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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи управління відеоресурсами на основі CMR”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Арсірій О.Ю.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

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

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

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

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

Арсірію Олегу Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи управління відеоресурсами на основі CMR*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи управління відеоресурсами на основі CMR*

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

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

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Лисенко І.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Арсірій О.Ю.
(прізвище та ініціали)

АНОТАЦІЯ

Арсірій О.Ю. Програмне забезпечення системи управління відеоресурсами на основі CMR. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління відеоресурсами на основі CMR.

Метою розробки є програмне забезпечення системи управління відеоресурсами на основі CMR.

Результат роботи – програмна реалізація системи управління відеоресурсами на основі CMR.

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

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

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

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, управління відео ресурсами, CMR

ABSTRACT

Arsiriy O.Yu. Software for a video resource management system based on CMR. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a video resource management system based on CMR.

The purpose of the development is software for a video resource management system based on CMR.

The result of the work is a software implementation of a video resource management system based on CMR.

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 are provided.

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

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

Keywords: computer engineering, video resource management, CMR

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БПД	–	бездротова передача даних
ПЗ	–	програмне забезпечення
СПД	–	системи передачі даних
ACK	–	повідомлення підтвердження прийому
ARQ	–	протокол повторної передачі даних
BPSK	–	Binary phase-shift keying
FFD	–	повнофункціональний пристрій
GFSK	–	Gaussian frequency-shift keying
MAC	–	шар механізму доступу
NACK	–	повідомлення непідтвердження прийому
OSI	–	мережна модель
P2P	–	однорангові мережі
PAN	–	персональна мережа
PPS	–	Portable Protocol Stack
RFD	–	пристрій з полегшеними функціями
TDMA	–	часовий поділ
Wi-Fi	–	бездротова технологія

ВСТУП

Актуальність теми. Відеозв'язок – найбільш природний і ефективний спосіб комунікацій, причому в побуті він застосовується усе ширше – навіть люди літнього віку все частіше спілкуються «по відео», наприклад за допомогою сервісу Skype. У бізнес-середовищу багато хто продовжує вважати, що відео – це складно, а проблема його більше широкого поширення замикається сама на себе (часто відео не використовується, тому що цей інструмент доступний не всім колегам, його немає в партнерів та ін.). У середньому типовий менеджер проводить три роки свого життя в літаку, приблизно стільки ж часу – чекаючи рейсів і пересаджень, три місяці – у пошуках паркування. Використання відеокомунікацій скорочує потребу в поїздках, а більшість співробітників, які стали використовувати цей вид дистанційного спілкування, відзначають, що стали краще висипатися, більше займатися спортом, більш правильно харчуватися, перестали випробовувати стрес. Іншим словами, якість їхнього життя істотно покращилося. Все це, природно, позитивно позначається й на ефективності роботи співробітників, а також підвищує їхню лояльність – багатьох тільки обрадує можливість попрацювати, не залишаючи свого заміського будинку. Крім того, завдяки більше широкому використанню відео-зв'язку, компанії одержують можливість скоротити витрати на відрядження співробітників, оренду офісних площ і т.д. Відео істотно підвищує й ефективність сприйняття інформації. 95% тих, хто переглянув відео, пам'ятають його основне посилання, у той час як після прочитання тексту даний показник становить лише 10%. Крім того перегляд відео поліпшує розуміння продукту або сервісу на 74%. 80% онлайн-відвідувачів будуть дивитися відео й тільки 20% прочитають тексти цілком.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи управління відеоресурсами на основі CMR.

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

– Огляд існуючих систем управління відеоресурсами на основі CMR.

– Дослідження системи управління відеоресурсами на основі CMR.

– Програмна реалізація системи управління відеоресурсами на основі CMR.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі управління відеоресурсами на основі CMR.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління відеоресурсами на основі CMR, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Одна з тенденцій – вихід систем відеоспівробітництва за межі переговірних кімнат. Співробітникам все частіше доводиться брати участь у нарадах з будинку, готелів, перебуваючи в дорозі, і завдання постачальників рішень – забезпечити їм можливість повноцінної участі в роботі компанії незалежно від місцезнаходження. Ефективна робота в команді – гарантія динамічного розвитку бізнесу, а сьогодні члени команди всі частіше розташовуються за межами периметра компанії.

Забезпечення участі в сеансах відеозв'язку (ВКЗ) з мобільних пристроїв – етап уже реалізований: всі провідні виробники мають у своєму портфелі програмні клієнти для основних мобільних платформ і усе більше користувачів підключаються до ВКЗ із мобільних пристроїв. Пальма першості – за платформою iOS (відповідні пристрої як термінали ВКЗ використовують близько 28% замовників), на другому місці – Android (20%), на третьому – Windows (9%). Представлено ефектне рішення, що дозволяє перевести сеанс ВКЗ із планшета iPad на великий екран – наприклад, на встановлену в кімнаті групову систему ВКЗ. При цьому планшет перетворюється в пульт керування системою, крім того, його можна використовувати для передачі діаграм, таблиць і іншої додаткової інформації в ході сеансу відеозв'язку.

Важливим кроком розвитку ВКЗ є поява (завдяки технології WebRTC) можливості здійснення сеансу відеозв'язку із браузера без необхідності установки якого-небудь спеціального додатка. Хоча поки WebRTC задіють тільки 10% користувачів ВКЗ, експерти єдині в оцінці, що ця технологія дуже вплине на галузь ВКЗ і уніфікованих комунікацій у цілому. Зокрема, WebRTC дозволить програмістам легко розробляти нові додатки, що запускаються безпосередньо з

Web-сторінки, а також вбудовувати можливості мультимедіа-комунікацій у різні програмні системи. Не випадково підтримка WebRTC реалізується в усі більшому числі продуктів постачальників відеосистем.

Як показують дослідження, замовники все частіше хочуть проводити сеанси ВКЗ не в спеціально обладнаних кімнатах, а в непідготовлених приміщеннях – наприклад, в офісах, організованих за принципом відкритого простору. У цьому зв'язку виникає проблема, як позбутися від сторонніх шумів. Компанія Polycom представила рішення по формуванню «акустичного міхура» – Acoustic Bubble, завдяки якому співробітник, що перебуває на «відкритому просторі», відчує себе так, начебто він сидить у закритому кабінеті. Цього року компанія анонсувала новий варіант – акустичний купол Acoustic Fence, що призначений для захисту вже не персональної системи ВКЗ, а груповий і реалізований як частина функціонала кодеків Group Series.

1.2 Область застосування

Для спільної роботи, сучасна інфраструктура конференц-зв'язку повинна функціонувати непомітно для користувача, а процес організації конференції повинен бути інтуїтивно зрозумілий. Ідуть у минуле часи, коли для організації й проведення конференцій було потрібно втручання спеціального адміністратора, – сучасні рішення здатні забезпечити проведення конференцій у повністю автоматичному режимі.

Споконвічно системи ВКЗ використовувалися в основному для проведення територіально розподілених нарад, при цьому співробітники звичайно збиралися в конференц-залах або переговірних кімнатах, а час і процедура конференцій жорстко регламентувалися. Відповідні ВКЗ одержали назву запланованих (Scheduled Conference). Вони дозволяли гарантувати, що ресурси сервера конференцій будуть доступні для заданих абонентів у певний час.

Але сьогодні бізнес-конференції повинні надавати більше волі учасникам, у тому числі у виборі місця й часу проведення. Відповідно, все більшою популярністю користуються конференції на вимогу (Ad-hoc Conference), які збираються «на льоту». Це, як правило, невеликі конференції, які створюються об'єднанням (перекладом на сервер ВКЗ) декількох з'єднань «крапка – крапка». Росте популярність і ще одного, нового варіанта, що часто називають «віртуальною кімнатою». Така кімната звичайно використовується для зустрічей, проведених на регулярній основі, а для входу (підключення) потрібно набрати заздалегідь певний номер або ввести URL.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління відеоресурсами на основі CMR, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Системи відеоконференцій різних типів уже активно використовується багатьма компаніями. Незалежно від типу бізнесу, керівники вже оцінили не тільки віддачу від економії на відрядженнях, а й підвищення ефективності бізнесу від поліпшення якості комунікацій між співробітниками.

Приміром, грамотно підібране рішення створює ефект присутності потрібного співробітника в переговірній кімнаті, навіть коли він перебуває віддалено й бере участь у конференції зі свого смартфона або планшета. А швидке створення робочої групи співробітників для оперативної наради стає питанням декількох натискань клавіш на ноутбуці або мобільному пристрої.

Всі ці зручності працюють тільки в тому випадку, коли система грамотно підібрана під особливості бізнесу даної конкретної компанії. Тому, питання правильного проектування системи є пріоритетним на самому початковому етапі підбора послуги або встаткування.

Незважаючи на заяви виробників відеоконференцій про те, що все встаткування сумісне на рівні протоколів, все-таки зустрічаються тут і «підводні камені». А саме: сумісність відбувається на рівні «спілкування по відео в HD-Якості», а от щоб реалізувати безліч додаткових опцій (досить важливих!), все-таки краще підбирати комплексне рішення від одного виробника.

Перевагами рішення від одного виробника можуть бути: використання єдиного планувальника відеонарад із системою нагадувань по електронній пошті, відображення доступності для відеодзвінка або неприступності співробітника в

інтерфейсі користувача, єдиний чат, можливість створення власних конференцій кожним користувачем віртуальних переговірних кімнат) і інші.

Досить вибрати підходящий тип зі списку основних, що зустрічається на ринку:

– Повністю хмарне рішення.

– Хмарно-апаратне рішення.

– Хмарне або хмарно-апаратне рішення з підтримкою публічних засобів відеоспілкування.

– Програмне серверне рішення (для установки в мережі замовника або в дата-центрі провайдеру).

– Повністю апаратне рішення (класична інфраструктура).

Розглянемо докладніше переваги й недоліки кожного типу рішень і приведемо приклади вдалої реалізації.

Повністю хмарне рішення

Рішення покликане об'єднати всіх співробітників компанії в єдину ВКЗ-платформу. На 100% реалізує концепцію BYOD (Bring Your Own Device), що уможливлює використання особистих або корпоративних комп'ютерів/ноутбуків, смартфонів і планшетів як пристрої ВКЗ-зв'язку.

Переваги хмарного рішення: відсутні витрати на апаратну інфраструктуру, а також не потрібне участь технічного фахівця для налаштування й обслуговування системи. Звичайно ліцензується передплата на 1 рік із продовженням.

Відмінний приклад успішного хмарного рішення: Lifesize Cloud.

Вартість мінімального рішення на 10 постійних учасників = $5624 у рік ($47 на 1 чоловік на місяць).

Lifesize Cloud заміняє апаратну інфраструктуру в 7-10 разів більшої вартості початкової закупівлі. А також виключає витрати на регулярне обслуговування, відновлення сервісних ліцензій (10% вартості встаткування в рік), зарплату системному адміністраторові.

Таким чином, витрати на хмарне рішення прийнятні при всіх перевагах такого підходу.

Хмарно-апаратне рішення

Для інсталяції в конференц-зали та переговорні кімнати часто потрібні якісні зовнішні камери – потрібний гарний оптичний зум, керування наведенням камери на спікера, а також якісні мікрофонні системи з можливістю охвату середнього й великого приміщення. У таких випадках неможливо обійтися без апаратних класичних кодеків ВКЗ із якісними комплектними камерами й мікрофонами.

При цьому, такі системи можуть прекрасно працювати із хмарними рішеннями, поєднуючись із ними в єдине ціле. Хмарне рішення в такому випадку створює можливість багатобічної конференції й підключає учасників з мобільними пристроями.

Приклад кодека з камерою й мікрофоном: Lifesize Icon 400.

Вартість Lifesize Icon 400 для переговірної кімнати або конференц-залу становить $3770 (у вартість включена сервісна ліцензія на 1 рік). Додаємо хмарне рішення Lifesize Cloud на 10 учасників $5624.

Такий підхід поєднує зручність класичних апаратних систем і простоту з розширеною функціональністю хмарного рішення.

Хмарне або хмарно-апаратне рішення з підтримкою публічних засобів відеоспілкування

Самий новий і самий затребуваний на сьогодні тип ВКЗ-системи. До всіх переваг, описаним у п. 1-3, додається можливість підтримки публічних засобів відеоспілкування (Microsoft Lync, Google Hangouts, Cisco Jabber, Skype, Lifesize Cloud, WebEx, GoToMeeting і т.п.)

Для того, щоб усе ця працювало, до хмарного рішення Lifesize Cloud на 10 учасників $5624 додаємо кодек (з камерою й мікрофоном): Lifesize Icon Flex $3040 (у вартість включена сервісна ліцензія на 1 рік).

Програмне серверне рішення (для установки в мережі замовника або в дата-центрі провайдеру)

Таке рішення звичайно вимагає наявності власного сервера в замовника, куди можна встановити ПЗ для ВКЗ (звичайно на основі віртуальної машини).

Альтернативні варіанти: покупка серверних віртуальних ресурсів у провайдеру й розміщення там віртуальної машини або установка власного сервера в дата-центрі провайдеру. Можуть бути використані обидва варіанти одночасно.

Фактично, при такому підході замовник одержує повністю контрольовану хмару у власній мережі.

Використовуючи кілька віртуальних машин, як усередині мережі, так і зовні, можна поєднувати їхні ресурси й гнучко управляти інтернет-трафіком і маршрутизацією.

Переваги такого підходу оцінять великі корпорації, сервіси-провайдери, компанії із сотнями людин персоналу, які повинні бути об'єднані відеозв'язком. А також ті, хто активно використовує Microsoft Lync і хоче розширити його функціональність якісним відеозв'язком. Всі ці переваги використовує рішення Pexip Infinity.

Pexip Infinity також сумісно із традиційними апаратними системами по стандартних протоколах SIP/H.323. Працює з найвідомішими публічними засобами відеоспілкування Microsoft Lync і Skype.

Pexip Infinity ліцензується залежно від кількості учасників або використовуваних портів. Розрахунок надається індивідуально під поставлене завдання.

Повністю апаратне рішення (класична інфраструктура)

Доступний великий вибору різних кодеків, серверів і іншого встаткування, що дозволяє побудувати повністю апаратний комплекс. Устаткування Lifesize відмінно вирішує дані завдання.

Таким чином, важливо зрозуміти особливості експлуатації системи замовником і запропонувати найбільш підходяще рішення під індивідуальні особливості бізнесу. Використання хмарних і програмних рішень гарантує оперативне відновлення для підтримки всіх нових функцій, що з'являються, нових версій відеокодеків, протоколів і т.п. Використання ж окремих апаратних елементів додає зручності експлуатації в переговірні кімнати й конференц-зали.

TrueConf

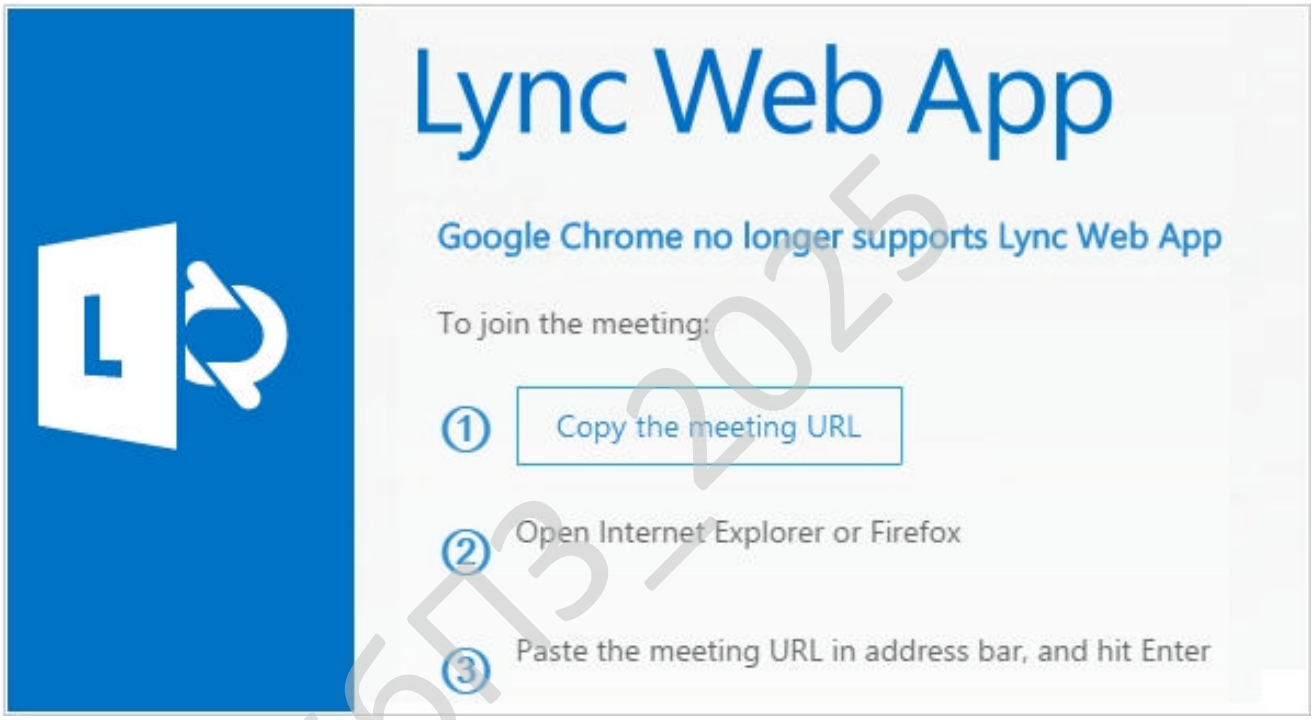

Рисунок 2.1 – Інтерфейс користувача TrueConf

На початку квітня 2025 року компанія Google відключила в браузерах на базі Chromium (Chrome, Яндекс Браузер) підтримку технології NPAPI, що використовується в багатьох системах відеоконференцій, що працюють через браузер (наприклад, VideoMost від компанії Spirit DSP, а також веб-версії Lync і Skype for Business від Microsoft). Таким чином, багато користувачів тепер не можуть повноцінно використовувати звичні відеозв'язки. Тому TrueConf пропонує для потерпілих особливі умови. До закінчення дії плагіна NPAPI вони

надають всім користувачам VideoMost, Lync, Skype for Business або іншого виробника відеозв'язку знижку 30% при переході на системи відеоконференцій Trueconf. Розроблювач систем ВКЗ, компанія TrueConf, представила 5 прогнозів про розвиток технологій ВКЗ:

– Домінування архітектури масштабованого відеокодування (SVC), над класичною схемою з мікшуванням (MCU).

– Універсальні ПК замінять вузькоспеціалізовані ВКЗ термінали в переговірних кімнатах.

– У користувачів з'явиться можливість вибирати аудіо/відео периферію (камери, мікрофони, динаміки й т.п.).

– Одержать поширення системи високої чіткості UltraHD 4K і мобільні відеоконференції.

– Популярними стануть нові сценарії роботи – крім ретельного планування зустрічей, планування відеопереговірних, користувачі почнуть використовувати й режим швидкого відеозв'язку по-потреби.

Тому в 2025 році конференції через браузер, адресні книги зі статусами, інтеграція з телефонією, різними календарями й платформами для віщання, а також приємний інтерфейс стануть обов'язковими атрибутами для будь-якого поважаючі себе виробника ВКЗ рішень. Компанія Trueconf опублікувала свої прогнози по розвитку технологій відеоконференцій цього року. Говорять, збережеться тенденція витиснення апаратних ВКЗ рішень програмними, а найбільший ріст очікує хмарні системи відеоконференцій. Причому цього року вони повинні здорово додати в забезпеченні мір безпеки. Велика увага в Trueconf приділили проблемі несумісності стандартів ВКЗ систем від різних вендоров, і виразили надію, що впровадження стандарту WebRTC допоможе частково вирішити цю проблему, тому що зовнішні учасники відеозустрічей зможуть підключатися через стандартний браузер. Правда, цей варіант не підходить для переговірних кімнат доти, поки в стандарті WebRTC немає підтримки H.264 кодека.

Google Hangouts інтегрували з Polycom, Cisco і іншими відеоконференціями

Рисунок 2.2 – Інтерфейс користувача Vidyo

Google випустив систему відеоконференцій для переговорних кімнат Chromebox for Meetings. Виявляється компанія Vidyo (виробник корпоративних систем відеозв'язку) надає платформу для сервісу Google Hangouts. Отож, ця

компанія прикрутила до цієї своєї платформи сервіс Vidyo2O, що тепер дозволяє користувачам Google Hangouts брати участь у відеозустрічах з користувачами дорослих корпоративних систем відеоконференцій – Polycom, Cisco, Avaya, Lifesize і інших (підтримуючі стандарти H.323/SIP). Таким чином, відкриваються широкі можливості по організації якісного міжкорпоративного відеозв'язку. Правда, сервіс ц не безкоштовний, а коштує від $99/міс за користувача.

Безпечні відеоконференції

Компанія Aventail випустила пристрій, що одержав назву Secure Collaboration, що разом з підтримуючу технологію SSL VPN продуктами забезпечує проведення IP-конференцій, що сполучають передачу голосу, відео й миттєвих повідомлень. Secure Collaboration підтримує такі додатки, як конференції, спільна робота й служби технічної підтримки, дані яких передаються в захищених VPN-тунелях. Це дозволяє організувати взаємодію авторизованих користувачів, що застосовують для доступу в Internet будь-які пристрої – ПК, ноутбуки, кишенькові комп'ютери або телефони. Secure Collaboration установлюється між шлюзом SSL VPN і корпоративною мережею, дозволяючи вилученим користувачам ініціювати конференції тільки після їх попередньої автентифікації. Створення захищених SSL-каналів запобігає можливість несанкціонованого доступу до зраджуваного контенту.

VideoMost – повна конфіденційність переговорів

Збільшення проникнення інтернет-мереж і активне використання їх у комерційному й державному секторах загострює питання захисту конфіденційної інформації, особливо якщо мова йде про переговори, організованих з використанням програмних продуктів. У багатьох державних установах України часто використовуються безкоштовні іноземні хмарні VoIP-сервіси, як Skype і Google, які піддані контролю з боку закордонних спецслужб. Ці хмарні сервіси управляються й фізично перебувають в інших країнах, що створює ризик витоку конфіденційної інформації, а відповідно пряму погрозу національної безпеки України.

Програмний продукт VideoMost створений з урахуванням найвищих вимог до конфіденційності веб-відеоконференцій і забезпечує інформаційну безпеку організації. Програмний продукт для багатоточкових відеоконференцій VideoMost не тільки дозволяє шифрувати передані дані, але й що важливіше, дає можливість розташувати комунікаційний сервер VideoMost як у ЦОД стороннього сервісу-провайдеру, так і безпосередньо усередині самої організації клієнта. У цьому випадку комунікаційний сервер VideoMost повністю підконтрольний IT-службі організації-клієнта, що кардинально знижує ризики несанкціонованого зовнішнього підключення. Найбільш вимогливі до забезпечення конфіденційності переговорів клієнти використовують VPN (virtual private network), тобто додатково шифрують весь переданий трафік програмними або апаратними засобами, які не впливають на роботу VideoMost. Сьогодні більшість державних і великих приватних корпоративних клієнтів воліє купувати ліцензії на сервера VideoMost для установки усередині своїх організацій, саме з міркувань контролю й інформаційної безпеки.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка додатків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при

програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка додатків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки додатків Windows дозволяють автоматизувати процес створення додатка. Для цього використовуються генератори додатків. Програміст відповідає на питання генератора додатків і визначає властивості додатка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор додатків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої додатки. Подібні засоби автоматизованого створення додатків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики додатка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні додатки, а також додатки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину додатка програмістові прийдеться розробляти самостійно. Вихідний текст додатка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон додатка – це вже половина всієї роботи. Вихідні тексти додатків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти додатків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої додатки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека

містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-додатки на базі об'єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-додатків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька сотень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами

на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи управління відеоресурсами на основі CMR.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності

розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Віртуальний простір для зустрічей одержав назву Collaboration Meeting Room (CMR). Цей персоналізований ресурс належить конкретному користувачеві – хазяїнові кімнати. Персоніфікація виражається, зокрема, у тому, що до CMR прив'язана його база контактів, у тому числі й людей з інших організацій. Персональна віртуальна кімната доступна в будь-який час і з будь-якого пристрою, у неї є постійний URL для підключення через WebEx і адреса для підключення по відеозв'язку. Хазяїн може запросити у свою кімнату користувачів з інших (зовнішніх) систем, у тому числі Lync.

Підхід, заснований на використанні персональних віртуальних кімнат, зручний і гнучкий. Один із ключових інфраструктурних елементів, необхідних для його реалізації, – система, що забезпечує пріоритизацію запитів, виділення ресурсів і розподіл (балансування) навантаження між серверами конференц-зв'язку, включаючи сервери MCU, – у випадку, якщо один із серверів вийде з ладу, навантаження буде перенесена на інші. Крім того, система зберігає всі налаштування віртуальних кімнат CMR – якщо таких кімнат багато, обсяг інформації виходить значний.

У контексті підвищення зручності й автоматизації проведення ВКЗ у реальних кімнатах слід зазначити систему автоматичного підстроювання камер, у якій автоматичне наведення на обличчя учасників здійснюється по голосі. Система автоматично знаходить всіх учасників дискусії, масштабує їхнє зображення й навіть змінює формат, коли хто-небудь входить або виходить із конференц-залу. Новинка працює з усіма наявними камерами, а при покупці нових кодеків замовник одержує її безкоштовно.

Відповідно до досліджень, висока вартість – головна перешкода на шляху більш масштабного впровадження ВКЗ. В 2025 році цей фактор указали 64,2% респондентів. Зниженню ціни сприяє жорсткість конкуренції на ринку ВКЗ за рахунок виходу на нього нових, у першу чергу азіатських, виробників. Yealink наголошує на простоті налаштування й використання, при цьому характеристики продуктів відповідають тим, що пропонують лідери ринку. Всю функціональність замовник одержує відразу – ніяких додаткових ліцензій здобувати не потрібно. Серед характеристик терміналів Yealink відзначимо підтримку кодека H.264 High Profile і наявність алгоритмів по оптимізації використання канальних ресурсів, включаючи пряму корекцію помилок (висока якість відео підтримується при втраті більше 8% пакетів). У ВКЗ-термінал Yealink убудовані два транскодера, що забезпечує незалежну обробку з'єднань із високим (HD) і більше низькою якістю.

При виборі рішення експерти компанії проаналізували наявні на ринку пропозиції. Типовий варіант припускав інвестиції порядку 500 тис. гривень за кожну крапку плюс ліцензії (близько 15%) кожний наступний рік. Рішення виявилося істотно дешевше (абонентська крапка від 90 тис. гривень), крім того, його використання не вимагало витрат на додаткові ліцензії. По своїй функціональності ВКЗ-пристрої відповідають всім основним вимогам, включаючи убудовані в термінали MCU, при цьому у використанні вони не складніше DVD-програвача.

Лідери ринку ВКЗ також намагаються знизити ціни й запропонувати більше вигідні умови на придбання своїх рішень. Так, наприклад, з виходом продуктів нового покоління персональні термінали ВКЗ від Cisco стали коштувати на 60% менше, при цьому молодша модель, за словами Максима Рєпіна, порівнянна за вартістю з гарним монітором.

Серед нових економічних моделей відзначимо можливість придбання інфраструктури ВКЗ, по суті, у лізинг. Замовник може узяти під оренду на певний час (рік, два, три) повні інфраструктури ВКЗ, включаючи сервер MCU. За

договором замовникові також надаються сервіс по відновленню ПЗ й технічна підтримка.

Cisco же взагалі пропонує інфраструктуру ВКЗ... безкоштовно. Справа в тому, що, відповідно до нової ліцензійної політики, програмні (віртуалізовані) сервери можуть бути отримані безкоштовно. Однак, щоб скористатися ними, необхідно придбати ліцензії, які продаються по числу користувачів, що дозволяє вкладати кошти рівномірно в міру охвату нових співробітників. При цьому користувач при наявності ліцензії може організувати конференцію з будь-якого пристрою. Скажемо, ліцензія Personal Multiparty Advanced вартістю 625 доларів дозволяє йому проводити конференції будь-якого типу (миттєві, заплановані конференції або персональні CMR) на необмежене число учасників з якістю до 1080p30. Інакше кажучи, у своїй віртуальній кімнаті користувач може зібрати стільки учасників, скільки здатна підтримати інфраструктура.

Правда, мінімальне число таких ліцензій – 25 штук, і апаратна платформа, звичайно, коштує грошей. Скажемо, спеціалізований UCS-сервер, що підтримує 36 HD-портів, обійдеться в суму близько 21,5 тис. доларів. Але при наявності в компанії вільних (апаратних) серверних ресурсів (програмні) сервери TelePresence можна розгорнути, по суті, безкоштовно.

Відео із хмари

Мабуть, самою обговорюваною моделлю зниження вартості «вхідного квитка» у мир ВКЗ є хмарна модель.

ВКЗ як послуга зв'язку

Крім відеодзвінків, всі оператори пропонують у тім або іншому виді послуги для проведення вебінарів і відеоконференцій. Однак, для того щоб скористатися такими послугами, компанії необхідно, як правило, укласти окремий контракт із оператором.

Користувачеві не потрібно ніякого спеціального устаткування – досить персонального комп'ютера з підключеної Web-камерою, виходом в Інтернет і браузером однієї з останніх версій (з підтримкою WebRTC). Як термінал можна

використовувати й мобільний пристрій на платформі Android (Apple, як відомо, не підтримує WebRTC). Для організації конференцій застосовується принцип виділення кожному співробітникові віртуальної переговірної кімнати, що дозволяє не резервувати загальний ресурс для проведення персональних відеоконференцій.

Послуга надається на підставі діючого договору зв'язку. Відповідно, оплата здійснюється з мобільного рахунку. Для корпоративних клієнтів це дуже зручно, тому що відсутня необхідність висновку додаткового контракту. При наявності бюджету на зв'язок витрати на ВКЗ можна віднести до цієї статті, і тоді не прийде погоджувати з бухгалтерією закупівлю встаткування або введення якоїсь нової категорії, послугою можуть скористатися не тільки корпоративні клієнти, але й ті компанії, у яких немає корпоративного контракту, а також звичайні користувачі. Клієнт може замовити послугу на один день, провести конференцію (або кілька конференцій) і відключитися.

Рішення базується на медіасервері й забезпечує проведення конференцій з якістю HD. Крім властиво відеоконференцій, підтримується широкий набір функцій для спільної роботи й керування конференцією: показ робочого стола, демонстрація документів, запис нарад, проведення опитувань, керування правами учасників, модерація й ін. Планується реалізувати підключення до конференції по телефоні (коли доступний тільки голосовий зв'язок).

Рішення володіє на даний момент унікальним набором функціональних можливостей. Ця хмара дозволяє проводити конференції за участю до 40 чоловік (для кожного користувача сервісу) при одночасній передачі відео й презентації (1080p), а також забезпечує інтеграцію з Lync, сумісність із пристроями H.323/SIP, підключення з міських телефонів і т.д. До хмарних сервісів Cloud можна підключитися із традиційних терміналів, включаючи нові пристрої Icon 400 і Icon Flex, які пропонуються по досить демократичній ціні (менш 3000 доларів).

Одна із ключових особливостей системи полягає в тому, що вона оптимізована для роботи в єдиному середовищі з такими додатками, як Microsoft Lync, Skype, Google Hangouts або Cisco Jabber. При підключенні користувальницьких пристроїв до термінала (через USB-порти) автоматично зорієнтуються всі доступні профільні додатки, за рахунок чого створюється загальне комунікаційне середовище, що дозволяє комфортно спілкуватися поза залежністю від програмної й апаратної платформи конкретного мобільного пристрою.

Оптимальним споживання послуг ВКЗ із хмари вважають поки тільки 14,5% замовників, тоді як переважна більшість (62%) воліють, щоб компоненти системи ВКЗ установлювалися в компанії, а також управлялися й обслуговувалися її фахівцями. Проте останній показник виявився нижче, ніж в опитуванні 2013 року, коли він становив 69%. При цьому стрімко набирає популярність гібридна модель, коли частина інфраструктури залишається в компанії, а при епізодичному сплеску навантаження задіються зовнішні (хмарні) ресурси. Її вважають оптимальної 30,5%.

Протокол SIP, що дозволяє елегантно погодити в одну систему IP-УАТМ, засоби IP-Відеоспостереження й рішення ВКЗ.

Рішення нового класу – Enterprise video Resource Planning (EvRP), що дозволяє централізовано управляти відеоматеріалами в масштабах всієї організації. Засноване на єдиній технологічній платформі рішення EvRP поєднує всі основні компоненти для створення, зберігання, керування, відображення й потокової передачі відео в будь-яку крапку корпоративної мережі або мережі Інтернет, у тому числі на мобільні пристрої.

У якості одного з найбільш актуальних завдань, що може бути ефективно вирішена за допомогою EvRP, називається трансляцію обігу керівників до співробітників – незалежно від їхньої кількості й територіального розташування. Крім того, пропонована система дозволяє практично необмежено розширити аудиторію нарад, проведених за допомогою систем ВКЗ, організувати

дистанційне навчання й підвищення кваліфікації співробітників, транслювати корпоративне телебачення, підключити відеопортал і т.д.

Ми хочемо зробити систему Digital Signage, але щоб через неї транслювалися й конференції, і виступу керівників, і інша інформація. Така система була розроблена, багато в чому вона націлена на ті ж завдання, що й рішення EvRP. По суті, у цій системі об'єднані сервіси Digital Signage (показ документів, роликів, віджетів та ін.), ВКЗ (трансляції конференцій) і властиво телебачення (зовнішні й внутрішні ТБ-канали). Серед розв'язуваних з її допомогою завдань – показ різної бізнес-інформації (новини галузі, динамічні діаграми показників діяльності компанії й т. Д.), проведення навчання персоналу, надання доступу до розкладу переговірних кімнат (інтеграція із системою бронювання переговірних), функції навігації й оповіщення. Крім того, корпоративне телебачення є відмінним інструментів для реалізації внутрішнього піару: поздоровлення співробітників, проведення конкурсів (наприклад, «співробітник місяця»), анонс корпоративних заходів, фото- і відеозвіти про заходи.

Розроблене рішення – мультиекране, з можливістю трансляції на мобільні пристрої, у тому числі, що перебувають за межами офісу. При цьому керування всім контентом здійснюється з єдиного центра (сервер керування може розміщатися як у локальній серверній кімнаті, так і в хмарі). Система наділена певним «інтелектом», у тому числі по автоматизованому перемиканню контента на дисплеї залежно від події. Наприклад, якщо на вулиці піде дощ, тло може помінятися на максимально позитивний, щоб підняти настрій співробітників. Система інтегрується із засобами Wi-Fi-аналітики й відеоаналітики, а також «розуміє» RFID-мітки. Така мітка може перебувати в співробітника в пропуску, при піднесенні якого до дисплея на екрані буде відображатися персоніфікована інформація.

Відео в промисловості

З'являються й активно розвиваються нові напрямки. Це, зокрема, оптична дефектоскопія, адаптивна автоматизація процесів, технічний зір роботів і безпілотних літальних апаратів.

Оборотний увага на усе більше активне використання відеотехнологій на виробничих комплексах. Один з напрямків – інформування й навчання персоналу, що працює на виробничих лініях. Відповідні рішення багато в чому нагадують системи корпоративного ТБ або Digital Signage, тільки інформаційні панелі встановлюються безпосередньо поруч із конвеєром і/або іншими робітниками місцями. Крім різних інструкцій і підказок, на них виводяться термінові повідомлення й інша інформація. Інший новий напрямок – системи технічного зору. Вони вже застосовуються в сільському господарстві, машинобудуванні, металургії й інших галузях промисловості й засновані на автоматичному аналізі форм, кольору, структури об'єктів, наприклад, по синтаксисі потоку MPEG.

3.2 Розробка структурної схеми

CMR – це новий сервіс, що дозволяє Інтернет-користувачам зв'язуватися не тільки з персональних комп'ютерів, мобільних пристроїв і стаціонарних телефонів, але й з відеотерміналів і програмних клієнтів, що підтримують SIP-протокол.

Даний сервіс як і колись дозволяє легко й швидко обмінюватися документами, презентаціями (тепер і на відеотерміналах), працювати над спільним контентом між учасниками наради з різним місцем розташування.

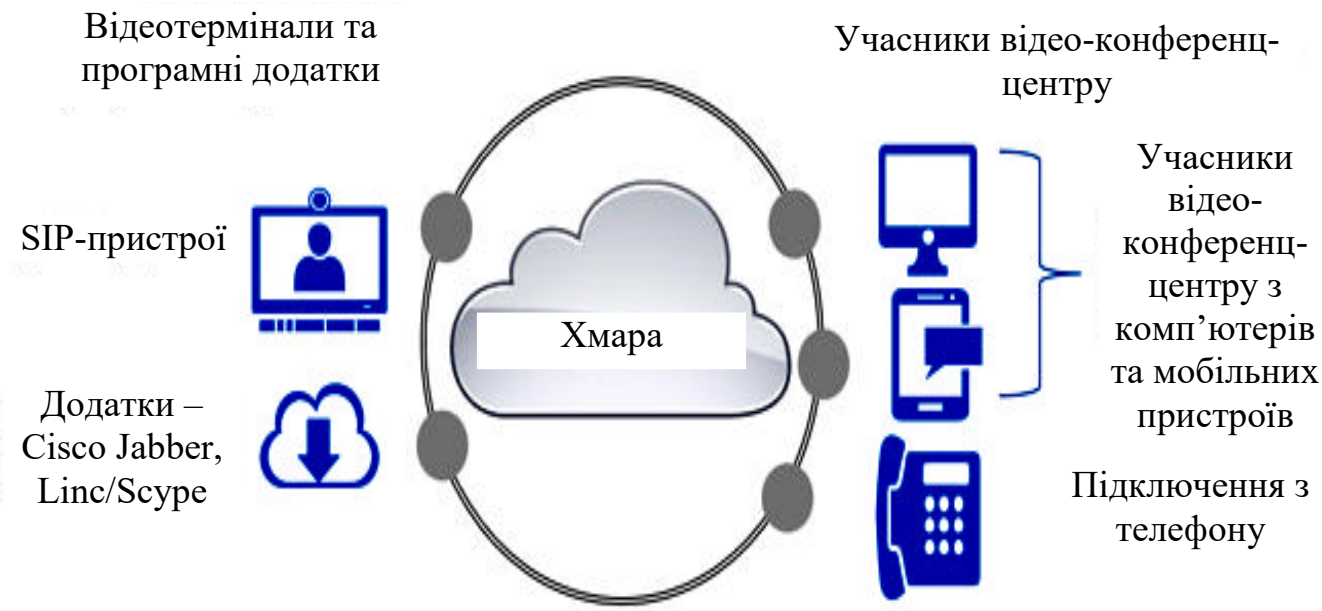

Рисунок 3.1 – Структурна схема системи

Бізнес-завдання, розв'язувані за допомогою CMR**:**

– Проведення нарад з колегами, партнерами й замовниками з будь-якої крапки миру з різних пристроїв: ПК, мобільний телефон, планшет, відеотермінал, програмний клієнт.

– Скорочення операційних витрат на відрядження, оскільки тепер можна взаємодіяти з колегами не відходячи від робочого місця

– Забезпечення доступності всіх співробітників і колег у будь-який час.

– Робота й зміна документів у реальному режимі часу.

Стандарти ВКЗ

У технічному описі будь-якої групової системи ВКЗ можна зустріти довгий перелік підтримуваних стандартів: відеокодеки, аудіокодеки, стандарти спільної роботи з даними, зв'язки й керування. Тим часом, більша частина характеристик, що приводяться в описах, не має практичного значення.

Мережа з маршрутизацією пакетів IP принципово підтримує одночасно цілий ряд різноманітних протоколів маршрутизації. Такими протоколами на сьогодні є: RIP, IGRP, EIGRP, IS-IS, OSPF, BGP і ін. Точно так само й для IP-

телефонії розроблений цілий ряд протоколів

Найпоширенішим є протокол H.323, зокрема, тому, що він став застосовуватися раніше інших протоколів. Інший протокол площини керування обслуговуванням виклику – SIP – орієнтований на те, щоб зробити кінцеві пристрої й шлюзи більш інтелектуальними й підтримувати додаткові послуги для користувачів. Ще один протокол – SGCP – розроблявся, для того, щоб зменшити вартість шлюзів за рахунок реалізації функцій інтелектуальної обробки виклику в централізованому встаткуванні. Протокол IPDC дуже схожий на SGCP, але має більше, ніж SGCP, механізмів експлуатаційного керування (OAM&P). Існує більш функціональний, ніж MGCP, протокол MEGACO. Його адаптований до H.323 варіант в рекомендації H.248.

Щоб стало зрозуміло, чим конкретно відрізняються один від одного перераховані в попередньому параграфі протоколи, розглянемо архітектуру мереж, побудованих на базі цих протоколів, і процедури встановлення й завершення з'єднання з їхнім використанням.

Мережі на базі протоколів H.323 орієнтовані на інтеграцію з телефонними мережами й можуть розглядатися як мережі ISDN, накладені на мережі передачі даних. Рекомендація H.323 передбачає досить складний набір протоколів, що призначений не просто для передачі мовної інформації з IP-мереж з комутацією пакетів. Його мета – забезпечити роботу мультимедійних додатків у мережах з негарантованою якістю обслуговування. Мовний трафік – це тільки один з додатків H.323, поряд з відеоінформацією й даними. Варіант побудови мереж IP-телефонії в рекомендації H.323, добре підходить тим операторам місцевих телефонних мереж, які зацікавлені у використанні мережі з комутацією пакетів (IP-мережі) для надання послуг міжміського й міжнародного зв'язку.

Для впровадження ВКЗ необхідна підтримка обмеженої кількості стандартів:

– Відеокодеки. Повноцінна робота будь-яких групових систем (незалежно від виробника) можлива за підтримкою всього лише двох стандартів кодування

відео: H.263 і H.264. Перший (більше старий) необхідний для сумісності з більше раннім устаткуванням. Деякі сучасні групові системи ВКЗ із убудованим MCU використовують H.263 при проведенні багатобічних відеоконференцій. Це пов'язане з тим, що він менш вимогливий до апаратних ресурсів, і перехід на нього часто здійснюється автоматично при збільшенні числа учасників.

– H.264 – самий зроблений стандарт відеокодування, що представляє собою подальший розвиток комп'ютерного формату mpeg4. Цей кодек став стандартом де-факто практично для всіх виробників систем ВКЗ. Він забезпечує значно більше високу якість зображення, чим H.263, при тій же пропускній здатності.

– Інші відеокодеки, підтримка яких заявлена виробниками групових систем ВКЗ, потрібні, по-перше, для сумісності з більше раннім устаткуванням (кодек H.261), а по-друге, для забезпечення максимально ефективної роботи встаткування однієї марки. Скажемо, у ситуації, коли групові системи ВКЗ різних вендорів з якихось причин (наприклад, через велику кількість учасників) були б змушені перейти з відеокодека H.264 на кодек H.263, системи одного виробника, можливо, зможуть працювати на іншій, більше зробленого різновиду цього стандарту: H.263+ або H.263++.

– Аудіокодеки. Для забезпечення аудіосумісності систем ВКЗ різних виробників у всіх групових терміналах повинна бути реалізована підтримка єдиного стандарту кодування звуку -G.711. Цей алгоритм кодування вузькополосного звуку (3,1 кГц) у каналі 48, 56 або 64 Кбіт/с забезпечує кращу якість звуку в порівнянні з іншими аудіокодеками – не гірше, ніж при звичайному телефонному зв'язку. Він обходиться мінімумом апаратних ресурсів, але має потребу в значній пропускній здатності.

– Всі інші аудіокодеки використовують алгоритми стиску інформації із втратами, що практично не впливає на якість звуку, але при цьому необхідна пропускна здатність каналу в 5-10 разів нижче, ніж в G.711. Так, стандарт кодування звуку G.729 цілком підходить для використання в Internet – йому потрібно всього 8 Кбіт/с. Однак кодеки із сильним стиском звуку нестійкі до

втрат пакетів, а крім того, для них характерна висока складність обробки звуку й, як наслідок, підвищений навантаження на апаратні ресурси встаткування.

– Інші стандарти. Ще один важливий стандарт – H.239, також відомий як функція DualVideo, – дозволяє паралельно із зображенням учасників передавати статичне, рідко обновлюване відео (двопотокова передача): знімки з документ-камери, слайди презентації, електронні таблиці або текстові файли з комп'ютера. Для виводу статичної «картинки» звичайно використовується окремий монітор з високим дозволом (1024x768 точок і вище). Без DualVideo відеоконференції не мають змісту, тому з 2002 р. ця функція підтримується майже всіма виробниками встаткування для ВКЗ.

Основні характеристики групових систем ВКЗ

Кожна групова система ВКЗ має широкий набір технічних характеристик. Якісь із них по-справжньому важливі, хоча необізнаному користувачеві можуть представлятися несуттєвими. Інші, навпаки, на перший погляд, вражають, але реального виграшу не дають.

Дизайн. Як не дивно, але зовнішній вигляд групової системи був і залишається одним із ключових критеріїв вибору: це встаткування завжди встановлюється на очах, підкреслює статус організації й повинне виглядати відповідним чином. Статистика світових продажів свідчить, що «непрезентабельні» групові системи ВКЗ не користуються популярністю, навіть незважаючи на найширшого функціонала або низьку ціну. Саме тому всі великі гравці ринку приділяють зовнішньому вигляду особливу увагу й навіть прибігають до послуг сторонніх дизайнерів.

Швидкість з'єднання по IP. Значення цього параметра в групових систем ВКЗ варіюється від 768 Кбіт/с до 4 Мбіт/с. Але чи не так уже потрібна підтримка високих швидкостей? Практика показує, що незалежно від використовуваного кодека швидкість передачі відео ніколи не перевищує 768 Кбіт/с. Більше того, навіть якщо відеоконференція проводиться між двома учасниками, термінали яких підтримують швидкості передачі до 4 Мбіт/с, реальна швидкість передачі

навряд чи перевищить 1 Мбіт/с, тому якість зображення залишиться таким же, як і при 768 Кбіт/с.

Зовсім інша ситуація, якщо групова система має убудований MCU. У такому випадку якість відео буде залежати від максимальної швидкості з'єднання по IP. Очевидно, що при загальній швидкості з'єднання, наприклад, 2 Мбіт/с, груповий термінал ВКЗ фізично не зможе забезпечити проведення чотирибічної конференції зі швидкостями 768 Кбіт/с (3 х 768 Кбіт/с = 2,3 Мбіт/с). Швидше за все, швидкість передачі даних буде знижена до 384 кбіт/с, а виходить, постраждає якість.

Таким чином, для групових систем початкового рівня без MCU підтримка швидкостей з'єднання по IP більше 768 Кбіт/с не дає реальних переваг. З іншого боку, більша швидкість – це певний запас на майбутнє, наявність якого дозволить реалізувати нові, більше вимогливі до пропускної здатності стандарти шляхом простого «перепрошивання» устаткування.

Для групових систем бізнес-класу підтримка високої пропускної здатності по IP – необхідність і застава того, що багатобічна відеоконференц-зв'язок буде мати гарну якість.

Швидкість з'єднання по ISDN. У групових систем ВКЗ швидкість з'єднання по ISDN становить від 384 Кбіт/с до 2 Мбіт/с. И це, мабуть, самий малозначимий параметр для сучасних систем ВКЗ. Сеанси відеозв'язку по телефонних каналах проводяться вкрай рідко, тому багато виробників взагалі відмовилися від підтримки ISDN у групових системах.

Дозвіл відео. У відеоконференц-зв'язку є кілька стандартних дозволів відеосигналу: SQCIF (128x96), QCIF (176x144), CIF (352x288), 4CIF (704x576). Перші два (SQCIF і QCIF) забезпечують занадто низька якість зображення, щоб їх мало сенс використовувати, а от підтримка дозволів CIF і 4CIF обов'язкова.

Убудований сервер MCU. Убудовані MCU не є прерогативою систем бізнес рівня й звичайно забезпечують проведення відеоконференцій від 4 до 9 учасників. Як визначити, MCU наскільки буде оптимальний? У першу чергу

необхідно оцінити реальні потреби. Нерозумно вибирати систему, MCU якої розраховано на чотирьох користувачів, якщо в компанії є вісім представництв по всій країні. При рівному числі підтримуваних учасників кращим вибором будуть моделі з найбільшою швидкістю з'єднання по IP, оскільки вони забезпечують більше високу якість відеозв'язку.

Відеовходи й відеовиходи. У будь-якої групової системи ВКЗ повинні бути як відеовходи для підключення джерел відеосигналів, так і відеовиходи для виводу відеозображення.

Однак, коли число одних досягає п'яти, а інших дев'яти, виникає резонне питання: навіщо так багато?

Велика кількість відеовходів/виходів обумовлено наявністю різних типів переданих відеосигналів. Найбільше поширення одержали композитний (рознімання RCA) і S-Video. Незважаючи на сумісність практично з усіма видами апаратури (ТБ, відеокамери й т.п.), вони забезпечують найнижчий дозвіл і якість. Тому в багатьох групових системах вивід відеозображення можливий також у вигляді більше якісного компонентного сигналу (3xRCA рознімання на канал). Крім того, як уже говорилося, практично у всіх системах ВКЗ підтримується функція DualVideo, і в цьому випадку відеосигнал з комп'ютера передається у високому дозволі через рознімання DVI/XGA.

Для організації відеоконференції буде потрібно не менш двох відеовходів і двох відеовиходів:

− один вхід для підключення додаткової відеокамери;

− один вхід DVI/XGA для підключення ноутбука (проведення презентацій) або документа-камери. Як додаткові джерела відеосигналу до групових систем ВКЗ часто підключають DVD- або відеопрогравачі;

− один вихід для виводу зображення з основної відеокамери;

− один вихід DVI/XGA для виводу презентації й зображення з документ-камери або комп'ютера.

При проведенні великих відеоконференцій до додаткових відеовиходів часто підключають монітори провідного інженера, що обслуговує сеанс ВКЗ, а також відеомагнітофони або DVD-рекодери, призначені для запису (протоколювання) конференцій.

Висока чіткість (High Definition, HD). Останнім часом особливий інтерес викликають відеоконференції, що забезпечують високу якість зображення – з дозволом 1280x720 (стандарт 720p) або 1920x1080 (стандарт 1080i/p) точок.

Практично всі провідні виробники систем ВКЗ уже випустили встаткування з підтримкою HD. Однак попит на подібні системи поки невеликий через їхню дорожнечу й високу вимогливість до пропускної здатності, а реальний виграш від застосування HD майже непомітний.

Додаткові функції. Одна з найбільш корисних – автоматичне наведення камери на мовець у цей момент учасника відеоконференції, зображення якого виводиться на екран замість загального плану приміщення з усіма присутніми в ньому людьми. На жаль, ця функція часто надається за додаткову плату.

Дуже цікавої, але практично не використовуваної є підтримка каскадування. Вона дозволяє збільшити число учасників відеоконференції за рахунок застосування декількох групових систем ВКЗ із убудованим MCU. Таке рішення має цілий ряд обмежень, наприклад, на екрані відображаються не всі учасники, а тільки виступаючі. Тому при необхідності проведення більше багатолюдних відеоконференцій найчастіше встановлюють додаткові MCU.

3.3 Розробка функціональної схеми

Нажаль сучасні лінії зв’язку далекі від ідеальних. Тому, дані, які передаються у системі управління відеоресурсами на основі CMR, потребують перешкодостійкого кодування.

Для реалізації перешкодостійкого кодування для управління відеоресурсами на основі CMR обрано циклічний код.

Циклічним кодом називається лінійний блоковий (n,k)-код, що характеризується властивістю циклічності, тобто зрушення вліво на один крок будь-якого дозволеного кодового слова дає також дозволене кодове слово, що належить цьому ж коду й у якого, множина кодових слів представляється сукупністю багаточленів ступеня (n-1) і менш, що діляться на деякий багаточлен $g(x)$ ступеня $r = n$-k, що є співмножником двочлена x^n+1.

Багаточлен $g(x)$ називається породжуючим.

Як треба з визначення, у циклічному коді кодові слова представляються у вигляді багаточленів:

$$a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x^1 + a_0x^0, \qquad (3.1)$$

де n – довжина коду;

a_i – коефіцієнти з поля $GF(q)$.

Якщо код побудований над полем $GF(2)$, то коефіцієнти приймають значення 0 або 1 і код називається двійковим.

Наприклад, якщо код побудований над полем $GF(q)=GF(2^3)$, що є розширенням $GF(2)$ по модулі багаточлена, що не *приводиться,* $f(z)=z^3+z+1$, а елементи цього поля мають вигляд, представлений у таблиці 3.1.

То коефіцієнти $a_i(x)$ приймають значення елементів цього поля й тому вони самі відображаються у вигляді багаточленів наступного виду:

$$a_i(z) = a_{m-1} \cdot z^{m-1} + a_{m-2} \cdot z^{m-2} + \ldots + a_1 \cdot z^1 + a_0 \cdot z^0, \qquad (3.2)$$

де m – ступінь багаточлена, по якому отримане розширення поля $GF(2)$;

α_i – коефіцієнти, що приймають значення елементів $GF(2)$, тобто 0 і 1. Такий код називається q-ним.

Таблиця 3.1 – Елементи поля $GF(2)$

0	000	0	α^3	011	Z+1
α^0	001	1	α^4	110	Z^2+Z
α^1	010	Z	α^5	111	Z^2+Z+1
α^2	100	Z^2	α^6	101	Z^2+1

Довжина циклічного коду називається примітивної й сам код називається примітивним, якщо його довжина $n= q^{m-1}$ на $GF(q)$.

Якщо довжина коду менше довжини примітивного коду, то код називається вкороченим або непримітивним.

Як треба з визначення загальна властивість кодових слів циклічного коду – це їхня подільність без остачі на деякий багаточлен $g(x)$, називаний породжуючим.

Результатом ділення двочлена x^n+1 на багаточлен $g(x)$ є перевірочний багаточлен $h(x)$.

Матричне завдання кодів

Циклічний код може бути заданий породжуючий й перевірочною матрицями. Для їхньої побудови досить знати породжуючий $g(x)$ і перевірочний $h(x)$ багаточлени.

Для несистематичного циклічного коду матриці будуються циклічним зрушенням породжуючого й перевірочного багаточленів, тобто шляхом їхнього множення на x:

$$G_{(n,k)} = \begin{vmatrix} g(x) \\ x \cdot g(x) \\ x^2 \cdot g(x) \\ \bullet\bullet\bullet \\ x^{k-1} \cdot g(x) \end{vmatrix},$$

та:

$$H_{(n,k)} = \begin{vmatrix} h(x) \\ x \cdot h(x) \\ x^2 \cdot h(x) \\ \bullet\bullet\bullet \\ x^{r-1} \cdot h(x) \end{vmatrix}.$$

При побудові матриці $H_{(n,k)}$ старший коефіцієнт багаточлена $h(x)$ розташовується праворуч.

Для систематичного циклічного коду матриця $G_{(n,k)}$ визначається з вираження:

$$G_{(n,k)} = \left| I_k , R_{k,r} \right|, \tag{3.3}$$

де I_k – одинична матриця;

$R_{k,r}$ – прямокутна матриця.

Рядки матриці $R_{k,r}$ визначаються з виражень:

$$r_i(x) = R_{g(x)} \left| a_i(x) \cdot x^r \right|, \tag{3.4}$$

або:

$$r_i(x) = R_{g(x)} \left| x^{n-i} \right|, \tag{3.5}$$

де $a_i(x)$ – значення і-*того* рядка матриці I_k;

i – номер рядка матриці $R_{k,r}$.

Використовуючи вираження:

$$r_i(x) = R_{g(x)} \left| x^{n-i} \right|, \tag{3.6}$$

одержимо той же результат.

Рядка матриці $G_{(n,k)}$ можна визначити безпосередньо з вираження:

$$r_i(x) = a_i(x) \cdot x^r + r_i(x), \tag{3.7}$$

де:

$$r_i(x) = R_{g(x)} \left| a_i(x) \cdot x^r \right|. \tag{3.8}$$

Перевірочна матриця в систематичному виді будується на основі матриці $G_{(n,k)}$, а саме:

$$H_{(n,k)} = \left| R_{k,r}^T , I_r \right|, \tag{3.9}$$

де I_r – одинична матриця;

$R_{k,r}^T$ – матриця з $G_{(n,k)}$ у транспонованому виді.

Одна з основних задач, що коштують перед розроблювачами пристроїв захисту від помилок при передачі дискретних повідомлень по каналах зв'язку є вибір багаточлена, породжуючий, $g(x)$ для побудови циклічного коду, що забезпечує необхідну мінімальну кодову відстань для гарантійного виявлення й виправлення t-кратних помилок.

Існують спеціальні таблиці на вибір $g(x)$ залежно від пропонованих вимог до коригувальних можливостей коду. Однак у кожного циклічного коду є свої особливості формування $g(x)$. Тому при вивченні конкретних циклічних кодів будуть розглядатися відповідні способи побудови $g(x)$.

Задача кодування полягає у формуванні по інформаційних словах $a(x)$ кодових слів (x) циклічного (n,k)-коду, що по своїй структурі може бути несистематичним і систематичним.

Формування кодових слів несистематичного коду полягає в множенні багаточлена $a(x)$, що відображає інформаційну послідовність довжини k, на породжуючий багаточлен, тобто $(x)=a(x)\times g(x)$. Формування кодових слів систематичного коду полягає в перетворенні інформаційної послідовності $a(x)$ відповідно до вираження $(x)=a(x)\oplus x^r+r(x)$.

Перевірочна послідовність $r(x)$ визначається двома способами:

– при використанні "класичного" способу кодування:

$$r(x) = R_{g(x)}\left[a(x)\cdot x^r\right]; \tag{3.10}$$

– при використанні способу кодування, рекомендованого МККТТ:

$$r(x) = \overline{R}_{g(x)}\left[a(x)\cdot x^r + x(1)^{r-1}\cdot x^k\right], \tag{3.11}$$

де $x(1)^{r-1}$ – одиничний багаточлен ступеня $(r-1)$.

Зазначені вище математичні операції виконують кодерыи несистематичного й систематичного кодів.

Способи декодування з виявленням помилок

Процедура декодування циклічного коду з виявленням помилок, за аналогією із процесом кодування, використовує два способи:

– При кодуванні "класичним" способом декодування засноване на використанні властивості подільності без остачі кодового багаточлена (x) циклічного (n,k)-коду на породжуючий багаточлен $g($x$)$. Тому алгоритм декодування містить у собі ділення прийнятого кодового слова, описуваного багаточленом $z^n(x)$ на $g(x)$, обчислення й аналіз остачі $r(x)$. Якщо $r(x)=0$, то

прийняте кодове слово вважається неспотвореним. Якщо $r(x)<>0$, то прийняте кодове слово стирається й формується сигнал "помилка".

– При кодуванні способом МККТТ декодування засноване на властивості одержання певної контрольної остачі $R_0(x)$ при діленні прийнятого кодового багаточлена (x) на породжуючий багаточлен. Тому, якщо отриманий при діленні остача $\overline{r(x)} = R_0(x)$, то прийняте кодове слово вважається неспотвореним. Якщо остача $\overline{r(x)} \neq R_0(x)$, то прийняте кодове слово стирається й формується сигнал "помилка". Значення контрольної остачі визначається з вираження:

$$R_0(x) = R_{g(x)}\left[x(1)^{r-1} \cdot x^k\right]. \quad (3.12)$$

Способи декодування з виправленням помилок і схемна реалізація декодувальних пристроїв

Декодування циклічного коду в режимі виправлення помилок можна здійснювати різними способами. Нижче викладаються два способи, що є найбільш простими.

В основу першого способу покладене використання таблиці синдромів (декодування), у якій кожному багаточлену або зразку помилок $e_i(x)$, відповідає певний синдром $S_i(x)$, що представляє остачу від ділення прийнятого кодового слова $a_i'(x)$ й відповідного йому $e_i(x)$ на $g(x)$. Процедура декодування наступна. Прийняте кодове слово $a_i'(x)$ ділиться на $g(x)$, визначається $S_i(x)$ і відповідний йому багаточлен $e_i(x)$, а потім $a_i'(x)$ підсумується з $e_i(x)$. У результаті одержуємо виправлене кодове слово, тобто:

$$a_i(x) = a_i'(x) + e_i(x). \quad (3.13)$$

До складу декодера входять: обчислювач синдрому (ВР), два регістри зрушення *RG*1 і *RG*2, постійний запам'ятовувальний пристрій (ПЗП), що містить $\sum_{i=1}^{t_u} C_n^i$ слова довжини n, що відповідають багаточленам помилок $e_i(x)$.

Прийняте кодове слово $a_i'(x)$ надходить на вхід обчислювача синдрому, де здійснюється ділення його на $g(x)$ і формування $S_i(x)$, і одночасно – на вхід *RG*2,

де $v_i'(x)$ накопичується. Синдром $S_i(x)$ використовується як адреса, по якому із ПЗП в регістр $RG1$ записується $e_i(x)$, що відповідає синдрому $S_i(x)$. Перераховані операції завершуються за n тактів. Протягом наступних n тактів відбувається заелементне підсумовування вмісту $RG2$ і $RG1$, тобто операція:

$$v_i(x) = v_i'(x) + e_i(x), \tag{3.14}$$

і виправлення помилок.

В основі другого способу виправлення помилок, що дозволяє значно скоротити об'єм використовуваних табличних синдромів і істотно спростити схему декодера, лежать наступні положення:

1. Синдром $S_i(x)$, що відповідає прийнятому кодовому слову дорівнює остачі від ділення $v_i'(x)$ на $g(x)$, а також остачі від ділення відповідного багаточлена помилок $ei(x)$ на $g(x)$, тобто:

$$S_i(x) = R_{g(x)}\left[v_i'(x)\right] = R_{g(x)}\left[e_i(x)\right]. \tag{3.15}$$

2. Якщо $S_i(x)$ відповідає $v_i'(x)$ й $e_i(x)$, то $x \in S_i(x)$ є синдромом, що відповідає:

$$R_{g(x)}\left[x \cdot S_i(x)\right] = R_{g(x)}\left[x \cdot v_i'(x) \bmod (x^n - 1)\right] = \\ = R_{g(x)}\left[x \cdot e_i(x) \bmod (x^n - 1)\right], \tag{3.16}$$

і:

$$x \cdot v_i'(x) \bmod (x^n - 1), \tag{3.17}$$

або:

$$x \cdot e_i(x) \bmod (x^n - 1). \tag{3.18}$$

3. При виправленні помилок використовуються синдроми зразків помилок тільки з ненульовим коефіцієнтом у старшому розряді.

Тому при реалізації цього способу множина всіх зразків помилок розбивається на класи еквівалентності. Кожний клас представляє циклічне зрушення одного зразка помилок, а синдром цього класу відповідає зразку помилок з ненульовим старшим розрядом. Якщо обчислений синдром належить одному із класів еквівалентності зразків помилок, що виправляються, то старший

символ кодового слова виправляється. Потім прийняте слово й синдром циклічно зрушується, а процес знаходження в попередній по старшинству позиції повторюється.

Для виправлення помилок, що належать даному класу еквівалентності, потрібно зробити *n* циклічних зрушень.

Найпростішим є декодер Меггітта. До складу декодера входять: обчислювач синдрому, що здійснює ділення кодового слова $a'(x)$ на $g(x)$ і формування відповідного синдрому; блок декодерів (ДК), що настроєний на синдроми всіх зразків помилок, що виправляються, з ненульовими старшими розрядами; регістр зрушення *RG*.

При надходженні на вхід схеми кодового слова $a'(x)$ його символи заповнюють регістр *RG*, а в обчислювачі формується відповідний синдром $S_i(x)$. Обчислений синдром рівняється з усіма табличними синдромами, закладеними в схему блоку ДК, і у випадку збігу з одним з них на його виході формується сигнал, що виправляє помилковий символ, що перебуває в старшому розряді регістра. Після цього вміст обчислювача й *RG* циклічно зрушується на один крок. Це зрушення реалізує операції $R_{g(x)}[x \cdot S_i(x)]$ й $x \cdot a'(x) \bmod(x^n - 1)$. Якщо новий синдром збігається з одним з табличних синдромів, то це означає, що відбулася помилка в другому по старшинству символі кодового слова, що, перейшовши в старший розряд *RG*, виправляється. Потім виробляється нове циклічне зрушення на одну позицію й нову перевірку на збіг синдромів. Після повторення цього процесу *n* раз в *RG* буде сформоване виправлене кодове слово. Введення зворотного зв'язка для *RG* не обов'язково, тому що в процесі виправлення помилок символи кодового слова надходять на вихід декодера.

При декодуванні циклічних кодів використовуються багаточлен помилок $e(x)$ і синдромний багаточлен $S(x)$.

Багаточлен помилок ступеня не більше (n-1) визначається з вираження:

$$e(x) = a'(x) + a(x), \tag{3.19}$$

де $v'(x)$ и $v(x)$ – багаточлени, що відображають відповідно прийняте (з помилкою) і передане кодові слова.

Ненульові коефіцієнти в $e(x)$ займають позиції, які відповідають помилкам.

Синдромний багаточлен, використовуваний при декодуванні циклічного коду, визначається як остача від ділення прийнятого кодового слова на породжуючий багаточлен, тобто:

$$S_i(x) = R_{g(x)}[v'(x)], \tag{3.20}$$

або:

$$S_i(x) = R_{g(x)}[v'(x) + e_i(x)] = R_{g(x)}[e_i(x)]. \tag{3.21}$$

Отже, синдромний багаточлен залежить безпосередньо від багаточлена помилок $e(x)$.Це положення використовується при побудові таблиці синдромів, застосовуваної в процесі декодування. Ця таблиця містить список багаточленів помилок і список відповідних синдромів, обумовлених з вираження:

$$S_i(x) = R_{g(x)}[e_i(x)] \tag{3.22}$$

Таблиця 3.2 – Список багаточленів помилок і список відповідних синдромів

(x)	$S(x)$
1	$R_{g(x)}[1]$
X	$R_{g(x)}[x]$
X^2	$R_{g(x)}[x^2]$
X+1	$R_{g(x)}[x+1]$
X^2+1	$R_{g(x)}[x^2+1]$

У процесі декодування по прийнятому кодовому слову обчислюється синдром, потім у таблиці перебуває відповідний багаточлен $e(x)$, підсумовування якого із прийнятим кодовим словом дає виправлене кодове слово, тобто:

$$v_i(x) = v_i'(x) + e_i(x). \tag{3.23}$$

Перераховані багаточлени $v(x), v'(x), g(x), h(x), e(x)$ і $S(x)$ можна складати, множити й ділити, використовуючи відомі правила алгебри, але із приведенням результату по mod 2, а потім по mod x^n+1, якщо ступінь результату перевищує ступінь (n-1).

При побудові й декодуванні циклічних кодів у результаті ділення багаточленів звичайно необхідно мати не частка, а остача від ділення.

Тому рекомендується більш простий спосіб ділення, використовуючи не багаточлени, а тільки його коефіцієнти.

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

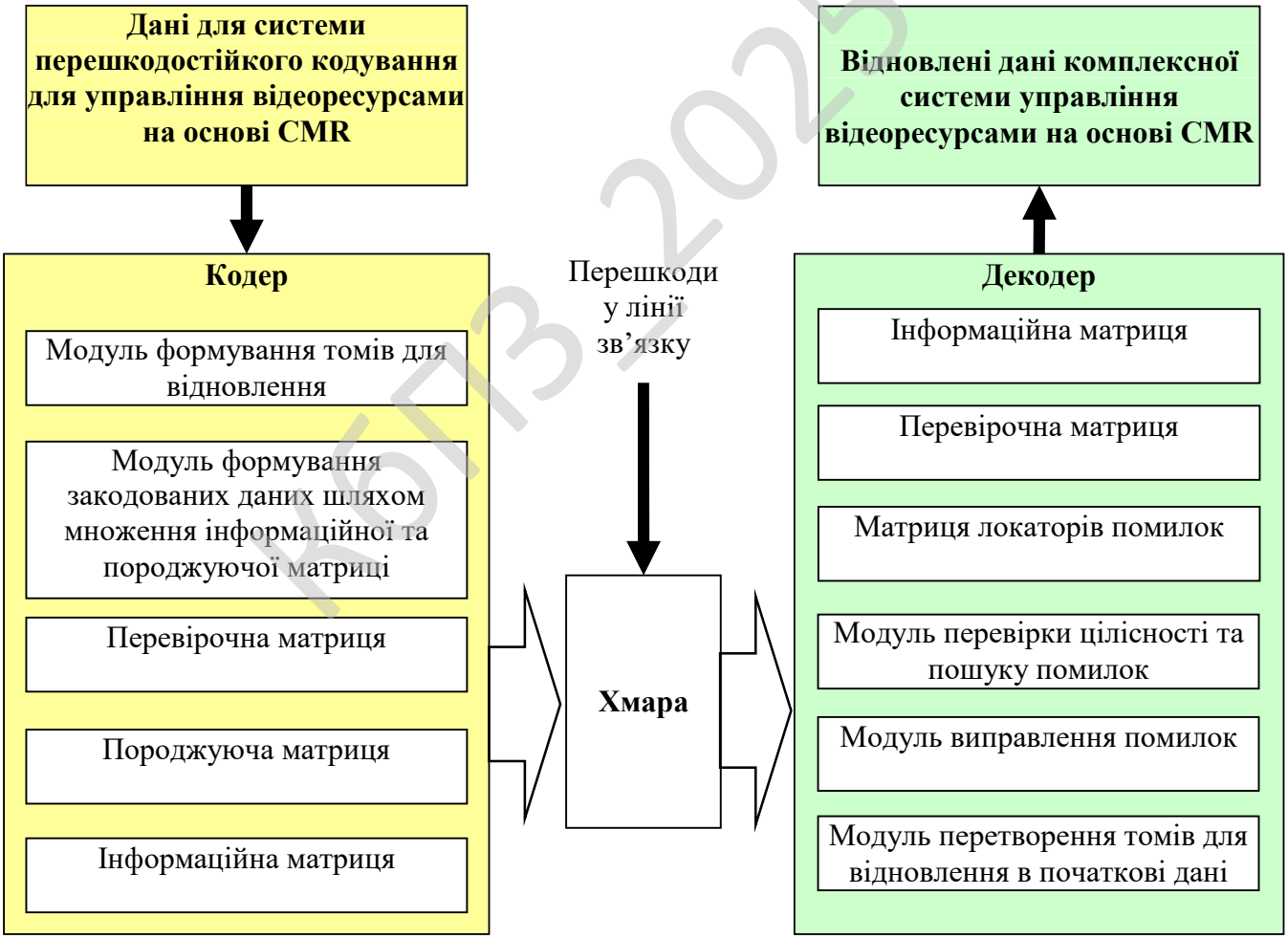

Рисунок 3.2 – Функціональна схема системи

З неї бачимо, що розроблена система перешкодостійкого кодування для управління відеоресурсами на основі CMR складається з наступних основних функціональних блоків:

– клієнт відеоконференцзв’язку;

– кодер циклічного коду, який використовується, коли відбувається відправлення інформації у системі перешкодостійкого кодування для управління відеоресурсами на основі CMR;

– декодер циклічного коду, який використовується при читанні даних з мережі;

– дані для перешкодостійкого збереження

– відновлені дані.

Розглянемо більш детально перераховані функціональні блоки кодування та декодування за допомогою циклічного кодування.

Блок кодування складається з наступних підблоків:

– Модуль формування пакетів для відновлення.

– Модуль формування закодованих даних шляхом множення інформаційної та породжуючої матриці.

– Перевірочна матриця.

– Породжуюча матриця.

– Інформаційна матриця.

Блок декодування складається з наступних підблоків:

– Інформаційна матриця.

– Перевірочна матриця.

– Матриця локаторів помилок.

– Модуль перевірки цілісності та пошуку помилок.

– Модуль виправлення помилок.

– Модуль перетворення пакетів для відновлення в початкові дані.

Коди циклічного коду базуються на спеціальному розділі математики – полях Галуа (GF) або кінцевих полях. Арифметичні дії (+, -, x, / і т.д.) над

елементами кінцевого поля дають результат, що також є елементом цього поля. Кодер та декодер циклічного коду повинні вміти виконувати ці арифметичні операції. Ці операції для своєї реалізації вимагають спеціального устаткування або спеціалізованого програмного забезпечення.

Кодове слово циклічного коду формується із залученням спеціального полінома. Всі коректні кодові слова повинні ділитися без залишку на ці утворюючі поліноми. Загальна форма утворюючого полінома має вигляд:

$$g(x) = (x\text{-}a^{i})(x\text{-}ai^{+1})\ldots(x\text{-}a^{i+2t}), \quad (3.24)$$

а кодове слово формується за допомогою операції:

$$c(x) = g(x)i(x), \quad (3.25)$$

де – $g(x)$ є утворюючим поліномом;

– $i(x)$ являє собою інформаційний блок;

– $c(x)$ – кодове слово, що називається простим елементом поля.

$2t$ символів парності в кодовому слові циклічного коду визначаються з наступного співвідношення:

$$p(x) = i(x)\cdot x^{n-k} \bmod g(x). \quad (3.26)$$

Перейдемо до розгляду іншого функціонального блоку – декодеру циклічного коду.

Декодер працює наступним чином.

Введемо позначення:

– $r(x)$ – Отримане кодове слово.

– S_i – Синдроми.

– $L(x)$ – Поліном локації помилок.

– X_i – Положення помилок.

– Y_i – Значення помилок.

– $c(x)$ – Відновлене кодове слово.

– v – Число помилок.

Отримане кодове слово r(x) являє собою вихідне (передане) кодове слово c(x) плюс помилки: $r(x) = c(x) + e(x)$.

Декодер циклічного коду намагається визначити позицію й значення помилки для числа *t* помилок (або 2*t* втрат) і виправити помилки й втрати.

Обчислення синдрому

Обчислення синдрому схоже на обчислення парності. Кодове слово циклічного коду має 2*t* **синдромів**, це залежить тільки від помилок (а не переданих кодових слів). Синдроми можуть бути обчислені шляхом підстановки 2*t* коріння утворюючого полінома *g*(*x*) в *r*(*x*).

Знаходження позицій символьних помилок

Це робиться шляхом рішення системи рівнянь із t невідомими. Існує кілька швидких алгоритмів для рішення цього завдання. Ці алгоритми використовують особливості структури матриці кодів циклічного коду й сильно скорочують необхідну обчислювальну потужність. Робиться це у два етапи:

1. Визначення полінома локації помилок

Це може бути зроблене за допомогою алгоритму Berlekamp-Massey або алгоритму Евкліда. Алгоритм Евкліда використовується частіше на практиці, тому що його легше реалізувати, однак, алгоритм Berlekamp-Massey дозволяє одержати більш ефективну реалізацію встаткування й програм.

2. Знаходження кореня цього полінома. Це робиться із залученням алгоритму пошуку Chien.

Знаходження значень символьних помилок

Тут також потрібно вирішити систему рівнянь із t невідомими. Для рішення використовується швидкий алгоритм Forney.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій

системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

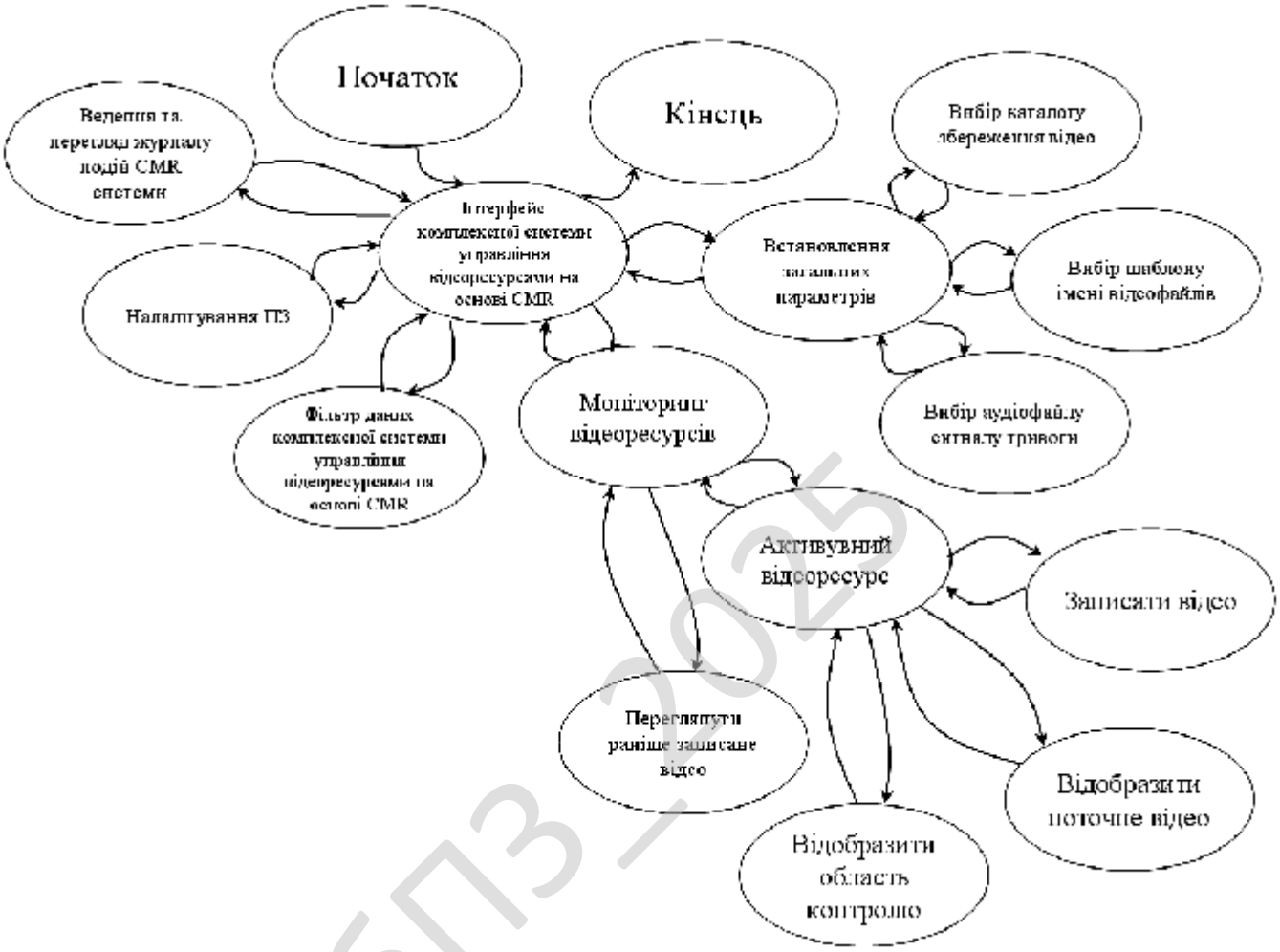

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

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

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

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

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

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

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

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

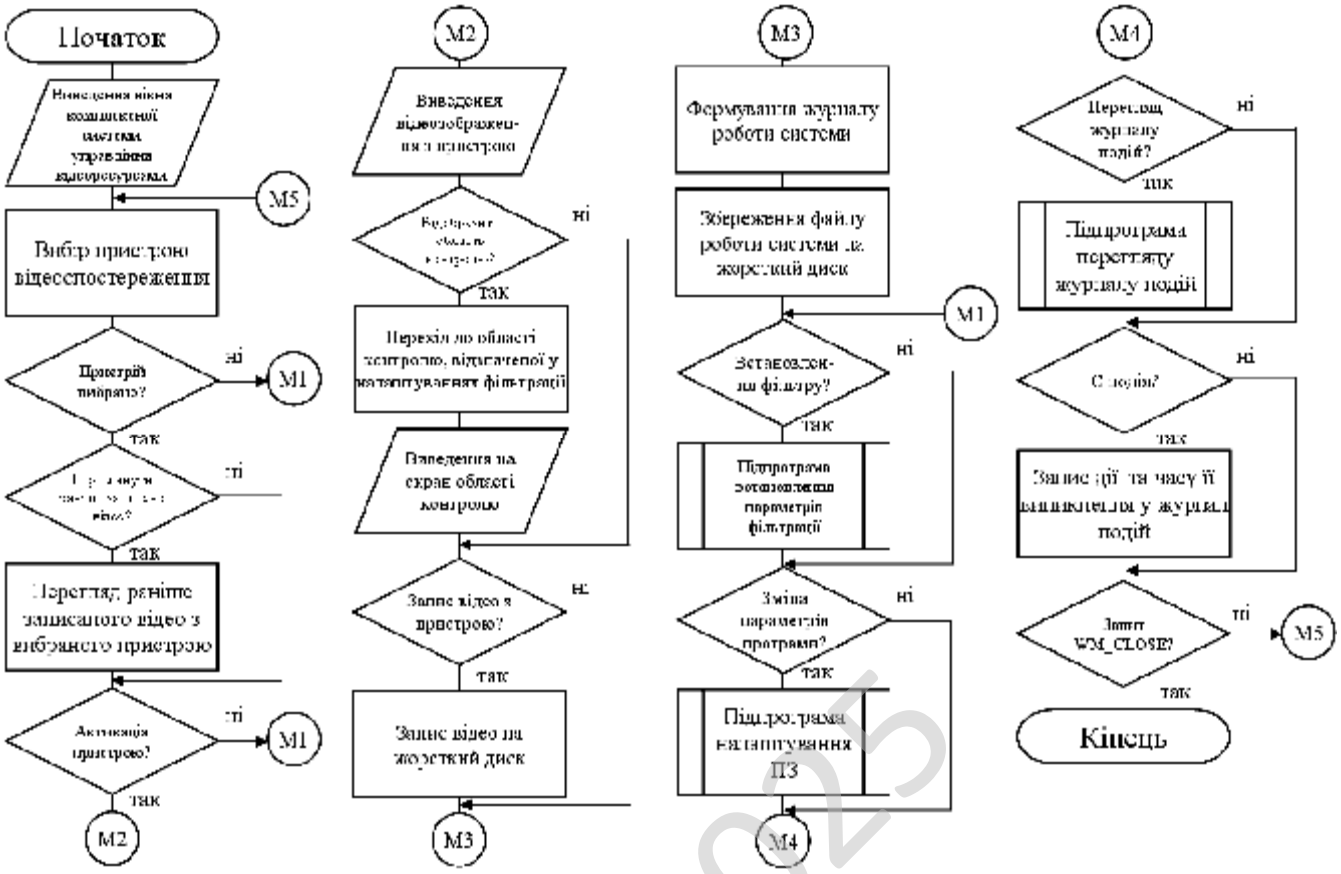

Рисунок 4.1 – Блок-схема основної програми

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних

аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

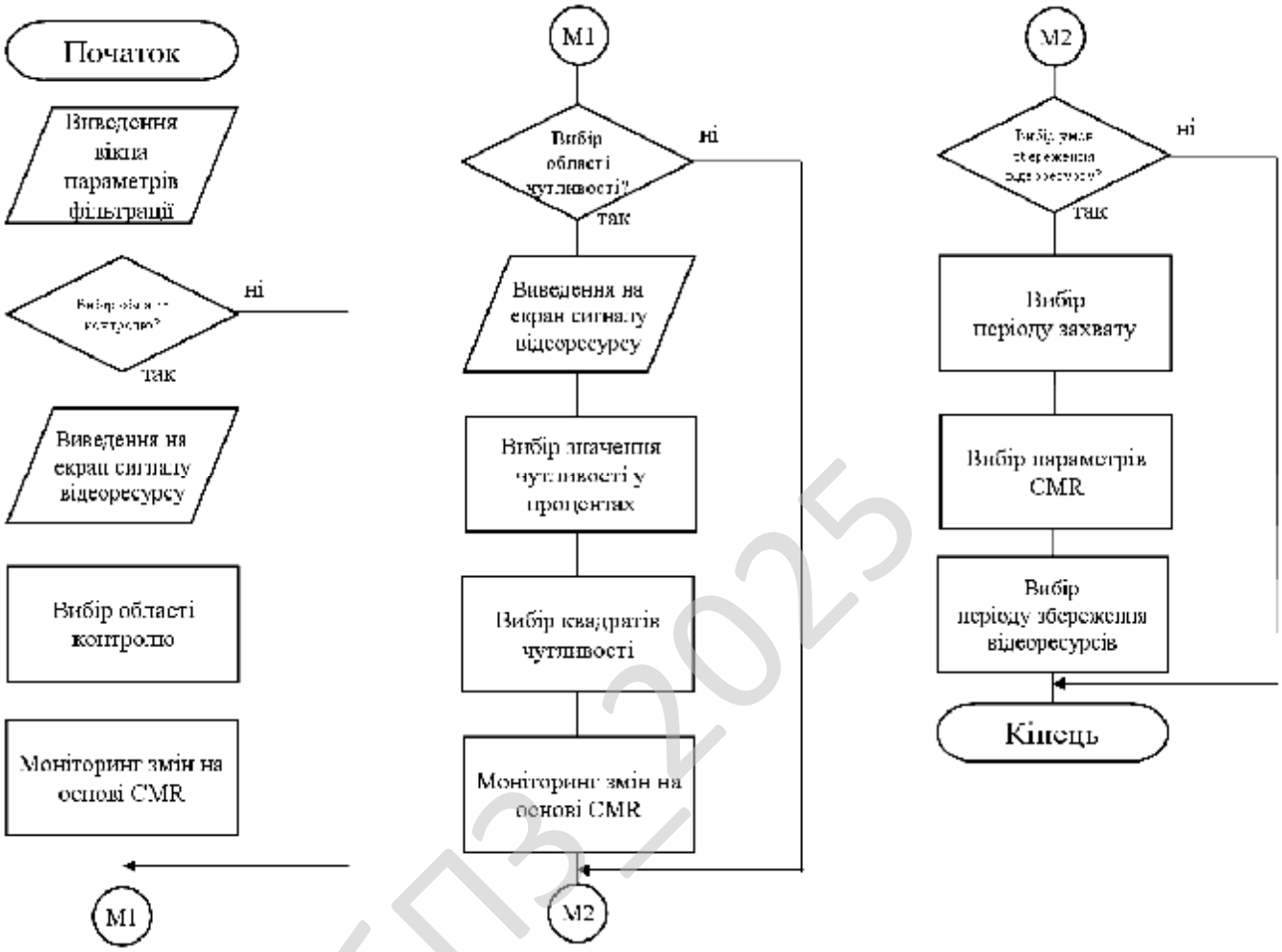

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною

одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

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

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

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

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

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

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

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

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

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

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

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

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

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

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

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого

програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

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

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

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

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

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

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

можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

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

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

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

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та

об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Наведемо частину коду підпрограми вибору пристрою для відеоспостереження:

```
TSetDeviceForm *SetDeviceForm;
void __fastcall TSetDeviceForm::FormShow(TObject *Sender)
{
  //зберегти поточні налаштування
  Sets.SaveToFile("prev.cfg");
  UpdateState();
  // перерахування пристроїв
  char szDeviceName[80];
  char szDeviceVersion[80];
  DeviceSelect->Items->Clear();
  for (int wIndex = 0; wIndex < 10; wIndex++)
  {
    if (capGetDriverDescription (wIndex, szDeviceName,
        sizeof (szDeviceName), szDeviceVersion,
        sizeof (szDeviceVersion)))
    {
// Додати ім’я до списку встановлених драйверів захвату
// та потім дозволити користувачеві вибрати драйвер для використання
DeviceSelect->Items->Add(AnsiString(szDeviceName)+"("
+AnsiString(szDeviceVersion)+")");
if(Sets.DevIndex == wIndex) DeviceSelect->Text = AnsiString(szDeviceName)+"("
+AnsiString(szDeviceVersion)+")";
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::DeviceSelectSelect(TObject *Sender)
{
   //update device index
   Sets.DevIndex = DeviceSelect->ItemIndex;
   UpdateState();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::OkBitBtnClick(TObject *Sender)
{
  DeleteFile("prev.cfg");
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::CancelBitBtnClick(TObject *Sender)
{
  Sets.LoadFromFile("prev.cfg");
  DeleteFile("prev.cfg");
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::VideoSrcBitBtnClick(TObject *Sender)
{
   capDlgVideoSource(Sets.hCaptureW);
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::FormatBitBtnClick(TObject *Sender)
{
   capDlgVideoFormat(Sets.hCaptureW);
   capGetStatus(Sets.hCaptureW, &Sets.CapStatus, sizeof (CAPSTATUS));
   Sets.VideoFormatSize = capGetVideoFormatSize(Sets.hCaptureW);
   if(Sets.VideoFormat){
      delete[] Sets.VideoFormat;
      Sets.VideoFormat = NULL;
   }
   if(Sets.VideoFormatSize > 0){
      Sets.VideoFormat = (LPBITMAPINFO) new BYTE[Sets.VideoFormatSize];
      capGetVideoFormat(Sets.hCaptureW, Sets.VideoFormat, Sets.VideoFormatSize);
   }
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::DisplayBitBtnClick(TObject *Sender)
{
   capDlgVideoDisplay(Sets.hCaptureW);
}
//---------------------------------------------------------------------------
void TSetDeviceForm::UpdateState()
{
```

```
//під'єднати новий пристрій
   capDriverDisconnect (Sets.hCaptureW);
   bool fOK = capDriverConnect(Sets.hCaptureW, Sets.DevIndex);
   if(fOK){
//отримати заголовки драйверу
      capDriverGetCaps(Sets.hCaptureW, &Sets.CapDrvCaps, sizeof (CAPDRIVERCAPS));
//отримати заголовки вікна
      capGetStatus(Sets.hCaptureW, &Sets.CapStatus, sizeof (CAPSTATUS));
//отримати доступні діалоги для драйверу
// діалогове вікно для джерела відеосигналу
      VideoSrcBitBtn->Visible = false;
      if (Sets.CapDrvCaps.fHasDlgVideoSource)
         VideoSrcBitBtn->Visible = true;
// діалогове вікно для формату відео
      FormatBitBtn->Visible = false;
      if (Sets.CapDrvCaps.fHasDlgVideoFormat)
         FormatBitBtn->Visible = true;
// діалогове вікно для відеозображення
      DisplayBitBtn->Visible = false;
      if (Sets.CapDrvCaps.fHasDlgVideoDisplay)
         DisplayBitBtn->Visible = true;
   }
}
```

Наведемо частину коду підпрограми для фільтрації відеозображення:

```
TSetFilterForm *SetFilterForm;
int SRX1, SRX2, SRY1, SRY2;
int CurSens = -1;
//---------------------------------------------------------------------------
__fastcall TSetFilterForm::TSetFilterForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TSetFilterForm::OkBitBtnClick(TObject *Sender)
{
   DeleteFile("prev.cfg");
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::CancelBitBtnClick(TObject *Sender)
```

```
{
   Sets.LoadFromFile("prev.cfg");
   DeleteFile("prev.cfg");
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SRPBPaint(TObject *Sender)
{
   DrawSR();
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SRPBMouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
  if(DrawRectButton->Down){
     SRPB->Cursor = crCross;
     if(Shift.Contains(ssLeft)){//натиснута ліва клавіша миші
        SRX2 = X;
        SRY2 = Y;
        DrawSR();
     }
  }
  else{
     SRPB->Cursor = crDefault;
  }
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SRPBMouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   if(Button == mbLeft && DrawRectButton->Down){
      SRX1 = SRX2 = X;
      SRY1 = SRY2 = Y;
   }
}
//---------------------------------------------------------------------------
void TSetFilterForm::DrawSR()
{
    TPicture *tPic = new TPicture;
    tPic->Bitmap->Width = SRPB->Width;
    tPic->Bitmap->Height =  SRPB->Height;
    tPic->Bitmap->Canvas->Brush->Color = (TColor)RGB(100,100,100);
```

```
    tPic->Bitmap->Canvas->FillRect(tPic->Bitmap->Canvas->ClipRect);
    tPic->Bitmap->Canvas->Brush->Color = clWhite;
    tPic->Bitmap->Canvas->FillRect(Rect(SRX1, SRY1, SRX2, SRY2));
    tPic->Bitmap->Canvas->CopyMode = cmSrcAnd;
    tPic->Bitmap->Canvas->StretchDraw(tPic->Bitmap->Canvas->ClipRect,
Sets.FramePic->Bitmap);
    SRPB->Canvas->Draw(0, 0, tPic->Bitmap);
    delete tPic;
}
void __fastcall TSetFilterForm::FormShow(TObject *Sender)
{
   //зберегти поточні налаштування
   Sets.SaveToFile("prev.cfg");
   SRPB->Width = Sets.pW;
   SRPB->Height = Sets.pH;
   SensPB->Width = Sets.pW;
   SensPB->Height = Sets.pH;
   if(Sets.ClipRect.bottom > 0){
      SRX1 = Sets.ClipRect.left;
      SRX2 = Sets.ClipRect.right;
      SRY1 = Sets.ClipRect.top;
      SRY2 = Sets.ClipRect.bottom;
   }else{
      SRX1 = SRY1 = 0;
      SRX2 = Sets.pW;
      SRY2 = Sets.pH;
   }
   ActivePointsEdit->Text = IntToStr(Sets.MaxActivePoints);
   AlarmTimeEdit->Text = IntToStr(Sets.AlarmMotionTime);
   CycleTimeEdit->Text = IntToStr(Sets.CycleTime);
   ForceSaveEdit->Text = IntToStr(Sets.ForceSaveTime);
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::RectSaveButtonClick(TObject *Sender)
{
   int t;
   if(SRX1 > SRX2){
      t = SRX1; SRX1 = SRX2; SRX2 = t;
   }
   if(SRY1 > SRY2){
      t = SRY1; SRY1 = SRY2; SRY2 = t;
   }
```

```
   Sets.ClipRect.left = SRX1;
   Sets.ClipRect.right = SRX2;
   Sets.ClipRect.top = SRY1;
   Sets.ClipRect.bottom = SRY2;
   if(Sets.bConnected) capGrabFrame(Sets.hCaptureW);
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SensPBPaint(TObject *Sender)
{
   DrawSensRegions();
}
//---------------------------------------------------------------------------
void TSetFilterForm::DrawSensRegions()
{
   int i;
   SensRegion *sR;
   TPicture *tPic = new TPicture;
   tPic->Bitmap->Width = SensPB->Width;
   tPic->Bitmap->Height =  SensPB->Height;
   for(i = 0; i < Sets.SensList->Count; i++){
      sR = (SensRegion*)Sets.SensList->Items[i];
      int cl = sR->sens * 255.0 / 100.0;
      tPic->Bitmap->Canvas->Brush->Color = (TColor)RGB(cl, cl, cl);
      tPic->Bitmap->Canvas->FillRect(*sR);
   }
   tPic->Bitmap->Canvas->CopyMode = cmSrcAnd;
   tPic->Bitmap->Canvas->Draw(0, 0, Sets.GrayPic->Bitmap);
   tPic->Bitmap->Canvas->Font->Color = clRed;
   tPic->Bitmap->Canvas->Pen->Width = 1;
   for(i = 0; i < Sets.SensList->Count; i++){
      sR = (SensRegion*)Sets.SensList->Items[i];
      tPic->Bitmap->Canvas->Brush->Style = bsSolid;
      tPic->Bitmap->Canvas->Brush->Color = clRed;
      tPic->Bitmap->Canvas->FrameRect(*sR);
      tPic->Bitmap->Canvas->Brush->Style = bsClear;
      tPic->Bitmap->Canvas->TextOut((sR->left+sR->right)*0.5-10, (sR->top+sR-
>bottom)*0.5-10, IntToStr(sR->sens));
   }
   SensPB->Canvas->Draw(0, 0, tPic->Bitmap);
   delete tPic;
  // SensPB->Canvas->Draw(0, 0, Sets.GrayPic->Bitmap);
}
```

```
void __fastcall TSetFilterForm::SensSpeedButtonClick(TObject *Sender)
{
   if(!((TSpeedButton*)Sender)->Down){
      CurSens = -1;
   }else{
      CurSens = StrToInt(((TSpeedButton*)Sender)->Caption);
   }
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SensPBMouseUp(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   int i;
   SensRegion *sR;
   if(CurSens != -1 && Button == mbLeft){
      for(i = 0; i < Sets.SensList->Count; i++){
         sR = (SensRegion*)Sets.SensList->Items[i];
         if(sR->left < X && sR->right > X && sR->top < Y && sR->bottom > Y){
            sR->sens = CurSens;
         }
      }
      DrawSensRegions();
   }
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SensPBMouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
   if(CurSens != -1){
      SensPB->Cursor = crHandPoint;
   }else{
      SensPB->Cursor = crDefault;
   }
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SensClearButtonClick(TObject *Sender)
{
   int i;
   SensRegion *sR;
   for(i = 0; i < Sets.SensList->Count; i++){
      sR = (SensRegion*)Sets.SensList->Items[i];
      sR->sens = 90;
```

```
    }
    DrawSensRegions();
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::AlarmApplyBitBtnClick(TObject *Sender)
{
    try{
        Sets.MaxActivePoints = StrToInt(ActivePointsEdit->Text.Trim());
        Sets.AlarmMotionTime = StrToInt(AlarmTimeEdit->Text.Trim());
        Sets.CycleTime = StrToInt(CycleTimeEdit->Text.Trim());
        Sets.ForceSaveTime = StrToInt(ForceSaveEdit->Text.Trim());
    }catch(...){
        ShowMessage("Введене число не входить в допустимий діапазон значень!");
    }
}
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використати фіналіста конкурсу AES – шифр Rijndael. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають строге математичне обґрунтування. Сама структура й послідовність операцій дозволяють виконувати даний алгоритм ефективно як на 16-бітних так і на 64-бітних процесорах. У структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцесорних робочих станціях може ще підняти швидкість шифрування в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку й довжини ключа), у яких послідовно виконуються наступні операції :

ByteSub – Таблична підстановка 8x8 біт (рисунок 4.3).

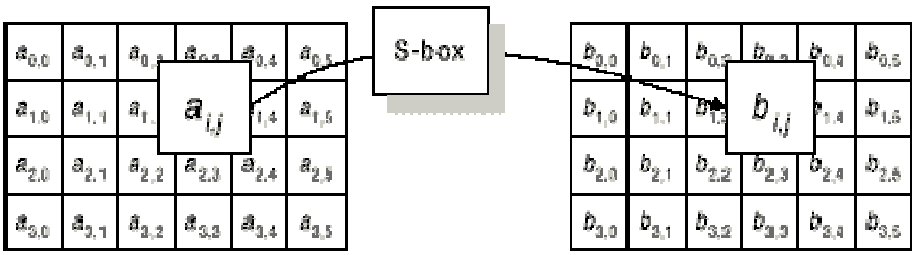

Рисунок 4.3 – Таблична підстановка 8x8 біт

ShiftRow – зрушення рядків у двовимірному масиві на різні зсуви (рисунок 4.4).

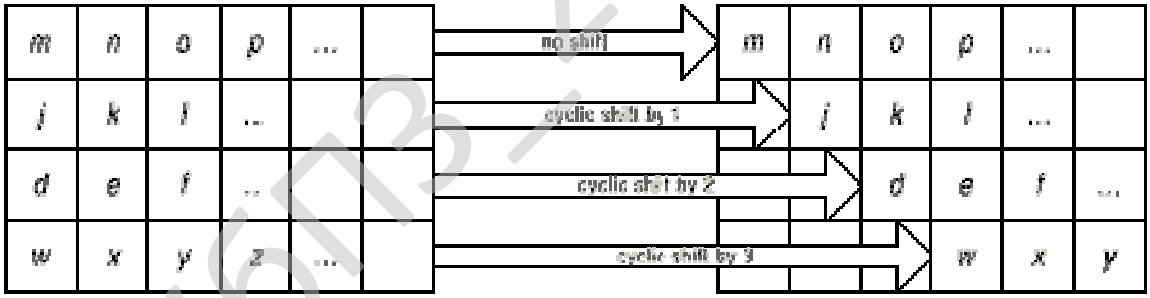

Рисунок 4.4 – Зрушення рядків у двовимірному масиві на різні зсуви

MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рисунок 4.5).

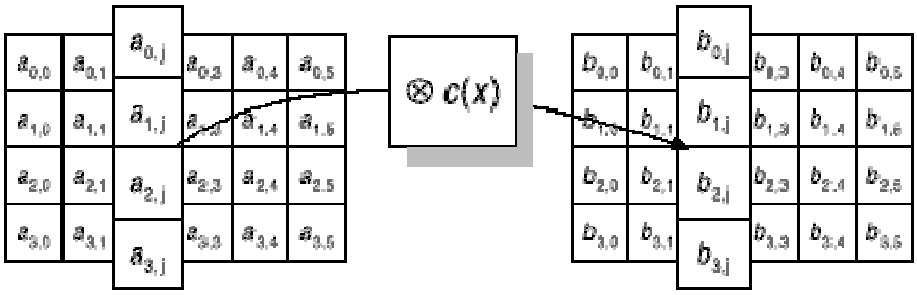

Рисунок 4.5 – Математичне перетворення, що перемішує дані усередині стовпця

AddRoundKey – додавання матеріалу ключа операцією XOR (рисунок 4.6).

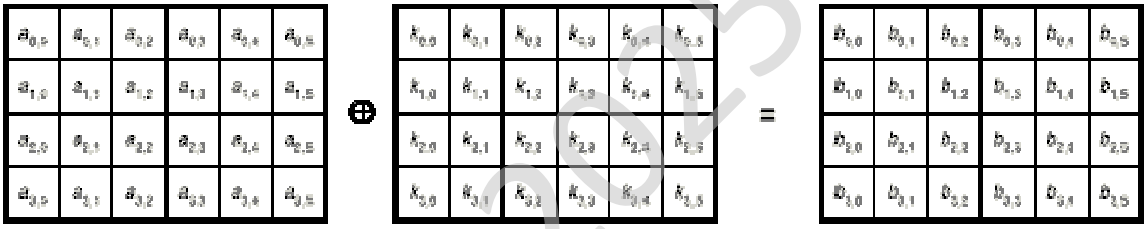

Рисунок 4.6 – Додавання матеріалу ключа операцією XOR

В останньому раунді операція перемішування стовпців відсутня, що робить всю послідовність операцій симетричною.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення комплексної системи управління відеоресурсами на основі CMR складається з наступних функціональних блоків:

– Навігаційне меню: Дії; Вид; Довідка.

– Блоку виведення даних відеоресурсу.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ: Активувати; Зняти; Переглянути; Пристрій; Фільтр; Загальні.

Рисунок 5.1 – Головне вікно розробленого ПЗ

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

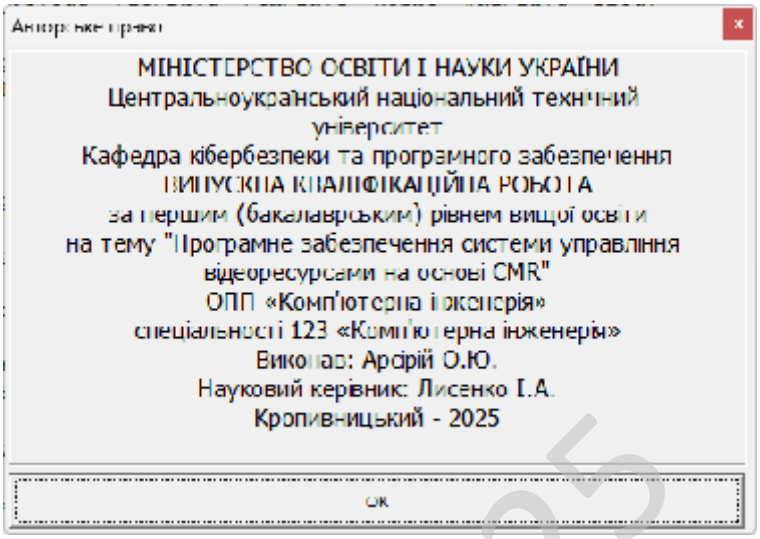

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

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

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

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

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

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

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

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

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – proprietary software.

Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж.

Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію.

Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт.

Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами.

Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення. Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи управління відеоресурсами на основі CMR.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління відеоресурсами на основі CMR.

– Досліджена система управління відеоресурсами на основі CMR.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління відеоресурсами на основі CMR.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління відеоресурсами на основі CMR.

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 призначені для системи управління відеоресурсами на основі CMR. 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 AES.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. - Д.: НГУ, 2016. - 187 с.

2. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

3. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

4. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

5. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447

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

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

8. 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.

9. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

10. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

11. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

12. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

13. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

14. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

16. 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.

17. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

18. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

19. 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.

20. 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.

21. 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.

22. 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.

23. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

24. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. Smirnov A.A., Kuznetsov 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. – 2025. – P. 61-78.

33. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

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

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

36. 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.

37. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція

“Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

38. 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.

39. 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.

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. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп’ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

42. 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.

43. 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).

44. 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

45. 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.

46. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

47. 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.

48. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

49. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". - Випуск 2 (118). т.2. - Х.: ХУПС - 2014. - С. 64-67

50. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції "Проблеми та перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. - 2014. - С. 240.

51. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник. – Кіровоград: КНТУ 2013. – 257с.

52. Смірнов О.А., Дрєєв О.М., Доренський О.П. «Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі. Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи управління відеоресурсами на основі CMR.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи управління відеоресурсами на основі CMR.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи управління відеоресурсами на основі CMR;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 81 аркуш.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Лисенко І.А.

Програмне забезпечення системи управління відеоресурсами на основі CMR

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 30

Літера: РП

Кропивницький – 2025 року

// MainUnit.cpp – основна програма

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "MainUnit.h"
#include "SetDeviceUnit.h"
#include "SetFilterUnit.h"
#include "SetCommonUnit.h"
#include "ReConst.hpp"
#include "reinit.hpp"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "trayicon"
#pragma resource "*.dfm"
TMainForm *MainForm;
DevSettings Sets;
bool bWork = false;
AnsiString ConfigName;
//---------------------------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner)
        : TForm(Owner)
{
   ConfigName = ExtractFilePath(Application->ExeName) + "va.cfg";
}
//---------------------------------------------------------------------------


void __fastcall TMainForm::ActivateButtonClick(TObject *Sender)
{
  bWork = !bWork;
  int worktime = GetTickCount(), frametime = 0;
  int lastsavetime = 0;

  if(bWork && !Sets.Connect()) return;

  Graphics::TBitmap *tbmp = new Graphics::TBitmap;

  if(bWork) ActivateButton->Caption = "Стоп";
  else ActivateButton->Caption = "Активувати";

  while (bWork){
    capGrabFrame(Sets.hCaptureW);
    switch (MainPageControl->TabIndex){
      case 0: // вхідний кадр
         PrevPB->Canvas->StretchDraw(PrevPB->ClientRect, Sets.FramePic->Bitmap);
         break;
      case 1: // контрольний кадр
         Sets.PaintArray(Sets.GrayData, tbmp);
         FramePB->Canvas->StretchDraw(FramePB->ClientRect, tbmp);
         break;
      case 2: // відфільтрований кадр
         Sets.PaintArray(Sets.FiltData, tbmp);
         FiltPB->Canvas->StretchDraw(FiltPB->ClientRect, tbmp);
         break;
      case 3: // кадр із рухом
         Sets.PaintArray(Sets.DifData, tbmp);
         MotionPB->Canvas->StretchDraw(MotionPB->ClientRect, tbmp);
         break;
    }

    if(worktime - lastsavetime > Sets.ForceSaveTime * 1000){
       lastsavetime = worktime;
       Sets.SaveFrame();
```

```
    }

    StatusBar->Panels->Items[0]->Text = "FPS: " + IntToStr(1000/(frametime ==
0?1:frametime)) + " (" + IntToStr(frametime) + "ms)";
    StatusBar->Panels->Items[1]->Text = "Кадрів оброблено: " +
IntToStr(Sets.FramesCaptured);
    StatusBar->Panels->Items[2]->Text = "Рухомих точок: " +
IntToStr(Sets.ActivePoints);

    frametime = GetTickCount() - worktime;
    while(bWork && frametime < Sets.CycleTime){
       frametime = GetTickCount() - worktime;
       Application->ProcessMessages();
       Sleep(5);
    }
    worktime = GetTickCount();
    Application->ProcessMessages();
  }
  delete tbmp;
  Sets.Disconnect();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::DeviceItemClick(TObject *Sender)
{
  if(!Sets.bConnected) Sets.Connect();
  SetDeviceForm->ShowModal();
  Sets.Connect();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::FormShow(TObject *Sender)
{
  Sets.hParentW = CaptPanel->Handle;
  Sets.LogMemo = LogMemo;
  Sets.LoadFromFile(ConfigName);

  AnsiString asLcid;
  int iLcid = 0;
  asLcid = LoadStr(_SLcid.id);
  if(asLcid != "")iLcid = asLcid.ToInt();
  if((iLcid != Sets.LocalId) && (Sets.LocalId == ENGLISH || Sets.LocalId ==
RUSSIAN)){
     if(LoadNewResourceModule(Sets.LocalId) != 0)
     {
        ReinitializeForms();
     }
  }
  asLcid = LoadStr(_SLcid.id);
  if(asLcid != "") iLcid = asLcid.ToInt();
  EnglishItem->Checked = iLcid == EnglishItem->Tag;
  RussianItem->Checked = iLcid == RussianItem->Tag;

  if(!Sets.bConnected && Sets.Connect()) capGrabFrame(Sets.hCaptureW);

}
//---------------------------------------------------------------------------

void DevSettings::SaveToFile(AnsiString FN){
  int FH = FileCreate(FN);
  if(FH != -1){
     FileWrite(FH, &DevIndex, sizeof(DevIndex));
     FileWrite(FH, &CaptureParms, sizeof (CAPTUREPARMS));
     FileWrite(FH, &CapStatus, sizeof (CAPSTATUS));
     FileWrite(FH, &VideoFormatSize, sizeof(VideoFormatSize));
     FileWrite(FH, VideoFormat, VideoFormatSize);
     FileWrite(FH, &ClipRect, sizeof(ClipRect));
     FileWrite(FH, &((int)SensList->Count), sizeof(int));
     for(int i=0; i < SensList->Count; i++){
       FileWrite(FH, SensList->Items[i], sizeof(SensRegion));
```

```
      }
      FileWrite(FH, &CycleTime, sizeof(CycleTime));
      FileWrite(FH, &MaxActivePoints, sizeof(MaxActivePoints));
      FileWrite(FH, &AlarmMotionTime, sizeof(AlarmMotionTime));
      FileWrite(FH, &ForceSaveTime, sizeof(ForceSaveTime));
      FileWrite(FH, &JPEGCompression, sizeof(JPEGCompression));
      char cstr[255];
      strcpy(cstr, ImagesFolder.c_str());
      FileWrite(FH, cstr, sizeof(cstr));
      strcpy(cstr, AlarmExecuteFN.c_str());
      FileWrite(FH, cstr, sizeof(cstr));
      FileWrite(FH, &LocalId, sizeof(LocalId));
      strcpy(cstr, FNTemplate.c_str());
      FileWrite(FH, cstr, sizeof(cstr));
      FileClose(FH);
   }
}


void DevSettings::LoadFromFile(AnsiString FN){
   int FH = FileOpen(FN, fmOpenRead);
   if(FH != -1){
      FileRead(FH, &DevIndex, sizeof(DevIndex));
      FileRead(FH, &CaptureParms, sizeof (CAPTUREPARMS));
      FileRead(FH, &CapStatus, sizeof (CAPSTATUS));
      FileRead(FH, &VideoFormatSize, sizeof(VideoFormatSize));
      if(VideoFormat){
         delete[] VideoFormat;
         VideoFormat = NULL;
      }
      if(VideoFormatSize > 0){
         VideoFormat = (LPBITMAPINFO) new BYTE[VideoFormatSize];
         FileRead(FH, VideoFormat, VideoFormatSize);
      }
      FileRead(FH, &ClipRect, sizeof(ClipRect));
      int senscnt;
      FileRead(FH, &senscnt, sizeof(senscnt));
      for(int i=0; i < senscnt && i < SensList->Count; i++){
        FileRead(FH, SensList->Items[i], sizeof(SensRegion));
      }
      FileRead(FH, &CycleTime, sizeof(CycleTime));
      FileRead(FH, &MaxActivePoints, sizeof(MaxActivePoints));
      FileRead(FH, &AlarmMotionTime, sizeof(AlarmMotionTime));
      FileRead(FH, &ForceSaveTime, sizeof(ForceSaveTime));
      FileRead(FH, &JPEGCompression, sizeof(JPEGCompression));
      char cstr[255];
      FileRead(FH, cstr, sizeof(cstr));
      ImagesFolder = cstr;
      FileRead(FH, cstr, sizeof(cstr));
      AlarmExecuteFN = cstr;
      FileRead(FH, &LocalId, sizeof(LocalId));
      FileRead(FH, cstr, sizeof(cstr));
      FNTemplate = cstr;
      if(FNTemplate == "") FNTemplate = "{dd}_{mm}_{yyyy}\\{dd}{mmm}{yy}-
{hh}{nn}{ss}{zzz}";
      FileClose(FH);
   }

}


// StatusCallbackProc: стан функції зворотнього виклику
// hWnd:               дескриптор вікна захвату
// nID:                код стану для поточного стану
// lpStatusText:       рядок повідомлення для поточного стану
//
LRESULT PASCAL DevSettings::StatusCallbackProc(HWND hWnd, int nID,
    LPSTR lpStatusText)
{
```

```
    if (nID == 0) {           // Видалити старі повідомлення про стан
        return (LRESULT) TRUE;
    }
    // Показати ID стану та текстове повідомлення про стан
    Sets.AddToLog("Status("+IntToStr(nID)+"): "+lpStatusText);
    return (LRESULT) TRUE;
}


// ErrorCallbackProc: помилка функції зворотнього виклику
// hWnd:              дескриптор вікна захвату
// nErrID:            код помилки для неочікуваної помилки
// lpErrorText:       повідомлення про помилку для неочікуваної помилки
//
LRESULT PASCAL DevSettings::ErrorCallbackProc(HWND hWnd, int nErrID,
    LPSTR lpErrorText)
{
    if (nErrID == 0)            // Початок нової головної функції
        return (LRESULT)TRUE;            // Очистити старі помилки

    // Показати ідентифікатор помилки та текст
    Sets.AddToLog("Помилка("+IntToStr(nErrID)+"): "+lpErrorText);
    MessageBox(hWnd, lpErrorText, AnsiString("Помилка
#"+IntToStr(nErrID)).c_str(), MB_OK | MB_ICONEXCLAMATION);
    return (LRESULT) TRUE;
}



// FrameCallbackProc: функція зворотнього виклику кадру
// hWnd:              дескриптор вікна захвату
// lpVHdr:            вказівник на структуру, яка містить
//                    інформацію про кадр
//
LRESULT PASCAL DevSettings::FrameCallbackProc(HWND hWnd, LPVIDEOHDR lpVHdr)
{
    Sets.UpdateFrame(lpVHdr);
    return (LRESULT) TRUE ;
}


bool DevSettings::Connect(){
   int nID = 0;
   Disconnect();
   if(!hCaptureW) hCaptureW = capCreateCaptureWindow("Система комплексної
системи управління відеоресурсами на основі CMR",
                              WS_CHILD,    0, 0, 100, 100, hParentW, nID);
   bool fOK = false;
   if(hCaptureW && DevIndex != -1){
     //установка функції зворотнього виклику
     fOK = capSetCallbackOnError(hCaptureW, &DevSettings::ErrorCallbackProc);
     if(fOK) fOK = capSetCallbackOnStatus(hCaptureW,
&DevSettings::StatusCallbackProc);
     if(fOK) fOK = capSetCallbackOnFrame(hCaptureW,
&DevSettings::FrameCallbackProc);
     //під'єднатися до драйвера
     if(fOK) fOK = capDriverConnect(hCaptureW, DevIndex);
     //установка параметрів захвату
     if(fOK && CaptureParms.dwRequestMicroSecPerFrame > 0){
        fOK = capCaptureSetSetup(hCaptureW, &CaptureParms, sizeof
(CAPTUREPARMS));
        if(!fOK) fOK = capCaptureGetSetup(hCaptureW, &CaptureParms,
sizeof(CAPTUREPARMS));
     }else if(fOK) fOK = capCaptureGetSetup(hCaptureW, &CaptureParms,
sizeof(CAPTUREPARMS));
     //установка формату відео
     if(fOK && VideoFormat){
        fOK = capSetVideoFormat(hCaptureW, VideoFormat, VideoFormatSize);
```

```
      }
      if((!fOK && VideoFormat) || (fOK && !VideoFormat)){
         VideoFormatSize = capGetVideoFormatSize(hCaptureW);
         if(VideoFormat){
            delete[] VideoFormat;
            VideoFormat = NULL;
         }
         if(VideoFormatSize > 0){
            VideoFormat = (LPBITMAPINFO) new BYTE[VideoFormatSize];
            fOK = capGetVideoFormat(hCaptureW, VideoFormat, VideoFormatSize);
         }
      }
      if(fOK){
         //установка розміру вікна
         capGetStatus(hCaptureW, &CapStatus, sizeof (CAPSTATUS));
        // SetWindowPos(hCaptureW, NULL, 0, 0, CapStatus.uiImageWidth,
        //              CapStatus.uiImageHeight, SWP_NOZORDER | SWP_NOMOVE);
         FramePic->Bitmap->Width = CapStatus.uiImageWidth;
         FramePic->Bitmap->Height = CapStatus.uiImageHeight;
         FramePic->Bitmap->HandleType = bmDIB;
         FramePic->Bitmap->PixelFormat = pf24bit;
         FramePic->Bitmap->IgnorePalette = true;

         GrayPic->Bitmap->Width = pW;
         GrayPic->Bitmap->Height = pH;
         GrayPic->Bitmap->HandleType = bmDIB;
         GrayPic->Bitmap->PixelFormat = pf24bit;
         GrayPic->Bitmap->IgnorePalette = true;

         if(GrayData) delete[] GrayData;
         GrayData = new BYTE[pW * pH];
         memset(GrayData, 0, pW*pH);

         //if(FiltData) delete[] FiltData;
         //FiltData = new BYTE[pW * pH];
      }

   }
   if(!fOK){
      AddToLog("Помилка: неможливо під'єднатися!");
      bConnected = false;
   }else{
      bConnected = true;
      AddToLog("Стан: Драйвер захвату під'єднано");
   }
   CurBufFrame = 0;
   FramesCaptured = 0;
   return fOK;
}



bool DevSettings::Disconnect(){
   if(!bConnected && !hCaptureW) return true;
   if(hCaptureW){
     // вивільнити функції зворотнього виклику
     capSetCallbackOnError(hCaptureW, NULL);
     capSetCallbackOnStatus(hCaptureW, NULL);
     capSetCallbackOnVideoStream(hCaptureW, NULL);
     capSetCallbackOnFrame(hCaptureW, NULL);
     //від'єднатися від пристрою
     capDriverDisconnect(hCaptureW);
     DestroyWindow(hCaptureW);
     hCaptureW = NULL;
   }
   bConnected = false;
   AddToLog("Стан: Драйвер захвату від'єднано");
   return true;
}
```

```
void DevSettings::UpdateFrame(LPVIDEOHDR lpVHdr)
{
   HDC hdc;
   HDRAWDIB hdd;
   hdd = DrawDibOpen();

   //виводити вихідний кадр в FramePic
   hdc = FramePic->Bitmap->Canvas->Handle ;
   FramePic->Bitmap->Canvas->TryLock();
   DrawDibBegin(hdd, hdc,
                VideoFormat->bmiHeader.biWidth, //dxDest
                VideoFormat->bmiHeader.biHeight, //dyDest
                &VideoFormat->bmiHeader, //lpbi
                VideoFormat->bmiHeader.biWidth, //dxSrc
                VideoFormat->bmiHeader.biHeight, //dySrc
                NULL);
   DrawDibDraw(hdd, hdc,
               0, //xDst
               0, //yDst
               VideoFormat->bmiHeader.biWidth, //dxDst
               VideoFormat->bmiHeader.biHeight, //dyDst
               &VideoFormat->bmiHeader, //lpbi
               lpVHdr->lpData, //lpBits
               0, //xSrc
               0, //ySrc
               VideoFormat->bmiHeader.biWidth, //dxSrc
               VideoFormat->bmiHeader.biHeight, //dySrc
               NULL);
   DrawDibEnd(hdd);
   FramePic->Bitmap->Canvas->Unlock();


   //виводити вихідний кадр в GrayPic (для обробки)
   DWORD xSrc = 0, ySrc = 0,
         dxSrc = VideoFormat->bmiHeader.biWidth,
         dySrc = VideoFormat->bmiHeader.biHeight;
   if(ClipRect.bottom > 0){
      //використати відсікання
      float MX = dxSrc / (float)pW,
            MY = dySrc / (float)pH;
      xSrc = ClipRect.left * MX;
      ySrc = ClipRect.top * MX;
      dxSrc = (ClipRect.right - ClipRect.left) * MX;
      if(dxSrc > (DWORD)VideoFormat->bmiHeader.biWidth - xSrc) dxSrc =
VideoFormat->bmiHeader.biWidth - xSrc;
      dySrc = (ClipRect.bottom - ClipRect.top) * MY;
      if(dySrc > (DWORD)VideoFormat->bmiHeader.biHeight - ySrc) dySrc =
VideoFormat->bmiHeader.biHeight - ySrc;
   }
   hdc = GrayPic->Bitmap->Canvas->Handle ;
   GrayPic->Bitmap->Canvas->TryLock();
   DrawDibBegin(hdd, hdc,
                pW, //dxDest
                pH, //dyDest
                &VideoFormat->bmiHeader, //lpbi
                dxSrc,
                dySrc,
                NULL);
   //DrawDibRealize(hdd, hdc, fBackground);
   DrawDibDraw(hdd, hdc,
               0, //xDst
               0, //yDst
               pW, //dxDst
               pH, //dyDst
               &VideoFormat->bmiHeader, //lpbi
               lpVHdr->lpData, //lpBits
               xSrc,
               ySrc,
```

```
                dxSrc,
                dySrc,
                NULL);
   DrawDibEnd(hdd);
   DrawDibClose(hdd);
   GrayPic->Bitmap->Canvas->Unlock();

   // створити зображення у відтінках сірого і автоматично збалансувати його
   BITMAP bm;
   GetObject(GrayPic->Bitmap->Handle,sizeof(BITMAP),&bm);
   register DWORD offs;
   DWORD maxoffs = bm.bmHeight * bm.bmWidthBytes ;
   register int Y;
   BYTE minY = 255, maxY = 0;
   for(offs = 0; offs < maxoffs; offs+= 3){
         Y =
((int)((BYTE*)bm.bmBits)[offs]+(int)((BYTE*)bm.bmBits)[offs+1]+(int)((BYTE*)bm.b
mBits)[offs+2]) / 3;
         if(minY > Y) minY = Y;
         if(maxY < Y) maxY = Y;
         GrayData[offs/3] = Y;
   }

   maxoffs = maxoffs/3;
   if(maxY - minY == 0){
     AddToLog(AnsiString("Помилка захвату зображення ")+__LINE__+" "+__FILE__);
     return;
   }
   static float M = 255/(maxY - minY);
   static float avminY = minY, avmaxY = maxY;
   avmaxY = 0.95*avmaxY + 0.05*maxY;
   if(avmaxY > 255) avmaxY = 255;

   avminY = 0.95*avminY + 0.05*minY;
   if(avminY > 250) avminY = 250;

   if(avmaxY <= avminY) avmaxY = avminY + 1;
   M = 255/(avmaxY - avminY);
   for(offs = 0; offs < maxoffs; offs++){
      Y = ((int)GrayData[offs] - avminY) * M ;
      if(Y < 0) GrayData[offs] = 0;
      else if(Y > 255) GrayData[offs] = 255;
      else GrayData[offs] = Y;
   }


   CurBufFrame++;
   CurBufFrame = CurBufFrame % (FrameBufCnt-1);

   FiltData = (BYTE*)FrameBuffer->Items[CurBufFrame];
   FilterFrame(GrayData, FiltData);
   FramesCaptured ++;

   if(FramesCaptured > 2){
      CompareFrames(FiltData,
                    (BYTE*)FrameBuffer->Items[CalcBufIndex(CurBufFrame-1)],
                    DifData );
   }

}



void __fastcall TMainForm::FormClose(TObject *Sender, TCloseAction &Action)
{
  bWork = false;
  Sets.SaveToFile(ConfigName);
  Sets.Disconnect();
}
```

```
//---------------------------------------------------------------------------


void __fastcall TMainForm::FilterItemClick(TObject *Sender)
{
   if(!Sets.bConnected) Sets.Connect();
   if(Sets.bConnected) capGrabFrame(Sets.hCaptureW);
   SetFilterForm->ShowModal();
}
//---------------------------------------------------------------------------


void DevSettings::FilterFrame(BYTE* Src, BYTE* Dst)
{
   register int idx, idy, x, y, i, offs, toffs, cnt;
   int ytop, ybot, dx, dy, srw, srh;
   float SY;
   DWORD sum;
   SensRegion *sr = (SensRegion*)SensList->Items[0];
   srw = sr->right - sr->left;
   srh = sr->bottom - sr->top;

   for(i=0; i<SensList->Count; i++){
      sr = (SensRegion*)SensList->Items[i];
      ybot = pH-sr->top; ytop = pH-sr->bottom;
      dx = (100 - sr->sens) * (srw)/400;
      dy = (100 - sr->sens) * (srh)/300;
      SY =  sr->sens/100.0*0.4 + 0.6;
      //dx=0; dy=5;
      for(y = ytop; y < ybot; y++){
         for(x = sr->left; x < sr->right; x++){
            offs = pW*y + x;
            if(sr->sens == 0){
               Dst[offs] = 0;
            }else if(sr->sens == 100){
               Dst[offs] = Src[offs];
            }
            else{
               sum = 0;
               cnt = 0;
               for(idy = y-dy; idy <= y+dy; idy++){
                  for(idx = x-dx; idx <= x+dx; idx++){
                    toffs = pW*((idy < ytop)? idy+srh : (idy >= ybot)? idy-srh
: idy) +
                                ((idx < sr->left)? idx+srw : ((idx >= sr->right)?
idx-srw : idx));
                   //  toffs = pW*((idy < 0)? idy+srh : (idy >= pH)? idy-srh :
idy) +
                   //          ((idx < 0)? idx+srw : ((idx >= pW)? idx-srw :
idx));
                     sum+= Src[toffs];
                     cnt++;
                  }
               }
               Dst[offs] = sum / cnt * SY;
            }
         }
      }
   }
}

void DevSettings::PaintArray(BYTE * arr, Graphics::TBitmap * bmp)
{
   BITMAP bm;
   bmp->Width = pW;
   bmp->Height = pH;
   bmp->PixelFormat = pf24bit;
   GetObject(bmp->Handle, sizeof(BITMAP), &bm);
   DWORD maxoffs = bm.bmHeight * bm.bmWidthBytes,
```

```
        offs;
   BYTE Y;
   if(!arr) return;
   for(offs = 0; offs < maxoffs; offs+= 3){
        Y = arr[offs/3];
        ((BYTE*)bm.bmBits)[offs] = Y;
        ((BYTE*)bm.bmBits)[offs+1] = Y;
        ((BYTE*)bm.bmBits)[offs+2] = Y;
   }
}



void DevSettings::CompareFrames(BYTE * f1, BYTE * f2, BYTE * dest)
{
    register int offs, maxoffs = pW * pH, Y;
    static int StartMotionTime;
    ActivePoints = 0;
    for(offs = 0; offs < maxoffs; offs++){
       Y = abs(f2[offs] - f1[offs]) * 5;
       if(Y >= 255){
          Y = 255;
          ActivePoints ++;
       }
       dest[offs] = Y;
    }

    if(ActivePoints >= MaxActivePoints){
       if(MotionTime == 0) StartMotionTime = GetTickCount();
       MotionTime = GetTickCount() - StartMotionTime + 1;
       SaveFrame();
       if(MotionTime > AlarmMotionTime * 1000){
          DoAlarm();
          MotionTime = 0;
       }
    }else{
       MotionTime = 0;
    }

}


int inline DevSettings::CalcBufIndex(int arg)
{
   return ((arg >= FrameBufCnt-1)? arg%(FrameBufCnt-1) : ((arg < 0)? arg +
FrameBufCnt - 1 : arg));
}


void __fastcall TMainForm::PrevPBPaint(TObject *Sender)
{
   Graphics::TBitmap *tbmp = new Graphics::TBitmap;

   switch (MainPageControl->TabIndex){
      case 0: //вхідний кадр
         PrevPB->Canvas->StretchDraw(PrevPB->ClientRect, Sets.FramePic->Bitmap);
         break;
      case 1: //контрольний кадр
         Sets.PaintArray(Sets.GrayData, tbmp);
         FramePB->Canvas->StretchDraw(FramePB->ClientRect, tbmp);
         break;
      case 2: //відфільтрований кадр
         Sets.PaintArray(Sets.FiltData, tbmp);
         FiltPB->Canvas->StretchDraw(FiltPB->ClientRect, tbmp);
         break;
      case 3: //кадр із рухом
         Sets.PaintArray(Sets.DifData, tbmp);
         MotionPB->Canvas->StretchDraw(MotionPB->ClientRect, tbmp);
```

```
         break;
   }

   delete tbmp;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::SaveItemClick(TObject *Sender)
{
   Sets.SaveToFile(ConfigName);
}
//---------------------------------------------------------------------------



void __fastcall TMainForm::LogMemoChange(TObject *Sender)
{
   if(LogMemo->Lines->Count > 0)
      StatusBar->Panels->Items[3]->Text = LogMemo->Lines->Strings[LogMemo-
>Lines->Count-1];
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Show1Click(TObject *Sender)
{
   Application->Restore();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Hide1Click(TObject *Sender)
{
   Application->Minimize();
}
//---------------------------------------------------------------------------


void DevSettings::DoAlarm()
{
    AddToLog("ТРИВОГА!");

    if(AlarmExecuteFN != "" && FileExists(AlarmExecuteFN)){
       if(ExtractFileExt(AlarmExecuteFN).LowerCase() == ".wav"){
          PlaySound(AlarmExecuteFN.c_str(), NULL, SND_ASYNC|SND_FILENAME);
       }else{
          ShellExecute(
                       hParentW,    // дескриптор батьківського вікна
                       "open",      // вказівник на рядок, який визначає
операцію для виконання
                       AlarmExecuteFN.c_str(), // вказівник  на рядок із іменем
файлу або папки
                       NULL,  // вказівник на рядок, який вихначає параметри
виконуваних файлів
                       NULL,  // вказівник на рядок, який визначає папку по
замовчуванню
                       SW_SHOWDEFAULT          // чи показується файл, коли його
відкрили
                       );
       }
    }

}


void DevSettings::SaveFrame()
{
  AnsiString FN = EvalImgPath();
  if(!DirectoryExists(ExtractFilePath(FN)))
ForceDirectories(ExtractFilePath(FN));
```

```
  FramePic->Bitmap->Canvas->TextOut(0, 0, DateTimeToStr(Now()));

  TJPEGImage *jp = new TJPEGImage();
  try
  {
    jp->Assign(FramePic->Bitmap);
    jp->CompressionQuality = JPEGCompression;
    jp->Compress();
    jp->SaveToFile(FN);
  }
  catch(...){
    delete jp;
    return;
  }
  delete jp;
  AddToLog("Кадр збережено");
}

AnsiString DevSettings::EvalImgPath(){
  static unsigned int sn = 0;
  sn++ ;
  TDateTime dt = Now();
  AnsiString tS, res = FNTemplate;
  AnsiString dd, ddd, mm, mmm, yy, yyyy, hh, nn, ss, zzz;
  DateTimeToString(dd, "dd", dt);
  DateTimeToString(ddd, "ddd", dt);
  DateTimeToString(mm, "mm", dt);
  DateTimeToString(mmm, "mmm", dt);
  DateTimeToString(yy, "yy", dt);
  DateTimeToString(yyyy, "yyyy", dt);
  DateTimeToString(hh, "hh", dt);
  DateTimeToString(nn, "nn", dt);
  DateTimeToString(ss, "ss", dt);
  DateTimeToString(zzz, "zzz", dt);
  DateTimeToString(tS, "dd_mmm_yyyy", dt);  // {dd}_{mmm}_{yyyy}\\{dd}{mmm}{yy}-
{hh}{nn}{ss}{zzz}.jpg
  AnsiString FN = ImagesFolder + "\\" + tS + "\\";
  //if(!DirectoryExists(FN)) CreateDir(FN);
  DateTimeToString(tS, "ddmmmyy-hhnnsszzz", Now());
  FN = FN + tS + ".jpg";
  TReplaceFlags rf; rf << rfReplaceAll;
  res = StringReplace(res, "{dd}", dd, rf);
  res = StringReplace(res, "{ddd}", ddd, rf);
  res = StringReplace(res, "{mm}", mm, rf);
  res = StringReplace(res, "{mmm}", mmm, rf);
  res = StringReplace(res, "{yy}", yy, rf);
  res = StringReplace(res, "{yyyy}", yyyy, rf);
  res = StringReplace(res, "{hh}", hh, rf);
  res = StringReplace(res, "{nn}", nn, rf);
  res = StringReplace(res, "{ss}", ss, rf);
  res = StringReplace(res, "{zzz}", zzz, rf);
  res = StringReplace(res, "{sn}", IntToStr(sn), rf);
  return ImagesFolder + "\\" + res + ".jpg";
}

void __fastcall TMainForm::CommonItemClick(TObject *Sender)
{
   SetCommonForm->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::OpenImgFolderBitBtnClick(TObject *Sender)
{
    ShellExecute(
                Handle,      // дескриптор батьківського вікна
                "open",      // вказівник на рядок, який визначає операцію для
виконання
                "explorer.exe",    // вказівник  на рядок із іменем файлу або
папки
```

```
                Sets.ImagesFolder.c_str(),     // вказівник на рядок, який
вихначає параметри виконуваних файлів
                NULL,  // вказівник на рядок, який визначає папку по
замовчуванню
                SW_SHOWDEFAULT           // чи показується файл, коли його
відкрили
                );
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::ShootNowBitBtnClick(TObject *Sender)
{
   if(!Sets.bConnected) Sets.Connect();
   if(Sets.bConnected){
      capGrabFrame(Sets.hCaptureW);
      Sets.SaveFrame();
      PrevPBPaint(Sender);

   }
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::FormCreate(TObject *Sender)
{
  EnglishItem->Tag = ENGLISH;
  RussianItem->Tag = RUSSIAN;

  AnsiString asLcid;
  int iLcid = 0;
  asLcid = LoadStr(_SLcid.id);
  if(asLcid != "") iLcid = asLcid.ToInt();

  EnglishItem->Checked = iLcid == EnglishItem->Tag;
  RussianItem->Checked = iLcid == RussianItem->Tag;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::EnglishItemClick(TObject *Sender)
{
  if(LoadNewResourceModule(dynamic_cast<TComponent*>(Sender)->Tag) != 0)
  {
    bWork = false;
    Sets.Disconnect();
    Application->ProcessMessages();
    Sets.LocalId = dynamic_cast<TComponent*>(Sender)->Tag;
    Sets.SaveToFile(ConfigName);
    ReinitializeForms();
    EnglishItem->Checked = EnglishItem == Sender;
    RussianItem->Checked  = RussianItem == Sender;
  }

}
//---------------------------------------------------------------------------
```

// MainUnit.h – заголовочний файл основної програми

```
//---------------------------------------------------------------------------

#ifndef MainUnitH
#define MainUnitH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include "vfw.h"
#include <ExtCtrls.hpp>
#include <ComCtrls.hpp>
#include <Graphics.hpp>
#include <Buttons.hpp>
#include <jpeg.hpp>
#include <ActnList.hpp>
#include <ExtActns.hpp>
//---------------------------------------------------------------------------
struct SensRegion:public RECT{
   int sens;
};

class DevSettings{
  public:
    HWND hParentW,
         hCaptureW;
    TMemo *LogMemo;
    TPicture *FramePic;
    TPicture *GrayPic;
    TList *SensList;
    TList *FrameBuffer;
    BYTE *GrayData,
         *FiltData, *DifData;
    int pW, pH;
    int SensRectNx, SensRectNy;
    int FrameBufCnt;
    DWORD FramesCaptured;
    int ActivePoints;
    int CurBufFrame;
    int DevIndex;
    bool bConnected;
    int CycleTime;
    int MaxActivePoints;
    int AlarmMotionTime;
    int ForceSaveTime;
    int MotionTime;
    AnsiString AlarmExecuteFN;
    AnsiString ImagesFolder;
    AnsiString FNTemplate;
    int JPEGCompression;
    int LocalId;

    CAPDRIVERCAPS CapDrvCaps;
    CAPSTATUS CapStatus;
    CAPTUREPARMS CaptureParms;
    LPBITMAPINFO VideoFormat;
    DWORD VideoFormatSize;
    RECT ClipRect;

    DevSettings(){
       JPEGCompression = 70;
       ImagesFolder = ".";
       FNTemplate = "{dd}_{mm}_{yyyy}\\{dd}{mmm}{yy}-{hh}{nn}{ss}{zzz}";
       MotionTime = 0;      //  мс
       CycleTime = 300;     //  мс
```

```
        MaxActivePoints = 10;
        AlarmMotionTime = 1; // в секундах
        ForceSaveTime = 600; // секунди
        DevIndex = -1;
        VideoFormatSize = 0;
        pW = 320; pH = 240;
        SensRectNx = SensRectNy = 8;
        ClipRect.bottom = 0;
        FrameBufCnt = 10;
        ActivePoints = 0;
        bConnected = false;
        hParentW = hCaptureW = VideoFormat = NULL;
        LogMemo = NULL;
        GrayData = FiltData = NULL;
        FramePic = new TPicture;
        GrayPic = new TPicture;
        SensList = new TList;
        FrameBuffer = new TList;
        int nx, ny, dx, dy;
        SensRegion *sR;
        dx = pW / SensRectNx;
        dy = pH / SensRectNy;
        for(nx = 0; nx < SensRectNx; nx++){
           for(ny = 0; ny < SensRectNy; ny++){
              sR = new SensRegion;
              sR->left = nx * dx;
              sR->right = sR->left + dx;
              sR->top = ny*dy;
              sR->bottom = sR->top + dy;
              sR->sens = 100;
              SensList->Add(sR);
           }
        }
        BYTE *tbuf;
        for(nx = 0; nx < FrameBufCnt; nx++){
           tbuf = new BYTE[pW*pH];
           memset(tbuf, 0, pW*pH);
           FrameBuffer->Add(tbuf);
        }
        DifData = (BYTE*)FrameBuffer->Items[FrameBufCnt - 1];
    }
    ~DevSettings(){
        if(VideoFormat) delete[] VideoFormat;
        if(GrayData) delete[] GrayData;
        //if(FiltData) delete[] FiltData;
        delete FramePic;
        delete GrayPic;
        for(int i = 0; i < SensList->Count; i++){
          delete (SensRegion*)(SensList->Items[i]);
        }
        delete SensList;

        for(int i = 0; i < FrameBuffer->Count; i++){
          delete (BYTE*)(FrameBuffer->Items[i]);
        }
        delete FrameBuffer;
    }
    void SaveToFile(AnsiString FN);
    void LoadFromFile(AnsiString FN);
    bool Connect();
    bool Disconnect();
    //callback functions
    static LRESULT PASCAL StatusCallbackProc(HWND hWnd, int nID, LPSTR
lpStatusText);
    static LRESULT PASCAL ErrorCallbackProc(HWND hWnd, int nErrID, LPSTR
lpErrorText);
    static LRESULT PASCAL FrameCallbackProc(HWND hWnd, LPVIDEOHDR lpVHdr);
    void AddToLog(AnsiString S){
      if(LogMemo) LogMemo->Lines->Add("["+TimeToStr(Time())+"] "+S);
```

```
    }
    void UpdateFrame(LPVIDEOHDR lpVHdr);
        void FilterFrame(BYTE* Src, BYTE* Dst);
        void PaintArray(BYTE * arr, Graphics::TBitmap * bmp);
        void CompareFrames(BYTE * f1, BYTE * f2, BYTE * dest);
        void DoAlarm();
        void SaveFrame();
        AnsiString EvalImgPath();

private:
        int inline CalcBufIndex(int arg);

} extern Sets;



class TMainForm : public TForm
{
__published:      // IDE-managed Components
        TMainMenu *MainMenu1;
        TMenuItem *SetItem;
        TMenuItem *DeviceItem;
        TPageControl *MainPageControl;
        TTabSheet *TabSheet1;
        TTabSheet *TabSheet2;
        TPanel *CaptPanel;
        TMemo *LogMemo;
        TTabSheet *TabSheet3;
        TPaintBox *FramePB;
        TMenuItem *FilterItem;
        TPopupMenu *TrayPopupMenu;
        TMenuItem *Show1;
        TTabSheet *TabSheet4;
        TPaintBox *FiltPB;
        TTabSheet *TabSheet5;
        TPaintBox *MotionPB;
        TPaintBox *PrevPB;
        TMenuItem *N1;
        TMenuItem *SaveItem;
        TMenuItem *CommonItem;
        TGroupBox *GroupBox1;
        TBitBtn *SetDeviceBitBtn;
        TBitBtn *SetFilterBitBtn;
        TBitBtn *SetCommonBitBtn;
        TStatusBar *StatusBar;
        TMenuItem *Hide1;
        TBitBtn *OpenImgFolderBitBtn;
        TActionList *ActionList1;
        TBrowseURL *BrowseURL1;
        TMenuItem *WebSite1;
        TGroupBox *GroupBox2;
        TBitBtn *ShootNowBitBtn;
        TButton *ActivateButton;
        TMenuItem *Language1;
        TMenuItem *EnglishItem;
        TMenuItem *RussianItem;
        void __fastcall ActivateButtonClick(TObject *Sender);
        void __fastcall DeviceItemClick(TObject *Sender);
        void __fastcall FormShow(TObject *Sender);
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall FilterItemClick(TObject *Sender);
        void __fastcall PrevPBPaint(TObject *Sender);
        void __fastcall SaveItemClick(TObject *Sender);
        void __fastcall LogMemoChange(TObject *Sender);
        void __fastcall Show1Click(TObject *Sender);
        void __fastcall Hide1Click(TObject *Sender);
        void __fastcall CommonItemClick(TObject *Sender);
        void __fastcall OpenImgFolderBitBtnClick(TObject *Sender);
        void __fastcall ShootNowBitBtnClick(TObject *Sender);
```

```
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall EnglishItemClick(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TMainForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TMainForm *MainForm;
//---------------------------------------------------------------------------
#endif
```

// SetFilterUnit.cpp – модуль фільтрації відеозображення

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "SetFilterUnit.h"
#include "MainUnit.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSetFilterForm *SetFilterForm;
int SRX1, SRX2, SRY1, SRY2;
int CurSens = -1;

//---------------------------------------------------------------------------
__fastcall TSetFilterForm::TSetFilterForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::OkBitBtnClick(TObject *Sender)
{
   DeleteFile("prev.cfg");
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::CancelBitBtnClick(TObject *Sender)
{
   Sets.LoadFromFile("prev.cfg");
   DeleteFile("prev.cfg");
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetFilterForm::SRPBPaint(TObject *Sender)
{
   DrawSR();
}
//---------------------------------------------------------------------------

void __fastcall TSetFilterForm::SRPBMouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
  if(DrawRectButton->Down){
     SRPB->Cursor = crCross;
     if(Shift.Contains(ssLeft)){//натиснута ліва клавіша миші
        SRX2 = X;
        SRY2 = Y;
        DrawSR();
     }
  }
  else{
     SRPB->Cursor = crDefault;
  }
}
//---------------------------------------------------------------------------

void __fastcall TSetFilterForm::SRPBMouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   if(Button == mbLeft && DrawRectButton->Down){
      SRX1 = SRX2 = X;
      SRY1 = SRY2 = Y;
   }

}
//---------------------------------------------------------------------------
```

```
void TSetFilterForm::DrawSR()
{
    TPicture *tPic = new TPicture;
    tPic->Bitmap->Width = SRPB->Width;
    tPic->Bitmap->Height =  SRPB->Height;
    tPic->Bitmap->Canvas->Brush->Color = (TColor)RGB(100,100,100);
    tPic->Bitmap->Canvas->FillRect(tPic->Bitmap->Canvas->ClipRect);
    tPic->Bitmap->Canvas->Brush->Color = clWhite;
    tPic->Bitmap->Canvas->FillRect(Rect(SRX1, SRY1, SRX2, SRY2));


    tPic->Bitmap->Canvas->CopyMode = cmSrcAnd;
    tPic->Bitmap->Canvas->StretchDraw(tPic->Bitmap->Canvas->ClipRect,
Sets.FramePic->Bitmap);
    SRPB->Canvas->Draw(0, 0, tPic->Bitmap);

    delete tPic;
}
void __fastcall TSetFilterForm::FormShow(TObject *Sender)
{
   //зберегти поточні налаштування
   Sets.SaveToFile("prev.cfg");

   SRPB->Width = Sets.pW;
   SRPB->Height = Sets.pH;
   SensPB->Width = Sets.pW;
   SensPB->Height = Sets.pH;
   if(Sets.ClipRect.bottom > 0){
      SRX1 = Sets.ClipRect.left;
      SRX2 = Sets.ClipRect.right;
      SRY1 = Sets.ClipRect.top;
      SRY2 = Sets.ClipRect.bottom;
   }else{
      SRX1 = SRY1 = 0;
      SRX2 = Sets.pW;
      SRY2 = Sets.pH;
   }

   ActivePointsEdit->Text = IntToStr(Sets.MaxActivePoints);
   AlarmTimeEdit->Text = IntToStr(Sets.AlarmMotionTime);
   CycleTimeEdit->Text = IntToStr(Sets.CycleTime);
   ForceSaveEdit->Text = IntToStr(Sets.ForceSaveTime);

}
//---------------------------------------------------------------------------

void __fastcall TSetFilterForm::RectSaveButtonClick(TObject *Sender)
{
   int t;
   if(SRX1 > SRX2){
      t = SRX1; SRX1 = SRX2; SRX2 = t;
   }

   if(SRY1 > SRY2){
      t = SRY1; SRY1 = SRY2; SRY2 = t;
   }

   Sets.ClipRect.left = SRX1;
   Sets.ClipRect.right = SRX2;
   Sets.ClipRect.top = SRY1;
   Sets.ClipRect.bottom = SRY2;
   if(Sets.bConnected) capGrabFrame(Sets.hCaptureW);
}
//---------------------------------------------------------------------------

void __fastcall TSetFilterForm::SensPBPaint(TObject *Sender)
{
```

```
   DrawSensRegions();
}
//---------------------------------------------------------------------------


void TSetFilterForm::DrawSensRegions()
{
   int i;
   SensRegion *sR;
   TPicture *tPic = new TPicture;
   tPic->Bitmap->Width = SensPB->Width;
   tPic->Bitmap->Height =  SensPB->Height;

   for(i = 0; i < Sets.SensList->Count; i++){
      sR = (SensRegion*)Sets.SensList->Items[i];
      int cl = sR->sens * 255.0 / 100.0;
      tPic->Bitmap->Canvas->Brush->Color = (TColor)RGB(cl, cl, cl);
      tPic->Bitmap->Canvas->FillRect(*sR);
   }

   tPic->Bitmap->Canvas->CopyMode = cmSrcAnd;
   tPic->Bitmap->Canvas->Draw(0, 0, Sets.GrayPic->Bitmap);

   tPic->Bitmap->Canvas->Font->Color = clRed;
   tPic->Bitmap->Canvas->Pen->Width = 1;

   for(i = 0; i < Sets.SensList->Count; i++){
      sR = (SensRegion*)Sets.SensList->Items[i];
      tPic->Bitmap->Canvas->Brush->Style = bsSolid;
      tPic->Bitmap->Canvas->Brush->Color = clRed;
      tPic->Bitmap->Canvas->FrameRect(*sR);
      tPic->Bitmap->Canvas->Brush->Style = bsClear;
      tPic->Bitmap->Canvas->TextOut((sR->left+sR->right)*0.5-10, (sR->top+sR-
>bottom)*0.5-10, IntToStr(sR->sens));
   }

   SensPB->Canvas->Draw(0, 0, tPic->Bitmap);

   delete tPic;
  // SensPB->Canvas->Draw(0, 0, Sets.GrayPic->Bitmap);

}

void __fastcall TSetFilterForm::SensSpeedButtonClick(TObject *Sender)
{
   if(!((TSpeedButton*)Sender)->Down){
      CurSens = -1;
   }else{
      CurSens = StrToInt(((TSpeedButton*)Sender)->Caption);
   }

}
//---------------------------------------------------------------------------


void __fastcall TSetFilterForm::SensPBMouseUp(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   int i;
   SensRegion *sR;
   if(CurSens != -1 && Button == mbLeft){
      for(i = 0; i < Sets.SensList->Count; i++){
         sR = (SensRegion*)Sets.SensList->Items[i];
         if(sR->left < X && sR->right > X && sR->top < Y && sR->bottom > Y){
            sR->sens = CurSens;
         }
      }
      DrawSensRegions();
   }
```

```
}
//---------------------------------------------------------------------------

void __fastcall TSetFilterForm::SensPBMouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
   if(CurSens != -1){
      SensPB->Cursor = crHandPoint;
   }else{
      SensPB->Cursor = crDefault;
   }
}
//---------------------------------------------------------------------------

void __fastcall TSetFilterForm::SensClearButtonClick(TObject *Sender)
{
   int i;
   SensRegion *sR;
   for(i = 0; i < Sets.SensList->Count; i++){
      sR = (SensRegion*)Sets.SensList->Items[i];
      sR->sens = 90;
   }
   DrawSensRegions();
}
//---------------------------------------------------------------------------



void __fastcall TSetFilterForm::AlarmApplyBitBtnClick(TObject *Sender)
{
   try{
      Sets.MaxActivePoints = StrToInt(ActivePointsEdit->Text.Trim());
      Sets.AlarmMotionTime = StrToInt(AlarmTimeEdit->Text.Trim());
      Sets.CycleTime = StrToInt(CycleTimeEdit->Text.Trim());
      Sets.ForceSaveTime = StrToInt(ForceSaveEdit->Text.Trim());
   }catch(...){
      ShowMessage("Введене число не входить в допустимий діапазон значень!");
   }
}
//---------------------------------------------------------------------------
```

// SetFilterUnit.h – заголовочний файл модуля фільтрації відеозображення

```
//---------------------------------------------------------------------------

#ifndef SetFilterUnitH
#define SetFilterUnitH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Buttons.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TSetFilterForm : public TForm
{
__published:      // IDE-компоненти
        TBitBtn *OkBitBtn;
        TBitBtn *CancelBitBtn;
        TPageControl *PageControl1;
        TTabSheet *TabSheet1;
        TPaintBox *SRPB;
        TTabSheet *TabSheet2;
        TBitBtn *RectSaveButton;
        TSpeedButton *DrawRectButton;
        TPaintBox *SensPB;
        TBitBtn *SensClearButton;
        TSpeedButton *SensSpeedButton;
        TSpeedButton *SpeedButton2;
        TSpeedButton *SpeedButton3;
        TSpeedButton *SpeedButton4;
        TSpeedButton *SpeedButton5;
        TSpeedButton *SpeedButton6;
        TSpeedButton *SpeedButton7;
        TSpeedButton *SpeedButton8;
        TSpeedButton *SpeedButton9;
        TSpeedButton *SpeedButton10;
        TSpeedButton *SpeedButton11;
        TTabSheet *TabSheet3;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TEdit *CycleTimeEdit;
        TEdit *ActivePointsEdit;
        TEdit *AlarmTimeEdit;
        TEdit *ForceSaveEdit;
        TLabel *Label8;
        TBitBtn *AlarmApplyBitBtn;
        void __fastcall OkBitBtnClick(TObject *Sender);
        void __fastcall CancelBitBtnClick(TObject *Sender);
        void __fastcall SRPBPaint(TObject *Sender);
        void __fastcall SRPBMouseMove(TObject *Sender, TShiftState Shift,
          int X, int Y);
        void __fastcall SRPBMouseDown(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y);
        void __fastcall FormShow(TObject *Sender);
        void __fastcall RectSaveButtonClick(TObject *Sender);
        void __fastcall SensPBPaint(TObject *Sender);
        void __fastcall SensSpeedButtonClick(TObject *Sender);
        void __fastcall SensPBMouseUp(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y);
        void __fastcall SensPBMouseMove(TObject *Sender, TShiftState Shift,
          int X, int Y);
        void __fastcall SensClearButtonClick(TObject *Sender);
```

```
        void __fastcall AlarmApplyBitBtnClick(TObject *Sender);
private:    // користувацькі об'яви
public:           // користувацькі об'яви
        __fastcall TSetFilterForm(TComponent* Owner);
        void DrawSR();
        void DrawSensRegions();
};
//---------------------------------------------------------------------------
extern PACKAGE TSetFilterForm *SetFilterForm;
//---------------------------------------------------------------------------
#endif
```

// SetDeviceUnit.cpp – модуль вибору пристрою для комплексної системи управління відеоресурсами на основі CMR

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "SetDeviceUnit.h"
#include "MainUnit.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSetDeviceForm *SetDeviceForm;
//---------------------------------------------------------------------------
__fastcall TSetDeviceForm::TSetDeviceForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::FormShow(TObject *Sender)
{
  //зберегти поточні налаштування
  Sets.SaveToFile("prev.cfg");

  UpdateState();

  // перерахування пристроїв
  char szDeviceName[80];
  char szDeviceVersion[80];

  DeviceSelect->Items->Clear();
  for (int wIndex = 0; wIndex < 10; wIndex++)
  {
    if (capGetDriverDescription (wIndex, szDeviceName,
        sizeof (szDeviceName), szDeviceVersion,
        sizeof (szDeviceVersion)))
    {
        // Додати ім’я до списку встановлених драйверів захвату
        // та потім дозволити користувачеві вибрати драйвер для використання
        DeviceSelect->Items-
>Add(AnsiString(szDeviceName)+"("+AnsiString(szDeviceVersion)+")");
        if(Sets.DevIndex == wIndex) DeviceSelect->Text =
AnsiString(szDeviceName)+"("+AnsiString(szDeviceVersion)+")";
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::DeviceSelectSelect(TObject *Sender)
{
   //update device index
   Sets.DevIndex = DeviceSelect->ItemIndex;
   UpdateState();


}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::OkBitBtnClick(TObject *Sender)
{
  DeleteFile("prev.cfg");
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::CancelBitBtnClick(TObject *Sender)
{
  Sets.LoadFromFile("prev.cfg");
  DeleteFile("prev.cfg");
  Close();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::VideoSrcBitBtnClick(TObject *Sender)
{
   capDlgVideoSource(Sets.hCaptureW);
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::FormatBitBtnClick(TObject *Sender)
{
   capDlgVideoFormat(Sets.hCaptureW);
   capGetStatus(Sets.hCaptureW, &Sets.CapStatus, sizeof (CAPSTATUS));

   Sets.VideoFormatSize = capGetVideoFormatSize(Sets.hCaptureW);
   if(Sets.VideoFormat){
      delete[] Sets.VideoFormat;
      Sets.VideoFormat = NULL;
   }
   if(Sets.VideoFormatSize > 0){
      Sets.VideoFormat = (LPBITMAPINFO) new BYTE[Sets.VideoFormatSize];
      capGetVideoFormat(Sets.hCaptureW, Sets.VideoFormat, Sets.VideoFormatSize);
   }
}
//---------------------------------------------------------------------------
void __fastcall TSetDeviceForm::DisplayBitBtnClick(TObject *Sender)
{
   capDlgVideoDisplay(Sets.hCaptureW);
}
//---------------------------------------------------------------------------


void TSetDeviceForm::UpdateState()
{
   //під'єднати новий пристрій
   capDriverDisconnect (Sets.hCaptureW);
   bool fOK = capDriverConnect(Sets.hCaptureW, Sets.DevIndex);

   if(fOK){
      //отримати заголовки драйверу
      capDriverGetCaps(Sets.hCaptureW, &Sets.CapDrvCaps, sizeof
(CAPDRIVERCAPS));
      //отримати заголовки вікна
      capGetStatus(Sets.hCaptureW, &Sets.CapStatus, sizeof (CAPSTATUS));
      //отримати доступні діалоги для драйверу
      // діалогове вікно для джерела відеосигналу
      VideoSrcBitBtn->Visible = false;
      if (Sets.CapDrvCaps.fHasDlgVideoSource)
         VideoSrcBitBtn->Visible = true;
      // діалогове вікно для формату відео
      FormatBitBtn->Visible = false;
      if (Sets.CapDrvCaps.fHasDlgVideoFormat)
         FormatBitBtn->Visible = true;
      // діалогове вікно для відеозображення
      DisplayBitBtn->Visible = false;
      if (Sets.CapDrvCaps.fHasDlgVideoDisplay)
         DisplayBitBtn->Visible = true;

   }
}
```

// SetDeviceUnit.h – заголовочний файл модуля вибору пристрою для комплексної системи управління відеоресурсами на основі CMR

```
//---------------------------------------------------------------------------

#ifndef SetDeviceUnitH
#define SetDeviceUnitH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Buttons.hpp>
//---------------------------------------------------------------------------
class TSetDeviceForm : public TForm
{
__published:      // IDE-managed Components
        TComboBox *DeviceSelect;
        TBitBtn *OkBitBtn;
        TBitBtn *CancelBitBtn;
        TBitBtn *VideoSrcBitBtn;
        TBitBtn *FormatBitBtn;
        TBitBtn *DisplayBitBtn;
        TLabel *Label1;
        void __fastcall FormShow(TObject *Sender);
        void __fastcall DeviceSelectSelect(TObject *Sender);
        void __fastcall OkBitBtnClick(TObject *Sender);
        void __fastcall CancelBitBtnClick(TObject *Sender);
        void __fastcall VideoSrcBitBtnClick(TObject *Sender);
        void __fastcall FormatBitBtnClick(TObject *Sender);
        void __fastcall DisplayBitBtnClick(TObject *Sender);
private:    // користувацькі об'яви
public:           // користувацькі об'яви
        __fastcall TSetDeviceForm(TComponent* Owner);
        void UpdateState();
};
//---------------------------------------------------------------------------
extern PACKAGE TSetDeviceForm *SetDeviceForm;
//---------------------------------------------------------------------------
#endif
```

// SetCommonUnit.cpp – модуль вибору параметрів комплексної системи управління відеоресурсами на основі CMR

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "SetCommonUnit.h"
#include "MainUnit.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "cdiroutl"
#pragma link "CSPIN"
#pragma resource "*.dfm"
TSetCommonForm *SetCommonForm;
//---------------------------------------------------------------------------
__fastcall TSetCommonForm::TSetCommonForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::OkBitBtnClick(TObject *Sender)
{
   Sets.ImagesFolder = ImageDirEdit->Text;
   Sets.JPEGCompression = JPEGCompressionEdit->Value;
   Sets.AlarmExecuteFN = AlarmOpenEdit->Text;
   DeleteFile("prev.cfg");
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::CancelBitBtnClick(TObject *Sender)
{
   Sets.LoadFromFile("prev.cfg");
   DeleteFile("prev.cfg");
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::FormShow(TObject *Sender)
{
   //зберегти поточні налаштування
   Sets.SaveToFile("prev.cfg");

   DLB->Directory = Sets.ImagesFolder;
   ImageDirEdit->Text = Sets.ImagesFolder;
   FNTemplateEdit->Text = Sets.FNTemplate;
   FNExampleLabel->Caption = Sets.EvalImgPath();
   JPEGCompressionEdit->Value = Sets.JPEGCompression;
   AlarmOpenEdit->Text = Sets.AlarmExecuteFN;
   AlarmOpenDialog->FileName = Sets.AlarmExecuteFN;
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::ImagePathSpeedButtonClick(TObject *Sender)
{
   DLPanel->Show();
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::DLOkButtonClick(TObject *Sender)
{
  ImageDirEdit->Text = DLB->Directory;
  DLPanel->Hide();
  Sets.ImagesFolder = ImageDirEdit->Text;
  Sets.FNTemplate = FNTemplateEdit->Text;
  FNExampleLabel->Caption = Sets.EvalImgPath();
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::DLCancelButtonClick(TObject *Sender)
```

```
{
   DLPanel->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::AlarmExecuteSpeedButtonClick(
      TObject *Sender)
{
   if(AlarmOpenDialog->Execute()){
      AlarmOpenEdit->Text = AlarmOpenDialog->FileName;
   }
}
//---------------------------------------------------------------------------
void __fastcall TSetCommonForm::OnAlarmNOExecuteSpeedButtonClick(
      TObject *Sender)
{
   AlarmOpenEdit->Clear();
}
//---------------------------------------------------------------------------

void __fastcall TSetCommonForm::FNTemplateEditKeyUp(TObject *Sender,
      WORD &Key, TShiftState Shift)
{
   Sets.FNTemplate = FNTemplateEdit->Text;
   FNExampleLabel->Caption = Sets.EvalImgPath();
}
//---------------------------------------------------------------------------


void __fastcall TSetCommonForm::HideFNHelpButtonClick(TObject *Sender)
{
  FNHelpBox->Visible = false;
}
//---------------------------------------------------------------------------

void __fastcall TSetCommonForm::SpeedButton2Click(TObject *Sender)
{
  FNHelpBox->Visible = true;
}
//---------------------------------------------------------------------------

void __fastcall TSetCommonForm::ClearFNTemplateButtonClick(TObject *Sender)
{
  FNTemplateEdit->Text = "{dd}_{mm}_{yyyy}\\{dd}{mmm}{yy}-{hh}{nn}{ss}{zzz}";
  Sets.FNTemplate = FNTemplateEdit->Text;
  FNExampleLabel->Caption = Sets.EvalImgPath();
}
//---------------------------------------------------------------------------
```

// SetCommonUnit.h – заголовочний файл модуля вибору параметрів комплексної системи управління відеоресурсами на основі CMR

```
//---------------------------------------------------------------------------

#ifndef SetCommonUnitH
#define SetCommonUnitH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Buttons.hpp>
#include "cdiroutl.h"
#include <Grids.hpp>
#include <Outline.hpp>
#include <FileCtrl.hpp>
#include <ExtCtrls.hpp>
#include "CSPIN.h"
#include <Dialogs.hpp>


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


class TSetCommonForm : public TForm
{
__published:      // IDE-managed Components
        TBitBtn *OkBitBtn;
        TBitBtn *CancelBitBtn;
        TEdit *ImageDirEdit;
        TSpeedButton *ImagePathSpeedButton;
        TPanel *DLPanel;
        TDirectoryListBox *DLB;
        TDriveComboBox *DCB;
        TButton *DLOkButton;
        TButton *DLCancelButton;
        TLabel *Label1;
        TCSpinEdit *JPEGCompressionEdit;
        TLabel *Label2;
        TLabel *Label3;
        TEdit *AlarmOpenEdit;
        TSpeedButton *AlarmExecuteSpeedButton;
        TOpenDialog *AlarmOpenDialog;
        TSpeedButton *OnAlarmNOExecuteSpeedButton;
        TGroupBox *GroupBox1;
        TMemo *AboutMemo;
        TLabel *Label4;
        TEdit *FNTemplateEdit;
        TLabel *FNExampleLabel;
        TSpeedButton *ClearFNTemplateButton;
        TSpeedButton *SpeedButton2;
        TGroupBox *FNHelpBox;
        TMemo *Memo1;
        TButton *HideFNHelpButton;
        void __fastcall OkBitBtnClick(TObject *Sender);
        void __fastcall CancelBitBtnClick(TObject *Sender);
        void __fastcall FormShow(TObject *Sender);
        void __fastcall ImagePathSpeedButtonClick(TObject *Sender);
        void __fastcall DLOkButtonClick(TObject *Sender);
        void __fastcall DLCancelButtonClick(TObject *Sender);
        void __fastcall AlarmExecuteSpeedButtonClick(TObject *Sender);
        void __fastcall OnAlarmNOExecuteSpeedButtonClick(TObject *Sender);
        void __fastcall FNTemplateEditKeyUp(TObject *Sender, WORD &Key,
          TShiftState Shift);
        void __fastcall HideFNHelpButtonClick(TObject *Sender);
        void __fastcall SpeedButton2Click(TObject *Sender);
        void __fastcall ClearFNTemplateButtonClick(TObject *Sender);
```

```
private:    // користувацькі об’яви
public:           // користувацькі об’яви
        __fastcall TSetCommonForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TSetCommonForm *SetCommonForm;
//---------------------------------------------------------------------------
#endif
```