mess.push_back(arh.mBuf[i_m]);
            if( ++i_m >= arh.mBuf.size() ) i_m = 0;
          }
          arh.headLstread = new_headLstread;

          res.release();

          //>> Архівуємо
          vector<string> t_lst, o_lst;
          arh.modList(t_lst);
          for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
          {
            arh.at(t_lst[i_t]).at().messList(o_lst);
            for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
                if(arh.at(t_lst[i_t]).at().messAt(o_lst[i_o]).at().startStat())
                  arh.at(t_lst[i_t]).at().messAt(o_lst[i_o]).at().put(o_mess);
          }
      }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(arh.nodePath().c_str(),_("Помилка читання повідомлення з
буфера."));
      }

    arh.prcStMess = false;
}

void *TArchiveS::ArhValTask( void *param )
{
    TArchiveS &arh = *(TArchiveS *)param;
    arh.endrunReqVal = false;
    arh.prcStVal = true;

    while( !arh.endrunReqVal )
    {
      int64_t work_tm = SYS->curTime();

      arh.vRes.resRequestR( );
      for(unsigned i_arh = 0; i_arh < arh.actUpSrc.size(); i_arh++)
          try
          {
            if( work_tm/arh.actUpSrc[i_arh].at().period() >
arh.actUpSrc[i_arh].at().end()/arh.actUpSrc[i_arh].at().period() )
                arh.actUpSrc[i_arh].at().getActiveData();
          }
          catch(TError err)
          { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }
      arh.vRes.resRelease( );

      TSYS::taskSleep((int64_t)arh.valPeriod()*1000000);
    }

    arh.prcStVal = false;

    return NULL;
```

```
}

TVariant TArchiveS::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    // Array messGet(int btm, int etm, string cat = "", int lev = 0, string arch
= ""); - запит системних повідомлень, для часу з <btm>
    //       до <etm> для категорії <cat>, рівня <lev> та архіву <arch>
    //  btm - час початку
    //  etm - час закінчення
    //  cat - категорія повідомлення
    //  lev - рівні повідомлень
    //  arch - архівація повідомлення
    if( iid == "messGet" && prms.size() >= 2 )
    {
      vector<TMess::SRec> recs;
      messGet( prms[0].getI(), prms[1].getI(), recs, ((prms.size()>=3) ?
prms[2].getS() : string("")),
          ((prms.size()>=4) ? prms[3].getI() : 0), ((prms.size()>=5) ?
prms[4].getS() : string("")) );
      TArrayObj *rez = new TArrayObj();
      for(unsigned i_m = 0; i_m < recs.size(); i_m++)
      {
          TVarObj *am = new TVarObj();
          am->propSet("tm",(int)recs[i_m].time);
          am->propSet("utm",recs[i_m].utime);
          am->propSet("categ",recs[i_m].categ);
          am->propSet("level",recs[i_m].level);
          am->propSet("mess",recs[i_m].mess);
          rez->propSet(TSYS::int2str(i_m),am);
      }
      return rez;
    }

    return TCntrNode::objFuncCall(iid,prms,user);
}

void TArchiveS::cntrCmdProc( XMLNode *opt )
{
    string a_path = opt->attr("path");
    //> Сервіс управління процесами
    if(a_path == "/serv/mess")            //Повідомлення доступу
    {
      if(ctrChkNode(opt,"info",RWRWRW,"root",SARH_ID,SEC_RD))
      //Інформаційні повідомлення
      {
          string arch = opt->attr("arch");
          opt->setAttr("end",TSYS::uint2str(messEnd(arch)));
          opt->setAttr("beg",TSYS::uint2str(messBeg(arch)));
      }
      else if(ctrChkNode(opt,"get",RWRWRW,"root",SARH_ID,SEC_RD))//Запит
значення дати
      {
          time_t tm      = strtoul(opt->attr("tm").c_str(),0,10);
          time_t tm_grnd = strtoul(opt->attr("tm_grnd").c_str(),0,10);
          string arch    = opt->attr("arch");
          string cat     = opt->attr("cat");
          int    lev     = atoi(opt->attr("lev").c_str());
          vector<TMess::SRec> rez;
          messGet( tm_grnd, tm, rez, cat, (TMess::Type)lev, arch );
          for(unsigned i_r = 0; i_r < rez.size(); i_r++)
            opt->childAdd("el")->
                setAttr("time",TSYS::uint2str(rez[i_r].time))->
                setAttr("utime",TSYS::uint2str(rez[i_r].utime))->
                setAttr("cat",rez[i_r].categ)->
                setAttr("lev",TSYS::int2str(rez[i_r].level))->
                setText(rez[i_r].mess);
      }
      return;
```

```
    }

    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TSubSYS::cntrCmdProc(opt);
      ctrMkNode("grp",opt,-1,"/br/va_",_("Архів
значень"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      if(ctrMkNode("area",opt,1,"/m_arch",_("Архів
повідомлень"),R_R_R_,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-1,"/m_arch/size",_("Повідомлення розміру
буферу"),RWRWR_,"root",SARH_ID,2,"tp","dec","min",TSYS::int2str(BUF_SIZE_DEF).c_
str());
          ctrMkNode("fld",opt,-1,"/m_arch/per",_("Період архівування
(s)"),RWRWR_,"root",SARH_ID,1,"tp","dec");
          if(ctrMkNode("area",opt,-1,"/m_arch/view",_("Перегляд
повідомлень"),R_R___,"root",SARH_ID))
          {
            ctrMkNode("fld",opt,-
1,"/m_arch/view/tm",_("Time"),RWRW__,"root",SARH_ID,1,"tp","time");
            ctrMkNode("fld",opt,-1,"/m_arch/view/size",_("Розмір
(s)"),RWRW__,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("fld",opt,-1,"/m_arch/view/cat",_("Зразок
категорії"),RWRW__,"root",SARH_ID,2,"tp","str","help",
                _("Шаблон категорії повідомлень або регулярне вираження.\n"
                    "Використовуйте тимчасові символи для групового
виділення:\n  '*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                  "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
            ctrMkNode("fld",opt,-
1,"/m_arch/view/lvl",_("Рівень"),RWRW__,"root",SARH_ID,4,"tp","dec","min","-
7","max","7",
                "help",_("Отримуємо повідомлення для рівня більшого і
дорівнюючого цьому."));
            ctrMkNode("fld",opt,-
1,"/m_arch/view/archtor",_("Архіватор"),RWRW__,"root",SARH_ID,4,"tp","str","dest
","select","select","/m_arch/lstAMess",
                "help",_("Архівація повідомлення.\n Не встановлюємо архіватор
для процесів у буфері та інших архіваторів.\Встановлюємо '<buffer>' для процесів
у буфері ."));
            if(ctrMkNode("table",opt,-
1,"/m_arch/view/mess",_("Messages"),R_R___,"root",SARH_ID))
            {
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0",_("Time"),R_R___,"root",SARH_ID,1,"tp","time");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0a",_("mcsec"),R_R___,"root",SARH_ID,1,"tp","dec");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/1",_("Category"),R_R___,"root",SARH_ID,1,"tp","str");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/2",_("Lev."),R_R___,"root",SARH_ID,1,"tp","dec");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/3",_("Message"),R_R___,"root",SARH_ID,1,"tp","str");
            }
          }
      }
      if(ctrMkNode("area",opt,2,"/v_arch",_("Архів
значеньь"),R_R_R_,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-1,"/v_arch/per",_("Беремо дані періоду
(ms)"),RWRWR_,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-1,"/v_arch/prior",_("Беремо дані завдання
пріорітетного рівня "),RWRWR_,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-
1,"/v_arch/nmb",_("Number"),R_R_R_,"root",SARH_ID,1,"tp","str");
          ctrMkNode("list",opt,-1,"/v_arch/archs",_("Архів
значень"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_pref
","va_","idSz","20");
```

```
      }
      ctrMkNode("fld",opt,-1,"/help/g_help",_("Опції
допомоги"),R_R___,"root",SARH_ID,3,"tp","str","cols","90","rows","10");
      return;
    }

    //> Процес управління на сторінці
    if(a_path == "/m_arch/per")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(messPeriod()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setMessPeriod(atoi(opt->text().c_str()));
    }
    else if(a_path == "/m_arch/size")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(messBufLen()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setMessBufLen(atoi(opt->text().c_str()));
    }
    else if(a_path == "/m_arch/view/tm")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))
      {
          opt->setText(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")));
          if( !atoi(opt->text().c_str()) )    opt-
>setText(TSYS::int2str(time(NULL)));
      }
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
          TBDS::genDBSet(nodePath()+"messTm",(atoi(opt-
>text().c_str())>=time(NULL))?"0":opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/size")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messSize","60",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messSize",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/cat")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messCat",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/lvl")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messLev","0",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messLev",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/archtor")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messArch","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messArch",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/lstAMess" && ctrChkNode(opt,"get",R_R___))
    {
      opt->childAdd("el")->setText("");
      opt->childAdd("el")->setText(BUF_ARCH_NM);
      vector<string> lsm, lsa;
      modList(lsm);
      for(unsigned i_m = 0; i_m < lsm.size(); i_m++)
```

```
      {
          at(lsm[i_m]).at().messList(lsa);
          for(unsigned i_a = 0; i_a < lsa.size(); i_a++)
            opt->childAdd("el")->setText(lsm[i_m]+"."+lsa[i_a]);
      }
    }
    else if(a_path == "/m_arch/view/mess" &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))
    {
      vector<TMess::SRec> rec;
      time_t gtm = atoi(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")).c_str());
      if( !gtm ) gtm = time(NULL);
      int gsz = atoi(TBDS::genDBGet(nodePath()+"messSize","60",opt-
>attr("user")).c_str());
      messGet( gtm-gsz, gtm, rec,
             TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")),
             atoi(TBDS::genDBGet(nodePath()+"messLev","0",opt-
>attr("user")).c_str()),
             TBDS::genDBGet(nodePath()+"messArch","",opt->attr("user")) );

      XMLNode *n_tm     = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0","",R_R___,"root",SARH_ID);
      XMLNode *n_tmu    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0a","",R_R___,"root",SARH_ID);
      XMLNode *n_cat    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/1","",R_R___,"root",SARH_ID);
      XMLNode *n_lvl    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/2","",R_R___,"root",SARH_ID);
      XMLNode *n_mess   = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/3","",R_R___,"root",SARH_ID);
      for(int i_rec = rec.size()-1; i_rec >= 0; i_rec--)
      {
          if(n_tm)      n_tm->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].time));
          if(n_tmu)     n_tmu->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].utime));
          if(n_cat)     n_cat->childAdd("el")->setText(rec[i_rec].categ);
          if(n_lvl)     n_lvl->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].level));
          if(n_mess)    n_mess->childAdd("el")->setText(rec[i_rec].mess);
      }
    }
    else if(a_path == "/v_arch/per")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(valPeriod()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setValPeriod(atoi(opt->text().c_str()));
    }
    else if(a_path == "/v_arch/prior")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(valPrior()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setValPrior(atoi(opt->text().c_str()));
    }
    else if(a_path == "/v_arch/nmb" && ctrChkNode(opt))
    {
      vector<string> list;
      valList(list);
      unsigned e_c = 0;
      for(unsigned i_a = 0; i_a < list.size(); i_a++)
          if(valAt(list[i_a]).at().startStat()) e_c++;
      opt->setText(TSYS::strMess(_("All: %d; Enabled: %d"),list.size(),e_c));
    }
    else if(a_path == "/br/va_" || a_path == "/v_arch/archs")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
```

```
      {
          vector<string> list;
          valList(list);
          for(unsigned i_a=0; i_a < list.size(); i_a++)
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(valAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          valAdd(vid); valAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))
      chldDel(mAval,opt->attr("id"),-1,1);
    }
    else if(a_path == "/help/g_help" &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))    opt->setText(optDescr());
    else TSubSYS::cntrCmdProc(opt);
}

//************************************************
//* TTipArchivator                               *
//************************************************
TTipArchivator::TTipArchivator( const string &id ) : TModule(id)
{
    mVal = grpAdd("val_");
    mMess = grpAdd("mess_");
}

TTipArchivator::~TTipArchivator()
{
    nodeDelAll();
}

TArchiveS &TTipArchivator::owner()
{
    return (TArchiveS &)TModule::owner();
}

void TTipArchivator::messAdd(const string &name, const string &idb )
{
    chldAdd(mMess, AMess(name,idb));
}

void TTipArchivator::valAdd( const string &iid, const string &idb )
{
    chldAdd(mVal, AVal(iid,idb));
}

void TTipArchivator::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TModule::cntrCmdProc(opt);
      ctrMkNode("area",opt,0,"/arch",_("Архіватори"));
      ctrMkNode("grp",opt,-1,"/br/mess_",_("Повідомлення
архіватора"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      ctrMkNode("grp",opt,-1,"/br/val_",_("Значення
архіватора"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      ctrMkNode("list",opt,-1,"/arch/mess",_("Повідомлення
архіваторів"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_
pref","mess_","idSz","20");
      ctrMkNode("list",opt,-1,"/arch/val",_("Значення
архіваторів"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_
pref","val_","idSz","20");
      return;
    }
    //> Процес управління на сторінці
```

```
    string a_path = opt->attr("path");
    if(a_path == "/br/mess_" || a_path == "/arch/mess")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
      {
          vector<string> list;
          messList(list);
          for( unsigned i_a=0; i_a < list.size(); i_a++ )
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(messAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          messAdd(vid); messAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))     messDel(opt-
>attr("id"),true);
    }
    else if(a_path == "/br/val_" || a_path == "/arch/val")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
      {
          vector<string> list;
          valList(list);
          for(unsigned i_a=0; i_a < list.size(); i_a++)
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(valAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          valAdd(vid); valAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))     valDel(opt-
>attr("id"),true);
    }
    else TModule::cntrCmdProc(opt);
}

//************************************************
//* Повідомлення архіватора                      *
//************************************************

//************************************************
//* TMArchivator                                 *
//************************************************
TMArchivator::TMArchivator(const string &iid, const string &idb, TElem *cf_el) :
    TConfig( cf_el ), run_st(false),
    m_id(cfg("ID").getSd()), m_name(cfg("NAME").getSd()),
m_dscr(cfg("DESCR").getSd()), m_addr(cfg("ADDR").getSd()),
    m_cat_o(cfg("CATEG").getSd()), m_start(cfg("START").getBd()),
m_level(cfg("LEVEL").getId()), m_db(idb)
{
    m_id = iid;
}

TCntrNode &TMArchivator::operator=( TCntrNode &node )
{
    TMArchivator *src_n = dynamic_cast<TMArchivator*>(&node);
    if( !src_n ) return *this;

    //> Конфігурація копіювання
    string tid = id();
    *(TConfig*)this = *(TConfig*)src_n;
    cfg("MODUL").setS(owner().modId());
    m_id = tid;
    m_db = src_n->m_db;
```

```
    if( src_n->startStat() && toStart() && !startStat() )
        start( );

    return *this;
}

void TMArchivator::postEnable( int flag )
{
    cfg("MODUL").setS(owner().modId());
}

void TMArchivator::preDisable( int flag )
{
    if( startStat() )   stop( );
}

void TMArchivator::postDisable(int flag)
{
    try
    {
      if( flag )
          SYS->db().at().dataDel(fullDB(),SYS-
>archive().at().nodePath()+tbl(),*this,true);
    }catch(TError err)
    { mess_warning(err.cat.c_str(),"%s",err.mess.c_str()); }
}

TTipArchivator &TMArchivator::owner( )    { return *(TTipArchivator*)nodePrev();
}

string TMArchivator::workId( )          { return owner().modId()+"."+id(); }

string TMArchivator::name()         { return (m_name.size())?m_name:m_id; }

string TMArchivator::tbl( )         { return
owner().owner().subId()+"_mess_proc"; }

void TMArchivator::load_( )
{
    if( !SYS->chkSelDB(DB()) ) return;
    SYS->db().at().dataGet(fullDB(),SYS->archive().at().nodePath()+tbl(),*this);
}

void TMArchivator::save_( )
{
    SYS->db().at().dataSet(fullDB(),SYS->archive().at().nodePath()+tbl(),*this);
}

void TMArchivator::categ( vector<string> &list )
{
    list.clear();
    string c_vl;
    for( int i_off = 0; (c_vl=TSYS::strSepParse(m_cat_o,0,';',&i_off)).size(); )
      list.push_back(c_vl);
}

bool TMArchivator::chkMessOK( const string &icateg, TMess::Type ilvl )
{
    vector<string> cat_ls;

    categ(cat_ls);

    if(ilvl >= level())
      for(unsigned i_cat = 0; i_cat < cat_ls.size(); i_cat++)
          if(TRegExp(cat_ls[i_cat], "p").test(icateg))
            return true;
   return false;
}
```

```
TVariant TMArchivator::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    // bool status() - беремо статус архівування.
    if(iid == "status") return startStat();
    // int end() - беремо дані  архівування, час закінчення.
    if(iid == "end") return (int)end();
    // int begin() - беремо дані  архівування, час початку.
    if(iid == "begin") return (int)begin();

    //> Виклик функцій конфігурації
    TVariant cfRez = objFunc(iid, prms, user);
    if(!cfRez.isNull()) return cfRez;

    return TCntrNode::objFuncCall(iid, prms, user);
}

void TMArchivator::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      ctrMkNode("WSCADA_cntr",opt,-1,"/",_("Повідомлення архіватора:
")+name(),RWRWR_,"root",SARH_ID);
      if(ctrMkNode("area",opt,-1,"/prm",_("Архіватор")))
      {
          if(ctrMkNode("area",opt,-1,"/prm/st",_("Стан")))
          {
            ctrMkNode("fld",opt,-
1,"/prm/st/st",_("Running"),RWRWR_,"root",SARH_ID,1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/prm/st/db",_("Архіватор
БД"),RWRWR_,"root","root",4,

"tp","str","dest","select","select","/db/list","help",TMess::labDB());
            ctrMkNode("fld",opt,-
1,"/prm/st/end",_("End"),R_R_R_,"root","root",1,"tp","time");
            ctrMkNode("fld",opt,-
1,"/prm/st/beg",_("Begin"),R_R_R_,"root","root",1,"tp","time");
          }
          if(ctrMkNode("area",opt,-1,"/prm/cfg",_("Конфігурація")))
          {
            ctrMkNode("fld",opt,-
1,"/prm/cfg/id",cfg("ID").fld().descr(),R_R_R_,"root",SARH_ID,1,"tp","str");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/nm",cfg("NAME").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","str","le
n","50");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/dscr",cfg("DESCR").fld().descr(),RWRWR_,"root",SARH_ID,3,"tp","str",
"cols","90","rows","3");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/addr",cfg("ADDR").fld().descr(),RWRWR_,"root",SARH_ID,1,"tp","str");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/lvl",cfg("LEVEL").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","dec",
                "help",_("Отримуємо повідомлення для рівня більшого і
дорівнюючого цьому."));
            ctrMkNode("fld",opt,-
1,"/prm/cfg/cats",cfg("CATEG").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","str",
                "help",_("Шаблон категорії повідомлень або регулярне вираження у
процесі архівування, відокремлюються символом';'.\n"
                  "Використовуйте тимчасові символи для групового
виділення:\n  '*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                  "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
            ctrMkNode("fld",opt,-1,"/prm/cfg/start",_("To
start"),RWRWR_,"root",SARH_ID,1,"tp","bool");
          }
      }
```

```
      if(run_st && ctrMkNode("area",opt,-
1,"/mess",_("Messages"),R_R___,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-
1,"/mess/tm",_("Time"),RWRW__,"root",SARH_ID,1,"tp","time");
          ctrMkNode("fld",opt,-1,"/mess/size",_("Розмір
(s)"),RWRW__,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-1,"/mess/cat",_("Зразок
категорії"),RWRW__,"root",SARH_ID,2,"tp","str","help",
            _("Шаблон категорії повідомлень або регулярне вираження.\n"
                  "Використовуйте тимчасові символи для групового виділення:\n
'*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
          ctrMkNode("fld",opt,-
1,"/mess/lvl",_("Level"),RWRW__,"root",SARH_ID,4,"tp","dec","min","0","max","7",
            "help",_("Отримуємо повідомлення для рівня більшого і дорівнюючого
цьому."));
          if(ctrMkNode("table",opt,-
1,"/mess/mess",_("Messages"),R_R___,"root",SARH_ID))
          {
            ctrMkNode("list",opt,-
1,"/mess/mess/0",_("Час"),R_R___,"root",SARH_ID,1,"tp","time");
            ctrMkNode("list",opt,-
1,"/mess/mess/0a",_("mcsec"),R_R___,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/mess/mess/1",_("Категорія"),R_R___,"root",SARH_ID,1,"tp","str");
            ctrMkNode("list",opt,-
1,"/mess/mess/2",_("Рівень"),R_R___,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/mess/mess/3",_("Повідомлення"),R_R___,"root",SARH_ID,1,"tp","str");
          }
      }
      return;
    }
    //> Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/prm/st/st")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
startStat() ? "1" : "0" );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     atoi(opt-
>text().c_str()) ? start() : stop();
    }
    else if(a_path == "/prm/st/db")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
DB() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setDB( opt-
>text() );
    }
    else if(a_path == "/prm/st/end" && ctrChkNode(opt))          opt->setText(
TSYS::int2str(end()) );
    else if(a_path == "/prm/st/beg" && ctrChkNode(opt))          opt->setText(
TSYS::int2str(begin()) );
    else if(a_path == "/prm/cfg/id" && ctrChkNode(opt))          opt->setText(
id() );
    else if(a_path == "/prm/cfg/nm" )
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
name() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setName( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/dscr")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
dscr() );
```

```
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setDscr( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/addr")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
addr() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setAddr( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/lvl")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(level()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setLevel(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/prm/cfg/start")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
toStart() ? "1" : "0" );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setToStart(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/prm/cfg/cats")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(m_cat_o);
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     { m_cat_o =
opt->text(); modif(); }
    }
    else if(a_path == "/mess/tm")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))
      {
          opt->setText(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")));
          if(!atoi(opt->text().c_str()))  opt-
>setText(TSYS::int2str(time(NULL)));
      }
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
          TBDS::genDBSet(nodePath()+"messTm",(atoi(opt-
>text().c_str())>=time(NULL))?"0":opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/size")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messSize","10",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messSize",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/cat")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messCat",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/lvl")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messLev","0",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messLev",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/mess" && run_st &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))
    {
      vector<TMess::SRec> rec;
```

```
      time_t end = atoi(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")).c_str());
      if( !end ) end = time(NULL);
      time_t beg = end - atoi(TBDS::genDBGet(nodePath()+"messSize","10",opt-
>attr("user")).c_str());
      string cat = TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user"));
      char   lev = atoi(TBDS::genDBGet(nodePath()+"messLev","0",opt-
>attr("user")).c_str());

      get( beg, end, rec, cat, lev );

      XMLNode *n_tm     = ctrMkNode("list",opt,-
1,"/mess/mess/0","",R_R___,"root",SARH_ID);
      XMLNode *n_tmu    = ctrMkNode("list",opt,-
1,"/mess/mess/0a","",R_R___,"root",SARH_ID);
      XMLNode *n_cat    = ctrMkNode("list",opt,-
1,"/mess/mess/1","",R_R___,"root",SARH_ID);
      XMLNode *n_lvl    = ctrMkNode("list",opt,-
1,"/mess/mess/2","",R_R___,"root",SARH_ID);
      XMLNode *n_mess   = ctrMkNode("list",opt,-
1,"/mess/mess/3","",R_R___,"root",SARH_ID);
      for(int i_rec = rec.size()-1; i_rec >= 0; i_rec--)
      {
          if(n_tm)      n_tm->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].time));
          if(n_tmu)     n_tmu->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].utime));
          if(n_cat)     n_cat->childAdd("el")->setText(rec[i_rec].categ);
          if(n_lvl)     n_lvl->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].level));
          if(n_mess)    n_mess->childAdd("el")->setText(rec[i_rec].mess);
      }
    }
    else TCntrNode::cntrCmdProc(opt);
}
```

tmodule.cpp – робота з модулями

```
//WebSCADA системний файл: tmodule.cpp

#include <sys/types.h>
#include <sys/stat.h>
#include <stdarg.h>
#include <unistd.h>
#include <dlfcn.h>
#include <string.h>
#include <libintl.h>

#include "tsys.h"
#include "terror.h"
#include "tmess.h"
#include "tsubsys.h"
#include "tmodule.h"

using namespace WSCADA;

//************************************************
//* TModule                                      *
//************************************************
const char *TModule::l_info[] =
    {"Модуль","Ім'я","Тип","Джерело","Версія","Автор","Дескриптор","Ліцензія"};

TModule::TModule( const string &id ) : mId(id)
{
    lc_id = string("oscd_")+mId;
    bindtextdomain(lc_id.c_str(),LOCALEDIR);

    //> Переведення динамічного рядка
#if 0
    char mess[][100] = { _("Автор"), _("Ліцензія") };
#endif
}

TModule::~TModule(  )
{
    //> Очищення списку експортуємих функцій
    for(unsigned i = 0; i < m_efunc.size(); i++)
        delete m_efunc[i];
}

string TModule::modName()
{
    return mName;
}

void TModule::postEnable( int flag )
{
    if(flag&TCntrNode::NodeRestore) return;

    mess_info(nodePath().c_str(),_("З'єднання з модулем!"));
}

TSubSYS &TModule::owner( )    { return *(TSubSYS*)nodePrev(); }

void TModule::modFuncList( vector<string> &list )
{
    list.clear();
    for(unsigned i = 0; i < m_efunc.size(); i++)
      list.push_back(m_efunc[i]->prot);
}

bool TModule::modFuncPresent( const string &prot )
{
    for(unsigned i = 0; i < m_efunc.size(); i++)
```

```
      if(m_efunc[i]->prot == prot)
          return true;
    return false;
}

TModule::ExpFunc &TModule::modFunc( const string &prot )
{
    for(unsigned i = 0; i < m_efunc.size(); i++)
      if(m_efunc[i]->prot == prot) return *m_efunc[i];
    throw TError(nodePath().c_str(),_("Функція '%s' не представлена у
модулі!"),prot.c_str());
}

void TModule::modFunc( const string &prot, void (TModule::**offptr)() )
{
    *offptr = modFunc(prot).ptr;
}

void TModule::modInfo( vector<string> &list )
{
    for(unsigned i_opt = 0; i_opt < sizeof(l_info)/sizeof(char *); i_opt++)
      list.push_back(l_info[i_opt]);
}

string TModule::modInfo( const string &name )
{
    string info;

    if(name == l_info[0])     info = mId;
    else if(name == l_info[1]) info = mName;
    else if(name == l_info[2]) info = mType;
    else if(name == l_info[3]) info = mSource;
    else if(name == l_info[4]) info = mVers;
    else if(name == l_info[5]) info = mAuthor;
    else if(name == l_info[6]) info = mDescr;
    else if(name == l_info[7]) info = mLicense;

    return info;
}

void TModule::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      ctrMkNode("WSCADA_cntr",opt,-1,"/",_("Module: ")+modId(),R_R_R_);
      ctrMkNode("branches",opt,-1,"/br","",R_R_R_);
      if(TUIS::icoPresent(owner().subId()+"."+modId())) ctrMkNode("img",opt,-
1,"/ico","",R_R_R_);
      if(ctrMkNode("area",opt,-1,"/help",_("Help")))
          if(ctrMkNode("area",opt,-1,"/help/m_inf",_("Модуль інформації")))
          {
            vector<string> list;
            modInfo(list);
            for(unsigned i_l = 0; i_l < list.size(); i_l++)
                ctrMkNode("fld",opt,-
1,(string("/help/m_inf/")+list[i_l]).c_str(),_(list[i_l].c_str()),R_R_R_,"root",
"root",1,"tp","str");
          }
      return;
    }

    //> Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/ico" && ctrChkNode(opt))
    {
      string itp;
```

```
      opt-
>setText(TSYS::strEncode(TUIS::icoGet(owner().subId()+"."+modId(),&itp),TSYS::ba
se64));
      opt->setAttr("tp",itp);
    }
    else if(a_path.substr(0,11) == "/help/m_inf" && ctrChkNode(opt))
      opt->setText(modInfo(TSYS::pathLev(a_path,2)));
    else TCntrNode::cntrCmdProc(opt);
}

const char *TModule::I18N( const char *mess )
{
    const char *rez = Mess->I18N(mess,lc_id.c_str());
    if( !strcmp(mess,rez) ) rez = _(mess);
    return rez;
}
```

tparamcontr.cpp – параметри управління системою

```
//WebSCADA системний файл: tparamcontr.cpp

#include "tbds.h"
#include "tsys.h"
#include "tmess.h"
#include "tdaqs.h"
#include "tcontroller.h"
#include "ttipdaq.h"
#include "ttiparam.h"
#include "tparamcontr.h"

using namespace WSCADA;

//************************************************
//* TParamContr                                  *
//************************************************
TParamContr::TParamContr( const string &name, TTipParam *tpprm ) :
TConfig(tpprm), m_en(false), tipparm(tpprm)
{
    setId(name);
    setName(name);
}

TParamContr::~TParamContr( )
{
    nodeDelAll();
}

TCntrNode &TParamContr::operator=( TCntrNode &node )
{
    TParamContr *src_n = dynamic_cast<TParamContr*>(&node);
    if(!src_n) return *this;

    //> Перевіряємо тип параметрів й змінюємо їх, якщо вони змінні, або нижчі
    if(type().name != src_n->type().name && owner().owner().tpPrmToId(src_n-
>type().name) >= 0)
    {
      if(enableStat()) disable();
      setType(src_n->type().name);
    }

    //> Конфігурація копіювання
    string tid = id();
    *(TConfig*)this = *(TConfig*)src_n;
    setId(tid);

    //> Дозволяємо нові параметри
    if(src_n->enableStat() && toEnable( ) && !enableStat())enable();

    return *this;
}

TController &TParamContr::owner( ) { return *(TController*)nodePrev(); }

string TParamContr::name( )   { string nm = cfg("NAME").getS(); return nm.size()
? nm : id(); }

void TParamContr::setName( const string &inm ) { cfg("NAME").setS(inm); }

string TParamContr::descr( ) { return cfg("DESCR").getS(); }

void TParamContr::setDescr( const string &idsc ){ cfg("DESCR").setS(idsc); }

void TParamContr::postEnable(int flag)
{
    TValue::postEnable(flag);
```

```
    if(!vlCfg())  setVlCfg(this);
    if(!vlElemPresent(&SYS->daq().at().errE()))
      vlElemAtt(&SYS->daq().at().errE());
}

void TParamContr::preDisable(int flag)
{
    //> Видаляємо або зупиняємо архівування
    vector<string> a_ls;
    vlList(a_ls);

    for(unsigned i_a = 0; i_a < a_ls.size(); i_a++)
      if(!vlAt(a_ls[i_a]).at().arch().freeStat())
      {
          string arh_id = vlAt(a_ls[i_a]).at().arch().at().id();
          if(flag) SYS->archive().at().valDel(arh_id,true);
          else SYS->archive().at().valAt(arh_id).at().stop();
      }

    if(enableStat())    disable();
}

void TParamContr::postDisable(int flag)
{
    if(flag)
    {
      //> Видаляємо параметри з БД
      try
      {
          SYS->db().at().dataDel(owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this,true);
      }catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }
    }
}

void TParamContr::load_( )
{
    if(!SYS->chkSelDB(owner().DB())) return;

    cfgViewAll(true);
    SYS->db().at().dataGet(owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this);
}

void TParamContr::save_( )
{
    SYS->db().at().dataSet( owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this );

    //> Зберігаємо архіви
    vector<string> a_ls;
    vlList(a_ls);
    for(unsigned i_a = 0; i_a < a_ls.size(); i_a++)
      if(!vlAt(a_ls[i_a]).at().arch().freeStat())
          vlAt(a_ls[i_a]).at().arch().at().save();
}

bool TParamContr::cfgChange( TCfg &cfg )  { modif( ); return true; }

TParamContr & TParamContr::operator=( TParamContr & PrmCntr )
{
    TConfig::operator=(PrmCntr);

    return *this;
}
```

```
void TParamContr::enable()
{
    m_en = true;
}

void TParamContr::disable()
{
    m_en = false;
}

void TParamContr::vlGet( TVal &val )
{
    if( val.name() == "err" )
    {
      if( !enableStat() ) val.setS(_("1:Параметри заборонені."),0,true);
      else if( !owner().startStat( ) ) val.setS(_("2:Контролер
зупинено."),0,true);
      else val.setS("0",0,true);
    }
}

void TParamContr::setId( const string &vl )
{
    cfg("SHIFR").setS(vl);
}

void TParamContr::setType( const string &tpId )
{
    if(enableStat() || tpId == type().name ||
!owner().owner().tpPrmPresent(tpId))     return;

    setNodeMode(TCntrNode::Disable);

    try
    {
      //> Чекаємо поки роз'єднаються інші
      while(nodeUse(true) > 1) usleep(1000);
      //> Видаляємо з БД
      postDisable(true);

      //> Створюємо тимчасову структуру
      TConfig tCfg(&type());
      tCfg = *(TConfig*)this;

      //> Встановлюємо нову конфігурацію структури
      tipparm = &owner().owner().tpPrmAt(owner().owner().tpPrmToId(tpId));
      setElem(tipparm);

      //> Відновлюємо конфігурацію
      *(TConfig*)this = tCfg;
    }catch(...) { }
    setNodeMode(TCntrNode::Enable);
    setVlCfg(this);
    modif();
}
TVariant TParamContr::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    //> Виклик функцій конфігурації
    TVariant cfRez = objFunc(iid, prms, user);
    if(!cfRez.isNull()) return cfRez;
    return TValue::objFuncCall(iid, prms, user);
}
void TParamContr::cntrCmdProc( XMLNode *opt )
{
    string a_path = opt->attr("path");
    //> Сервіс управління процесами
    if(a_path.substr(0,6) == "/serv/")    { TValue::cntrCmdProc(opt); return; }
```

```
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TValue::cntrCmdProc(opt);
      ctrMkNode("WSCADA_cntr",opt,-1,"/",_("Параметр:
")+name(),RWRWR_,"root",SDAQ_ID);
      if(ctrMkNode("area",opt,0,"/prm",_("Parameter")))
      {
          if(ctrMkNode("area",opt,-1,"/prm/st",_("Стан")))
          {
            if(!enableStat() && owner().owner().tpPrmSize() > 1)
                ctrMkNode("fld",opt,-
1,"/prm/st/type",_("Тип"),RWRWR_,"root",SDAQ_ID,4,"tp","str","dest","select","se
lect","/prm/tpLst",
                  "help",_("Змінюємо тип керівництва до останіх даних для
специфічних конфігурацій."));
            else ctrMkNode("fld",opt,-
1,"/prm/st/type",_("Тип"),R_R_R_,"root",SDAQ_ID,1,"tp","str");
            if(owner().enableStat())
                ctrMkNode("fld",opt,-
1,"/prm/st/en",_("Дозволено"),RWRWR_,"root",SDAQ_ID,1,"tp","bool");
          }
          if(ctrMkNode("area",opt,-1,"/prm/cfg",_("Configuration")))
            TConfig::cntrCmdMake(opt,"/prm/cfg",0,"root",SDAQ_ID,RWRWR_);
      }
        return;
    }
    //> Процес управління на сторінці
    if(a_path == "/prm/st/type")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SDAQ_ID,SEC_RD))     opt-
>setText(type().name);
      if(ctrChkNode(opt,"set",RWRWR_,"root",SDAQ_ID,SEC_WR))     setType(opt-
>text());
    }
    else if(a_path == "/prm/st/en")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SDAQ_ID,SEC_RD))     opt-
>setText(enableStat()?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root",SDAQ_ID,SEC_WR))
      {
          if(!owner().enableStat()) throw TError(nodePath().c_str(),_("Контролер
не запустився!"));
          else atoi(opt->text().c_str())?enable():disable();
      }
    }
    else if(a_path.substr(0,8) == "/prm/cfg")
TConfig::cntrCmdProc(opt,TSYS::pathLev(a_path,2),"root",SDAQ_ID,RWRWR_);
    else if(a_path == "/prm/tmplList" && ctrChkNode(opt))
    {
        vector<string> lls, ls;
      SYS->daq().at().tmplLibList(lls);
      for(unsigned i_l = 0; i_l < lls.size(); i_l++)
      {
          SYS->daq().at().tmplLibAt(lls[i_l]).at().list(ls);
          for(unsigned i_t = 0; i_t < ls.size(); i_t++)
            opt->childAdd("el")->setText(lls[i_l]+"."+ls[i_t]);
      }
    }
    else if(a_path == "/prm/tpLst" && ctrChkNode(opt))
      for(unsigned i_tp = 0; i_tp < owner().owner().tpPrmSize(); i_tp++)
          opt->childAdd("el")->setAttr("id",owner().owner().tpPrmAt(i_tp).name)-
>setText(owner().owner().tpPrmAt(i_tp).descr);
    else TValue::cntrCmdProc(opt);
}
```

main.cpp – головна програма

```
//Файл системи збору, обробки та відображення інформації об’єкту моніторингу у
мережі: main.cpp

#include <getopt.h>
#include "terror.h"
#include "tmess.h"
#include "tsys.h"

using namespace WSCADA;

int main(int argc, char *argv[], char *envp[] )
{
    int rez = 0;

    //Перевірка початку роботи режиму основного процесу
    int next_opt;
    optind=opterr=0;
    struct option long_opt[] = { {"Режим основного процесу" ,0,NULL,'d'}, {NULL
,0,NULL,0  } };
    while((next_opt=getopt_long(argc,argv,"",long_opt,NULL)) != -1)
      if( next_opt == 'd' )
      {
          printf("Початок роботи режиму основного процесу!\n");
          int pid = fork();
          if( pid == -1 )
          {
            printf("Помилка: неможливо створити новий процес!\n");
            return -1;
          }
          if( pid != 0 )      return 0;

          //Готується оточення режиму основного процесу
          setsid();
          break;
      }

    try
    {
      SYS = new TSYS(argc,argv,envp);

      SYS->load();
      if( (rez=SYS->stopSignal()) > 0 ) return rez;
      rez = SYS->start();

      delete SYS;
    }catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }

    printf("Система збору, обробки та відображення інформації об’єкту
моніторингу у мережі коректно працює з даними %d.\n",rez);

    return rez;
}
```

tsys.cpp – встановлення та запуск системи

```
//WebSCADA системний файл: tsys.cpp

#include <features.h>
#include <ieee754.h>
#include <syscall.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/utsname.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <getopt.h>
#include <stdio.h>
#include <signal.h>
#include <stdarg.h>
#include <stdlib.h>
#include <langinfo.h>
#include <zlib.h>

#include "terror.h"
#include "tmess.h"
#include "tsys.h"

using namespace WSCADA;

//Поточна зміна доступу
TMess *WSCADA::Mess;
TSYS  *WSCADA::SYS;
bool TSYS::finalKill = false;
pthread_key_t TSYS::sTaskKey;

TSYS::TSYS( int argi, char ** argb, char **env ) : argc(argi), argv((const char
**)argb), envp((const char **)env),
    mUser("root"), mConfFile("/etc/WSCADA.xml"), mId("EmptySt"),
mName(_("Порожня Станція")), mIcoDir("./icons/"), mModDir("./"),
    mWorkDB("<cfg>"), mSaveAtExit(false), mSavePeriod(0), rootModifCnt(0),
mStopSignal(-1), mMultCPU(false)
{
    finalKill = false;
    SYS = this;         //Ініціалізуємо значення глобальних зміних доступу
    mSubst = grpAdd("sub_",true);
    nodeEn();
    pthread_key_create(&sTaskKey, NULL);

    Mess = new TMess();

    if(getenv("USER")) mUser = getenv("USER");

    //> Ініціалізуємо системний годинник
    clkCalc();

#if __GLIBC_PREREQ(2,4)
    //> Multi CPU дозволяють перевірку
    cpu_set_t cpuset;
    CPU_ZERO(&cpuset);
    CPU_SET(1,&cpuset);
    mMultCPU = !pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t),
&cpuset);
#endif

    //> Встановлюємо сигнальні програми обробки
    signal(SIGINT,sighandler);
    signal(SIGTERM,sighandler);
```

```
    //signal(SIGCHLD,sighandler);
    signal(SIGALRM,sighandler);
    signal(SIGPIPE,sighandler);
    //signal(SIGFPE,sighandler);
    //signal(SIGSEGV,sighandler);
    signal(SIGABRT,sighandler);
}

TSYS::~TSYS( )
{
    finalKill = true;

    //Видаляємо всі вузли в команді управління
    del("ModSched");
    del("UI");
    del("Special");
    del("Archive");
    del("DAQ");
    del("Protocol");
    del("Transport");
    del("Security");
    del("BD");

    delete Mess;
    pthread_key_delete(sTaskKey);
}

string TSYS::host( )
{
    utsname ubuf; uname(&ubuf);
    return ubuf.nodename;
}

string TSYS::workDir( )
{
    char buf[STR_BUF_LEN];
    return getcwd(buf,sizeof(buf));
}

void TSYS::setWorkDir( const string &wdir )
{
    if(workDir() == wdir) return;
    if(chdir(wdir.c_str()) != 0)
      mess_warning(nodePath().c_str(),_("Змініть робочу директорію'%s' Помилка:
%s. Можливо поточний каталог вже встановлено правильно'%s'."),
          wdir.c_str(),strerror(errno),workDir().c_str());
    modif( );
}

XMLNode *TSYS::cfgNode( const string &path, bool create )
{
    string s_el, ndNm;

    XMLNode *t_node = &rootN;
    if(t_node->name() != "WebSCADA")
    {
      if(!create) return NULL;
      t_node->setName("WebSCADA");
    }

    for(int l_off = 0, nLev = 0; true; nLev++)
    {
        s_el = TSYS::pathLev(path,0,true,&l_off);
        if(s_el.empty()) return t_node;
        bool ok = false;
        for(unsigned i_f = 0; !ok && i_f < t_node->childSize(); i_f++)
            if(t_node->childGet(i_f)->attr("id") == s_el)
            {
                t_node = t_node->childGet(i_f);
```

```
                ok = true;
              }
            if(!ok)
            {
              if(!create)   return NULL;
              ndNm = "prm";
              switch(nLev)
              {
                case 0: ndNm = "station";     break;
                case 1: if(s_el.compare(0,4,"sub_") == 0) ndNm = "node";   break;
                case 2: if(s_el.compare(0,4,"mod_") == 0) ndNm = "node";   break;
              }
              if(ndNm == "prm") t_node = t_node->childIns(0,ndNm)-
>setAttr("id",s_el);
              else t_node = t_node->childAdd(ndNm)->setAttr("id",s_el);
            }
        }
        return t_node;
}

string TSYS::int2str( int val, TSYS::IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%d",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%o",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%x",val);

    return buf;
}

string TSYS::uint2str( unsigned val, IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%u",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%o",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%x",val);

    return buf;
}

string TSYS::ll2str( int64_t val, IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%lld",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%llo",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%llx",val);

    return buf;
}

string TSYS::real2str( double val, int prec, char tp )
{
    char buf[STR_BUF_LEN];
    if(tp == 'g') snprintf(buf,sizeof(buf),"%.*g",prec,val);
    else if(tp == 'e') snprintf(buf,sizeof(buf),"%.*e",prec,val);
    else snprintf(buf,sizeof(buf),"%.*f",prec,val);

    return buf;
}

string TSYS::time2str( time_t itm, const string &format )
{
    struct tm tm_tm;
    localtime_r(&itm,&tm_tm);
    char buf[100];
    int ret = strftime(buf, sizeof(buf), format.empty()?"%d-%m-%Y
%H:%M:%S":format.c_str(), &tm_tm);
    return (ret > 0) ? string(buf,ret) : string("");
}
```

```
string TSYS::time2str( double utm )
{
    if(utm < 1e-6) return "0";
    int lev = 0;
    int days = (int)floor(utm/(24*60*60*1e6));
    int часи = (int)floor(utm/(60*60*1e6))%24;
    int mins = (int)floor(utm/(60*1e6))%60;
    double usec = utm - 1e6*(days*24*60*60 + часи*60*60 + mins*60);

    string rez;
    if(days)           { rez += TSYS::int2str(days)+_("day"); lev =
vmax(lev,6); }
    if(часи)           { rez += (rez.size()?"
":"")+TSYS::int2str(часи)+_("годин"); lev = vmax(lev,5); }
    if(mins && lev < 6) { rez += (rez.size()?"
":"")+TSYS::int2str(mins)+_("хвилин"); lev = vmax(lev,4); }
    if((1e-6*usec) > 0.5 && lev < 5)     { rez += (rez.size()?"
":"")+TSYS::real2str(1e-6*usec,3)+_("секунд"); lev = vmax(lev,3); }
    else if((1e-3*usec) > 0.5 && !lev)   { rez += (rez.size()?"
":"")+TSYS::real2str(1e-3*usec,4)+_("мікросекунд"); lev = vmax(lev,2); }
    else if(usec > 0.5 && !lev)          { rez += (rez.size()?"
":"")+TSYS::real2str(usec,4)+_("us"); lev = vmax(lev,1); }
    else if(!lev) rez += (rez.size()?" ":"")+TSYS::real2str(1e3*usec,4)+_("ns");
    return rez;
}

string TSYS::cpct2str( double cnt )
{
    if(cnt > 0.2*pow(2,80))  return
TSYS::real2str(cnt/pow(2,80),3,'g')+_("YiB");
    if(cnt > 0.2*pow(2,70))  return
TSYS::real2str(cnt/pow(2,70),3,'g')+_("ZiB");
    if(cnt > 0.2*pow(2,60))  return
TSYS::real2str(cnt/pow(2,60),3,'g')+_("EiB");
    if(cnt > 0.2*pow(2,50))  return
TSYS::real2str(cnt/pow(2,50),3,'g')+_("PiB");
    if(cnt > 0.2*pow(2,40))  return
TSYS::real2str(cnt/pow(2,40),3,'g')+_("TiB");
    if(cnt > 0.2*pow(2,30))  return
TSYS::real2str(cnt/pow(2,30),3,'g')+_("GiB");
    if(cnt > 0.2*pow(2,20))  return
TSYS::real2str(cnt/pow(2,20),3,'g')+_("MiB");
    if(cnt > 0.2*pow(2,10))  return
TSYS::real2str(cnt/pow(2,10),3,'g')+_("KiB");
    return TSYS::real2str(cnt,3,'g')+_("B");
}

string TSYS::addr2str( void *addr )
{
    char buf[sizeof(void*)*2+3];
    snprintf(buf,sizeof(buf),"%p",addr);

    return buf;
}

void *TSYS::str2addr( const string &str )
{
    return (void *)strtoul(str.c_str(),NULL,16);
}

string TSYS::strNoSpace( const string &val )
{
    int beg = -1, end = -1;

    for(unsigned i_s = 0; i_s < val.size(); i_s++)
      if(val[i_s] != ' ' && val[i_s] != '\n' && val[i_s] != '\t')
      {
          if(beg < 0) beg = i_s;
```

```
            end = i_s;
        }

    return (beg>=0) ? val.substr(beg,end-beg+1) : "";
}

string TSYS::strMess( const char *fmt, ... )
{
    char str[STR_BUF_LEN];
    va_list argptr;

    va_start(argptr,fmt);
    vsnprintf(str,sizeof(str),fmt,argptr);
    va_end(argptr);

    return str;
}

string TSYS::optDescr( )
{
    utsname buf;
    uname(&buf);
    return TSYS::strMess(_(
      "********************************************************************************
**\n"
      "********** %s v%s (%s-%s). *********\n"
      "********************************************************************************
**\n\n"
      "================================================================================
==\n"
      "======================== Загальносистемні опції
======================\n"
      "================================================================================
==\n"
      "-h, --help             Інформаційні повідомлення про опції системи.\n"
      "    --Config=<path>    Шлях до конфігураційного файлу.\n"
      "    --Station=<id>     Ідентифікатор станції.\n"
      "    --режим основного процесу          Початок роботи режиму основного
процесу.\n"
      "    --MessLev=<level>  Повідомлення процесів<level> (0-7).\n"
      "    --log=<direct>     Повідомлення про :\n"
      "                         <direct> & 1 - події у системі;\n"
      "                         <direct> & 2 - ввод/вивід у системі;\n"
      "                         <direct> & 4 - помилки у системі;\n"
      "                         <direct> & 8 - архівація у системі;\n"
      "----------- Конфігураційний файл параметрів робочої станції'%s' ---------
--\n"
      "StName     <nm>  Ім’я робочої станції.\n"
      "WorkDB     <Type.Name> Робоча база даних (тип та ім’я).\n"
      "Workdir    <path>      Робоча директорія.\n"
      "IcoDir     <path>      Директорія іконок.\n"
      "ModDir     <path>      Директорія модулів.\n"
      "MessLev    <level>     Повідомлення <рівні> (0-7).\n"
      "LogTarget  <direction> Повідомлення про :\n"
      "                           <direct> & 1 - події у системі;\n"
      "                           <direct> & 2 - ввод/вивід у системі;\n"
      "                           <direct> & 4 - помилки у системі;\n"
      "                           <direct> & 8 - архівація у системі;\n"
      "Lang2CodeBase <lang>   Базова мова для переведення тексту, двосимвольний
код .\n"
      "SaveAtExit <true>      Збереження змін у системі перед виходом.\n"
      "SavePeriod <sec> Збереження періоду роботи  у системі .\n\n"),
      PACKAGE_NAME,VERSION,buf.sysname,buf.release,nodePath().c_str());
}

bool TSYS::cfgFileLoad( )
{
    bool cmd_help = false;
```

```
    //================ Редагування параметрів=========================
    int next_opt;
    const char *short_opt="h";
    struct option long_opt[] =
    {
      {"help"     ,0,NULL,'h'},
      {"Config"   ,1,NULL,'f'},
      {"Station"  ,1,NULL,'s'},
      {NULL       ,0,NULL,0  }
    };

    optind=opterr=0;
    do
    {
      next_opt=getopt_long(argc,(char * const *)argv,short_opt,long_opt,NULL);
      switch(next_opt)
      {
          case 'h':
            fprintf(stdout,"%s",optDescr().c_str());
            Mess->setMessLevel(7);
            cmd_help = true;
            break;
          case 'f': mConfFile = optarg; break;
          case 's': mId = optarg; break;
          case -1 : break;
      }
    } while(next_opt != -1);

    //Завантажуємо кофігураційний файл
    int hd = open(mConfFile.c_str(),O_RDONLY);
    if(hd < 0) mess_err(nodePath().c_str(),_("Кофігураційний файл '%s' Помилка:
%s"),mConfFile.c_str(),strerror(errno));
    else
    {
      string s_buf;
      int cf_sz = lseek(hd,0,SEEK_END);
      if(cf_sz > 0)
      {
          lseek(hd,0,SEEK_SET);
          char *buf = (char *)malloc(cf_sz+1);
          read(hd,buf,cf_sz);
          buf[cf_sz] = 0;
          s_buf = buf;
          free(buf);
      }
      close(hd);

      try
      {
          ResAlloc res(nodeRes(),true);
          rootN.load(s_buf,true);
          if(rootN.name() == "WebSCADA")
          {
            XMLNode *stat_n = NULL;
            for(int i_st = rootN.childSize()-1; i_st >= 0; i_st--)
                if(rootN.childGet(i_st)->name() == "station")
                {
                  stat_n = rootN.childGet(i_st);
                  if(stat_n->attr("id") == mId) break;
                }
            if(stat_n && stat_n->attr("id") != mId)
            {
                mess_warning(nodePath().c_str(),_("Робоча станція '%s' не
представлена у  кофігураційному файлі Використайте '%s' конфігурацію робочої
станції!"),
                  mId.c_str(), stat_n->attr("id").c_str());
                mId     = stat_n->attr("id");
            }
            if(!stat_n) rootN.clear();
```

```
          } else rootN.clear();
          if(!rootN.childSize()) mess_err(nodePath().c_str(),_("Помилка
конфігурації '%s'!"),mConfFile.c_str());
          rootModifCnt = 0;
      }
      catch(TError err) { mess_err(nodePath().c_str(),_("Завантажуємо
кофігураційний файл Помилка: %s"),err.mess.c_str() ); }
    }

    return cmd_help;
}

void TSYS::cfgFileSave( )
{
    ResAlloc res(nodeRes(),true);
    if(!rootModifCnt) return;
    int hd = open(mConfFile.c_str(), O_CREAT|O_TRUNC|O_WRONLY, 0664);
    if(hd < 0) mess_err(nodePath().c_str(),_("Кофігураційний файл '%s' Помилка:
%s"),mConfFile.c_str(),strerror(errno));

    string rezFile = rootN.save(XMLNode::XMLHeader);
    int rez = write(hd, rezFile.data(), rezFile.size());
    if(rez != (int)rezFile.size()) mess_err(nodePath().c_str(),_("Помилка запису
конфігурації. %s"),mConfFile.c_str(),((rez<0)?strerror(errno):""));
    rootModifCnt = 0;
    rootFlTm = time(NULL);
}

void TSYS::cfgPrmLoad( )
{
    //Системні параметри
    mName =
TBDS::genDBGet(nodePath()+"StName",name(),"root",TBDS::UseTranslate);
    mWorkDB = TBDS::genDBGet(nodePath()+"WorkDB",workDB(),"root",TBDS::OnlyCfg);
    setWorkDir(TBDS::genDBGet(nodePath()+"Workdir").c_str());
    setIcoDir(TBDS::genDBGet(nodePath()+"IcoDir",icoDir()));
    setModDir(TBDS::genDBGet(nodePath()+"ModDir",modDir()));
    setSaveAtExit(atoi(TBDS::genDBGet(nodePath()+"SaveAtExit","0").c_str()));
    setSavePeriod(atoi(TBDS::genDBGet(nodePath()+"SavePeriod","0").c_str()));
}

void TSYS::load_()
{
    static bool first_load = true;

    bool cmd_help = cfgFileLoad();
    mess_info(nodePath().c_str(),_("Load!"));
    cfgPrmLoad();
    Mess->load(); //Завантажуємо повідомлення

    if( first_load )
    {
      //> Створюємо підсистему
      add( new TBDS() );
      add( new TSecurity() );
      add( new TTransportS() );
      add( new TProtocolS() );
      add( new TDAQS() );
      add( new TArchiveS() );
      add( new TSpecialS() );
      add( new TUIS() );
      add( new TModSchedul() );

      //> Завантажуємо модулі
      modSchedul().at().load();
      if( !modSchedul().at().loadLibS() )
      {
          mess_err(nodePath().c_str(),_("Жоден модуль не завантажений. Ваша
конфігурація перервана!"));
```

```
            stop();
        }

        //> Завантажуємо базу даних першої  підсистеми
        db().at().load();
        if( !cmd_help ) modSchedul().at().modifG();     // Для перевантаження
спроби від бази даних

        //> Друге завантаження для завантаження від родової БД
        Mess->load();
        cfgPrmLoad();
    }

    //> Пряме завантаження підсистем та модулів
    vector<string> lst;
    list(lst);
    for( unsigned i_a=0; i_a < lst.size(); i_a++ )
      try { at(lst[i_a]).at().load(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка завантаження підсистеми
'%s'."),lst[i_a].c_str());
      }

    if( cmd_help ) stop();
    first_load = false;
}

void TSYS::save_( )
{
    char buf[STR_BUF_LEN];

    mess_info(nodePath().c_str(),_("Save!"));

    //> Системні параметри
    getcwd(buf,sizeof(buf));
    TBDS::genDBSet(nodePath()+"StName",mName,"root",TBDS::UseTranslate);
    TBDS::genDBSet(nodePath()+"Workdir",buf);
    TBDS::genDBSet(nodePath()+"IcoDir",icoDir());
    TBDS::genDBSet(nodePath()+"ModDir",modDir());
    TBDS::genDBSet(nodePath()+"SaveAtExit",TSYS::int2str(saveAtExit()));
    TBDS::genDBSet(nodePath()+"SavePeriod",TSYS::int2str(savePeriod()));

    Mess->save(); //Завантажуємо повідомлення
}

int TSYS::start( )
{
    vector<string> lst;
    list(lst);

    mess_info(nodePath().c_str(),_("Start!"));
    for(unsigned i_a=0; i_a < lst.size(); i_a++)
      try { at(lst[i_a]).at().subStart(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка запуску
підсистеми'%s'."),lst[i_a].c_str());
      }

    cfgFileScan( true );

    mess_info(nodePath().c_str(),_("Завершення запуску!"));

    unsigned int i_cnt = 1;
    mStopSignal = 0;
    while(!mStopSignal)
```

```
    {
      //> CPU підрахунок частоти
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))     clkCalc( );

      //> Кофігураційний файл змін періодичних перевірок
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))     cfgFileScan( );

      //> Періодична загальна перевірка бібліотек
      if(modSchedul( ).at().chkPer() && !(i_cnt%(modSchedul(
).at().chkPer()*1000/STD_WAIT_DELAY)))
          modSchedul( ).at().libLoad(modDir(),true);

      //> Періодичний запис змін до  БД
      if(savePeriod() && !(i_cnt%(savePeriod()*1000/STD_WAIT_DELAY))) save();

      //> Кофігураційний файл зберігає необхідні зміни
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))     cfgFileSave();

      //> Викликаємо підсистему кожні 10 сек.
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))
          for(unsigned i_a=0; i_a < lst.size(); i_a++)
            try { at(lst[i_a]).at().perSYSCall(i_cnt/(1000/STD_WAIT_DELAY)); }
            catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str());
}

      usleep(STD_WAIT_DELAY*1000);
      i_cnt++;
    }

    mess_info(nodePath().c_str(),_("Stop!"));
    if(saveAtExit() || savePeriod())     save();
    cfgFileSave();
    for(int i_a=lst.size()-1; i_a >= 0; i_a--)
      try { at(lst[i_a]).at().subStop(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка остановки
підсистеми'%s'."),lst[i_a].c_str());
      }

    return mStopSignal;
}

void TSYS::stop( )
{
    mStopSignal = SIGUSR1;
}

bool TSYS::chkSelDB( const string& wDB,  bool isStrong )
{
    if(selDB().empty() && !isStrong) return true;
    if(SYS->selDB() == TBDS::realDBName(wDB)) return true;
    return false;
}

void TSYS::sighandler( int signal )
{
    switch(signal)
    {
      case SIGINT:
          SYS->mStopSignal=signal;
          break;
      case SIGTERM:
          mess_warning(SYS->nodePath().c_str(),_("Отриманий сигнал переривання
роботи. Сервер зупиняється!"));
          SYS->mStopSignal=signal;
          break;
      case SIGFPE:
```

```
          mess_warning(SYS->nodePath().c_str(),_("Виключення плаваючої крапки
спіймане!"));
            exit(1);
          break;
      case SIGCHLD:
      {
          int status;
          pid_t pid = wait(&status);
          if(!WIFEXITED(status) && pid > 0)
            mess_info(SYS->nodePath().c_str(),_("Вивільнено процес-
потомок%d!"),pid);
          break;
      }
      case SIGPIPE:
          //mess_warning(SYS->nodePath().c_str(),_("Сигнал переривання
PIPE!"));
          break;
      case SIGSEGV:
          mess_emerg(SYS->nodePath().c_str(),_("Сигнал помилки від сегменту!"));
          break;
      case SIGABRT:
          mess_emerg(SYS->nodePath().c_str(),_("WebSCADA перервала роботу!"));
          break;
      case SIGALRM:     break;
      default:
          mess_warning(SYS->nodePath().c_str(),_("Невизначений
сигнал%d!"),signal);
    }
}

void TSYS::cfgFileScan( bool first )
{
    struct stat f_stat;

    if(stat(cfgFile().c_str(),&f_stat) != 0) return;
    bool up = false;
    if(rootCfgFl != cfgFile() || rootFlTm != f_stat.st_mtime) up = true;
    rootCfgFl = cfgFile();
    rootFlTm = f_stat.st_mtime;

    if(up && !first)
    {
      modifG();
      setSelDB("<cfg>");
      load();
      setSelDB("");
    }
}

int64_t TSYS::curTime( )
{
    timeval cur_tm;
    gettimeofday(&cur_tm,NULL);
    return (int64_t)cur_tm.tv_sec*1000000 + cur_tm.tv_usec;
}

bool TSYS::eventWait( bool &m_mess_r_stat, bool exempl, const string &loc,
time_t tm )
{
    time_t t_tm, s_tm;

    t_tm = s_tm = time(NULL);
    while( m_mess_r_stat != exempl )
    {
      time_t c_tm = time(NULL);
      //Контролюємо перерву
      if( tm && ( c_tm > s_tm+tm) )
      {
          mess_crit(loc.c_str(),_("Timeouted !!!"));
```

```
            return true;
        }
        //Створюємо повідомлення
        if( c_tm > t_tm+1 )  //1sec
        {
            t_tm = c_tm;
            mess_info(loc.c_str(),_("Чекаємо подію..."));
        }
        usleep(STD_WAIT_DELAY*1000);
    }
    return false;
}

string TSYS::strSepParse( const string &path, int level, char sep, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    if(an_dir >= (int)path.size()) return "";
    while(true)
    {
      t_dir = path.find(sep,an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? path.substr(an_dir) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir+1;
          return path.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir+1;
      t_lev++;
    }
    return "";
}

string TSYS::strParse( const string &path, int level, const string &sep, int
*off, bool mergeSepSymb )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    if(an_dir >= (int)path.size() || sep.empty()) return "";
    while(true)
    {
      t_dir = path.find(sep,an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? path.substr(an_dir) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir+sep.size();
          return path.substr(an_dir,t_dir-an_dir);
      }
      if( mergeSepSymb && sep.size() == 1 )
          for(an_dir = t_dir; an_dir < (int)path.size() && path[an_dir] ==
sep[0]; ) an_dir++;
      else an_dir = t_dir+sep.size();
      t_lev++;
    }
    return "";
}
```

```
string TSYS::strLine( const string &str, int level, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0, edLnSmbSz = 1;
    size_t t_dir;

    if(an_dir >= (int)str.size()) return "";
    while(true)
    {
      for(t_dir = an_dir; t_dir < str.size(); t_dir++)
          if(str[t_dir] == '\x0D' || str[t_dir] == '\x0A')
          { edLnSmbSz = (str[t_dir] == '\x0D' && ((t_dir+1) < str.size()) &&
str[t_dir+1] == '\x0A') ? 2 : 1; break; }
      if(t_dir >= str.size())
      {
          if(off) *off = str.size();
          return (t_lev==level) ? str.substr(an_dir) : "";
      }
      else if(t_lev == level)
      {
          if(off) *off = t_dir+edLnSmbSz;
          return str.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir+edLnSmbSz;
      t_lev++;
    }
    return "";
}

string TSYS::pathLev( const string &path, int level, bool encode, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    //> Перший роздільний прохід
    while(an_dir < (int)path.size() && path[an_dir]=='/') an_dir++;
    if(an_dir >= (int)path.size()) return "";
    //> Шлях рівня процесу
    while(true)
    {
      t_dir = path.find("/",an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? ( encode ?
TSYS::strDecode(path.substr(an_dir),TSYS::PathEl) : path.substr(an_dir) ) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir;
          return encode ? TSYS::strDecode(path.substr(an_dir,t_dir-
an_dir),TSYS::PathEl) : path.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir;
      t_lev++;
      while(an_dir < (int)path.size() && path[an_dir]=='/') an_dir++;
    }
}

string TSYS::path2sepstr( const string &path, char sep )
{
    string rez, curv;
    int off = 0;
    while( !(curv=TSYS::pathLev(path,0,false,&off)).empty() )
      rez+=curv+sep;
    if(!rez.empty())    rez.resize(rez.size()-1);

    return rez;
```

```
}

string TSYS::sepstr2path( const string &str, char sep )
{
    string rez, curv;
    int off = 0;
    while( !(curv=TSYS::strSepParse(str,0,sep,&off)).empty() )
      rez+="/"+curv;

    return rez;
}

string TSYS::strEncode( const string &in, TSYS::Code tp, const string &symb )
{
    int i_sz;
    string sout;

    switch(tp)
    {
      case TSYS::PathEl:
          sout = in;
          for( i_sz = 0; i_sz < (int)sout.size(); i_sz++ )
            switch( sout[i_sz] )
            {
                case '/': sout.replace(i_sz,1,"%2f"); i_sz+=2; break;
                case '%': sout.replace(i_sz,1,"%25"); i_sz+=2; break;
            }
          break;
      case TSYS::HttpURL:
          sout = in;
          for( i_sz = 0; i_sz < (int)sout.size(); i_sz++ )
            switch( sout[i_sz] )
            {
                case '%': sout.replace(i_sz,1,"%25"); i_sz+=2; break;
                case ' ': sout.replace(i_sz,1,"%20"); i_sz+=2; break;
                case '\t': sout.replace(i_sz,1,"%09"); i_sz+=2; break;
                default:
                  if( sout[i_sz]&0x80 )
                  {
                    char buf[4];
                    snprintf(buf,sizeof(buf),"%%%02X",(unsigned
char)sout[i_sz]);
                    sout.replace(i_sz,1,buf);
                    i_sz+=2;
                    break;
                  }
            }
          break;
      case TSYS::Html:
          sout.reserve(in.size()+10);
          for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            switch( in[i_sz] )
            {
                case '>':     sout+="&gt;";     break;
                case '<':     sout+="&lt;";     break;
                case '"':     sout+="&quot;";   break;
                case '&':     sout+="&amp;";    break;
                case '\'':    sout+="&apos;";   break;
                default:      sout+=in[i_sz];
            }
          break;
      case TSYS::JavaSc:
          sout.reserve(in.size()+10);
          for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            switch( in[i_sz] )
            {
                case '\n':    sout+="\\n";      break;
                default:      sout+=in[i_sz];
            }
```

```
            break;
        case TSYS::SQL:
            sout.reserve(in.size()+10);
            for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
              switch( in[i_sz] )
              {
                  case '\'':    sout+="\\'";      break;
                  case '\"':    sout+="\\\"";     break;
                  case '`':     sout+="\\`";      break;
                  case '\\':    sout+="\\\\";     break;
                  default:      sout+=in[i_sz];
              }
            break;
        case TSYS::Custom:
            sout.reserve(in.size()+10);
            for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            {
              unsigned i_smb;
              for(i_smb = 0; i_smb < symb.size(); i_smb++)
                  if(in[i_sz] == symb[i_smb])
                  {
                    char buf[4];
                    sprintf(buf,"%%%02X",(unsigned char)in[i_sz]);
                    sout += buf;
                    break;
                  }
              if(i_smb >= symb.size()) sout += in[i_sz];
            }
            break;
        case TSYS::base64:
        {
            sout.reserve(in.size()+in.size()/4+in.size()/57+10);
            const char *base64alph =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
            for( i_sz = 0; i_sz < (int)in.size(); i_sz+=3 )
            {
              if(i_sz && !(i_sz%57)) sout.push_back('\n');
              sout.push_back(base64alph[(unsigned char)in[i_sz]>>2]);
              if((i_sz+1) >= (int)in.size())
              {
                  sout.push_back(base64alph[((unsigned char)in[i_sz]&0x03)<<4]);
                  sout += "==";
              }
              else
              {
                  sout.push_back(base64alph[(((unsigned
char)in[i_sz]&0x03)<<4)|((unsigned char)in[i_sz+1]>>4)]);
                  if((i_sz+2) >= (int)in.size())
                  {
                    sout.push_back(base64alph[((unsigned
char)in[i_sz+1]&0x0F)<<2]);
                    sout.push_back('=');
                  }
                  else
                  {
                    sout.push_back(base64alph[(((unsigned
char)in[i_sz+1]&0x0F)<<2)|((unsigned char)in[i_sz+2]>>6)]);
                    sout.push_back(base64alph[(unsigned char)in[i_sz+2]&0x3F]);
                  }
              }
            }
            break;
        }
        case TSYS::FormatPrint:
            sout = in;
            for(i_sz = 0; i_sz < (int)sout.size(); i_sz++)
              if(sout[i_sz] == '%') { sout.replace(i_sz,1,"%%"); i_sz++; }
            break;
        case TSYS::oscdID:
```

```
            sout.reserve(in.size());
            for(i_sz = 0; i_sz < (int)in.size(); i_sz++)
              switch(in[i_sz])
              {
                  case ' ': case '/': case '\\': case '&': case '(':
                  case ')': case '[': case ']': case '!': case '~':
                  case '`': case '@': case '%': case '^': case '-':
                  case '+': case '=': case '*': case '{': case '}':
                  case ':': case ';': case '"': case '\'': case '<':
                  case '>': case '?': case '.': case ',':
                    sout+="_";  break;
                  default:      sout+=in[i_sz];
              }
            break;
        case TSYS::Bin:
        {
            string svl, evl;
            sout.reserve(in.size());
            for(int off = 0; (svl=TSYS::strSepParse(in,0,'\n',&off)).size(); )
              for(int offE = 0; (evl=TSYS::strSepParse(svl,0,' ',&offE)).size(); )
                  sout+=(char)strtol(evl.c_str(),NULL,16);
            break;
        }
        case TSYS::Reverse:
            for(i_sz = in.size()-1; i_sz >= 0; i_sz--) sout += in[i_sz];
            break;
        case TSYS::ShieldSimb:
            sout.reserve(in.size());
            for(i_sz = 0; i_sz < (int)in.size(); i_sz++)
              if(in[i_sz] == '\\' && i_sz < ((int)in.size()-1))
              {
                  switch(in[i_sz+1])
                  {
                    case 'a':   sout += '\a';     break;
                    case 'b':   sout += '\b';     break;
                    case 'f':   sout += '\f';     break;
                    case 'n':   sout += '\n';     break;
                    case 'r':   sout += '\r';     break;
                    case 't':   sout += '\t';     break;
                    case 'v':   sout += '\v';     break;
                    case 'x': case 'X':
                      if((i_sz+3) < (int)in.size() && isxdigit(in[i_sz+2]) &&
isxdigit(in[i_sz+3]))
                        { sout +=
(char)strtol(in.substr(i_sz+2,2).c_str(),NULL,16); i_sz += 2; }
                      else sout += in[i_sz+1];
                            break;
                    default:
                            if((i_sz+3) < (int)in.size() && in[i_sz+1] >= '0' &&
in[i_sz+1] <= '7' &&
                                                            in[i_sz+2] >= '0' &&
in[i_sz+2] <= '7' &&
                                                            in[i_sz+3] >= '0' &&
in[i_sz+3] <= '7')
                            { sout +=
(char)strtol(in.substr(i_sz+1,3).c_str(),NULL,8); i_sz += 2; }
                            else sout += in[i_sz+1];
                  }
                  i_sz++;
              }else sout += in[i_sz];
            break;
    }
    return sout;
}

unsigned char TSYS::getBase64Code(unsigned char asymb)
{
    switch(asymb)
    {
```

```
      case 'A' ... 'Z': return asymb-(unsigned char)'A';
      case 'a' ... 'z': return 26+asymb-(unsigned char)'a';
      case '0' ... '9': return 52+asymb-(unsigned char)'0';
      case '+':         return 62;
      case '/':         return 63;
    }
    return 0;
}

string TSYS::strDecode( const string &in, TSYS::Code tp )
{
    unsigned i_sz;
    string sout;

    switch(tp)
    {
      case TSYS::PathEl: case TSYS::HttpURL: case TSYS::Custom:
          sout.reserve(in.size());
          for(i_sz = 0; i_sz < in.size(); i_sz++)
            switch(in[i_sz])
            {
                case '%':
                  if(i_sz+2 < in.size())
                  {
                      sout += (char)strtol(in.substr(i_sz+1,2).c_str(),NULL,16);
                      i_sz += 2;
                  }else sout += in[i_sz];
                  break;
                default: sout += in[i_sz];
            }
          break;
      case TSYS::base64:
          sout.reserve(in.size());
          for( i_sz = 0; i_sz < in.size(); )
          {
            if(in[i_sz] == '\n')    i_sz+=sizeof('\n');
            if((i_sz+3) < in.size())
                if( in[i_sz+1] != '=' )
                {
                  char w_code1 = TSYS::getBase64Code(in[i_sz+1]);

      sout.push_back((TSYS::getBase64Code(in[i_sz])<<2)|(w_code1>>4));
                  if( in[i_sz+2] != '=' )
                  {
                      char w_code2 = TSYS::getBase64Code(in[i_sz+2]);
                      sout.push_back((w_code1<<4)|(w_code2>>2));
                      if( in[i_sz+3] != '=' )

      sout.push_back((w_code2<<6)|TSYS::getBase64Code(in[i_sz+3]));
                  }
                }
            i_sz+=4;
          }
          break;
      case TSYS::Bin:
          sout.reserve(in.size());
          for( i_sz = 0; i_sz < in.size(); i_sz++ )
            sout += TSYS::strMess(((i_sz+1)%16)?"%0.2x ":"%0.2x\n",(unsigned
char)in[i_sz]);
          break;
      default: sout = in;     break;
    }

    return sout;
}

string TSYS::strCompr( const string &in, int lev )
{
    z_stream strm;
```

```
    if( in.empty() )    return "";

    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;

    if( deflateInit(&strm,lev) != Z_OK ) return "";

    uLongf comprLen = deflateBound(&strm,in.size());
    char out[comprLen];

    strm.next_in = (Bytef*)in.data();
    strm.avail_in = (uInt)in.size();
    strm.next_out = (Bytef*)out;
    strm.avail_out = comprLen;

    if( deflate(&strm, Z_FINISH) != Z_STREAM_END )
    {
      deflateEnd(&strm);
      return "";
    }

    comprLen = strm.total_out;

    deflateEnd(&strm);

    return string(out,comprLen);
}

string TSYS::strUncompr( const string &in )
{
    int ret;
    z_stream strm;
    unsigned char out[STR_BUF_LEN];
    string rez;

    if( in.empty() )    return "";

    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;

    if( inflateInit(&strm) != Z_OK )      return "";

    strm.avail_in = in.size();
    strm.next_in = (Bytef*)in.data();
    do
    {
      strm.avail_out = sizeof(out);
      strm.next_out = out;
      ret=inflate(&strm,Z_NO_FLUSH);
      if( ret == Z_STREAM_ERROR || ret == Z_NEED_DICT || ret == Z_DATA_ERROR ||
ret == Z_MEM_ERROR )
          break;
      rez.append((char*)out,sizeof(out)-strm.avail_out);
    } while( strm.avail_out == 0 );

    inflateEnd(&strm);

    if( ret != Z_STREAM_END ) return "";

    return rez;
}

float TSYS::floatLE(float in)
{
#if __BYTE_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
```

```
    union ieee754_le
    {
      float f;
      struct
        {
          unsigned int mantissa:23;
          unsigned int exponent:8;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_be.f = in;
    ieee754_le.ieee.mantissa  = ieee754_be.ieee.mantissa;
    ieee754_le.ieee.exponent  = ieee754_be.ieee.exponent;
    ieee754_le.ieee.negative  = ieee754_be.ieee.negative;

    return ieee754_le.f;
#endif

    return in;
}

float TSYS::floatLErev(float in)
{
#if __BYTE_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      float f;
      struct
        {
          unsigned int mantissa:23;
          unsigned int exponent:8;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_le.f = in;
    ieee754_be.ieee.mantissa  = ieee754_le.ieee.mantissa;
    ieee754_be.ieee.exponent  = ieee754_le.ieee.exponent;
    ieee754_be.ieee.negative  = ieee754_le.ieee.negative;

    return ieee754_be.f;
#endif

    return in;
}

double TSYS::doubleLE(double in)
{
#if __BYTE_ORDER == __BIG_ENDIAN || __FLOAT_WORD_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      double d;
      struct
        {
          unsigned int mantissa1:32;
          unsigned int mantissa0:20;
          unsigned int exponent:11;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_be.d = in;
    ieee754_le.ieee.mantissa0 = ieee754_be.ieee.mantissa0;
    ieee754_le.ieee.mantissa1 = ieee754_be.ieee.mantissa1;
    ieee754_le.ieee.exponent  = ieee754_be.ieee.exponent;
    ieee754_le.ieee.negative  = ieee754_be.ieee.negative;
```

```
    return ieee754_le.d;
#endif

    return in;
}

double TSYS::doubleLErev(double in)
{
#if __BYTE_ORDER == __BIG_ENDIAN || __FLOAT_WORD_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      double d;
      struct
        {
          unsigned int mantissa1:32;
          unsigned int mantissa0:20;
          unsigned int exponent:11;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_le.d = in;
    ieee754_be.ieee.mantissa0 = ieee754_le.ieee.mantissa0;
    ieee754_be.ieee.mantissa1 = ieee754_le.ieee.mantissa1;
    ieee754_be.ieee.exponent  = ieee754_le.ieee.exponent;
    ieee754_be.ieee.negative  = ieee754_le.ieee.negative;

    return ieee754_be.d;
#endif

    return in;
}

long TSYS::HZ()
{
    return sysconf(_SC_CLK_TCK);
}

bool TSYS::cntrEmpty( )
{
    ResAlloc res( nodeRes(), false );
    return mCntrs.empty();
}

double TSYS::cntrGet( const string &id )
{
    ResAlloc res( nodeRes(), false );
    map<string,double>::iterator icnt = mCntrs.find(id);
    if( icnt == mCntrs.end() )      return 0;
    return icnt->second;
}

void TSYS::cntrSet( const string &id, double vl )
{
    ResAlloc res( nodeRes(), true );
    mCntrs[id] = vl;
}

void TSYS::taskCreate( const string &path, int priority, void
*(*start_routine)(void *), void *arg, int wtm, pthread_attr_t *pAttr, bool
*startSt )
{
    int detachStat = 0;
    pthread_t procPthr;
    pthread_attr_t locPAttr, *pthr_attr;
    map<string,STask>::iterator ti;
```

```
    ResAlloc res(taskRes, true);
    for(time_t c_tm = time(NULL); mTasks.find(path) != mTasks.end(); )
    {
      if(time(NULL) >= (c_tm+wtm)) throw TError(nodePath().c_str(),_("Завдання
'%s' вже присутнє!"),path.c_str());
      res.release();
      usleep(10000);
      res.request(true);
    }
    STask &htsk = mTasks[path];
    htsk.path = path;
    htsk.task = start_routine;
    htsk.taskArg = arg;
    htsk.flgs = 0;
    res.release();

    if(pAttr) pthr_attr = pAttr;
    else
    {
      pthr_attr = &locPAttr;
      pthread_attr_init(pthr_attr);
    }
    pthread_attr_setinheritsched(pthr_attr, PTHREAD_EXPLICIT_SCHED);
    struct sched_param prior;
    prior.sched_priority = 0;

    int policy = SCHED_OTHER;
#if __GLIBC_PREREQ(2,4)
    if(priority < 0)    policy = SCHED_BATCH;
#endif
    if(priority > 0 /*&& SYS->user() == "root"*/)    policy = SCHED_RR;
    pthread_attr_setschedpolicy(pthr_attr, policy);
    prior.sched_priority =
vmax(sched_get_priority_min(policy),vmin(sched_get_priority_max(policy),priority
));
    pthread_attr_setschedparam(pthr_attr,&prior);

    try
    {
      pthread_attr_getdetachstate(pthr_attr,&detachStat);
      if(detachStat == PTHREAD_CREATE_DETACHED) htsk.flgs |= STask::Detached;
      int rez = pthread_create(&procPthr, pthr_attr, taskWrap, &htsk);
      if(rez == EPERM)
      {
          mess_warning(nodePath().c_str(),_("No permission for create real-time
policy. Default thread is created!"));
          policy = SCHED_OTHER;
          pthread_attr_setschedpolicy(pthr_attr, policy);
          prior.sched_priority = 0;
          pthread_attr_setschedparam(pthr_attr,&prior);
          rez = pthread_create(&procPthr, pthr_attr, taskWrap, &htsk);
      }
      if(!pAttr) pthread_attr_destroy(pthr_attr);

      if(rez) throw TError(nodePath().c_str(), _("Завдання створило
помилку%d."), rez);

      //> Чекаємо закінчення ініціалізації структури потоку для не відривних
завдань
      while(!(htsk.flgs&STask::Detached) && !htsk.thr) pthread_yield();
      //> Чекаємо запуск статусу
      for(time_t c_tm = time(NULL); !(htsk.flgs&STask::Detached) && startSt &&
!(*startSt); )
      {
          if(time(NULL) >= (c_tm+wtm)) throw
TError(nodePath().c_str(),_("Завдання '%s' запуск відкладений!"),path.c_str());
          usleep(STD_WAIT_DELAY *1000);
      }
    }
```

```
    catch(TError)
    {
      res.request(true);
      mTasks.erase(path);
      res.release();
      throw;
    }
}

void TSYS::taskDestroy( const string &path, bool *endrunCntr, int wtm, bool
noSignal )
{
    ResAlloc res(taskRes, false);
    map<string,STask>::iterator it = mTasks.find(path);
    if(it == mTasks.end()) return;
    pthread_t thr = it->second.thr;
    res.release();

    if(endrunCntr) *endrunCntr = true;
    if(!noSignal) pthread_kill(thr, SIGALRM);

    //> Чекаємо завершення завдання та повторюємо відправлення SIGALRM
    time_t t_tm, s_tm;
    t_tm = s_tm = time(NULL);
    while(!(it->second.flgs&STask::FinishTask))
    {
        if(!noSignal) pthread_kill(thr, SIGALRM);
      time_t c_tm = time(NULL);
        //Контролюємо перерву
        if(wtm && (c_tm > (s_tm+wtm)))
        {
            mess_crit((nodePath()+path+": stop").c_str(),_("Timeouted !!!"));
          throw TError(nodePath().c_str(),_(«завдання '%s' is not
stopped!"),path.c_str());
        }
      //Створюємо повідомлення
        if(c_tm > t_tm+1)  //1sec
        {
            t_tm = c_tm;
            mess_info((nodePath()+path+": stop").c_str(),_("Чекаємо подію..."));

        }
        usleep(STD_WAIT_DELAY*1000);
    }

    if(!(it->second.flgs&STask::Detached)) pthread_join(thr, NULL);

    res.request(true);
    mTasks.erase(it);
}

void *TSYS::taskWrap( void *stas )
{
    //> Беремо тимчасову структуру завдання
    STask *tsk = (STask *)stas;
    pthread_setspecific(TSYS::sTaskKey, tsk);

    //> Запам’ятовуємо параметри виклику
    void *(*wTask) (void *) = tsk->task;
    void *wTaskArg = tsk->taskArg;

    //> Отримуємо поточні політику і пріорітет
    int policy;
    struct sched_param param;
    pthread_getschedparam(pthread_self(), &policy, &param);
    tsk->policy = policy;
    tsk->prior = param.sched_priority;

#if __GLIBC_PREREQ(2,4)
```

```
    //> Отримуємо і завантажуємо CPU установлення
    if(SYS->multCPU() && !(tsk->flgs & STask::Detached))
    {
      tsk->cpuSet = TBDS::genDBGet(SYS->nodePath()+"CpuSet:"+tsk->path);
      cpu_set_t cpuset;
      CPU_ZERO(&cpuset);
      string sval;
      bool cpuSetOK = false;
      for(int off = 0; (sval=TSYS::strParse(tsk->cpuSet,0,":",&off)).size();
cpuSetOK = true)
          CPU_SET(atoi(sval.c_str()),&cpuset);
      if(cpuSetOK) pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t),
&cpuset);
    }
    else if(SYS->multCPU() && (tsk->flgs & STask::Detached)) tsk->cpuSet = "NA";
#endif

    //> Закінчуємо установки та ініціалізуємо індікатор закінчення
    tsk->tid = syscall(SYS_gettid);
    tsk->thr = pthread_self();

    //> Викликаємо робочі завдання
    void *rez = NULL;
    try { rez = wTask(wTaskArg); }
    catch(TError err)
    {
      mess_err(err.cat.c_str(),err.mess.c_str());
      mess_err(SYS->nodePath().c_str(),_("Завдання %u несподівано закінчено
виключенням."),tsk->thr);
    }

    //> Відмічаємо закінчення завдання
    tsk->flgs |= STask::FinishTask;

    //> Переміщуємо об’єкти завдання окремо
    if(tsk->flgs & STask::Detached) SYS->taskDestroy(tsk->path, NULL);

    return rez;
}

void TSYS::taskSleep( int64_t per, time_t cron )
{
    struct timespec sp_tm;
    STask *stsk = (STask*)pthread_getspecific(sTaskKey);

    if(!cron)
    {
      if(!per) per = 1000000000;
      clock_gettime(CLOCK_REALTIME,&sp_tm);
      int64_t end_tm = (int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec;
      int64_t pnt_tm = (end_tm/per + 1)*per;
      do
      {
          sp_tm.tv_sec = pnt_tm/1000000000; sp_tm.tv_nsec = pnt_tm%1000000000;
          if(clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME,&sp_tm,NULL))
      return;
          clock_gettime(CLOCK_REALTIME,&sp_tm);
      }while(((int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec) < pnt_tm);

      if(stsk)
      {
          stsk->tm_beg = stsk->tm_per;
          stsk->tm_end = end_tm;
          stsk->tm_per = (int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec;
      }
    }
    else
    {
      time_t end_tm = time(NULL);
```

```
      while(time(NULL) < cron && usleep(1000000) == 0) ;
      if(stsk)
      {
          stsk->tm_beg = stsk->tm_per;
          stsk->tm_end = 1000000000ll*end_tm;
          stsk->tm_per = 1000000000ll*time(NULL);
      }
    }
}

time_t TSYS::cron( const string &vl, time_t base )
{
    string cronEl, tEl;
    int vbeg, vend, vstep, vm;

    time_t ctm = base?base:time(NULL);
    struct tm ttm;
    localtime_r(&ctm,&ttm);
    ttm.tm_sec = 0;

reload:
    bool isReload = false;

    //> Хвилини check
    cronEl = TSYS::strSepParse(vl,0,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=59; }
      if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_min>=vbeg)?60:0));
      else if((vbeg=vmax(0,vbeg)) < (vend=vmin(59,vend)))
      {
          if(ttm.tm_min < vbeg) vm = vmin(vm,vbeg);
          else if((vstep>1 && ttm.tm_min >= (vbeg+((vend-vbeg)/vstep)*vstep)) ||
(vstep <= 0 && ttm.tm_min >= vend))
            vm = vmin(vm,vbeg+60);
          else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*(((ttm.tm_min+1)-
vbeg)/vstep + ((((ttm.tm_min+1)-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_min+1);
      }
      if(vm == ttm.tm_min+1) break;
    }
    ttm.tm_min = vm;
    mktime(&ttm);

    //> Перевіряємо час
    cronEl = TSYS::strSepParse(vl,1,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=23; }
      if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_hour>vbeg)?24:0));
      else if((vbeg=vmax(0,vbeg)) < (vend=vmin(23,vend)))
      {
          if(ttm.tm_hour < vbeg) vm = vmin(vm,vbeg);
          else if((vstep>1 && ttm.tm_hour > (vbeg+((vend-vbeg)/vstep)*vstep)) ||
(vstep <= 0 && ttm.tm_hour > vend))
            vm = vmin(vm,vbeg+24);
          else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*((ttm.tm_hour-vbeg)/vstep
+ (((ttm.tm_hour-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_hour);
      }
      if(vm == ttm.tm_hour) break;
    }
    isReload = (vm != 200 && ttm.tm_hour!=vm);
```

```
    ttm.tm_hour = vm;
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; goto reload; }

    //> Перевіряємо день
    cronEl = TSYS::strSepParse(vl,2,' ');
    string cronElw = TSYS::strSepParse(vl,4,' ');
    vm = 200;
    if(cronEl != "*")
      for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
      {
          vbeg = vend = -1; vstep = 0;
          sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
          if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=1; vend=31; }
          if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_mday>vbeg)?31:0));
          else if((vbeg=vmax(1,vbeg)) < (vend=vmin(31,vend)))
          {
            if(ttm.tm_mday < vbeg) vm = vmin(vm,vbeg);
            else if((vstep>1 && ttm.tm_mday > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && ttm.tm_mday > vend))
                vm = vmin(vm,vbeg+31);
            else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*((ttm.tm_mday-
vbeg)/vstep + (((ttm.tm_mday-vbeg)%vstep)?1:0)));
            else vm = vmin(vm, ttm.tm_mday);
          }
          if(vm == ttm.tm_mday) break;
       }
    if(cronEl == "*" || (cronElw != "*" && !cronElw.empty()))
      for(int eoff = 0; (tEl=TSYS::strSepParse(cronElw,0,',',&eoff)).size(); )
      {
          vbeg = vend = -1; vstep = 0;
          sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
          if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=6; }
          if(vend < 0) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday +
vbeg+((ttm.tm_wday>vbeg)?7:0));
          else if((vbeg=vmax(0,vbeg)) < (vend=vmin(6,vend)))
          {
            if(ttm.tm_wday < vbeg) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday +
vbeg);
            else if((vstep>1 && ttm.tm_wday > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && ttm.tm_wday > vend))
                vm = vmin(vm, ttm.tm_mday - ttm.tm_wday + vbeg+7);
            else if(vstep>1) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday + vbeg +
vstep*((ttm.tm_wday-vbeg)/vstep + (((ttm.tm_wday-vbeg)%vstep)?1:0)));
            else vm = vmin(vm, ttm.tm_mday);
          }
          if(vm == ttm.tm_mday) break;
      }
    isReload = (vm!=200 && ttm.tm_mday!=vm);
    if(vm <= 31) ttm.tm_mday = vm;
    else { ttm.tm_mday = vm-31; ttm.tm_mon++; }
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; ttm.tm_hour = 0; goto reload; }

    //> Перевіряємо місяць
    cronEl = TSYS::strSepParse(vl,3,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=1; vend=12; }
      if(vend < 0) vm = vmin(vm,vbeg+(((ttm.tm_mon+1)>vbeg)?12:0));
      else if((vbeg=vmax(1,vbeg)) < (vend=vmin(12,vend)))
      {
          if((ttm.tm_mon+1) < vbeg) vm = vmin(vm,vbeg);
          else if((vstep>1 && (ttm.tm_mon+1) > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && (ttm.tm_mon+1) > vend))
            vm = vmin(vm,vbeg+12);
```

```
          else if(vstep>1) vm = vmin( vm, vbeg + vstep*(((ttm.tm_mon+1)-
vbeg)/vstep + ((((ttm.tm_mon+1)-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_mon+1);
      }
      if(vm == (ttm.tm_mon+1)) break;
    }
    isReload = (vm!=200 && ttm.tm_mon!=(vm-1));
    ttm.tm_mon = vm-1;
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; ttm.tm_hour = 0; ttm.tm_mday = 1; goto
reload; }

    return mktime(&ttm);
}

TVariant TSYS::objFuncCall( const string &iid, vector<TVariant> &prms, const
string &user )
{
    // int message(string cat, int level, string mess) - форматуємо системне
повідомлення  <mess> з категорією <cat>, рівнем <level>
    //  cat - повідомлення категорії
    //  level - повідомлення рівня
    //  mess - повідомлення тексту
    if(iid == "message" && prms.size() >= 3)    { message(
prms[0].getS().c_str(), (TMess::Type)prms[1].getI(), "%s",
prms[2].getS().c_str() ); return 0; }
    // int messDebug(string cat, string mess) - форматуємо системне повідомлення
<mess> з категорією <cat> і відповідний рівень
    //  cat - повідомлення категорії
    //  mess - повідомлення тексту
    if(iid == "messDebug" && prms.size() >= 2)  { mess_debug(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messInfo" && prms.size() >= 2)   { mess_info(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messNote" && prms.size() >= 2)   { mess_note(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messWarning" && prms.size() >= 2){ mess_warning(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messErr" && prms.size() >= 2)    { mess_err(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messCrit" && prms.size() >= 2)   { mess_crit(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messAlert" && prms.size() >= 2)  { mess_alert(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messEmerg" && prms.size() >= 2)  { mess_emerg(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    // string system(string cmd, bool noPipe = false) - викликаємо консольні
команди <cmd> для повернення у ОС  результатів з каналів
    //  cmd - текс команди
    //  noPipe - результат блокують для другорядного виклику
    if(iid == "system" && prms.size() >= 1)
    {
      if(prms.size() >= 2 && prms[1].getB()) return
system(prms[0].getS().c_str());
      FILE *fp = popen(prms[0].getS().c_str(),"r");
      if(!fp) return string("");

      char buf[STR_BUF_LEN];
      string rez;
      for(int r_cnt = 0; (r_cnt=fread(buf,1,sizeof(buf),fp)); )
          rez.append(buf,r_cnt);

      pclose(fp);
      return rez;
    }
    // string fileRead( string file ) - Повертаємо <file> контент у рядку.
    if(iid == "fileRead" && prms.size() >= 1)
    {
      char buf[STR_BUF_LEN];
```

```
      string rez;
        int hd = open(prms[0].getS().c_str(),O_RDONLY);
      if(hd != -1)
      {
          for(int len = 0; (len=read(hd,buf,sizeof(buf))) > 0; )
rez.append(buf,len);
          close(hd);
      }
      return rez;
    }
    // int fileWrite( string file, string str, bool append = false ) - Записуємо
<str> до <file>, переміщуємо представлення, або <append>.
    //        Return wrote bytes count.
    if(iid == "fileWrite" && prms.size() >= 2)
    {
      int wcnt = 0, wflags = O_WRONLY|O_CREAT|O_TRUNC;
      string val = prms[1].getS();
      if(prms.size() >= 3 && prms[2].getB()) wflags = O_WRONLY|O_CREAT|O_APPEND;
        int hd = open(prms[0].getS().c_str(), wflags, 0664);
      if(hd != -1)
      {
          wcnt = write(hd,val.data(),val.size());
          close(hd);
      }
      return wcnt;
    }
    // XMLNodeObj XMLNode(string name = "") - створюємо XML об’єкт вузлів з
іменем <name>
    //  name - XML ім’я вузлу
    if(iid == "XMLNode") return new XMLNodeObj((prms.size()>=1) ? prms[0].getS()
: "");
    // string cntrReq(XMLNodeObj req, string stat = "") - запит інтерфейсу
управління до системи через XML
    //  req - запити XML вузлів
    //  stat - переміщуємо WebSCADA-робочу станцію для запиту
    if(iid == "cntrReq" && prms.size() >= 1)
    {
      XMLNode req;
      if(!dynamic_cast<XMLNodeObj*>(prms[0].getO())) return string(_("1:Запит не
об’єктний!"));
      ((XMLNodeObj*)prms[0].getO())->toXMLNode(req);
      string path = req.attr("path");
      if(prms.size() < 2 || prms[1].getS().empty())
      {
          req.setAttr("user",user);
          cntrCmd(&req);
      }
      else
      {
          req.setAttr("path","/"+prms[1].getS()+path);
          transport().at().cntrIfCmd(req,"cntrReq");
          req.setAttr("path",path);
      }
      ((XMLNodeObj*)prms[0].getO())->fromXMLNode(req);
      return string("0");
    }
    // string sleep(int tm, int ntm = 0) – викликаємо завдання переходу до
сплячого режиму через <tm> секунд та <ntm> наносекунд.
    //  tm - чекаємо цей час у секундах
    //  ntm - чекаємо цей час у наносекундах
    if(iid == "sleep" && prms.size() >= 1)
    {
      struct timespec sp_tm;
      sp_tm.tv_sec = prms[0].getI();
      sp_tm.tv_nsec = (prms.size() >= 2) ? prms[1].getI() : 0;
      int rez = clock_nanosleep(CLOCK_REALTIME,0,&sp_tm,NULL);
      return rez;
    }
```

```
    // int time(int usec) – повертаємо абсолютний час у секундах від 1/1/1970 та
в мікросекундах, якщо <usec> задано
    //  usec - microсекунди of time
    if(iid == "time")
    {
      if(prms.empty()) return (int)time(NULL);
      int64_t tm = curTime();
      prms[0].setI(tm%1000000); prms[0].setModify();
      return (int)(tm/1000000);
    }
    // int localtime(int fullsec, int sec, int min, int hour, int mday, int
month, int year, int wday, int yday, int isdst)
    //      - повертаємо повну дату, базуємо на абсолютному часі у секундах
<fullsec> від 1.1.1970
    //  fullsec – час джерела у секундах від 1.1.1970
    //  sec - секунди
    //  min - хвилини
    //  hour - часи
    //  mday - дні місяця
    //  month - місяці
    //  year - роки
    //  wday - дні неділі
    //  yday - дні року
    //  isdst - відмітка про літній час
    if(iid == "localtime" && prms.size() >= 2)
    {
      time_t tm_t = prms[0].getI();
      struct tm tm_tm;
      localtime_r(&tm_t,&tm_tm);

      prms[1].setI(tm_tm.tm_sec); prms[1].setModify();
      if(prms.size() >= 3)    { prms[2].setI(tm_tm.tm_min); prms[2].setModify();
}
      if(prms.size() >= 4)    { prms[3].setI(tm_tm.tm_hour);
prms[3].setModify(); }
      if(prms.size() >= 5)    { prms[4].setI(tm_tm.tm_mday);
prms[4].setModify(); }
      if(prms.size() >= 6)    { prms[5].setI(tm_tm.tm_mon); prms[5].setModify();
}
      if(prms.size() >= 7)    { prms[6].setI(1900+tm_tm.tm_year);
prms[6].setModify(); }
      if(prms.size() >= 8)    { prms[7].setI(tm_tm.tm_wday);
prms[7].setModify(); }
      if(prms.size() >= 9)    { prms[8].setI(tm_tm.tm_yday);
prms[8].setModify(); }
      if(prms.size() >= 10)   { prms[9].setI(tm_tm.tm_isdst);
prms[9].setModify(); }
      return 0;
    }
    // string strftime(int sec, string form = "%Y-%m-%d %H:%M:%S") – перетворює
абсолютний час  <sec> у рядок формату <form>
    //  sec – час у секундах від 1.1.1970
    //  form – вихідний форматований рядок
    if(iid == "strftime" && !prms.empty())
    {
      time_t tm_t = prms[0].getI();
      struct tm tm_tm;
      localtime_r(&tm_t,&tm_tm);
      char buf[1000];
      int rez = strftime(buf, sizeof(buf), (prms.size()>=2) ?
prms[1].getS().c_str() : "%Y-%m-%d %H:%M:%S", &tm_tm);
      return (rez>0) ? string(buf,rez) : "";
    }
    // int strptime(string str, string form = "%Y-%m-%d %H:%M:%S") - повертає
час  у секундах від of 1/1/1970,
    //      базується на запису рядку часу <str>, відповідно до вказаного
шаблону  <form>
    //  str - джерело часу у рядку
    //  form - рядки часу у форматі POSIX-функцій "strptime"
```

```
    if(iid == "strptime" && !prms.empty())
    {
      struct tm stm;
      stm.tm_isdst = -1;
      strptime(prms[0].getS().c_str(), (prms.size()>=2) ? prms[1].getS().c_str()
: "%Y-%m-%d %H:%M:%S", &stm);
      return (int)mktime(&stm);
    }
    // int cron(string cronreq, int base = 0) - повертає час , планується у
форматі стандартного Cron <cronreq>,
    //      початок від основного часу<base> або від поточного, якщо основа не
вказана
    //  cronreq - розповсюджений у стандартному форматі Cron
    //  base - основний час
    if(iid == "cron" && !prms.empty())
      return (int)cron(prms[0].getS(), (prms.size()>=2) ? prms[1].getI() : 0);
    // string strFromCharCode(int char1, int char2, int char3, ...) - створює
рядок з кодових символів
    //  char1, char2. char3 - кодові символи
    if(iid == "strFromCharCode")
    {
      string rez;
      for(unsigned i_p = 0; i_p < prms.size(); i_p++)
          rez += (unsigned char)prms[i_p].getI();
      return rez;
    }
    // string strCodeConv( string src, string fromCP, string toCP ) - Текстовий
рядок перекодовує з кодової сторінки <fromCP> до кодової сторінки <toCP>.
    //  src - source text;
    //  fromCP - з кодової сторінки , порожньої для внутрішньої кодової сторінки
;
    //  toCP - до кодової сторінки , порожньої для внутрішньої кодової сторінки
.
    if(iid == "strCodeConv" && prms.size() >= 3)
      return Mess->codeConv((prms[1].getS().size() ? prms[1].getS() : Mess-
>charset()),
                  (prms[2].getS().size() ? prms[2].getS() : Mess->charset()),
prms[0].getS());

    return TCntrNode::objFuncCall(iid,prms,user);
}

void TSYS::cntrCmdProc( XMLNode *opt )
{
    char buf[STR_BUF_LEN];

    //Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      snprintf(buf,sizeof(buf),_("%s station:
\"%s\""),PACKAGE_NAME,name().c_str());
      ctrMkNode("WSCADA_cntr",opt,-1,"/",buf,R_R_R_);
      if(ctrMkNode("branches",opt,-1,"/br","",R_R_R_))
          ctrMkNode("grp",opt,-
1,"/br/sub_",_("Subsystem"),R_R_R_,"root","root",1,"idm","1");
      if(TUIS::icoPresent(id())) ctrMkNode("img",opt,-1,"/ico","",R_R_R_);
      if(ctrMkNode("area",opt,-1,"/gen",_("Station"),R_R_R_))
      {
          ctrMkNode("fld",opt,-
1,"/gen/id",_("ID"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/stat",_("Station"),RWRWR_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/prog",_("Program"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/ver",_("Version"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/host",_("Ім'я
хосту"),R_R_R_,"root","root",1,"tp","str");
```

```
          ctrMkNode("fld",opt,-1,"/gen/user",_("Користувач
системи"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/sys",_("Операційна
система"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/frq",_("Частота
(MHZ)"),R_R_R_,"root","root",1,"tp","real");
          ctrMkNode("fld",opt,-1,"/gen/clk_res",_("Значення годинника реального
часу"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/in_charset",_("Внутрішній набір
символів"),R_R___,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/config",_("Кофігураційний
файл"),R_R___,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/workdir",_("Робоча
директорія"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/icodir",_("Директорія
іконок"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/moddir",_("Директорія
модулів"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/wrk_db",_("Робоча база
даних"),RWRWR_,"root","root",4,"tp","str","dest","select","select","/db/list",
            "help",_("Адрес робочої бази даних у форматі [<БД module>.<БД
name>].\n Змінюємо ці поля якщо необхідно зберегти або завантажити усю систему з
іншої  БД."));
          ctrMkNode("fld",opt,-1,"/gen/saveExit",_("Збереження змін у системі
перед виходом"),RWRWR_,"root","root",2,"tp","bool",
            "help",_("Обираємо автоматичне збереження системи до  БД перед
виходом."));
          ctrMkNode("fld",opt,-1,"/gen/savePeriod",_("Збереження періоду роботи
у системі "),RWRWR_,"root","root",2,"tp","dec",
            "help",_("Використовуємо нульовий період (секунди) для періодичного
збереження  змін частин системи у БД."));
          ctrMkNode("fld",opt,-
1,"/gen/lang",_("Language"),RWRWR_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/baseLang",_("Базова мова тексту
змінних"),RWRWR_,"root","root",5,"tp","str","len","2","dest","sel_ed","select","
/gen/baseLangLs",
            "help",_("Мультимовність для змінного тексту підтримки для вибору
базової мови."));
          if(ctrMkNode("area",opt,-1,"/gen/mess",_("Повідомлення"),R_R_R_))
          {
            ctrMkNode("fld",opt,-1,"/gen/mess/lev",_("Найменший
рівень"),RWRWR_,"root","root",3,
                "tp","dec","len","1","help",_("Повідомлення найменшого рівня для
відображення процесів, які відбуваються у системі."));
            ctrMkNode("fld",opt,-1,"/gen/mess/log_sysl",_("To
syslog"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_stdo",_("To
stdout"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_stde",_("To
stderr"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_arch",_("До
архіву"),RWRWR_,"root","root",1,"tp","bool");
          }
      }
      if(ctrMkNode("area",opt,-1,"/subs",_("Підсистема")))
          ctrMkNode("list",opt,-
1,"/subs/br",_("Підсистеми"),R_R_R_,"root","root",3,"idm","1","tp","br","br_pref
","sub_");
      if(ctrMkNode("area",opt,-1,"/tasks",_("Tasks"),R_R___))
          if(ctrMkNode("table",opt,-
1,"/tasks/tasks",_("Завдання"),RWRW__,"root","root",2,"key","path",
            "help",!multCPU()?"":_("Для CPU встановлюємо рядок номерів
процесорів використання, відокремлений символом':'.\n"
                                   "CPU починаємо з 0.")))
          {
            ctrMkNode("list",opt,-
1,"/tasks/tasks/path",_("Path"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/thrd",_("Поток"),R_R___,"root","root",1,"tp","str");
```

```
            ctrMkNode("list",opt,-
1,"/tasks/tasks/tid",_("TID"),R_R___,"root","root",1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/stat",_("Статус"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/plc",_("Політика"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/prior",_("Prior."),R_R___,"root","root",1,"tp","dec");
#if __GLIBC_PREREQ(2,4)
            if(multCPU())
                ctrMkNode("list",opt,-1,"/tasks/tasks/cpuSet",_("CPU
встановлення"),RWRW__,"root","root",1,"tp","str");
#endif
          }
      if( !cntrEmpty() && ctrMkNode("area",opt,-1,"/cntr",_("Країна")) )
          if( ctrMkNode("table",opt,-
1,"/cntr/cntr",_("Counters"),R_R___,"root","root") )
          {
            ctrMkNode("list",opt,-
1,"/cntr/cntr/id","ID",R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/cntr/cntr/vl",_("Value"),R_R___,"root","root",1,"tp","real");
          }
      if( ctrMkNode("area",opt,-1,"/hlp",_("Help"),R_R___) )
          ctrMkNode("fld",opt,-1,"/hlp/g_help",_("Опції
допомоги"),R_R___,"root","root",3,"tp","str","cols","90","rows","10");
      return;
    }

    //Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/ico" && ctrChkNode(opt))
    {
      string itp;
      opt->setText(TSYS::strEncode(TUIS::icoGet(id(),&itp),TSYS::base64));
      opt->setAttr("tp",itp);
    }
    else if(a_path == "/gen/host" && ctrChkNode(opt))opt->setText(host());
    else if(a_path == "/gen/sys" && ctrChkNode(opt))
    {
      utsname ubuf; uname(&ubuf);
      opt->setText(string(ubuf.sysname)+"-"+ubuf.release);
    }
    else if(a_path == "/gen/user" && ctrChkNode(opt))opt->setText(mUser);
    else if(a_path == "/gen/prog" && ctrChkNode(opt))opt->setText(PACKAGE_NAME);
    else if(a_path == "/gen/ver" && ctrChkNode(opt))  opt->setText(VERSION);
    else if(a_path == "/gen/id" && ctrChkNode(opt))   opt->setText(id());
    else if(a_path == "/gen/stat")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(name());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setName(opt->text());
    }
    else if(a_path == "/gen/frq" && ctrChkNode(opt))  opt-
>setText(TSYS::real2str((float)sysClk()/1000000.,6));
    else if(a_path == "/gen/clk_res" && ctrChkNode(opt))
    {
      struct timespec tmval;
      clock_getres(CLOCK_REALTIME,&tmval);
      opt->setText(TSYS::time2str(1e-3*tmval.tv_nsec));//
TSYS::real2str((float)tmval.tv_nsec/1000000.,4));
    }
    else if(a_path == "/gen/in_charset" && ctrChkNode(opt))opt->setText(Mess-
>charset());
    else if(a_path == "/gen/config" && ctrChkNode(opt))          opt-
>setText(mConfFile);
    else if(a_path == "/gen/wrk_db" )
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt-
>setText(mWorkDB);
```

```
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setWorkDB(opt-
>text());
    }
    else if(a_path == "/gen/saveExit")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(
int2str(saveAtExit()) );
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setSaveAtExit(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/gen/savePeriod")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(
int2str(savePeriod()) );
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setSavePeriod(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/gen/workdir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(workDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setWorkDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/icodir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(icoDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setIcoDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/moddir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(modDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setModDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/lang")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(Mess-
>lang());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLang(opt-
>text());
    }
    else if(a_path == "/gen/baseLang")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(Mess-
>lang2CodeBase());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess-
>setLang2CodeBase(opt->text());
    }
    else if(a_path == "/gen/baseLangLs" && ctrChkNode(opt))
    {
      opt->childAdd("el")->setText(Mess->lang2Code());
      if(!Mess->lang2CodeBase().empty() && Mess->lang2CodeBase() != Mess-
>lang2Code())
          opt->childAdd("el")->setText(Mess->lang2CodeBase());
      opt->childAdd("el")->setText("");
    }
    else if(a_path == "/gen/mess/lev")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt-
>setText(TSYS::int2str(Mess->messLevel()));
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess-
>setMessLevel(atoi(opt->text().c_str()));
    }
    else if(a_path == "/gen/mess/log_sysl")
    {
```

```
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x01)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x01:Mess->logDirect()&(~0x01) );
    }
    else if(a_path == "/gen/mess/log_stdo")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x02)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x02:Mess->logDirect()&(~0x02) );
    }
    else if(a_path == "/gen/mess/log_stde")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x04)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x04:Mess->logDirect()&(~0x04) );
    }
    else if(a_path == "/gen/mess/log_arch")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x08)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x08:Mess->logDirect()&(~0x08) );
    }
    else if((a_path == "/br/sub_" || a_path == "/subs/br") &&
ctrChkNode(opt,"get",R_R_R_,"root","root",SEC_RD))
    {
      vector<string> lst;
      list(lst);
      for(unsigned i_a=0; i_a < lst.size(); i_a++)
          opt->childAdd("el")->setAttr("id",lst[i_a])-
>setText(at(lst[i_a]).at().subName());
    }
    else if(a_path == "/tasks/tasks")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root","root"))
      {
          XMLNode *n_path     = ctrMkNode("list",opt,-
1,"/tasks/tasks/path","",R_R___,"root","root");
          XMLNode *n_thr      = ctrMkNode("list",opt,-
1,"/tasks/tasks/thrd","",R_R___,"root","root");
          XMLNode *n_tid      = ctrMkNode("list",opt,-
1,"/tasks/tasks/tid","",R_R___,"root","root");
          XMLNode *n_stat     = ctrMkNode("list",opt,-
1,"/tasks/tasks/stat","",R_R___,"root","root");
          XMLNode *n_plc      = ctrMkNode("list",opt,-
1,"/tasks/tasks/plc","",R_R___,"root","root");
          XMLNode *n_prior    = ctrMkNode("list",opt,-
1,"/tasks/tasks/prior","",R_R___,"root","root");
          XMLNode *n_cpuSet   = (multCPU() ? ctrMkNode("list",opt,-
1,"/tasks/tasks/cpuSet","",RWRW__,"root","root") : NULL);

          ResAlloc res(taskRes,false);
          for(map<string,STask>::iterator it = mTasks.begin(); it !=
mTasks.end(); it++)
          {
            if(n_path)  n_path->childAdd("el")->setText(it->first);
            if(n_thr)   n_thr->childAdd("el")->setText(TSYS::uint2str(it-
>second.thr));
            if(n_tid)   n_tid->childAdd("el")->setText(TSYS::int2str(it-
>second.tid));
            if(n_stat)
            {
                int64_t tm_beg = 0, tm_end = 0, tm_per = 0;
                for(int i_tr = 0; tm_beg == tm_per && i_tr < 2; i_tr++)
                { tm_beg = it->second.tm_beg; tm_end = it->second.tm_end; tm_per
= it->second.tm_per; }
```

```
                XMLNode *cn = n_stat->childAdd("el");
                if(it->second.flgs&STask::FinishTask) cn->setText(_("Закінчено.
"));
                if(tm_beg && tm_beg < tm_per)
                  cn->setText(cn->text()+TSYS::strMess(_("Останій: %s.
Завантажено: %3.1f% (%s з %s)"),
                    time2str((time_t)(1e-9*tm_per),"%d-%m-%Y
%H:%M:%S").c_str(), 100*(double)(tm_end-tm_beg)/(double)(tm_per-tm_beg),
                    time2str(1e-3*(tm_end-tm_beg)).c_str(), time2str(1e-
3*(tm_per-tm_beg)).c_str()));
            }
            if(n_plc)
            {
                string plcVl = _("Стандартний");
                if(it->second.policy == SCHED_RR) plcVl = _("Кругова система
");
#if __GLIBC_PREREQ(2,4)
                if(it->second.policy == SCHED_BATCH) plcVl = _("Style
\"batch\"");
#endif
                n_plc->childAdd("el")->setText(plcVl);
            }
            if(n_prior) n_prior->childAdd("el")->setText(TSYS::int2str(it-
>second.prior));
            if(n_cpuSet)      n_cpuSet->childAdd("el")->setText(it-
>second.cpuSet);
          }
      }
#if __GLIBC_PREREQ(2,4)
      if(multCPU() && ctrChkNode(opt,"set",RWRW__,"root","root",SEC_WR) && opt-
>attr("col") == "cpuSet")
      {
          ResAlloc res(taskRes,true);
          map<string,STask>::iterator it = mTasks.find(opt->attr("key_path"));
          if(it == mTasks.end()) throw TError(nodePath().c_str(),_("Не
представлене завдання '%s'."));
          if(it->second.flgs & STask::Detached) return;

          it->second.cpuSet = opt->text();

          cpu_set_t cpuset;
          CPU_ZERO(&cpuset);
          string sval;
          for(int off = 0; (sval=TSYS::strParse(it-
>second.cpuSet,0,":",&off)).size(); )
            CPU_SET(atoi(sval.c_str()),&cpuset);
          int rez = pthread_setaffinity_np(it->second.thr, sizeof(cpu_set_t),
&cpuset);
          res.release();
          TBDS::genDBSet(nodePath()+"CpuSet:"+it->first,opt->text());
          if(rez == EINVAL && opt->text().size()) throw
TError(nodePath().c_str(),_("Не встановлено жодних дозволених процесорів."));
          if(rez && opt->text().size()) throw TError(nodePath().c_str(),_("CPU
установки для потоку помилкові."));
      }
#endif
    }
    if(!cntrEmpty() && a_path == "/cntr/cntr" &&
ctrChkNode(opt,"get",R_R___,"root","root"))
    {
      XMLNode *n_id     = ctrMkNode("list",opt,-
1,"/cntr/cntr/id","",R_R___,"root","root");
      XMLNode *n_vl     = ctrMkNode("list",opt,-
1,"/cntr/cntr/vl","",R_R___,"root","root");

      ResAlloc res( nodeRes(), false );
      for(map<string,double>::iterator icnt = mCntrs.begin(); icnt !=
mCntrs.end(); icnt++)
      {
```

```
          if(n_id)      n_id->childAdd("el")->setText(icnt->first);
          if(n_vl)      n_vl->childAdd("el")->setText(TSYS::real2str(icnt-
>second));
       }
    }
    else if(a_path == "/hlp/g_help" &&
ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD)) opt->setText(optDescr());
    else TCntrNode::cntrCmdProc(opt);
}
```