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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки стійкого відеоспостереження”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Чернат Є.С.
«____» _____ 2024 р.

Керівник проекту
докт. техн. наук, професор
_____ Смірнов О.А.
«____» _____ 2024 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 *125 "*Кібербезпека*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Кібербезпека*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Чернату Єгору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки стійкого відеоспостереження*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 135-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки стійкого відеоспостереження*

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 кібербезпеки в промислову експлуатацію.
6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2024 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	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Чернат Є.С.
(прізвище та ініціали)

АНОТАЦІЯ

Чернат Є.С. Програмне забезпечення системи кібербезпеки стійкого відеоспостереження. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки стійкого відеоспостереження.

Метою розробки є програмне забезпечення системи кібербезпеки стійкого відеоспостереження.

Результат роботи – програмна реалізація системи кібербезпеки стійкого відеоспостереження.

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 Delphi 10.4 Sydney.

Ключові слова: кібербезпека, відеоспостереження

ABSTRACT

Chernat E.S. Software of the cyber security system of persistent video surveillance. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of persistent video surveillance.

The purpose of the development is the software of the cyber security system of persistent video surveillance.

The result of the work is the software implementation of the cyber security system of sustainable video surveillance.

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

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

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

The program was developed in the Delphi 10.4 Sydney environment.

Keywords: cyber security, video surveillance

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. Відеоспостереження застосовується в сучасному світі повсюдно:

– для охорони об'єктів;

– боротьби з тероризмом;

– моніторингу дорожнього руху;

– у наукових дослідженнях.

Ці завдання зачіпають багато аспектів соціального життя і є надзвичайно актуальними.

З появою цифрових відеореєстраторів з'явилася можливість обробки відеоданих за допомогою персонального комп'ютера або спеціалізованих чипів, що привело до появи нового кола завдань у цифровій обробці сигналів. Збільшення швидкодії процесорів дозволило обробляти цифрові відеодані в реальному часі, завдяки чому коло завдань відеоспостереження, розв'язуваних за допомогою комп'ютерів, розширюється з кожним роком. Наприклад, до них відносяться:

– розпізнавання номерів автомобілів;

– виявлення й розпізнавання осіб;

– вивчення поводження тварин;

– виявлення об'єктів, що рухаються.

У даній роботі досліджується проблема надійного відеоспостереження в великих приміщеннях. Створені методи й програмна система становлять інтерес для організацій які займаються забезпеченням безпеки на складах, вокзалах, стоянках, у виставочних залах і на інших великих територіях. Для ведення відеоспостереження в таких приміщеннях потрібна велика кількість відеокамер. Завдяки їхньому здешевленню, це вже не є проблемою. Однак, операторові системи відеоспостереження доводиться відслідковувати величезний потік даних. Що складно навіть при наявності автоматичного виявлення об'єктів, що

рухаються. Для зменшення навантаження на оператора зроблений наступний крок у відеоспостереженні – системи відеоспостереження.

Завдання відеоспостереження полягає у визначенні положення об'єкта на плані спостережуваної території по відеоданим, отриманим від однієї або декількох камер. Завдання відеоспостереження не може ефективно вирішуватися без використання поворотних (PTZ) камер. Дана робота орієнтована головним чином на рішення проблем, що виникають при роботі з такими камерами.

Завдання відеоспостереження розділяється на три підзадачі:

– калібрування плану – визначення відповідності між точками в тривимірному просторі охоронюваної території й точками на двомірному плані;

– калібрування відеокамер – визначення положення й орієнтації відеокамер у просторі;

– виявлення областей інтересу – визначення областей кадру, що відповідають об'єктам, які рухаються, що раніше були відсутні або пропали.

Калібрування плану виконується шляхом завдання світових координат для двох точок плану й не викликають труднощів. У роботі досліджуються завдання калібрування відеокамер і виявлення областей інтересу.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки стійкого відеоспостереження.

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

– Огляд існуючих систем стійкого відеоспостереження.

– Дослідження системи кібербезпеки стійкого відеоспостереження.

– Програмна реалізація системи кібербезпеки стійкого відеоспостереження.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі стійкого відеоспостереження.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки стійкого відеоспостереження, 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

Система призначена для реалізації програмного забезпечення системи стійкого відеоспостереження. Система забезпечення безпеки – це камери відеоспостереження, які будуть записувати все, що відбувається в полі їх видимості: 24 години на добу, 7 днів на тиждень.

Системи відеоспостереження бувають зовнішні і внутрішні. За якістю системи відеоспостереження діляться на стандартні і професійні. У кожного типу є свої переваги і недоліки, а також потреби.

Стандартні камери відеоспостереження допомагають домогтися гарної якості зображення.

Професійні камери відеоспостереження, використовуючи сучасну технологію IP-відеоспостереження, дозволяють отримати відмінно деталізовану картинку, щоб можна було розгледіти обличчя злочинця, номер машини тощо.

Внутрішні камери відеоспостереження не потребують доповненнь і можуть бути встановлені без особливих проблем.

Зовнішні камери відеоспостереження повинні мати якісний корпус, який буде стійкий до можливого вандалізму, і не буде пропускати воду в разі дощу або снігу. Також зовнішня камера відеоспостереження повинна бути оснащена системою підігріву, щоб у разі сильних заморозків, всі системи були в робочому стані і продовжували роботу.

1.2 Область застосування

Областю застосування є системи стійкого відеоспостереження. В наш час в Україні спостерігається все більш зростаюче бажання громадян забезпечувати свою безпеку та безпеку особистого майна за допомогою інноваційних технологій, здатних задовольнити їх найвищі вимоги. Однією з таких технологій

є системи охоронного спостереження. Як відомо, поняття «відеоспостереження» давно є буденним. Для жителів міст давно вже не в дивину бачити камери відеоконтролю в торгових центрах, офісах, метрополітенах, супермаркетах та інших місцях масового скупчення народу. В офісах різних компаній, крім камер відкритого відеоспостереження, встановлюють приховані камери.

Можна нескінченно сперечатися про законність та моральність прихованого відеоспостереження, але багато керівників підприємств або звичайні громадяни стали перед необхідністю установки прихованих пристроїв відеоконтролю.

Приховані відеокамери монтують з наступних причин:

– захист відеокамер від вандалізму;

– захист від зазіхань на власність;

– контроль за персоналом;

– контроль за подіями вдома.

Як і будь-які охоронні заходи, установка прихованої відеокамери має свої переваги і свої недоліки. Розглянемо їх докладніше.

Позитивні аспекти

Приховану камеру не видно, а тому захищена від зловмисників або вандалів; порушник, якщо попереджений спеціальної інформаційної табличкою, не відчуває себе безкарним, крім цього він не знає напрямки і зони огляду камери, тому не може її обійти або зламати; прихована камера не «напружує» законослухняних громадян; крім цього є можливість установки прихованого мікрофона, що дає можливість виробляти не тільки відео огляд, але і записувати звук.

Негативні аспекти

Вартість установки такого пристрою більш висока (а якщо в офісі або квартирі був зроблений ремонт до установки прихованої відеокамери, то додайте ще вартість косметичного ремонту, якщо камера вмонтовується в стіну). Монтаж займає більшу кількість часу, ніж при установці камери відкритого спостереження; обслуговування таких камер більш трудомістке, а як наслідок –

більш дороге; якість переданого зображення ненабагато, але все ж гірше, ніж у звичайної відеокамери.

Правда, встановлювати приховані відеокамери можна не тільки в стіни і стелі. Щоб не псувати обробку офісів і квартир можна монтувати відеокамери в систему охоронно-пожежної сигналізації, протипожежні димові датчики, приховувати за підвісними стелями, за фотографіями в рамах і картинами. Бувають відеокамери, закамуфльовані під будь-які предмети офісного інтер'єру – годинник, квіткові горщики і т. п. Це стосується в основному бездротових камер з автономними джерелами живлення.

Але, виробляючи установку прихованої камери відеоспостереження, не варто забувати про правовий аспект прихованого відеоконтролю. Виробляти приховане відеоспостереження або прослуховування громадян, без них на те згоду, протизаконно. Працівники офісів або відвідувачі компаній повинні бути сповіщені про це заздалегідь. Згідно зі статтею 307 ЦК України – будь-яка зйомка фізичних осіб на вигляді – фото – теле – кіно – плівку (в т.ч. і прихована) без відповідного на те згоди цих осіб, може проводитися виключно у випадках, передбачених існуючим законодавством, так як в рамках оперативних заходів відповідними органами. Матеріали, отримані незаконним шляхом, не можуть бути доказом у суді. Мало того, якщо буде доведено, що відеоматеріали отримані за допомогою спец засобів стеження, заборонених для використання громадянами, що не мають відношення до державних оперативно-розшуковим службам, що то осуджують можуть легко перетворитися на обвинувачених.

Так, що, використовуючи наведену вище інформацію, Ви можете самі підбирати тип прихованої відеокамери, який найкращим чином може задовольнити потреби при виборі і установці системи відеоспостереження. І на етапі проектування і вибору краще проконсультуватися з фахівцями в області систем охорони і безпеки.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки стійкого відеоспостереження, 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

Для роботи з системами відеоспостереження необхідно програмне забезпечення (ПЗ), що дозволяє найбільш комфортно і доступно використовувати весь функціонал відеокамер для користувача. На жаль, стандартний софт, яким оснащені пристрої та компоненти відеосистеми, не завжди відповідає даним вимогам. Вашій увазі в рамках цієї статті ми спробуємо надати огляд деяких програм для відеоспостереження, доступних для скачування в мережі Інтернет безкоштовно. Отже, почнемо.

Програми для відеоспостереження

AbelCam – безкоштовна програма для відеоспостереження, що дозволяє проводити запис відео з веб-камери, а так само редагувати відеозаписи. Вона може працювати з будь-якими типами камер, звичайними веб-камерами, у тому числі бездротовими і відразу з декількома джерелами. AbelCam настроюється так, що запис з камери спостереження тільки у випадку, якщо в кадрі з'являється рухомий об'єкт. Так само за допомогою настроювань можна позначати «мертві зони». ПЗ може передавати відео та звук в режимі реалтайм, підтримуються Windows Media і Motion JPEG потоки. Дозволяється визначати різні параметри для редагування. Крім цього в AbelCam є функції веб-сервера, які підтримують PHP, CGI, SSI, Ajax і proxy.

AV100 – безкоштовна утиліта від Arecont Vision для систем з високим дозволом. Дозволяє здійснювати контроль до 16 відеокамер у високому дозволі, проводити запис з роздільною здатністю 1920 x 1200 (HDTV-якість) зі швидкістю 240 кадрів / сек.

КОДОС-відеомережі – програма відеоспостереження, яку можна скачати безкоштовно. В ній можливе підключення до 4 відеокамер. Можливості, які притаманні сучасним системам відеоконтролю, реалізовані в ній. Детектор звуку, детектор руху, керування доступом до перегляду відеоданих, регулювання часу включення режиму запису з камер і т.п. Дозволяє здійснювати настройку положення і величини вікон на моніторі. Крім цього у цієї програми досить зручний користувальницький інтерфейс. Вільно розповсюджується в мережі.

VN-RS800U – безкоштовне ПЗ для відеоспостереження, призначене для управління моніторингом та мережевою системою. Відеосистема, яка обслуговується цією програмою, включає в себе до 32-х IP-відеокамер. VN-RS800U досить знайшла широке поширення в малому та середньому бізнесі. Швидкість при одночасного запису і відображенні до 60 к / с, але в тому випадку, якщо активна тільки функція запису, то швидкість збільшується до 90 к / с. Читання файлів можливо в VGA-форматі зі швидкістю не менше 16 к / с.

Таким чином, ми розглянули чотири найпопулярніші безкоштовні програми для відеоспостереження, доступні в мережі. Хочемо зазначити, що функціонал обраної вами програми буде сильно залежати від її версії, тому перед остаточним вибором варто їх самостійно випробувати. Всі вони мають приємний інтерфейс і не дуже вимогливі до заліза і є повнофункціональними програмами, що дозволяють зручно працювати з відеоспостереженням. Використовують операційні системи сімейства Windows від XP до 10/11, так улюбленими в Україні.

Віддалене відеоспостереження: огляд можливостей

Віддалене відеоспостереження стало невід'ємною частиною процесу забезпечення охорони і безпеки будь-якого об'єкта. Так як застосування цієї технології не тільки забезпечує безпеку, але дозволяє економити кошти на фізичній охороні. В даний час ринок послуг відеоспостереження визначається стрімким технічним прогресом, який ознаменував появу пристроїв, що передають зображення по IP-мереж, систем для виявлення рухомих об'єктів,

відеореєстраторів, відеосерверів, цифрових відеокамер і ще багатьох інших технічних інновацій. У даній статті ми проведемо огляд можливостей і постараємося розповісти про переваги технології в цілому.

Після появи відеообладнання для роботи в IP-протоколі, з'єднувати віддалені пристрої з центральним комп'ютером провідними лініями вже немає необхідності. Зараз системи спостереження можна розглядати з тієї ж структурою, можливостями та особливостями, як і будь-яку іншу мережу передачі даних. На сьогоднішній день IP-мережі передають відеодані цілком самостійно, завдяки відеостиску та інших можливостей програмного забезпечення, які скорочують мережевий трафік. Технічні рішення і додаткові можливості дають можливість повноцінно, інтегруватися з охоронно-пожежною сигналізацією та системами доступу.

Простота модернізації

Одна з ключових переваг систем відеоконтролю на базі IP-технології. Здійснення відеозапису будь-якої ділянки мережі може бути автономним. Ця частина системи буде продовжувати фіксувати відеозображення, навіть якщо інша буде виведена з ладу. Центральне управління відновлюється, як тільки мережа знову запрацює і виникне доступ до всіх зафіксованим подіям. Ця конфігурація дозволяє розподіляти інформацію по всій мережі, зберігати і керувати нею.

Весь запис відеоспостереження може проводитися на місцях – це один з методів архівації. Локальна система створює відеофрагмент події, даючи знати про те, що трапилося співробітникам служби охорони центрального поста і місцевий персонал. Апаратура віддалених ділянок може бути запрограмована на здійснення дій, властивих даній дільниці контролю або відеокамери. Для того допомоги співробітникам безпеки, щоб розібратися в ситуації, є функція програмування пріоритету подій, як дистанційна, так і місцева.

Система віддаленого відеоспостереження

Вивільняє мережу від переповнення переданої відеоінформацією, демонструючи на моніторах центрального поста найбільш небезпечні пригоди. На віддалене записуючий пристрій буде надходити великий обсяг відеоінформації, так як будь-яка ділянка буде фіксувати дані з усіх камер відеоконтролю на місці. Цей обсяг інформації мережа віддаленого ділянки може не осилити, тому простіше буде комутувати аналогові камери спостереження проводами з віддаленим ділянкою і користуватися ними, а відео на центральний комп'ютер передавати за допомогою IP-технології.

Швидкість передачі даних по мережі відіграє значну роль в сукупній продуктивності у цифрових пристроїв, тим більше у великих відеосистемах з великим числом камер. Для цих цілей найбільш підходять системи зі швидкістю до 10 Гбіт, але це обладнання коштує досить дорого. Тому, правильне поєднання аналогових відеокамер і цифрових технологій, здатне задовольнити оперативні завдання системи відеоконтролю, не перевищуючи бюджетних рамок.

Огляд системи віддаленого відеоспостереження DROPCAM HD

Ми не раз піднімали тему віддаленого відеоспостереження і на те є причини. Досі доступність такого роду технологій простим користувачам здається чимось з ряду геть вихідним, так як широке поширення й активний розвиток в цій сфері розпочалося лише в останні пару-трійку років і який розвиток! Згадайте Ivideon, коли для побудови готової системи можна використовувати вже наявні веб-камери. Більше того, все популярнішими стають готові апаратні рішення, начебто Stem IZON. Але, хто цікавився питанням такого роду пристроїв, пам'ятає, що до професійних рішень побутовим далеко. Але час летить непомітно, а технології розвиваються швидко. І ось, вже цілком доступні рішення, які транслюють відео в дозволі HD з частотою 30 кадрів в секунду, причому зі звуком, здатні відтворювати голос господаря і цілком ефективно діючі в темряві.

Компанія Dropcam – це один з ветеранів на споживчому ринку систем віддаленого відеоспостереження і не перший рік вона радує людей такого роду пристроями. А починалося все з Dropcam Echo, створеної силами Axis і здатна транслювати картинку зі смішним за сучасними мірками QVGA-дозволом. Та й за розмірами «цеглини», був значним. Тим не менш, на безриб'ї і рак риба. Ось тільки зараз часи інші, конкуренція загострилася, плюс розвелося чимало китайських виробників, готових віддавати такого роду рішення за порівняно скромні суми. У Dropcam не розгубилися і видали на-гора одну з кращих за свої гроші систем, причому тепер її створювали вже всередині компанії.

Dropcam HD компактна, стильна, а при необхідності її нескладно замаскувати. Компактний модуль з діаметром близько 80 мм розташовується на функціональній алюмінієвої підставці, з якої при необхідності він легко витягується. Сама ж підставка може виконувати як роль настільного упору, так і настінного кріплення.

Сподобався монолітний корпус камери, в якому є лише один отвір – роз'єм micro-USB. Через нього пристрій живиться і підключається до ПЗ для базової установки, про яку мова піде трохи нижче, а поки давайте глянемо на технічні характеристики шпигунського гаджета.

У аксесуар використовується якісна матриця і ширококутний світлочутливий об'єктив. Така зв'язка дозволила вести запис в умовах досить слабкої освітленості і навіть в напівтемряві, причому підсумкова картинка виходить в дозволі 1280x720 пікселів і видається з частотою 30 кадрів в секунду, що дуже круто. У стандартних побутових рішень зазвичай дозвіл запису VGA (640x480), так і швидкість і якщо досягає 15 кадрів в секунду, то вже добре. Варто відзначити, що кодування відео відбувається з використанням кодекаH.264, тобто непогано скорочується, але з-за HD-дозволу потік все одно виходить немаленький і для його підтримки потрібна достатньо широкий канал. Рекомендується від 10 Мбіт хоча в ідеалі – 50 Мбіт.

Але з швидким Інтернетом зараз начебто проблем немає. До того ж Dropcam HD залишається функціональної навіть практично в повній темряві. Цю проблему вирішує вбудований набір з потужних інфрачервоних світлодіодів і наявність відповідного режиму роботи. Кут огляду становить 107°.

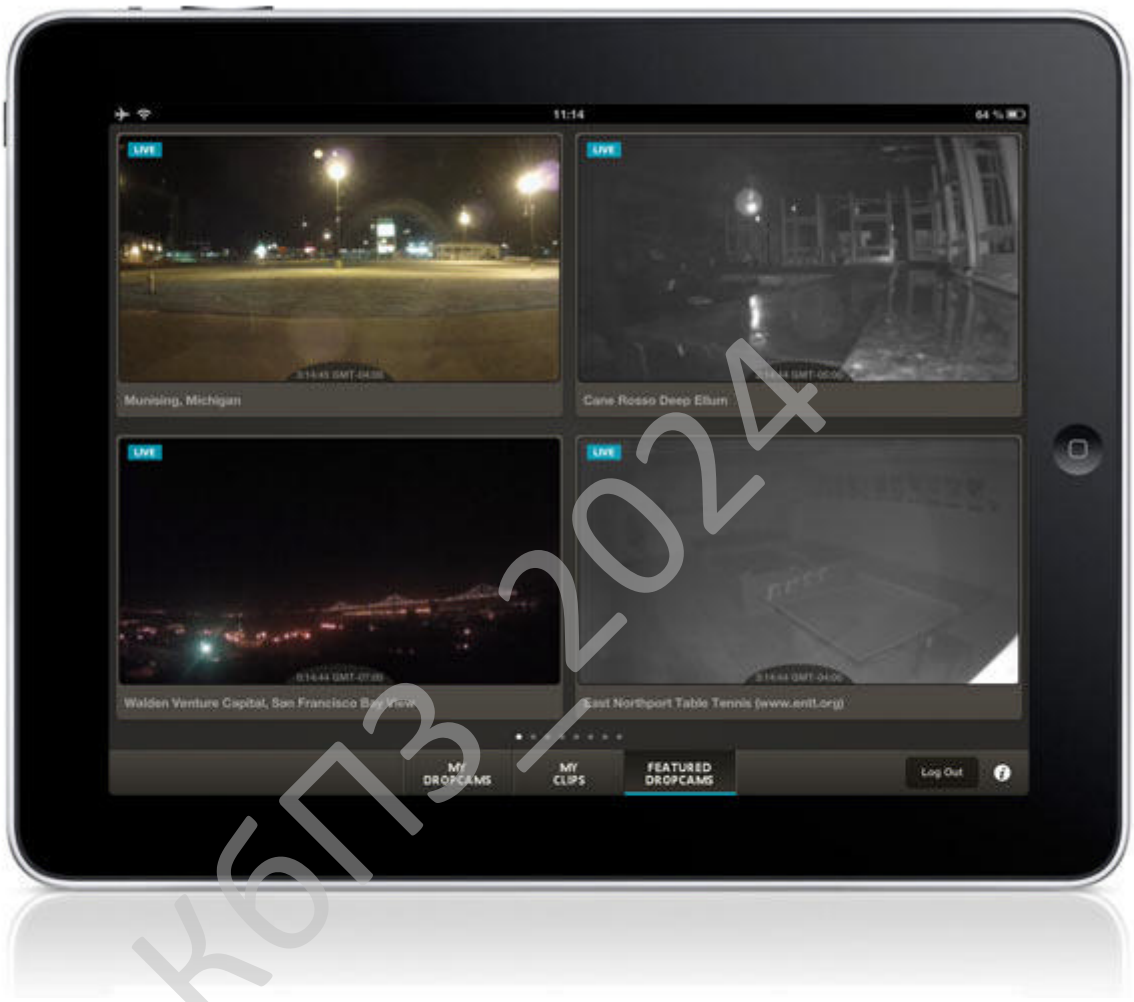

Рисунок 2.1 – Інтерфейс користувача Dropcam HD

Ще одна важлива фішка камери – це наявність власного якісного і чутливого мікрофона. Частенько на цьому інші виробники економлять. Більш того, навіть є динамік і при бажанні можна віддалено гаркнути на порушника або нахабного домашнього коте, творить шкоду. Вбудований динамік і функція передачі власного голосу в принципі дають масу цікавих можливостей.

Комплект поставки досить мізерний, але все необхідне для швидкої установки системи в ньому є, включаючи 3-метровий кабель USB.

Бездротова Камера, з домашньою мережею зв'язується через Wi-Fi, правда, власного акумулятора у неї немає. Так що за фактом один провід з пристрою буде стирчати постійно, а поблизу потрібна наявність електророзетки. Але перш ніж аксесуар встановлювати на постійне місце проживання, його доведеться попередньо підключити до комп'ютера і налаштувати, чим і займемося нижче.

Виробник запевняє, що на налаштування Dropcam HD піде не більше 60 секунд. Якщо дуже поспішати і на 100% знати алгоритм дій, то можна вкластися і в цей час. Але навіщо зайвий раз напружуватися? Витратимо не хвилину, а три, зате зробимо все спокійно. Підключивши пристрій до USB-порту отримаємо доступ до фірмового ПЗ для Windows і OS X. Пара кліків, після чого пропонується завести обліковий запис в онлайновому сервісі Dropcam – без цього ніяк, адже всі дані шифруються і зберігаються в хмарі. Реєстрація швидка і легка. Потім досить вибрати точку доступу Wi-Fi і все, можна користуватися.

Компанія пропонує кілька варіантів доступу до трансляції: через будь-який веб-браузер на ПК, з допомогою iOS-пристрої [iTunes link] або Android-гаджета [Google play]. Крім того, для мобільної ОС Applepідтримуються Push-повідомлення, плюс є оповіщення по E-mail. Такі приходять, якщо камера визначає рух або звук.

Взагалі, запис відбувається постійно, більше того, вся вона може зберігатися на серверах Dropcam, але тільки за гроші. Підтримувати 7-денний архів даних обійдеться в $10 в місяць, 30-денний – втричі дорожче. Для другої та всіх наступних камер діє 50-відсоткова знижка на сервіс. А ось базова підтримка обійдеться безкоштовно, але вона включає лише живу трансляцію. При бажанні відеопотоком можна поділитися з друзями через E-mail (але їм теж доведеться зареєструватися в сервісі) або ж опублікувати його у відкритому доступі.

З підключенням камери в захищених мережах проблем бути не повинно, так як підтримуються протоколи WEP (40-bit, 128-bit), WPA (TKIP, AES) та WPA2 (TKIP, AES). Сам бездротовий модуль працює в режимі 802.11b/g/n. Потік ж шифрується з використанням 256-бітного алгоритму AES і SSL-з'єднання.

Якість відео вельми на рівні, але не чекайте кристально чіткої картинки. Все-таки воно буде гірше, ніж у крутих HD-ріпів вытягиваемых з бездонних torrent-мереж фільмів. Але за рахунок високої частоти підсумковий результат виходить краще, ніж у більшості конкуруючих рішень, а в умовах гарної освітленості зображення просто шикарне. Оптичного зума або можливості рухати камеру віддалено немає. Доступний тільки цифровий зум – активна область ділиться на п'ять частин і кожну з них можна збільшити. Але картинка вийде каламутною – краще використовувати систему без цифрового збільшення, яке є лише іграшкою, не більше.

Отже, серед плюсів Dropcam HD виділимо просту установку і настройку, двосторонню аудіозв'язок, режим нічного бачення, безкоштовний стрімінг в режимі 24/7, можливість поділитися потоком з друзями і доступ з мобільних пристроїв через спеціалізовані клієнти.

Недоліки є теж. Так, функція запису і зберігання відео варто відчутних грошей, ні можливості локального зберігання знятих копій, немає вбудованого акумулятора (камера прив'язана до електромережі) і немає пило-вологозахищеністю (можна використовувати тільки всередині приміщень).

Серед конкурентів варто згадати систему відеоспостереження Logitech Alert 750e, захищену від будь-якої негоди, здатну харчуватися навіть від Ethernet-мережі і зберігати відео локально. Ось тільки коштує вона аж $350 і це в США. В Росії як мінімум буде те ж саме, але в євро, тобто сильно дорожче. Ще можна згадати про дійсно бездротову системі Avaak VueZone з вбудованим акумулятором, але вона записує відео в SD-як, а коштує дорожче ($200 в США).

Огляд системи віддаленого відеоспостереження STEM IZON

Професійні системи віддаленого відеоспостереження коштують дорого, причому йдеться не тільки устаткування, але і в оплаті праці фахівців, які все це будуть налаштовувати і підтримувати. Загалом, не проблема. Раз вже знадобилася організації така штука, то і гроші вона знайде. Проблема лише в тому, що далеко не завжди потрібен весь пакет послуг і висока надійність за всім стандартам індустрії. Та й можливості професійного обладнання для простого обивателя зайві. Насправді варіантів використання такої в побуті багато, починаючи з віддаленого спостереження за дитиною або дитиною постарше, так і шукають, де б нашкодив, і закінчуючи банальною стеженням за власним житлом. А раз є попит, то буде і пропозиція. Наприклад, в даному випадку це програмно-апаратний комплекс Stem IZON, про який і піде мова нижче.

Stem IZON являє собою бездротову камеру з вбудованим модулем Wi-Fi і набором датчиків, що дозволяють налаштувати умови реагування пристрої. У плані апаратної начинки – це продукт не найскладніший, але ж всі ми знаємо, що «залізо» без софта є лише марною оболонкою. Stem ж пропонує готове і, що найважливіше, просте у використанні рішення. Але перш ніж перейти до базової налаштування пристрою, давайте розберемося, що воно вміє і що пропонує за свої гроші.

Камера поставляється в прозорої акрилової коробці, в якій також є магнітна база для камери, пара дюбелів для кріплення системи на стіну, 2,7-метровий кабель miniUSB-USB для підзарядки і блок живлення. При необхідності пристрій можна живити не тільки від електромережі, але і від ПК.

Дуже сподобалося рішення з магнітною базою, що дозволяє змінювати кут камери без всяких шарнірів і кріплень. Просто вибираємо потрібне положення і примагнічуємо IZON одним рухом.

Технічно камера не хапає зірок з неба, але її можливостей достатньо, щоб впоратися з поставленим завданням. Вона здатна записувати відео в дозволі 640x480 пікселів з частотою 30 кадрів в секунду і бітрейтом 1,5 Мбіт/с. Кут

огляду становить 60°. При прямій трансляції в реальному часі якість відеопотоку знижується (320x240 пікселів, 10-кадрів в секунду, 300 Кбіт/с), але зате картинка без проблем передається навіть через мобільний EDGE-з'єднання, хоча 3G, звичайно, краще. Припускаю, що читач вже здогадався про можливість підключатися до пристрою з будь-якої точки планети, де є доступ до Інтернету.

Крім записує відео і транслюється звук в діапазоні від 40 Гц до 8 кГц. IZON реагує на рівень звуку від 35 до 95 дБ. На жаль, в камері немає вбудованого інфрачервоного ліхтаря і можливості працювати в повній темряві. Виробник заявляє, мовляв, застосування таких засобів негативно впливає на якість запису відео в звичайних умовах освітлення. Не впевнений, що це так, скоріше, справа в економії. Зате пристрій досить непогано працює навіть при слабкому освітленні. Достатньо світла від 7,5-ватної лампочки.

Природно, для роботи IZON потрібне підключення до Мережі і ось тут вже без софта не обійтися. Пристрій контролюється тільки через iOS-додаток і через нього забезпечується доступ до трансляції в реальному часі або ж до архіву записів. Однойменне безкоштовний додаток доступний в App Store [iTunes link].

Перш ніж приступити безпосередньо до моніторингу, доведеться трошки напружити звивини і підключити камеру до iPhone або iPad, до домашньої мережі та настроїти умови роботи. Процес нескладний, але недосвідченого користувача може дещо спантеличити, тому трішки його тут розпишу. Для початку додаємо камеру (можна використовувати одночасно кілька пристроїв):

Щоб система зрозуміла, що до гаджету під'єднується господар, а не сусід Вася, використовується проста система світлового коду. Світлодіод на камері промигає в певній послідовності оранжевим і зеленим кольором, після чого програма запропонує вибрати правильний варіант.

Потім підключіть камеру до мережі Wi-Fi, і після того, як на екрані з'явиться QR-код, його треба буде показати пристрою.

Все, ось тепер камера синхронізована з iPhone.

Важливою особливістю системи Stem IZON є гнучка система налаштування. Адже користі від такого роду гаджета буде мало, якщо пропустиш найцікавіше. Таким чином, пристрій можна настроїти, щоб воно активувався при певній зміні зовнішніх умов і посилало при цьому повідомлення на iPhone. Передбачена реакція на звук та на рух. В останньому випадку потрібно задати активну зону. Це може бути, наприклад, вхідні двері, ліжечко з немовлям, певна частина кімнати, де любить пустувати домашній вихованець і т. п.

Рисунок 2.2 – Інтерфейс користувача Stem IZON

Тепер, де б ви не знаходилися, то будете отримувати повідомлення при реакції камери відповідно до заданих умов, плюс можна просто підключитися до неї і поспостерігати за тим, що ж діється на підконтрольній території.

До речі, не обов'язково реагувати на повідомлення миттєво. Система здатна самостійно записати відео при активації камери і зберегти його на сервері Stem. Але для використання цієї функції необхідно зареєструватися. Процедура традиційна (email, пароль) і не займе більше хвилини.

На жаль, є обмеження. Так, довжина ролика не перевищує 30 секунд і таких роликів в день можна записувати до 25 штук. До того ж, перенести їх на ПК або зовнішній накопичувач не можна. Stem дозволяє лише переглядати контент через фірмовий додаток.

Що стосується якості відео, то воно залишає бажати кращого. Хоча, я вже писав про його параметри вище і про причини – економія трафіку. Тим не менше, видається картинки цілком достатньо для розуміння того, що відбувається на підконтрольній території. А якщо ведеться автоматичний запис, то тоді і картинка буде краще.

В огляді йдеться про систему відеоспостереження Stem другого покоління, але апаратно від оригіналу вона нічим не відрізняється. Хіба що назвою (у першому поколінні в назві літеру «і», у другому – «I»). Таким чином, якщо ви щасливий володар моделі першого покоління, то можете отримати всі описані вище плюшки безкоштовно. Достатньо оновити прошивку пристрою і встановити фірмове додаток в iPhone.

За це виробникові великий плюс. До речі, нове стало значно функціональніша і зручніше попереднього. З іншого боку, компанії ще є над чим працювати, включаючи поліпшення продуктивності як локального програми, так і онлайнового сервісу. Адже обмеження на пряму трансляцію не апаратне, а програмне. Хотілося б мати можливість записувати ролики довше 30 секунд і більше, ніж 25 штук в день.

З одного боку, Stem IZON справляється з поставленими завданнями і, що важливо, налаштувати комплект дуже легко. З іншого боку, занадто вже багато обмежень з серверної сторони. Можливо, виробник з часом їх зніме, але поки маємо те, що маємо.

До плюсів варто віднести відмінний дизайн і зручність у встановленні.

Що стосується альтернатив, то такі існують, наприклад, Ivideon. Правда, це лише програмне рішення, а апаратну частину доведеться знайти самому. Хоча підійде будь-яка веб-камера або IP-камера. Але в першому випадку потрібен ще й комп'ютер, у другому – доведеться розщедритися, що за витратами ставить такий варіант на одну сходинку з рішенням Stem.

Переваги:

– гарний дизайн;

– зручність в установці;

– гнучка настройка параметрів реагування;

– підтримка функції повідомлень;

– робота через стільникову мережу.

Недоліки:

– обмеження на довжину відео та щоденне кількість роликів;

– погана якість відео при прямій трансляції.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в

чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки стійкого відеоспостереження.

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 кібербезпеки 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 кібербезпеки 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

Опишемо математичні моделі фіксованої й поворотної камер і викладемо завдання калібрування.

Калібрування відеокамер – одне із центральних завдань в області машинного зору (Computer Vision). Завдання полягає у визначенні параметрів математичної моделі, що описує реально використовуваний пристрій відеореєстрації. Як правило, завдання калібрування розділяються на два класи:

– зовнішнє калібрування;

– внутрішнє калібрування.

Ціль зовнішнього калібрування полягає у визначенні положення й орієнтації відеокамери в просторі. Рішення даного завдання потрібно в різних областях:

– відеоспостереженні;

– тривимірній реконструкції;

– картографії;

– системах розпізнавання об'єктів;

– системах взаємодії з комп'ютером за допомогою визначення положення рук або напрямку погляду;

– системах керування роботами.

Внутрішнє калібрування орієнтоване на визначення таких характеристик камери як:

– фокусна відстань;

– розмір пікселя;

– величина дісторсії.

Дані параметри описують перекручування, викликані оптикою відеокамери, і їх необхідно враховувати при рішенні більшості завдань комп'ютерного зору.

Для математичного подання фіксованої камери використовується модель, описувана формулою (3.1), або скорочено (3.2). Ця модель досить точно відповідає процесу побудови зображення в більшості сучасних фото- і відеокамер.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} R^T & -R^T C \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad (3.1)$$

де (X, Y, Z) – координати точки в тривимірному просторі, а (x, y) – проекція цієї точки на картинну площину.

Внутрішні параметри моделі камери: f – фокусна відстань, (c_x,c_y) – положення принципової точки (точки перетинання оптичної осі з картинною площиною).

Зовнішні параметри моделі камери: $R \in R^{3x3}$ – матриця повороту, що задає напрямок об'єктива камери, $C \in R^3$ – положення камери.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \cdot P \cdot [R|C] \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (3.2)$$

Тут К – матриця внутрішнього калібрування, [R\C] – матриця зовнішнього калібрування, Р – матриця проектування.

Однак, через обмеження сучасної оптики реальний процес трохи відрізняється від представленої моделі. Одне з найпоширеніших перекручувань – дісторсія. Дісторсія (від лат. distorsio, distortio – скривлення) – аберація оптичних систем, при якій лінійне збільшення змінюється по полю зору. При цьому

порушується подоба між об'єктом і його зображенням. Найбільш частий випадок дісторсії – радіальна дісторсія. Ця модель використовується в даній роботі.

Найчастіше параметри дісторсії визначаються разом з параметрами внутрішнього калібрування камери.

Поворотні відеокамери (PTZ cameras) з можливістю віддаленого керування, завдяки здешевленню, знаходять усе більше поширення. При роботі з такими камерами (рисунок 3.1) за допомогою команд із пульта керування, можна змінювати напрямок об'єктива за азимутом (ф) на 360°, кутом місця (ф) на 90° і змінювати фокусну відстань (f). Такі камери дозволяють ефективніше вирішувати завдання відеоспостереження. Дослідження, проведені в бакалаврській роботі, головним чином орієнтовані на поворотні камери.

Рисунок 3.1 – Поворотна відеокамера

Модель поворотної камери є розширенням моделі фіксованої камери (3.2) і описується формулою (3.3).

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K(f) \cdot P \cdot \Phi(\phi) \cdot \Psi(\psi) \cdot [R|C] \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (3.3)$$

де K(f) – матриця внутрішнього калібрування зі змінною фокусною відстанню, Φ(φ) – поворот по азимуті на відомий кут φ , Ψ(φ) – поворот по куті місця на

відомий кут ф, [R\C] – матриця зовнішнього калібрування, P – матриця проектування.

Приведемо короткий огляд методів визначення внутрішніх параметрів моделі камери. Часто ці методи називають методами внутрішнього калібрування.

Існують чотири основних підходи до визначення параметрів внутрішнього калібрування:

- фотограмметричний підхід;
- калібрування по точках сходу;
- самокалібрування за зсувом камери;
- самокалібрування за поворотом камери.

Всі вони також дозволяють визначити параметри дісторсії.

Проведений аналіз різних способів калібрування внутрішніх параметрів камери й дісторсії показали, що немає необхідності розробляти новий спосіб калібрування, і цілком можна скористатися методом запропонованим у роботі Захана.

При подальшому викладі будемо вважати, що параметри внутрішнього калібрування камери відомі, а дісторсія відсутня. Якщо це не так, то дісторсія може бути компенсована по відомих параметрах.

Приведемо огляд методів визначення зовнішніх параметрів моделі камери (положення й орієнтації в просторі). Часто ці методи називають методами зовнішнього калібрування. Необхідність визначення зовнішніх параметрів моделі камери виникає при рішенні різних завдань. Це й фотограмметрія, і тривимірна реконструкція по зображеннях, і відеоспостереження. Якщо положення камери в просторі може бути безпосередньо обмірювано, то для визначення орієнтації необхідно зіставити зображення, отримане від камери, і координати, зображених на ньому об'єктів, у тривимірному просторі. Однак, вимір положення камери не завжди є простим рішенням. Найчастіше камери розташовані у важкодоступних місцях (наприклад, під дахом ангара), або положення камери часто змінюється (наприклад, у дослідницьких лабораторіях), тоді процедура виміру забирає багато часу. Тому виникла необхідність у розробці алгоритмів зовнішнього

калібрування камер, що дозволяють швидко визначати положення й орієнтацію камер навіть при утрудненому доступі до неї.

Методи зовнішнього калібрування можна розділити на дві групи:

– методи калібрування однієї камери;

– методи спільного калібрування групи камер.

Для визначення положення й орієнтації камери використовується набір відомих відповідностей $X_i – x_i$ (i = 1..N) між точками в тривимірному просторі X_i і їхніми зображеннями на кадрі X_i. У загальному випадку даний підхід дозволяє визначити узагальнену матрицю калібрування:

$$H = K \cdot P \cdot [R \backslash C].$$

Зі співвідношення $X_i \times HX_i = 0$ одержуємо:

$$\begin{pmatrix} 0^T & -X_i^T & x_i^y X_i^T \\ X_i^T & 0^T & -x_i^x X_i^T \\ -x_i^y X_i^T & -x_i^x X_i^T & 0^T \end{pmatrix} \begin{pmatrix} H^1 \\ H^2 \\ H^3 \end{pmatrix} = 0 \quad (3.4)$$

З (3.4), у силу лінійної залежності рядів, одержуємо систему з 2N лінійних рівнянь із 12-ю невідомими. Узагальнена матриця калібрування може бути визначена по 6 відомих відповідностях. Для підвищення точності визначення параметрів калібрування використовується більша кількість відповідностей і застосовується метод найменших квадратів. Матриця H може бути факторизована в добуток матриць K, H і [R\C].

Відповідності між точками тривимірного простору і їхніх зображень на кадрі встановлюються вручну, або за допомогою каліброваних маркерів. Описаний підхід до зовнішнього калібрування відеокамери дозволяє визначати положення й орієнтацію камери з високою точністю, але вимагає багато часу на проведення якісних вимірів.

Для проведення досліджень в області тривимірної реконструкції по зображеннях у різних університетах стали створюватися спеціальні лабораторії з більшою кількістю камер. Процедура калібрування в таких лабораторіях вимагає значного часу. Крім того, калібрування доводиться повторювати щораз при зміні

положення камер. Тому виникла необхідність у розробці нових алгоритмів зовнішнього калібрування камер, що дозволяють швидко й з мінімальною участю людини визначати положення й орієнтації камер. Для рішення цього завдання використовуються різні алгоритми спільного калібрування групи камер.

Проведене дослідження показало, що більшість методів калібрування групи камер розроблені для нерухливих камер. Вони можуть бути використані для роботи з поворотними камерами, але для цього камери повинні бути попередньо настроєні таким чином, щоб перекривалися їхні області видимості. Дана вимога, як правило, неможливо задовольнити при калібруванні в великих приміщеннях зі складною топологією. Алгоритми для зовнішнього калібрування поворотних камер вимагають використання камер певного виду, або орієнтовані на калібрування в невеликих приміщеннях. Тому, для рішення поставлених у роботі завдань, треба було розробити алгоритм, що дозволяє визначати положення й орієнтацію поворотних камер у великих приміщеннях з незначною участю людини.

Опишемо метод спільного калібрування групи камер у великих приміщеннях.

Для калібрування поворотних камер у великих приміщеннях пропонується використовувати стаціонарні графічні маркери. Маркери повинні бути невеликого розміру, для того щоб їх було легко розміщати, оскільки основна мета розробки полягає в спрощенні роботи людини. Також встає проблема ототожнення зображень одного маркера, отриманих від різних камер, оскільки це найважливіший момент будь-якого алгоритму калібрування. Отже, кожний маркер повинен бути унікальний.

Для забезпечення зазначених умов щонайкраще підходять маркери із двомірним баркодом. Баркод – це зразок, що кодує цифрову інформацію графічним способом. Існує два різновиди баркодів: лінійні й двомірні, надалі під баркодом будемо мати на увазі двомірний баркод.

Більшість способів побудови баркодов добре підходять для передачі інформації, але не стійкі до перспективних перекручувань: Maxicode, Data Matrix, Quick Responce. Система кодування ARToolKit позбавлена цього недоліку, але надійність розпізнавання маркера даною системою істотно залежить від умов висвітлення через виділення маркера по порозі. У системі ARTag стійкість виділення значно підвищена за рахунок виділення маркерів по границях.

Для виділення й розпізнавання маркерів у розробленому алгоритмі використовується метод ARTag, оскільки він найкраще підходить для завдання калібрування. При калібруванні не потрібно великої кількості маркерів, тому вихідний алгоритм ARTag використовується в модифікованому виді:

− маркер складається з 4 × 4 кліток, що дозволяє підвищити надійність розпізнавання на великій відстані від камери при тім же розмірі маркера;

− номер маркера кодується з надмірністю в 8 біт. Алгоритм калібрування групи камер складається із чотирьох етапів.

1. Розміщення каліброваних маркерів. Маркери роздруковуються на принтері на аркушах формату А4. Людина закріплює їх на стінах і підлозі приміщення.

2. Виявлення й ідентифікація маркерів. Для виявлення маркерів поворотні камери сканують оточення по азимуту й куту місця таким чином, щоб сусідні області видимості перекривалися. У кожному положенні виконується пошук і розпізнавання маркерів.

У результаті формується набір каліброваних даних:

– кути зсуву об'єктива камери по азимуту й куту місця;

– однорідні координати вершин маркера на зображенні (i – номер камери; j – номер маркера).

Камери можуть виявити не всі маркери. L_i – кількість маркерів виявлених ї-ю камерою.

3. Початкова оцінка.

На цьому етапі виробляється початкова оцінка взаємного розташування й орієнтації камер і маркерів. У процесі виконання оцінки вирішуються два завдання:

– визначення положення й орієнтації камери при відомому положенні спостережуваного маркера;

– визначення положення спостережуваного маркера при відомих положенні й орієнтації камери.

Визначення положення й орієнтації камери

Відомі: x_1,x_2,x_3,x_4 – однорідні координати вершин маркера на зображенні; X_1,X_2, X_3,X_4 – однорідні координати вершин маркера в просторі.

З (3.3) одержимо:

$$K(f)\cdot P\cdot\Phi(\phi)\cdot\Psi(\psi)\cdot[R|C]\cdot\begin{pmatrix} X_1 & X_2 & X_3 & X_4 \end{pmatrix}=\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}. \quad (3.5)$$

Матричне рівняння (3.5) зводиться до системи із дванадцяти лінійних рівнянь із дванадцятьома невідомими. Вирішуючи систему, одержуємо значення елементів матриці [R\C].

Визначення положення маркера

Відомі: x_1,x_2,x_3,x_4 – однорідні координати вершин маркера на зображенні; C, R – положення й орієнтація камери; W – ширина маркера в одиницях виміру світової системи координат. Нехай одиничні вектори $v_1(R), v_2(R), v_3(R), v_4(R)$ – задають прямі, що проходять через центр камери й точки відповідних вершин маркера на зображенні; одиничний вектор $v(R)$ – задає пряму, що проходить через центр маркера на зображенні. Тоді координати вершин маркера в просторі визначаються за формулою:

$$X_i = C + \frac{W}{\sqrt{2}\sin(\arccos(v(R)\times v_i(R)))}, \qquad i = 1..4 \quad (3.6)$$

Початкова оцінка взаємного розташування й орієнтації камер і маркерів виконується по наступному алгоритмі:

– вибирається маркер, видимий найбільшою кількістю камер;

– задаються координати вершин маркера, з урахуванням його форми й розміру;

– обчислюються положення й орієнтації камер, що бачать обраний маркер;

– вибирається маркер з невідомим положенням, видимий найбільшою кількістю невідомих камер і хоча б однієї відомої;

– обчислюється положення обраного маркера;

– якщо є камера з невідомим положенням і орієнтацією, – до пункту 3;

– якщо є маркери з невідомим положенням, – до пункту 4.

4. Калібрування камер.

Завдання калібрування полягає в тому, щоб по зібраним каліброваним даним визначити положення й орієнтацію камер у єдиній системі координат. Рішення одержуємо, мінімізуючи цінову функцію $F(R_i, C_i, M_j)$, побудовану на основі математичної моделі поворотної камери (3.3), де Ri – матриця повороту i-ї камери; C_i – положення i-ї камери в просторі; $M_j = (X_j, Y_j, Z_j)$ – положення j-го маркера в просторі; (x_{ij}, y_{ij}) – координати центра маркера на кадрі.

$$F(R_i, C_i, M_j) = \\ = \sum_{i,j} \left\| \begin{pmatrix} x_{ij} \\ y_{ij} \\ 1 \end{pmatrix} - K(f) \cdot P \cdot \Phi(\phi) \cdot \Psi(\psi) \cdot [R_i | C_i] \cdot \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} \right\|_E . \quad (3.7)$$

Для мінімізації цінової функції використовується ітераційний алгоритм нелінійної оптимізації на основі довірчих областей. Як початкове наближення параметрів використовуються значення, отримані на етапі початкової оцінки. У результаті одержуємо положення (C_i) і орієнтацію (R_i) всіх камер і положення всіх маркерів (M_j) у єдиній системі координат.

Для коректної роботи запропонованого алгоритму калібровані дані повинні задовольняти двом умовам:

– кожний калібрований маркер повинні виявити не менш ніж дві камери;

− якщо розділити всі камери довільним образом на дві непусті групи, то кількість маркерів, що виявляються камерами з обох груп, повинне бути не менш трьох.

У противному випадку визначення параметрів неможливо, що треба із простих геометричних міркувань.

Точність запропонованого методу спільного калібрування групи камер оцінювалася на синтетичних даних.

Дослідження показало, що при використанні для калібрування більше 15 маркерів помилка репроекции становить 0.15 пікселів. Ці показники підтверджуються експериментальними даними.

Запропонований метод спільного калібрування камер не уступає по точності визначення положення й орієнтації камер методам спільного калібрування для лабораторій. Процес калібрування не займає багато часу й не вимагає спеціальних знань і навичок. Також описана модифікація, що дозволяє виконувати калібрування гібридних систем, що складаються з фіксованих і поворотних камер.

3.2 Розробка структурної схеми

Опишемо завдання виявлення областей інтересу по відео. До областей інтересу відносяться області кадру, які відповідають об'єктам, що рухаються, залишеним об'єктам, що пропали об'єктам.

Приведемо огляд і аналіз методів виявлення об'єктів по відеоданим.

Існує два підходи до виявлення об'єктів у відеопотоці:

1. Виділення тла. Буває тло з моделлю й без. У першому випадку будується модель тла й аналізується відмінність кадру від моделі. У другому модель тла не будується, а оцінюється відмінність від одного або декількох попередніх кадрів. Даний підхід дозволяє виявляти залишені й зниклі об'єкти, але при зміні умов висвітлення працює хитливо.

2. Аналіз руху. Для кожного пікселя визначається вектор зсуву щодо положення в попередньому кадрі. Потім аналізується отримане поле векторів.

Аналіз руху стійко працює навіть при значній зміні умов висвітлення, але має високу обчислювальну складність. Підхід не дозволяє виявляти залишені й зниклі об'єкти.

Опишемо метод виявлення областей інтересу на відео. Метод поєднує достоїнства описаних вище підходів: стійкість до зміни висвітлення й виявлення нерухливих об'єктів:

– При незмінних умовах висвітлення використовується виявлення по моделі тла, що дозволяє виявляти як рухаються так і нерухливі об'єкти.

– У випадку зміни висвітлення виявлення здійснюється за допомогою аналізу руху, доти поки модель тла не обновиться повністю.

Для поворотних камер застосовується моделювання оточення у вигляді сферичної панорами. Для мінімізації обсягу використовуваної пам'яті модель накопичує тільки середню яскравість пікселів для фокусної відстані, що відповідає куту зору камери в 15°. Для зберігання такої моделі потрібно 38 мегабайт пам'яті. Модель дозволяє виявляти об'єкти на відстані до 50 метрів.

Для підвищення надійності виділення областей інтересу необхідно враховувати рівень шуму. Відповідно до досліджень, рівень шуму залежить від яскравості пікселя. Шкала яскравості розбивається на кілька непересічних інтервалів. З панорами вибирається трохи пікселів з яскравістю, що лежить у центрі одного з інтервалів. Для обраних пікселів відслідковується флуктуація яскравості й визначається рівень шуму. Рівень шуму для всіх значень яскравості з інтервалу обчислюється шляхом бікубічної інтерполяції.

Виділення областей інтересу здійснюється по наступному алгоритмі:

1. Позначаються пікселі кадру, відмінність яких від відповідних їм пікселів моделі перевищує рівень шуму.

2. Позначені пікселі поєднуються у зв'язні області.

3. Область позначається як об'єкт, якщо її розмір перевищує 600 пікселів.

Описаний алгоритм дозволяє на відстані 50 метрів і куті зору камери в 15° виявити об'єкт, площу якого перевищує 1 м^2.

Алгоритми аналізу руху по оптичному потоці мають високу обчислювальну складність, тому не підходять для систем відеоспостереження в великих приміщеннях, де використовується велика кількість камер. Тому в запропонованому методі виявлення об'єктів, аналіз руху виконується на основі векторів руху для блоків, що дозволяє істотно підвищити швидкість роботи.

Основна ідея пошуку векторів руху для блоків полягає в наступному. Поточний кадр розбивається сіткою на блоки фіксованого розміру. Для кожного блоку з поточного кадру перебуває блок у попередньому кадрі, на якому досягається мінімум функція відповідності. Вектор руху визначається із взаємного розташування цих блоків. У результаті роботи алгоритму одержуємо поле векторів руху для блоків.

Знайдений за допомогою мінімізації вектор руху для блоку може не відповідати реальному руху цього блоку. Назвемо такий вектор помилковим. Використання помилкових векторів негативно позначається на якості аналізу руху. Найчастіше такі вектори відповідають блокам зображення без деталей. Є дві основних ознаки помилковості вектора:

– велике значення функції відповідності блоків для цього вектора;

– значна відмінність цього вектора від сусідніх векторів.

Для виявлення помилкових векторів пропонується використовувати функцію довіри, описану у роботі [6]. При побудові функції враховувалися міркування, наведені в попередньому абзаці. Функція довіри має такий вигляд:

$$\Psi(I_1, I_2, B_{i,j}, v_{i,j}) = \\ = a_1 \cdot F(I_1, I_2, B_{i,j}, v_{i,j})^{-1} + a_2 \cdot G(v_{i,j}, \{V\})^{-1} - a_3 \cdot H(I_1, B_{i,j}). \quad (3.8)$$

Тут I_1, I_2 – поточний і попередній кадри; B_{ij}, V_{ij} – блок і вектор руху для нього; a_1, a_2, a_3 – вагарні коефіцієнти;

$F(I_1, I_2, B_{ij}, V_{ij})$ – функція відповідності блоків; $G(vij, \{V\})$ – функція, що характеризує близькість вектора v до сусідніх векторів {V};

$H(I_1, B_{ij})$ – Функція, що оцінює наявність деталей у блоці зображення.

Вектор v_x,у уважається помилковим, якщо значення функції довіри $\Psi(I_1, I_2, B_{xy}, v_{x,y})$ перевищує поріг T. Значення вагових коефіцієнтів і порогу визначаються емпірично.

Для виявлення областей інтересу за допомогою поворотних камер шляхом аналізу руху потрібно виключити зсув блоків зображення, пов'язане з рухом камери або зуммуванням. Для цього застосовується власний алгоритм компенсації глобального руху [3, 7].

1. Виконати фільтрацію поля векторів за значенням функції довіри.

2. Повторити N раз:

– вибрати випадковим образом три вектори з поля векторів руху;

– обчислити параметри афінного перетворення по трійці векторів;

– обновити гістограму по кожному параметрі афінной моделі глобального руху.

3. За допомогою алгоритму вододілу виділити піки гістограм, що лежать вище порога t_1.

4. Зі значень параметрів, що відповідають пікам, побудувати набір параметрів-кандидатів.

5. Вибрати параметри, на яких досягається мінімум міжкадрової різниці.

С допомогою знайдених параметрів глобального руху обчислюються вектори глобального руху для блоків. По векторах обчислюється помилка відповідності блоків. Блоки, для яких помилка відповідності перевищує поріг t_2, позначаються як приналежному об'єкту.

Аналіз руху виконується зі швидкістю 200 кадрів (320x240 пікселів) у секунду на Pentium 4 (2.8 Ггц), що дозволяє в реальному часі обробляти інформацію від 40 камер на одній машині.

Для виявлення областей інтересу за допомогою фіксованих камер використовуються моделювання тла у вигляді суміші гауссіанів і аналіз руху по відмінності векторів для блоків від нульових. Структурна схема системи представлена на рисунку 3.2.

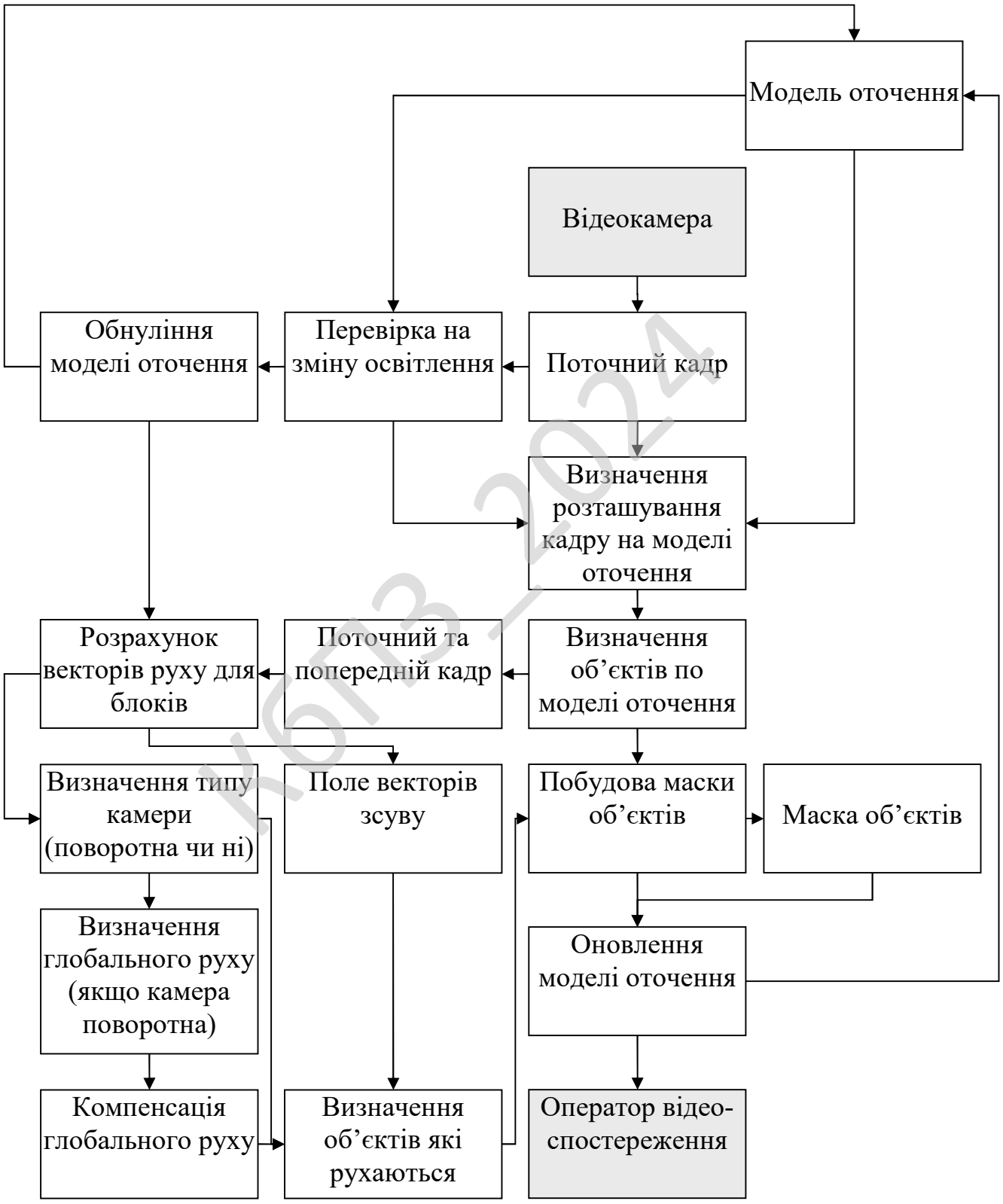

Рисунок 3.2 – Структурна схема системи

Метод дозволяє виявляти об'єкти, що рухаються, й нерухливі об'єкти, і зберігає працездатність навіть при сильних змінах умов висвітлення.

Розроблений метод виявлення областей інтересу за допомогою камер має стійкість до зміни висвітлення й дозволяє виявляти нерухливі об'єкти. Метод дозволяє виявляти об'єкти за допомогою поворотних камер під час сканування й зумування. Висока швидкість роботи дозволяє обробляти на одній машині дані, отримані від великої кількості камер.

3.3 Розробка функціональної схеми

Опишемо функції розробленої системи відеоспостереження, побудованої на базі розроблених алгоритмів спільного калібрування камер і виділення областей інтересу.

Функціональна схема системи складається з наступних блоків:

– Блок функцій оператора.

– Блок функцій інженера.

– Блок функцій адміністратора.

Блок функцій оператора, який включає в себе наступні функції:

1. Перегляд відео з камер.

2. Оцінка ситуації по плану:

– Відображення областей спостереження камер.

– Відображення доданих об’єктів.

– Відображення знайдених об’єктів.

3. Керування камерами:

– Клік по відео.

– Обертання камери.

– Клік по карті.

– Режим супроводу.

4. Зміна налаштувань системи.

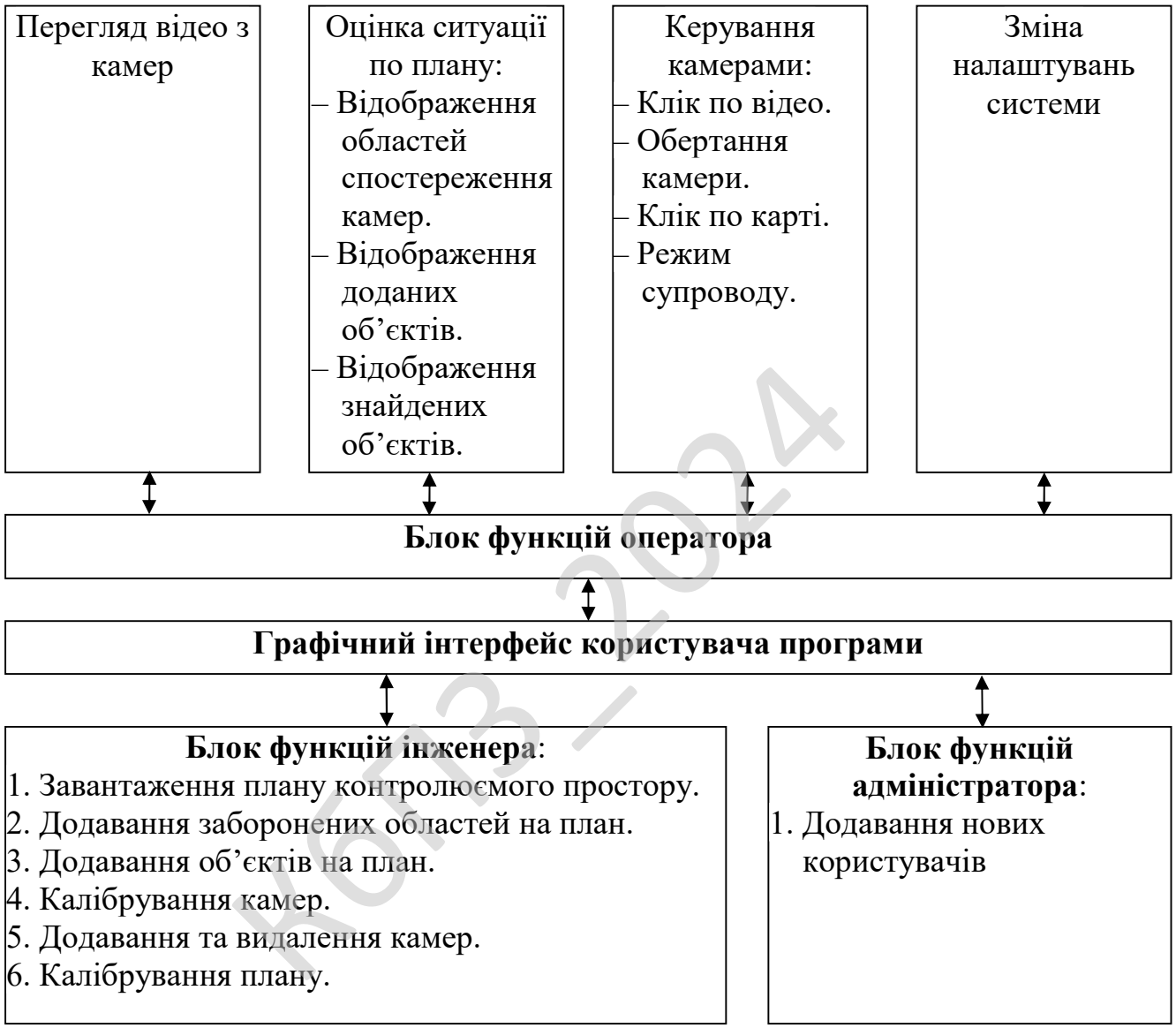

Рисунок 3.3 – Функціональна схема системи

Блок функцій інженера, який включає в себе наступні функції:

1. Завантаження плану контролюємого простору.

2. Додавання заборонених областей на план.

3. Додавання об'єктів на план.

4. Калібрування камер.

5. Додавання та видалення камер.

6. Калібрування плану.

Блок функцій адміністратора, який включає в себе наступні функції:

1. Додавання нових користувачів.

Система дозволяє не тільки контролювати обстановку по відеоряду, але й відслідковувати всі зміни на двомірному плані охоронюваного простору. На плані відображаються області видимості камер і виявлені об'єкти, що полегшує сприйняття загальної обстановки.

Так, наприклад, на плані легко помітити області контрольованого простору не видимі ні однією відеокамерою.

Крім того система надає користувальницький інтерфейс для керування камерами за допомогою кліків миші на компоненті головного вікна, із зображенням плану.

Система відеоспостереження має розвитий графічний інтерфейс і істотну обчислювальну частину.

Система має більш багату функціональність ніж існуючі комерційні розробки. Використання поворотних камер, розмаїтість можливостей по керуванню камерами й спостереженню за обстановкою дозволяє ефективніше вирішувати завдання відеоспостереження, ніж при використанні традиційних систем відеоспостереження з нерухливими камерами.

Таких можливостей удалося досягти завдяки використанню розроблених методів калібрування камер і виявлення областей інтересу.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.4. Після початку роботи розробленого ПЗ ми потрапляємо до головного вікна ПЗ звідки

можемо потрапити до моніторингу камер системи відеоспостереження через підключення камер системи відеоспостереження.

У подальшому провести перегляд наявного архіву системи відеоспостереження, через поточний кадр відеокамери провести визначення розташування кадру на моделі оточення та перейти до визначення об'єктів по моделі оточення, побудови маски об'єктів, розрахунку векторів руху для блоків, визначення об'єктів які рухаються.

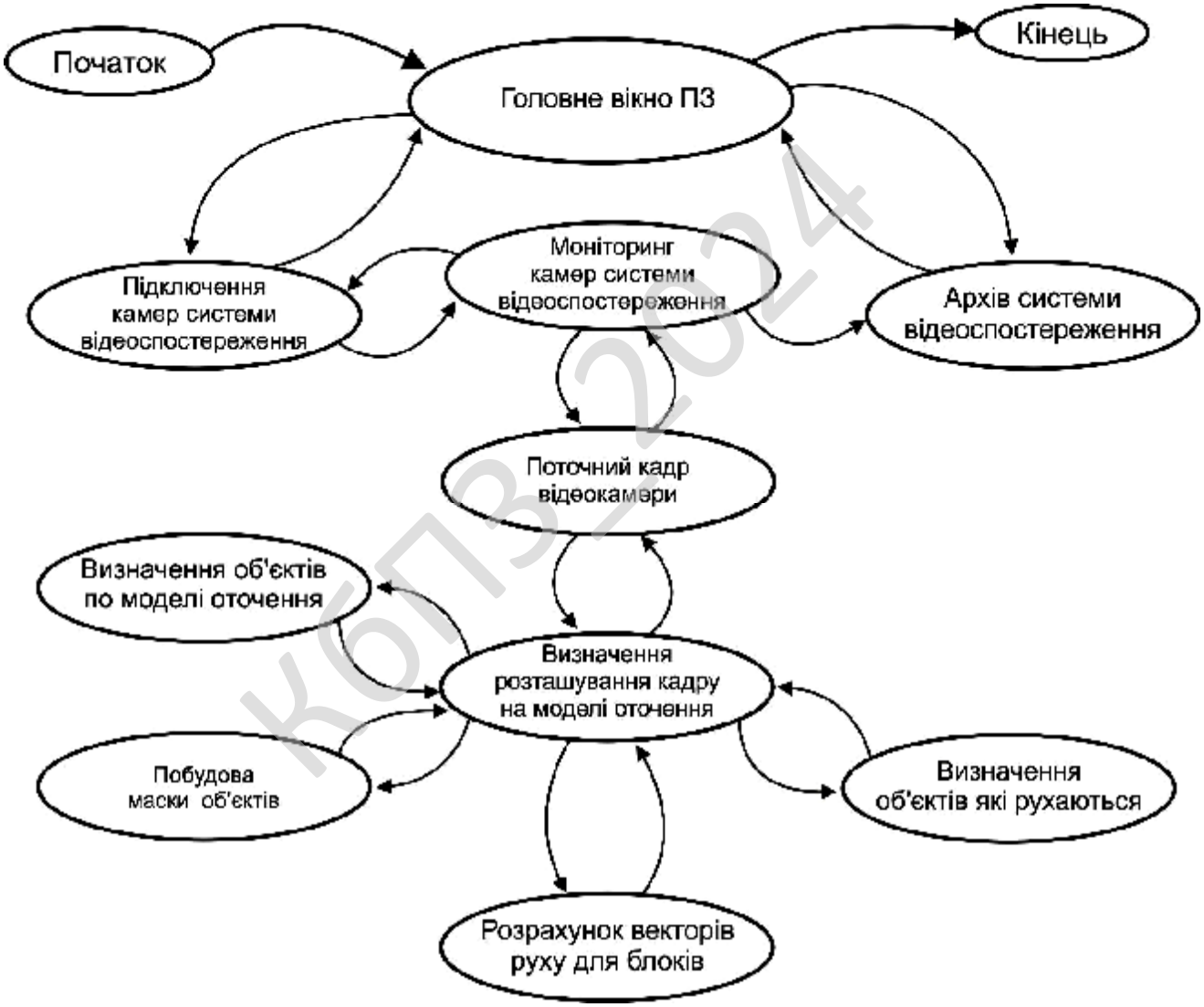

Рисунок 3.4 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну

схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що після запуску програми спочатку відбувається виведення вікна моніторингу. Потім здійснюється:

– Підключення камер системи відеоспостереження.

– Камери проініціалізовано (запит).

– Отримання поточного кадру.

– Читання моделі оточення та визначення розташування кадру на моделі оточення.

– Визначення об'єктів по моделі оточення.

– Аналіз поточного та попереднього кадру.

– Розрахунок векторів руху для блоків.

– Розрахунок поля векторів зсуву.

– Визначення об'єктів які рухаються.

– Побудова маски об'єктів.

– Оновлення моделі оточення, виведення поточних результатів.

– Запит перегляду відео архіву?

– Перегляд архіву.

– Запит обробки відео файлів?

– Підпрограма обробки відео файлів.

– Оновлення архівної підсистеми.

– Запит WM_CLOSE?

– Приховання вікон ПЗ.

– Підпрограма обробки запитів модулів.

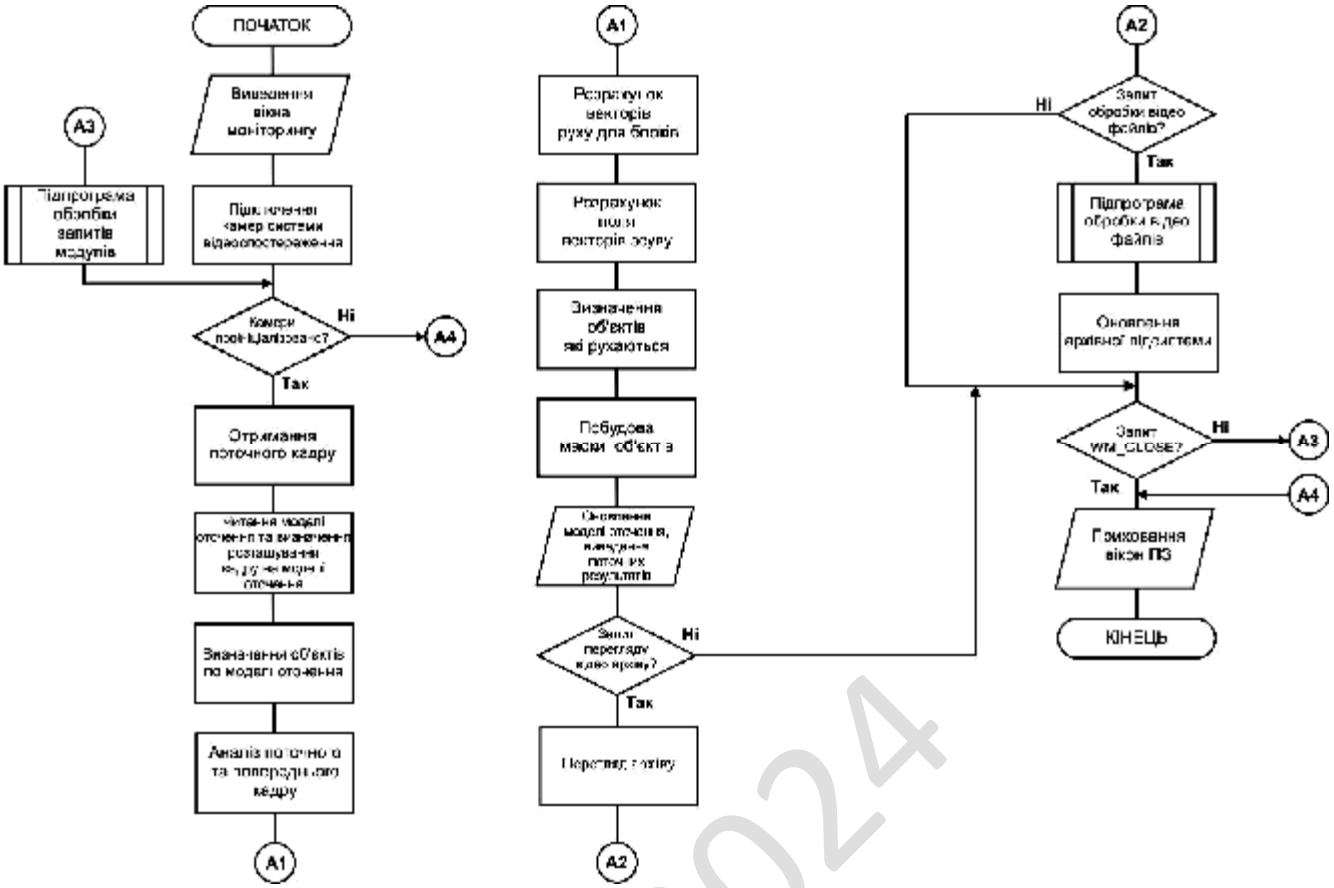

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображено роботу підпрограми обробки запитів модулів:

– Читання списку доданих модулів.

– Є запити на виконання?

– Читання поточного статусу модулів, формування списку на виконання.

– Запит оновлення документації?

– Завантаження нового файлу документації з Інтернет мережі.

– Запит оновлення моделі оточення?

– Завантаження нового файлу моделі оточення з Інтернет мережі.

– Запит перезавантаження ПЗ?

– Подача системного повідомлення WM_CLOSE.

– Запит оновлення маски об'єктів?

– Завантаження нового файлу маски об'єктів з Інтернет мережі.

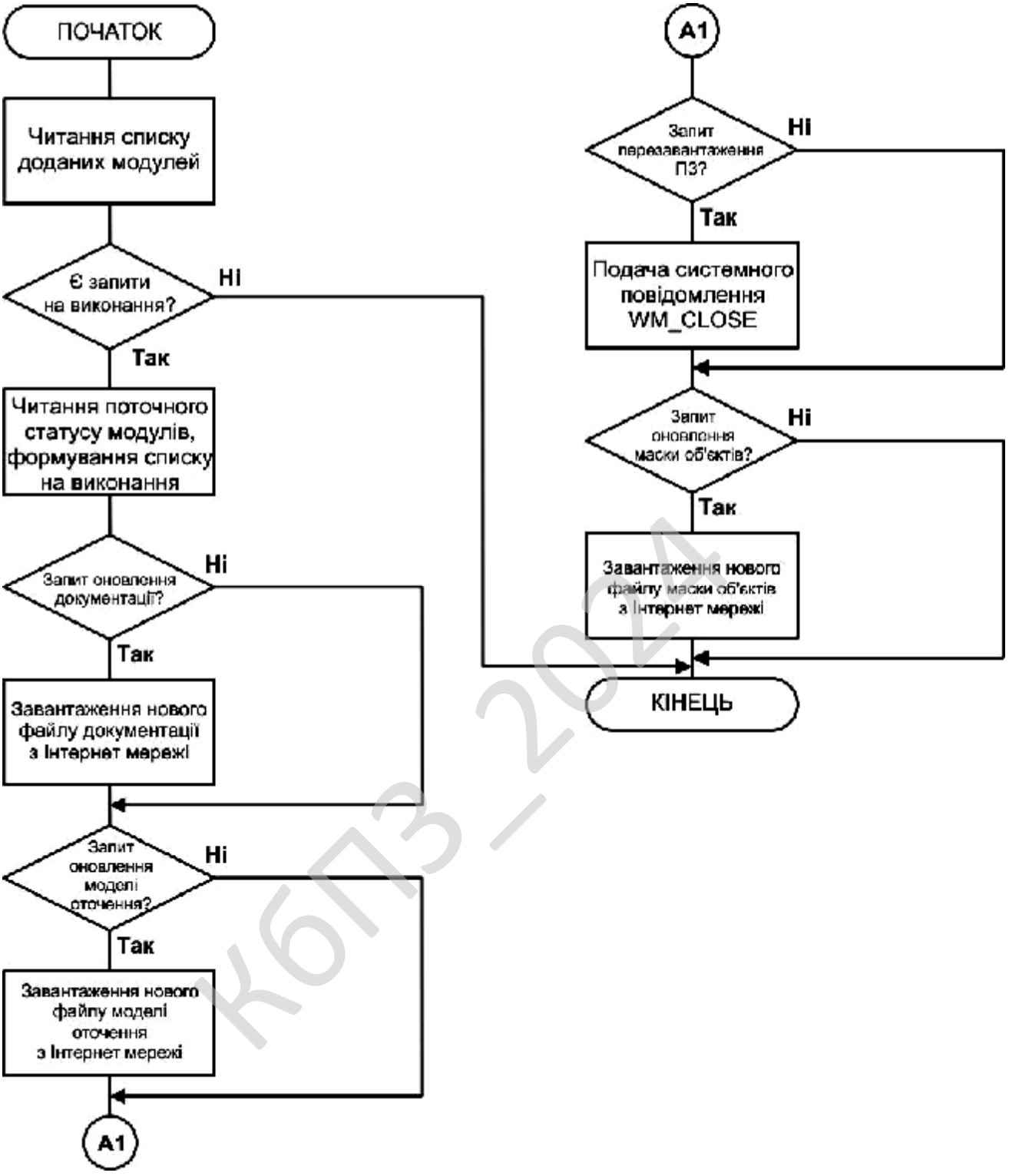

Рисунок 4.2 – Блок-схема підпрограми обробки запитів модулів

На рисунку 4.3 зображено роботу підпрограми обробки відео файлів:

– Формування списку відео архіву.

– Виведення файлів відео архіву.

– Запит видалення файлу?

– Видалення файлу відео архіву.

– Запит видалення папки?

– Читання архіву.

– Запит обробки відео файлу?

– Читання у налаштуваннях обраного обробника відеофайлів.

– Завантаження стороннього редактора обробки відеофайлів.

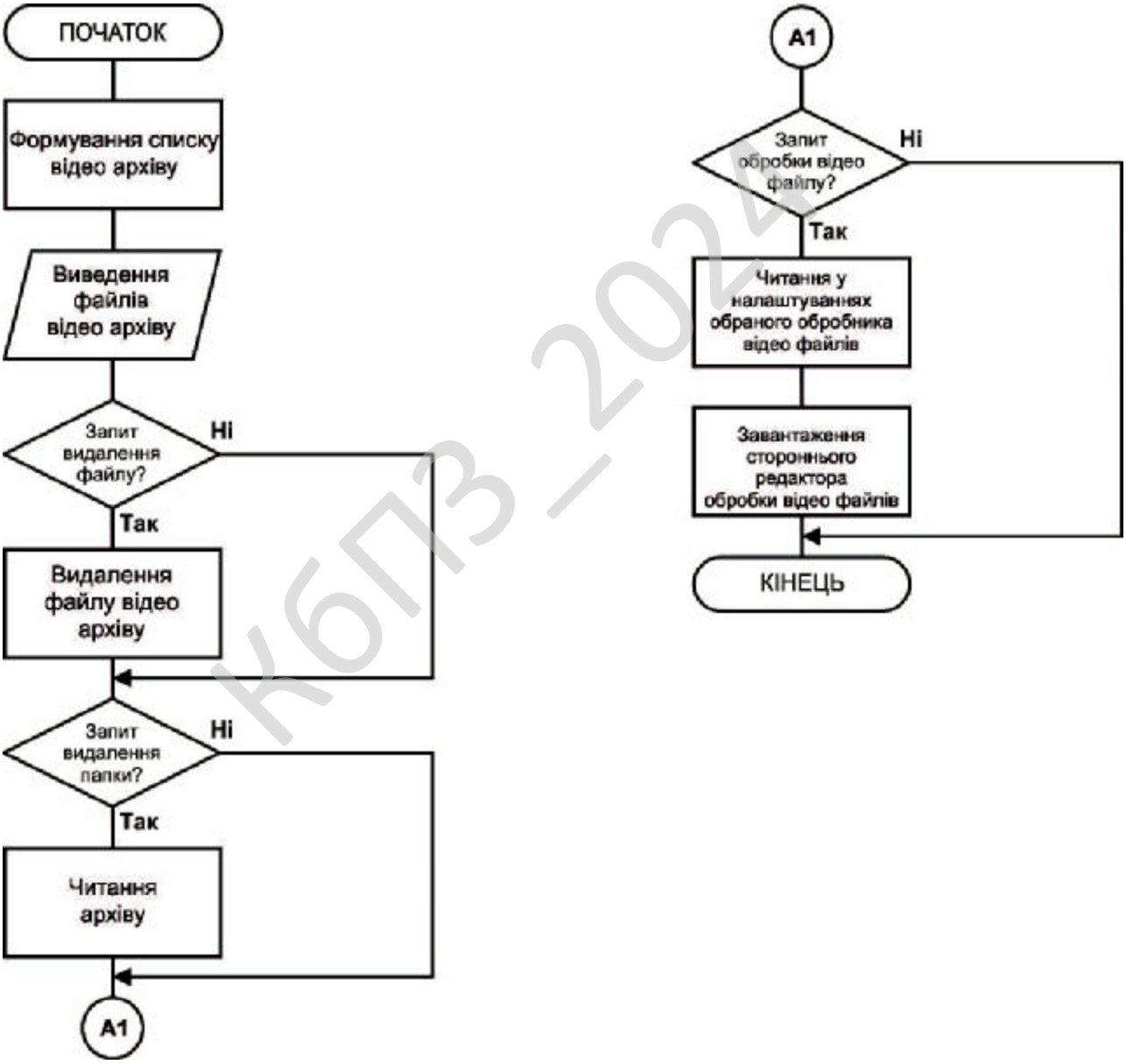

Рисунок 4.3 – Блок-схема підпрограми обробки відео файлів

Опис алгоритмів функціонування системи

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів взаємодії з відеоданими я зіткнулася 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. Крім цього були детально розглянуті протоколи відео, як приклад протокол керування відео потоками RTSP.

Real Time Streaming Protocol. RTSP являє собою протокол керування відеопотоками для використання в розважальних і комунікаційних системах.

Протокол RTSP дає можливість клієнтові запитувати потоки з мультимедійних серверів, подібно тому, як HTTP дозволяє клієнтам видавати запити до Web-серверів.

Фактично RTSP перейняв більшу частину свого синтаксису й семантики від HTTP/1.1, оскільки формально обидва протоколи виконують схожі функції. Подібність підкреслює загальний характер багатьох реалізованих в HTTP концепцій.

Однак протоколи мають ряд ключових відмінностей, які пов'язані з унікальними вимогами для мультимедійних потоків і обмеженістю можливостей HTTP/1.1 по передачі мультимедійних даних.

Перед розглядом специфікація RTSP, необхідно знати що таке SDP. SDP (Session Description Protocol) мережний протокол, призначений для опису сесії передачі потокових даних мультимедіа.

Сесія SDP може реалізовувати кілька потоків даних. У протоколі SDP у цей час визначені відео дані.

Повідомлення SDP, передане від одного вузла іншому може вказувати на:

– адреси місця призначення, які можуть бути для медіа-потоків мультикастінг–адресами;

– номера UDP портів для відправника й одержувача;

– медіа-формати (наприклад кодек), які можуть застосовуватися під час сесії;

– часу старту й зупинки. Використовується у випадку широкомовних сесій, наприклад, телевізійних або радіопрограм.

Можна внести час початку, завершення й часи повторів сесії. Незважаючи на те, що Session Description Protocol надає можливість опису мультимедіаданих, у ньому не вистачає механізмів узгодження параметрів сесії. Розглянемо приклад надання моделі узгодження на основі механізму відгуку.

У прикладі вузли обмінюються SDP повідомленнями з метою дійти згоди щодо формату даних.

```
v=0
o=- 1815849 0 IN IP4 194.67.15.181
s=Cisco SDP 0
c=IN IP4 194.67.15.181
t=0 0
m=audio 20062 RTP/AVP 99 18 101 100
a=rtpmap:99 G.729b/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=rtpmap:100 X-NSE/8000
a=fmtp:100 200-202
a=sqn:0
a=cdsc: 1 audio RTP/AVP 99 18 101 100
a=cdsc: 5 image udptl t38
a=cpar: a=T38Faxversion:0
a=cpar: a=T38Faxratemanagement:transferredtcf
a=cpar: a=T38Faxmaxdatagram:160
a=X-sqn:0
a=X-cap: 1 image udptl t38
```

Специфікація RTSP значною мірою перегукується зі специфікацією HTTP/1.1. На верхньому рівні обидва протоколи переслідують одну мету — дати можливість клієнтові запитувати зміст із сервера на основі URI запиту.

Історично розробка RTSP почалася після того, як була виконана частина робіт над специфікацією HTTP.

Оскільки HTTP послужив основою для RTSP, мети й завдання цих протоколів багато в чому збігаються.

Доставка мультимедійних потоків використовує таку ж саму модель, що й HTTP: ресурси, URI що ідентифікуються, взаємодія запит/відповідь, а також можливість передачі даних через одне або декількох проміжних ланок на шляху між клієнтом і сервером.

Використання готових концепцій, включення спеціальних заголовків протоколу й кодів відповідей зменшило витрати на розробку й реалізацію RTSP. Крім того, обидва протоколи відіграють роль в ініціюванні передачі мультимедійних потоків через WEB.

Подібність між двома протоколами забезпечує значну гнучкість при ухваленні рішення, який протокол буде обслуговувати конкретну функцію.

Розглянемо запит на інформацію з описом сеансу (наприклад, http://www.foo.com/bar.sdp). Відповідь Web-сервера буде включати інформацію, у форматі SDP:

```
HTTP/1.1 200 OK Content-Type: application/sdp
v=0
o=- 2890844526 2890842807 IN IP4 192.16.24.202 s=RTSP Session
m=audio 0 RTP/AVP 0
a=control:
rtsp://foo.com/bar/audio
m=video 0 RTP/AVP 31
a=control:
rtsp://foo.com/bar/video
```

У цей момент Web-браузер уже може активізувати медіаплейер для виконання інших дій. Крім того, медіаплейер може надати користувачеві інтерфейс для вибору мультимедійних сеансів.

У цьому випадку медіаплейер може прямо взаємодіяти з Rtsp-сервером для уточнення описів без залучення Web-браузера.

Методи запитів RTSP. RTSP-сервери повинні підтримувати чотири основні методи, використовуваних клієнтами для добування мультимедійних сеансів: OPTIONS, SETUP, PLAY і TEARDOWN.

На верхньому рівні заголовок OPTIONS дозволяє клієнтові визначати функціональні можливості сервера, такі як номер версії RTSP і підтримувані методи; цей метод поводиться так само, як метод OPTIONS HTTP/1.1.

Інші три методи маніпулюють збереженої на сервері інформацією про стан для координації передачі мультимедійних даних. Клієнт використовує метод SETUP для встановлення транспортного з'єднання для кожного потоку в сеансі. Метод PLAY використовується для ініціювання передачі потоку (потоків).

Метод TEARDOWN використовується для завершення передачі. Більшість методів запитів пов'язане з підтримкою клієнта, який запитує відтворення потоку з мультимедійного сервера.

Однак RTSP – сервер може також допускати запис мультимедійного змісту за допомогою необов'язкових методів ANNOUNCE і RECORD.

Допустимо, користувач прагне записати телеконференцію на Rtsp-сервері для подальшого відтворення. У цьому випадку клієнт відправляє опис конференції серверу в повідомленні ANNOUNCE.

Потім клієнт повинен відправити повідомлення SETUP для кожного з потоків у сеансі, щоб проінформувати сервер про транспортні параметри, таких як Ip-адреси й номера портів. URI запиту в повідомленні SETUP указує ім'я, яке клієнт прагне асоціювати із записуваним змістом.

Після цього клієнтові слід відправити запит RECORD для початку запису потоків. Через якийсь час клієнт може зв'язатися із сервером і переглянути записану конференцію.

Обробка відео потоку і виведення на екран в середовищі Windows при застосуванні основних методів виведення відео інформації на екран лінійки операційних систем Windows.

Виникає гостра проблема в швидкості обробки потокового кадру, що приводить до уповільнення процесу виведення інформації на екран. Як відомо для перегляду відеопотоку необхідно не менше 24 кадрів в секунду.

Перед розглядом подробиць схеми роботи програми розглянемо виконані основні напрацювання.

Розглянемо розроблені класи:

– Клас TMAG_AbstractAllocator

Оголошення класу:

Type AbstractAllocator = class()

Опис:

Абстрактне виділення покажчика для захоплення відео потоку.

– Клас TMAG_ASFWriter

Оголошення класу:

Type TMAG_ASFWriter = class(A,B)

Опис:

Цей клас розроблений, щоб створити тимчасовий файл ASF (журнал роботи ПЗ).

– Клас TColorControl

Оголошення класу:

type TMAG_ColorControl = class()

Опис:

Установка і отримання контролю кольору від Video вікна. Клас підтримує апаратні API функції операційної системи.

Одночасно можна провести перевірку на здібність доступу програми до API функцій операційної системи.

– Клас TrackBar

Оголошення класу:

type TMAG_TrackBar = class(A,B)

Опис:

Панель коректування відео потоку інформації.

– Клас TVideoWindowEx2

Оголошення класу:

type TMAG_VideoWindowEx2 = class(A,B,C)

Опис**:**

Альтернатива до стандартного відео вікна, який пропонує більш швидкий і легкий шлях висновку відео на екран в додатку.

– Клас TVideoWindowEx2Caps

Оголошення класу:

type TMAG_VideoWindowEx2Caps = class()

Опис:

Перевірка на сумісність висновку потокової інформації.

– Клас TFilter

Оголошення класу:

type TMAG_Filter = class(A,B)

Опис:

Клас пропонує можливість створення і підключення до програми додаткових фільтрів.

– Клас TFilterGraph

Оголошення класу:

type TMAG_FilterGraph = class()

Опис:

Центральний клас де відбувається синхронізація і протоколювання системи призначених для користувача повідомлень, а також повідомлень операційної системи.

– Клас TSampleGrabber

Оголошення класу:

type TMAG_SampleGrabber = class(A,B,C)

Опис:

Клас дозволяє створювати знімки екрану і знімки аудіо ділянок.

– Клас TVideoWindow

Оголошення класу:

type TMAG_VideoWindow = class(A,B,C)

Опис:

Управління висновком потокового відео на екран ПЗ.

– Клас TVMRBitmap

Оголошення класу:

type TMAG_VMRBitmap = class

Опис:

Коректування виду матриці формату BMP.

Розглянемо типи даних, що використовуються у бакалаврському ПЗ:

– PVMRPreferences. Приречений тип-покажчик на TVMRPreferences.

– TAbstractAllocatorClass . Тип даних абстрактного виділення для класу

– TGraphMode. Тип даних для вибору графічного режиму

– TSeekingCap. Конкретизує здібності пошуку медіа потоку.

– TSeekingCaps. Конкретизує здібності пошуку медіа потоків.

– TVideoMode. Тип відео установки для класу TVideoWindow.

– TBitmapOption. Тип даних настройок захоплення кадру

Розглянемо основні константи, що часто використовуються у бакалаврському ПЗ:

– rdOverlay – вказівка верхнього шару, що виводиться;

– rdSysMem – адреса початку системних функцій;

– rdVidMem – адреса початку відео масиву;

– WM_CAPTURE_BITMAP – Номер створеного призначеного для користувача повідомлення, що генерується при захопленні зображення

WM_GRAPHNOTIFY – Ідентифікатор фільтру повідомлень.

Розглянемо головну задачу яку необхідно було виконати при реалізації бакалаврської роботи – компенсація руху.

Компенсація руху це один з основних алгоритмів, які застосовуються при обробці й стиску відеоданих.

Алгоритм використовує подібність сусідніх кадрів у відео послідовності й знаходить вектора руху окремих частин зображення (звичайно блоків 16x16 і 8x8). Використання компенсації дозволяє при стиску багаторазово збільшити ступінь стиску за рахунок видалення надмірності у вигляді співпадаючих частин кадрів.

Використовується не тільки при стиску, але й при фільтрації відео, зміні частоти кадрів і т.д.

Рішення проблеми стиску стало першорядним завданням, починаючи із самої появи цифрового відео. Для оцінки візьмемо відеоряд з наступними параметрами:

– Розмір кадра: 720x576 (PAL, 414720 пікселей).

– Частота кадрів: 25 кадрів / сек (PAL).

– Кольори: YV12 (YUV 4:2:0) (16 біт на 4 пікселя + 8 біт на кожний = 12 біт на піксель).

У підсумку на запис або передачу однієї секунди такого відео без застосування стиску буде потрібно 14,8 мегабайта без звуку й службової інформації.

Для зберігання півторагодинного фільму вже буде потрібно 78 гігабайт, що зовсім неприпустимо у зв'язку з написанням програми для мобільної ОС Windows Mobile де виділені ресурси на програму сильно обмежені.

Практично в кожному відео сусідні кадри схожі, мають загальні об'єкти, які, як правило, зміщаються друг щодо друга. І зовсім природнє бажання закодувати відео так, щоб об'єкти не кодувалися багаторазово, а просто описувалися деякі їхні зсуви.

Навіть у цьому випадки якщо взяти й запакувати архіватором 1 кадр і всі зображення меж кадрової різниці, вийде помітний виграш при стиску. Але цей виграш можна суттєво збільшити.

У зв'язку з високою обчислювальною складністю алгоритмів розпізнавання образів і недостатньої точності їх роботи застосовують різні методи, що дозволяють швидко знаходити вектори руху.

1. Завантажується поточний кадр.

2. Кадр ділиться на блоки (наприклад 16x16).

3. Проводиться обхід блоків (кожний блок у цьому випадку обробляється окремо).

4. При розрахунку одного блоку проводиться обхід деякої околиці блоку в пошуку максимальної відповідності зображенню блоку на попередньому кадрові в межах цієї околиці.

5. Таким чином, після завершення пошуку ми одержуємо набір векторів, що вказує "рух" блоків зображення між кадрами. Ці вектори можуть бути природно використані для створення зображення скомпенсованого кадра, який краще наближає кадр для якого проводилася компенсація руху.

Алгоритми для порівняння кадрів:

1. Обчислення SSD (суми квадратичних відхилень). Для пари блоків дає гарні результати по якості, особливо при еталонних тестах, тому що метрика PSNR обчислюється на основі середнього квадратичного відхилення, але вимагає значних витрат ресурсів (множення операція повільна, навіть таблиця квадратів не сильно прискорює процес) і сильно чутливий до зміни яскравості. Чим менше SSD – тим більше схожі блоки.

2. Порівняння по характерних точках. Може виконуватися дуже швидко (за рахунок обходу лише невеликого числа точок), але може дуже погано корелювати з більш якісними метриками.

3. Обчислення SAD (суми абсолютних різниць). Виконується за розумний час і дає прийнятний результат по якості (але має низьку стійкість до шуму). Реально застосовується й має гарні швидкісні показники за рахунок використання SIMD розширень (які дозволяють виконувати безліч вирахувань одночасно без використання "інтелектуальних" засобів процесора по паралельним обчисленням).

Найбільше часто використовується обчислення SAD. Для цього необхідно знайти заданий блок:

1. Повний перебір (Full Search). У деякій області навколо оброблюваного блоку відбувається перебір координат шуканого блоку. Якщо маємо блок 16x16 і область пошуку ±32 х ±32, то нам потрібно буде 4096 раз порахувати SAD для кожного оброблюваного блоку. Це повільно, але дає гарантовано кращий результат по заданій метриці.

2. Пошук по шаблонові. Виконується швидко, дає не кращі результати.

3. Спіральний пошук. Вважається, що чим ближче блок до поточного, тим більше ймовірність того, що він шуканий. І його точність зменшується від центру до країв області пошуку.

На картинці з векторами руху видно довгі вектора, тому що використовувався повний перебір починаючи з лівого верхнього кута області пошуку, хоча очевидно, що з нульовими векторами немає практично ніякої різниці, але довгі вектора погіршують стискальність поля векторів, а нульові вектора немає.

Для розробки бакалаврського ПЗ використовувалась мова програмування Object Pascal в середовищі Borland Delphi XE5 (більш докладно див. додаток В). Але деякі модулі було реалізовано у вигляді динамічної бібліотек. Розглянемо реалізацію методу повного перебору розроблену для бакалаврської роботи у вигляді динамічної бібліотеки (ABC.DLL) на мові Builder C 7.0.

```
void ASD(BYTE* Currentframe, BYTE* Previousframe, int Width,
       int Height, MV *Motionvectors)
{
  int Blocksperheight = (Height + 15) >> 4;
//Число блоків по вертикалі
  int Blocksperwidth  = (Width  + 15) >> 4;
//Число блоків по горизонталі
  int Verticaloffset, Horizontaloffset, Tempoffset;
//Зсув
  int Offsetperline = Width + Border * 2;
//Зсув на один рядок
  int Startoffset = Offsetperline * Border + Border;
```

```
//Початковий зсув
  BYTE *Currptr, *Prevptr;
//Покажчики на попередній і на наступний кадри
        MV Probmv;
//Пробний вектор
        long Minerror, Error;
//Значення погрішності
        for (int i = 0; i < Blocksperheight; i++)
        {
                for (int j = 0; j < Blocksperwidth; j++)
                {
//Обчислюємо зсув
                    Verticaloffset = (i << 4) * Offsetperline + Startoffset;
                    Horizontaloffset = (j << 4);
//Встановлюємо зсуву на поточний блок
                    Currptr=Currentframe+Verticaloffset+Horizontaloffset;
                    Prevptr=Previousframe+Verticaloffset+Horizontaloffset;
                    Minerror = MAXLONG;
// Обраховуємо помилку
                for (int y = -Maxmotion; y < Maxmotion; y++ )
                {
//Заощаджуємо час на множеннях
                Tempoffset = y * Offsetperline;
                        for (int x = -Maxmotion; x < Maxmotion; x++ )
                        {
//Geterror – функція порівняння блоків, наприклад, SAD
     Error = Geterror (Currptr, Prevptr + Tempoffset + x, Offsetperline);
                  if (Error < Minerror)
                  {
                  Probmv.x = x;
                  Probmv.y = y;
                  Minerror = Error;
                  }
                       }
                      }
//Записуємо отриманий вектор
                        Motionvectors [i * Blocksperwidth + j] = Probmv;
                }
        }}
```

У новому стандарті MPEG-4 AVC/H.264 введені також не квадратні (прямокутні) блоки, розмір яких може дробитися до 4x4 пікселя.

У такий спосіб вдається досить ефективно використовувати подібність сусідніх кадрів, а завдяки більш складній формі блоків зростає точність компенсації руху на границях об'єктів, що рухаються. Крім компенсації руху для подальшого уточнення зображення (або для областей, що знову з'являються, яких не було в минулих кадрах) використовується стиск між кадрової інформації й незалежний стиск блоків.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм MMB, в основі якого лежить змішування операцій різних алгебраїчних груп. MMB – ітеративний алгоритм, що складається з лінійних дій (XOR і використання ключа) і паралельного застосування чотирьох великих оборотних нелінійних підстановок. Ці підстановки визначаються за допомогою множення по модулю 2^{32}-1 з постійними множниками. У підсумку з'являється алгоритм, що використовує 128-бітовий ключ і 128-бітовий блок.

Алгоритм MMB оперує 32-бітовими підблоками тексту (x_0, x_1, x_2, x_3) і 32-бітовими підблоками ключу (k_0, k_1, k_2, k_3). Це спрощує реалізацію алгоритму на сучасних 64-бітових процесорах. Чергуючись із операцією XOR, шість разів використовується нелінійна функція *f*. Запишемо операції алгоритму (всі операції з індексами виконуються по модулю 4):

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

Функція f виконується в три кроки:

1. $x_i = c_i * x_i$ для $i = 0..3$ (Якщо на вході множення одні одиниці, то на виході – теж одні одиниці).

2. Якщо молодший значущий біт $x_0 = 1$, то $x_0 = x_0 \oplus C$. Якщо молодший значущий байт $x_3 = 0$, то $x_3 = x_3 \oplus C$.

3. $x_i = x_{i-1} \oplus x_i \oplus x_{i+1}$ для $i = 0..3$.

Всі операції з індексами виконуються по модулю 4. Операція множення на кроці 1 виконується по модулі 2^{32}-1. Спеціальний випадок для даного алгоритму: якщо другий операнд дорівнює 2^{32}-1, результат теж дорівнює 2^{32}-1. В алгоритмі використовуються наступні константи:

$$C = 2aaaaaaa,\ c_0 = 025f1cdb,\ c_1 = 2*c_0,\ c_2 = 2^3 * c_0,\ c_3 = 2^7 * c_0.$$

Константа C – «найпростіша» константа без кругової симетрії, високою трійковою вагою й нульовим молодшим значущим бітом. У константи c_0 є інші особливі характеристики. Константи c_1, c_2 і c_3 – зрушені версії c_0, і служать для запобігання атак, заснованих на симетрії.

Розшифрування виконується у зворотному порядку, Етапи 2 і 3 інверсні їм самим. На етапі 1 замість c_i використовується c_i^{-1}. Значення c_0^{-1} = 0dad4694.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: Допомога; Архівні дані; Налаштування.

– Вікно виведення відео зображення;

– Функцій програми (кнопок): Сканувати; Обрати сигнал; Шаблон спостереження; Налаштування; Вікно відео архіву; Відображення областей спостереження камер: Off; Відображення доданих об’єктів: Off; Відображення знайдених об’єктів: On.

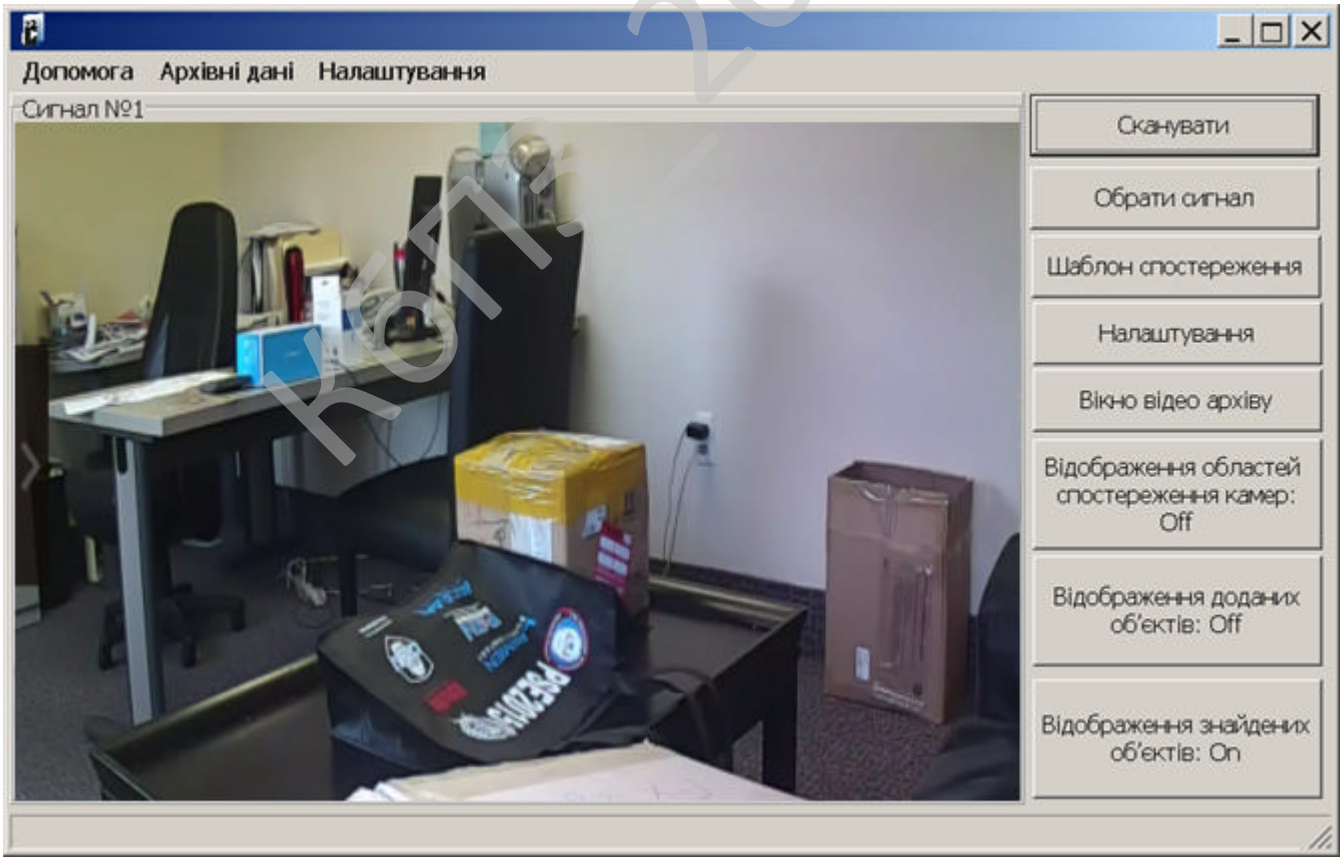

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено авторське право. Авторське право є ключовою галуззю права інтелектуальної власності; воно призначене захищати лише зовнішню форму вираження об'єкта, тобто їхнє матеріальне втілення.

Авторське право не може використовуватись для захисту абстрактних ідей, концепцій, фактів, стилів та технік, що можуть бути використані у творі. Захист авторського права — одна з важливих категорій теорії цивільного та цивільно-процесуального права.

Під захистом авторських прав слід розуміти передбачені законом заходи із їхнього визнання, припинення їхнього порушення, застосування до правопорушників заходів юридичної відповідальності. Захист особистих немайнових і майнових прав суб'єктів авторського права здійснюється в порядку, встановленому адміністративним, цивільним і кримінальним законодавством. Було обрано Shareware умову розповсюдження. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

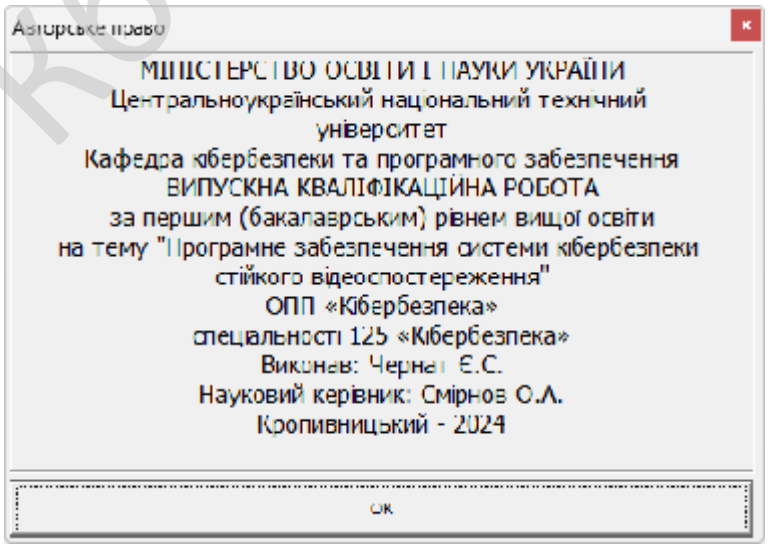

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки стійкого відеоспостереження.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем стійкого відеоспостереження.

– Досліджена система стійкого відеоспостереження.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки стійкого відеоспостереження.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання стійкого відеоспостереження.

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 Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки стійкого відеоспостереження. 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ММВ.

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. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. The MIT Press. 2022 1677 p.

2. Will Grant. 101 UX Principles. Packt Publishing. 2022. 432 p.

3. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

4. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

5. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

6. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

7. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009

8. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

9. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

10. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

11. 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.

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

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

14. 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.

15. 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.

16. 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.

17. 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,

18. 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.

19. 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

20. 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.

21. 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.

22. 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.

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

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

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

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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).

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки стійкого відеоспостереження.

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. № 135-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки стійкого відеоспостереження.

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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки стійкого відеоспостереження;

– 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 Delphi 10.4 Sydney.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 72 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2024 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

_____ Смірнов О.А.

Програмне забезпечення системи кібербезпеки стійкого відеоспостереження

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 52

Літера: РП

Кропивницький – 2024 року

РОЗРОБЛЕНИЙ ГОЛОВНИЙ ФАЙЛ ПЗ

```
// Назва розробленої програми
program My_Video;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Виконав: Чернат Єгор Сергійович, КБ-20
Керівник: Смірнов О.А.
}
{$R 'RES.res'}
// Підключення ресурсів
Uses
// Підключення бібліотек
  Forms, // форми
//  Головне вікно ПЗ
  Video_RTSP in 'Video_RTSP.pas' {Form1},
// Вікно налаштувань
  Settings in 'Settings.pas' {Form2},
// Вікно роботи з мережою
  LAN in 'LAN.pas'{Form3},
/ Вікно авторського права
  About in 'About.pas' {AboutBox};
begin
// З цього місця починається робота коду
// Проходить ініціалізіція ПЗ
  Application.Initialize;
// Встановлюється назва ПЗ, яка
// буде відображатись у рамці основного вікна
  Application.Title := 'VIDEO';
// Підключення форми 1
  Application.CreateForm(TForm1, Form1);
// Підключення форми 2
  Application.CreateForm(TForm2, Form2);
// Підключення форми 3
  Application.CreateForm(TForm3, Form3);
// Підключення форми авторського права
  Application.CreateForm(TAboutBox, AboutBox);
// Виведення на єкран головної форми ПЗ
  Application.Run;
 end.
```

Файл налаштувань

```
unit Settings;
// Назва модулю
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Виконав: Чернат Єгор Сергійович, КБ-20
Керівник: Смірнов О.А.
}

interface
{Інтерфейсна частина, тут проходе опис класів та типів}

Uses
// Підключення бібліотек
   Windows,
   WInSock,
   Mobile,
   SysUtils,
   Classes;

const
   mibLen = $0A;

type
   TMibId = array[1..mibLen] of integer;

   TIpConnInfo = class // Об'ява класу
   private // секція бачення
      FRemoteIp :TInAddr;
      FRemotePort :integer;
      FState :DWORD;
      FLocalIp :TInAddr;
      FLocalPort :integer;
      FProto :ShortString;
      function GetLocalPort :Integer;
      function GetLocalIpString :String;
      function Ip2Str( const Ip :TInAddr ):String;
      function GetRemotePort :Integer;
      function GetRemoteIpString :String;
      function GetStateString :String;
   public
      constructor Create( const aProto :String );
      property IpProtoName :ShortString
         read FProto;
      property LocalIp :TInAddr
         read FLocalIp;
      property LocalPort :Integer
         read GetLocalPort;
      property LocalIpString :String
         read GetLocalIpString;
```

```
   property RemoteIpString :String
      read GetRemoteIpString;
   property RemoteIp :TInAddr
      read FRemoteIp;
   property RemotePort :Integer
      read GetRemotePort;
   property State :DWORD
      read FState;
   property StateString :String
      read GetStateString;
end;


EConnListError = class(Exception); // Об'ява класу
   EConnListLockError = class(EConnListError); // Об'ява класу
   EConnListUnlockError = class(EConnListError); // Об'ява класу
   EConnListRefreshError = class(EConnListError); // Об'ява класу

TIpConnListStatus = ( connlist_ready, connlist_refreshing );

TIpProtocol = (udp_ip, tcp_ip);
TIpProtocols = set of TIpProtocol;

TFnugryIpConnectionList = class(TComponent) // Об'ява класу
private // секція бачення
   FConnections :TList;
   FWsInited :Bool;
   FMobileLib :THandle;
   FMobileQueryProc :Pointer;
   FMobileInitProc :Pointer;
   FProtocols :TIpProtocols;
   FStatus :TIpConnListStatus;
   FOnStatusChange :TNotifyEvent;
   FAccessMutex :THandle;
   FPollForTrapEvent :THandle;
   FSupportedViewRoot :TAsnObjectIdentifier;
   function GetConnCount :Integer;
   function GetConnections( Index :Integer ):TIpConnInfo;
protected
   procedure StatusChange; virtual;
   procedure SetStatus( Value :TIpConnListStatus ); virtual;
   procedure DoLock;
   procedure DoUnlock;
   procedure Clear;
public
   constructor Create( aOwner :TComponent ); override;
   destructor Destroy; override; // Деструктор
   procedure Refresh;
   function Lock( Timeout :DWORD ):Bool;
   function Unlock :Bool;
   property Status :TIpConnListStatus
```

```
      read FStatus;
   property Connections[ Index :Integer ] :TIpConnInfo
      read GetConnections; default;
   property ConnCount :Integer
      read GetConnCount;
published
   property Protocols :TIpProtocols
      read FProtocols write FProtocols;
   property OnStatusChange :TNotifyEvent
      read FOnStatusChange write FOnStatusChange;
end;

ENetstatError = class(Exception); // Об'ява класу

TVNstatCounterObject = class// Об'ява класу
private // секція бачення
   FName :String;
   FId   :TMibId;
   FDescription :String;
   FWsInited :Bool;
   FMobileLib :THandle;
   FMobileQueryProc :Pointer;
   FMobileInitProc :Pointer;
   function GetValue :DWORD;
public
   constructor Create( const aName, aDesc :String; const aId :TMibId );
   destructor Destroy; override; // Деструктор
   property Description :String
      read FDescription;
   property Name :String
      read FName;
   property Id :TMibId
      read FId;
   property Value :DWORD
      read GetValue;
end;

TCounterList = class(TComponent) // Об'ява класу
private // секція бачення
   FCounters :TStringList;
   function GetCount :Integer;
   function GetCounters( Index :Integer ):TVNstatCounterObject;
public
   constructor Create( aOwner :TComponent ); override;
   destructor Destroy; override; // Деструктор
   procedure Clear;
   procedure AddCounter( aCounter :TVNstatCounterObject );
   function IndexOf( const aName :String ):Integer;
   property Count :Integer
      read GetCount;
   property Counters[Index :Integer] :TVNstatCounterObject
```

```
        read GetCounters; default;
    end;

    TIpStats = class(TCounterList) // Об'ява класу
    public
       constructor Create( aOwner :TComponent ); override; //Конструктор
    end;

    TRtspStats = class(TCounterList) // Об'ява класу
    public
       constructor Create( aOwner :TComponent ); override; //Конструктор
    end;

    TTCPStats = class(TCounterList) // Об'ява класу
    public
       constructor Create( aOwner :TComponent ); override; //Конструктор
    end;

    TUdpStats = class(TCounterList) // Об'ява класу
    public
       constructor Create( aOwner :TComponent ); override; //Конструктор
    end;

implementation
{Секція де проходить реалізація того що описано в інтерфейс ній частині}

const
  DEFAULT_LOCK_TIMEOUT  = 100;
  MOBILE_LIB_NAME       = 'inetmib1.dll';
  MOBILE_INITPROC_NAME    = 'MobileExtensionInit';
  MOBILE_QUERYPROC_NAME   = 'MobileExtensionQuery';

  mib_tcpConnTable :TMibId =  (1, 3, 6, 1, 2, 1, 6, 13, 1, 1);
  mib_udpTable : TMibId = (1, 3, 6, 1, 2, 1, 7, 5, 1, 1);
  mib_ipDefaultTTL : TMibId = (1, 3, 6, 1, 2, 1, 4, 2, 1, 1);
  mib_ipInReceives : TMibId = (1, 3, 6, 1, 2, 1, 4, 3, 1, 1);
  mib_ipInHdrErrors : TMibId =  (1, 3, 6, 1, 2, 1, 4, 4, 1, 1);
  mib_ipInAddrErrors : TMibId = (1, 3, 6, 1, 2, 1, 4, 5, 1, 1);
  mib_ipForwDatagrams : TMibId = (1, 3, 6, 1, 2, 1, 4, 6, 1, 1);
  mib_InUnknownProtos : TMibId = (1, 3, 6, 1, 2, 1, 4, 7, 1, 1);
  mib_ipInDiscards : TMibId = (1, 3, 6, 1, 2, 1, 4, 8, 1, 1);
  mib_ipInDelivers : TMibId = (1, 3, 6, 1, 2, 1, 4, 9, 1, 1);
  mib_ipOutRequests : TMibId = (1, 3, 6, 1, 2, 1, 4, 10, 1, 1);
  mib_ipOutDiscards : TMibId = (1, 3, 6, 1, 2, 1, 4, 11, 1, 1);
  mib_ipOutNoRoutes : TMibId = (1, 3, 6, 1, 2, 1, 4, 12, 1, 1);
  mib_ipReasmTimeout : TMibId = (1, 3, 6, 1, 2, 1, 4, 13, 1, 1);
  mib_ipReasmReqds : TMibId = (1, 3, 6, 1, 2, 1, 4, 14, 1, 1);
  mib_ipReasmOKs : TMibId = (1, 3, 6, 1, 2, 1, 4, 15, 1, 1);
  mib_ipReasmFails : TMibId = (1, 3, 6, 1, 2, 1, 4, 16, 1, 1);
  mib_ipFragOKs : TMibId = (1, 3, 6, 1, 2, 1, 4, 17, 1, 1);
  mib_ipFragFails : TMibId = (1, 3, 6, 1, 2, 1, 4, 18, 1, 1);
```

```
  mib_ipFragCreates : TMibId = (1, 3, 6, 1, 2, 1, 4, 19, 1, 1);
  mib_rtspInMsgs : TMibId = (1, 3, 6, 1, 2, 1, 5, 1, 1, 1);
  mib_rtspInErrors : TMibId = (1, 3, 6, 1, 2, 1, 5, 2, 1, 1);
  mib_rtspInDestUnreachs : TMibId =  (1, 3, 6, 1, 2, 1, 5, 3, 1, 1);
  mib_rtspInTimeExcds : TMibId = (1, 3, 6, 1, 2, 1, 5, 4, 1, 1);
  mib_rtspInParmProbs : TMibId = (1, 3, 6, 1, 2, 1, 5, 5, 1, 1);
  mib_rtspInSrcQuenchs : TMibId = (1, 3, 6, 1, 2, 1, 5, 6, 1, 1);
  mib_rtspInRedirects : TMibId = (1, 3, 6, 1, 2, 1, 5, 7, 1, 1);
  mib_rtspInEchos : TMibId = (1, 3, 6, 1, 2, 1, 5, 8, 1, 1);
  mib_rtspInEchoReps : TMibId = (1, 3, 6, 1, 2, 1, 5, 9, 1, 1);
  mib_rtspInTimestamps : TMibId = (1, 3, 6, 1, 2, 1, 5, 10, 1, 1);
  mib_rtspInTimestampReps : TMibId = (1, 3, 6, 1, 2, 1, 5, 11, 1, 1);
  mib_rtspInAddrMasks : TMibId = (1, 3, 6, 1, 2, 1, 5, 12, 1, 1);
  mib_rtspInAddrMaskReps : TMibId = (1, 3, 6, 1, 2, 1, 5, 13, 1, 1);
  mib_rtspOutMsgs : TMibId = (1, 3, 6, 1, 2, 1, 5, 14, 1, 1);
  mib_rtspOutErrors : TMibId = (1, 3, 6, 1, 2, 1, 5, 15, 1, 1);
  mib_rtspOutDestUnreachs : TMibId = (1, 3, 6, 1, 2, 1, 5, 16, 1, 1);
  mib_rtspOutTimeExcds : TMibId = (1, 3, 6, 1, 2, 1, 5, 17, 1, 1);
  mib_rtspOutParmProbs : TMibId = (1, 3, 6, 1, 2, 1, 5, 18, 1, 1);
  mib_rtspOutSrcQuenchs : TMibId = (1, 3, 6, 1, 2, 1, 5, 19, 1, 1);

  // Константи статусів сокетів (Socket)

  CLOSED       = 1;
  LISTEN       = 2;
  SYN_SENT     = 3;
  SYN_RECEIVED = 4;
  ESTABLISHED  = 5;
  CLOSE_WAIT   = 6;
  FIN_WAIT_1   = 7;
  CLOSING      = 8;
  LAST_ACK     = 9;
  FIN_WAIT_2   = 10;
  TIME_WAIT    = 11;
  TCB_DISCARD  = 12;
type
  TMobileInitProc = function(
     dwTimeZeroReference : DWORD;
     Var hPollForTrapEvent : THandle;
     Var SupportedView : TAsnObjectIdentifier) : BOOL; stdcall;
  TMobileQueryProc = function(
     RequestType : Byte;
     Var VariableVindings : TRFC1157VarBindList;
     Var ErrorStatus : TAsnInteger;
     Var ErrorIndex : TAsnInteger) : BOOL; stdcall;

{ Реалізація TIpStats }

constructor TIpStats.Create( aOwner :TComponent );
//Конструктор
begin
```

```
   inherited;
   AddCounter( TVNstatCounterObject.Create( 'ipDefaultTTL', '',
               mib_ipDefaultTTL));
   AddCounter( TVNstatCounterObject.Create( 'ipInReceives', '',
               mib_ipInReceives));
   AddCounter( TVNstatCounterObject.Create( 'ipInHdrErrors', '',
               mib_ipInHdrErrors));
   AddCounter( TVNstatCounterObject.Create( 'ipInAddrErrors', '',
               mib_ipInAddrErrors));
   AddCounter( TVNstatCounterObject.Create( 'ipForwDatagrams', '',
               mib_ipForwDatagrams));
   AddCounter( TVNstatCounterObject.Create( 'ipInUnknownProtos', '',
               mib_InUnknownProtos));
   AddCounter( TVNstatCounterObject.Create( 'ipInDiscards', '',
               mib_ipInDiscards));
   AddCounter( TVNstatCounterObject.Create( 'ipInDelivers', '',
               mib_ipInDelivers));
   AddCounter( TVNstatCounterObject.Create( 'ipOutRequests', '',
               mib_ipOutRequests));
   AddCounter( TVNstatCounterObject.Create( 'ipOutDiscards', '',
               mib_ipOutDiscards));
   AddCounter( TVNstatCounterObject.Create( 'ipOutNoRoutes', '',
               mib_ipOutNoRoutes));

   AddCounter( TVNstatCounterObject.Create( 'ipReasmTimeout', '',
               mib_ipReasmTimeout));
   AddCounter( TVNstatCounterObject.Create( 'ipReasmReqds', '',
               mib_ipReasmReqds));
   AddCounter( TVNstatCounterObject.Create( 'ipReasmOKs', '',
                mib_ipReasmOKs));
   AddCounter( TVNstatCounterObject.Create( 'ipReasmFails', '',
                mib_ipReasmFails));
   AddCounter( TVNstatCounterObject.Create( 'ipFragOKs', '',
                mib_ipFragOKs));
   AddCounter( TVNstatCounterObject.Create( 'ipFragFails', '',
                mib_ipFragFails));
   AddCounter( TVNstatCounterObject.Create( 'ipFragCreates', '',
                mib_ipFragCreates));
end;

{ Реалізація TRtspStats }
constructor TRtspStats.Create( aOwner :TComponent ); //Конструктор
begin
   inherited;
AddCounter( TVNstatCounterObject.Create('rtspInMsgs','',mib_rtspInMsgs ));
AddCounter( TVNstatCounterObject.Create( 'rtspInErrors', '',
            mib_rtspInErrors ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInDestUnreachs', '',
mib_rtspInDestUnreachs ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInTimeExcds', '',
mib_rtspInTimeExcds ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInParmProbs', '',
mib_rtspInParmProbs ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInSrcQuenchs', '',
mib_rtspInSrcQuenchs ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInRedirects', '',
mib_rtspInRedirects ));
```

```
   AddCounter( TVNstatCounterObject.Create( 'rtspInEchos', '', mib_rtspInEchos
));
   AddCounter( TVNstatCounterObject.Create( 'rtspInEchoReps', '',
mib_rtspInEchoReps ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInTimestamps', '',
mib_rtspInTimestamps ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInTimestampReps', '',
mib_rtspInTimestampReps ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInAddrMasks', '',
mib_rtspInAddrMasks ));
   AddCounter( TVNstatCounterObject.Create( 'rtspInAddrMaskReps', '',
mib_rtspInAddrMaskReps ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutMsgs', '', mib_rtspOutMsgs
));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutErrors', '',
mib_rtspOutErrors ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutDestUnreachs', '',
mib_rtspOutDestUnreachs ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutTimeExcds', '',
mib_rtspOutTimeExcds ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutParmProbs', '',
mib_rtspOutParmProbs ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutSrcQuenchs', '',
mib_rtspOutSrcQuenchs ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutRedirects', '',
mib_rtspOutRedirects ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutEchos', '', mib_rtspOutEchos
));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutEchoReps', '',
mib_rtspOutEchoReps ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutTimestamps', '',
mib_rtspOutTimestamps ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutTimestampReps', '',
mib_rtspOutTimestampReps ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutAddrMasks', '',
mib_rtspOutAddrMasks ));
   AddCounter( TVNstatCounterObject.Create( 'rtspOutAddrMaskReps', '',
mib_rtspOutAddrMaskReps ));
end;


{ Реалізація TCP Stats }

constructor TTcpStats.Create( aOwner :TComponent ); //Конструктор
begin
   inherited;
  AddCounter( TVNstatCounterObject.Create( 'tcpRtoAlgorithm', '',
mib_tcpRtoAlgorithm));
  AddCounter( TVNstatCounterObject.Create( 'tcpRtoMin', '', mib_tcpRtoMin));
  AddCounter( TVNstatCounterObject.Create( 'tcpRtoMax', '', mib_tcpRtoMax));
  AddCounter( TVNstatCounterObject.Create( 'tcpMaxConn', '', mib_tcpMaxConn));
  AddCounter( TVNstatCounterObject.Create( 'tcpActiveOpens', '',
mib_tcpActiveOpens));
  AddCounter( TVNstatCounterObject.Create( 'tcpPassiveOpens', '',
mib_tcpPassiveOpens));
  AddCounter( TVNstatCounterObject.Create( 'tcpAttemptsFails', '',
mib_tcpAttemptsFails));
```

```
  AddCounter( TVNstatCounterObject.Create( 'tcpEstabResets', '',
mib_tcpEstabResets));
  AddCounter( TVNstatCounterObject.Create( 'tcpCurrEstab', '',
mib_tcpCurrEstab));
  AddCounter( TVNstatCounterObject.Create( 'tcpInSegs', '', mib_tcpInSegs));
  AddCounter( TVNstatCounterObject.Create( 'tcpOutSegs', '', mib_tcpOutSegs));
  AddCounter( TVNstatCounterObject.Create( 'tcpRetransSegs', '',
mib_tcpRetransSegs));
  AddCounter( TVNstatCounterObject.Create( 'tcpInErrs', '', mib_tcpInErrs));
  AddCounter( TVNstatCounterObject.Create( 'tcpOutRsts', '', mib_tcpOutRsts));
end;

constructor TUdpStats.Create( aOwner :TComponent ); //Конструктор
begin
   inherited;

  AddCounter( TVNstatCounterObject.Create( 'udpInDatagrams', '',
mib_udpInDatagrams ));
  AddCounter( TVNstatCounterObject.Create( 'udpNoPorts', '', mib_udpNoPorts ));
  AddCounter( TVNstatCounterObject.Create( 'udpInErrors', '', mib_udpInErrors
));
  AddCounter( TVNstatCounterObject.Create( 'udpOutDatagrams', '',
mib_udpOutDatagrams ));
end;

{ Реалізація TVNstatCounterObject }

function TVNstatCounterObject.GetValue :DWORD; // Реалізація функції
var // Об’ява змінних
   varBind     :TRFC1157VarBind;
   varBindList :TRFC1157VarBindList;
   errorStatus :TAsnInteger;
   errorIndex  :TAsnInteger;
begin
   varBindList.List := @varBind;
   varBindList.len := 1;
   fillchar(varBind, SizeOf(varBind), 0);
   varBind.Name.idLength := mibLen;
   varBind.Name.ids := @FId;
   if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
      varBindList, errorStatus, errorIndex) then
      raise ENetstatError.Create(m_err_query);
   if not (varBindList.list.value.asnType in
     [ASN_GAUGE32, ASN_INTEGER, ASN_INTEGER32, ASN_COUNTER32, ASN_UNSIGNED32])
then
      raise ENetstatError.Create(m_err_invtype);
   result := varBindList.list.value.Counter;
end;
//Конструктор
constructor TVNstatCounterObject.Create(
  const aName, aDesc :String; const aId :TMibId );
var
   WSData :TWSAData;
```

```
   PollForTrapEvent :THandle;
   SupportedViewRoot :TAsnObjectIdentifier;
begin
   inherited Create;
   FWsInited := WsaStartup($0101, WSData ) = 0;
   if not FWsInited then
      raise EConnListError.Create(m_err_wsstartup);
   FMobileLib := LoadLibrary(MOBILE_LIB_NAME);
   if FMobileLib = 0 then
      raise EConnListError.Create(m_err_loadlib);
   FMobileInitProc := GetProcAddress(FMobileLib, MOBILE_INITPROC_NAME);
   FMobileQueryProc := GetProcAddress(FMobileLib, MOBILE_QUERYPROC_NAME);
   if ( ( FMobileQueryProc = Nil ) or ( FMobileInitProc = Nil ) ) then
      raise EConnListError.Create(m_err_loadlib);
   if not TMobileInitProc(FMobileInitProc)(GetTickCount,
      PollForTrapEvent, SupportedViewRoot) then
      raise EConnListError.Create(m_err_initlib);
   FName := aname;
   FDescription := aDesc;
   FId := aid;
   dec(Fid[8]);
end;

destructor TVNstatCounterObject.Destroy; // Деструктор
begin
   if FMobileLib <> 0 then FreeLibrary(FMobileLib);
   if FWsInited then WSACleanup;
   inherited;
end;

{ Реалізація TCounterList }

procedure TCounterList.AddCounter( aCounter :TVNstatCounterObject );
// Реалізація процедури
begin
   assert( assigned( aCounter ));
   FCounters.AddObject( aCounter.Name, aCounter );
end;

function TCounterList.GetCount :Integer; // Реалізація функції
begin
   assert( assigned( FCounters ));
   result := FCounters.Count;
end;

function TCounterList.GetCounters( Index :Integer ):TVNstatCounterObject;
// Реалізація функції
begin
   assert( assigned( FCounters ));
   result := TVNstatCounterObject(FCounters.Objects[Index]);
end;
```

```
function TCounterList.IndexOf( const aName :String ):Integer;
// Реалізація функції
begin
   result := FCounters.IndexOf( aName );
end;

constructor TCounterList.Create( aOwner :TComponent ); //Конструктор
begin
   inherited;
   FCounters := TStringList.Create;
end;

destructor TCounterList.Destroy; // Деструктор
begin
   Clear;
   if assigned(FCounters) then FCounters.Free;
   inherited;
end;

procedure TCounterList.Clear; // Реалізація процедури
var // Об’ява змінних
   i :integer;
   c :TObject;
begin
   if assigned(FCounters) then
      for i := FCounters.Count-1 downto 0 do
         begin
            c := FCounters.Objects[i];
            FCounters.Delete(i);
            if assigned(c) then c.free;
         end;
end;

{ Реалізація TIpConNInfo }

constructor TIpConNInfo.Create( const aProto :String ); //Конструктор
begin
   inherited Create;
   FState := LISTEN;
   FProto := aProto;
end;

function TIpConNInfo.Ip2Str( const Ip :TInAddr ):String;
// Реалізація функції
begin
   result := format('%d.%d.%d.%d',
     [Integer(ip.s_un_b.s_b1),
      Integer(ip.s_un_b.s_b2),
      Integer(ip.s_un_b.s_b3),
      Integer(ip.s_un_b.s_b4)]);
```

```
end;

function TIpConNInfo.GetLocalPort :Integer; // Реалізація функції
begin
   result := FLocalPort;
end;

function TIpConNInfo.GetLocalIpString :String; // Реалізація функції
begin
   result := Ip2Str(FLocalIp);
end;

function TIpConNInfo.GetRemotePort :Integer; // Реалізація функції
begin
   if State = LISTEN then
      result := 0
   else
      result := FRemotePort;
end;

function TIpConNInfo.GetRemoteIpString :String; // Реалізація функції
begin
   result := Ip2Str(FRemoteIp);
end;

function TIpConNInfo.GetStateString :String; // Реалізація функції
const
   state_name :array[CLOSED..TCB_DISCARD] of string[16] =
   ('CLOSED',
    'LISTEN',
    'SYN_SENT',
    'SYN_RECEIVED',
    'ESTABLISHED',
    'CLOSE_WAIT',
    'FIN_WAIT_1',
    'CLOSING',
    'LAST_ACK',
    'FIN_WAIT_2',
    'TIME_WAIT',
    'TCB_DISCARD'
   );
begin
   if State in [CLOSED..TCB_DISCARD] then
      result := state_name[state]
   else
      result := 'UNKNOWN';
end;


{ Реалізація TFnugryIpConnectionList }
```

```
function TFnugryIpConnectionList.GetConnCount :Integer;
// Реалізація функції
begin
   assert(assigned(FConnections));
   result := FConnections.Count;
end;

function TFnugryIpConnectionList.GetConnections( Index :Integer ):TIpConnInfo;
// Реалізація функції
begin
   assert(assigned(FConnections));
   result := FConnections[Index];
end;

procedure TFnugryIpConnectionList.Clear; // Реалізація процедури
var // Об'ява змінних
   i :integer;
   p :TIpConnInfo;
begin
   if assigned(FConnections) then
      for i := FConnections.Count-1 downto 0 do
         begin
            p := FConnections[i];
            assert(assigned(p));
            FConnections.Delete(i);
            p.free;
         end;
end;

procedure TFnugryIpConnectionList.StatusChange; // Реалізація процедури
begin
   if assigned(FOnStatusChange) then FOnStatusChange(Self);
end;

procedure TFnugryIpConnectionList.SetStatus( Value :TIpConnListStatus );
// Реалізація процедури
begin
   if FStatus <> Value then
      begin
        FStatus := Value;
        StatusChange;
      end;
end;

procedure TFnugryIpConnectionList.DoLock; // Реалізація процедури
begin
   if not Lock(DEFAULT_LOCK_TIMEOUT) then
      raise ECOnnListLockError.Create(m_err_lock);
end;
procedure TFnugryIpConnectionList.DoUnlock;
// Реалізація процедури
```

```
begin
   if not UnLock then
      raise ECOnnListUnLockError.Create(m_err_unlock);
end;

constructor TFnugryIpConnectionList.Create( aOwner :TComponent );
// Конструктор
var // Об'ява змінних
   WSData :TWSAData;
begin
   inherited;
   FProtocols := [udp_ip, tcp_ip];
   FWsInited := WsaStartup($0101, WSData ) = 0;
   if not FWsInited then
      raise EConnListError.Create(m_err_wsstartup);
   FMobileLib := LoadLibrary(MOBILE_LIB_NAME);
   if FMobileLib = 0 then
      raise EConnListError.Create(m_err_loadlib);
   FMobileInitProc := GetProcAddress(FMobileLib, MOBILE_INITPROC_NAME);
   FMobileQueryProc := GetProcAddress(FMobileLib, MOBILE_QUERYPROC_NAME);
   if ( ( FMobileQueryProc = Nil ) or ( FMobileInitProc = Nil ) ) then
      raise EConnListError.Create(m_err_loadlib);
   if not TMobileInitProc(FMobileInitProc)(GetTickCount,
      FPollForTrapEvent, FSupportedViewRoot) then
      raise EConnListError.Create(m_err_initlib);
   FAccessMutex := CreateMutex(nil, false, nil);
   if FAccessMutex = 0 then
      raise ECOnnListError.Create(m_err_alloc);
   FCOnnections := TList.Create;
   FStatus := connlist_ready;
end;

destructor TFnugryIpConnectionList.Destroy; // Деструктор
begin
   Clear;
   if assigned(FConnections) then FCOnnections.Free;
   if FAccessMutex <> 0 then CloseHandle(FAccessMutex);
   if FMobileLib <> 0 then FreeLibrary(FMobileLib);
   if FWsInited then WSACleanup;
   inherited;
end;


function TFnugryIpConnectionList.Lock( Timeout :DWORD ):Bool;
// Реалізація функції
begin
   result := WaitForSingleObject(FAccessMutex, Timeout ) = WAIT_OBJECT_0;
end;

function TFnugryIpConnectionList.Unlock :Bool; // Реалізація функції
begin
```

```
  result := ReleaseMutex(FAccessMutex);
end;

procedure TFnugryIpConnectionList.Refresh;
// Реалізація процедури
   procedure ReadTcpTable;
   var // Об'ява змінних
      varBind     :TRFC1157VarBind;
      varBindList :TRFC1157VarBindList;
      errorStatus :TAsnInteger;
      errorIndex  :TAsnInteger;
      ConnIndex   :Integer;
      Info        :TIpConnInfo;
      ListTail    :Integer;
   begin
      fillchar( varBindList, SizeOf(varBindList), 0);
      varBindList.List := @varBind;
      varBindList.len := 1;
      fillchar(varBind, SizeOf(varBind), 0);
      varBind.Name.idLength := mibLen;
      varBind.name.ids := @mib_tcpConnTable;
      ListTail    := FConnections.Count;
      if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
         varBindList, errorStatus, errorIndex) then
         raise EConnListRefreshError.Create(m_err_query);
      if varBindList.list.value.asnType = ASN_NULL then
         exit;
      {читання з'єднань }
      while (varBindList.list.value.asnType = ASN_INTEGER) do
      begin
         Info := TIpConnInfo.Create('TCP');
         Info.FState := varBindList.list.value.Counter;
         FCOnnections.Add(Info);
         if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
            varBindList, errorStatus, errorIndex) then
            raise EConnListRefreshError.Create(m_err_query);
      end;
      if varBindList.list.value.asnType = ASN_NULL then
         raise EConnListRefreshError.Create(m_err_queryend);
      { читання значень ips }
      ConnIndex := ListTail;
      while (varBindList.list.value.asnType = ASN_RFC1155_IPADDRESS) do
      begin
         move(varBindList.list.value.address.stream^,
            Connections[ConnIndex].FLocalIP, sizeof(TInAddr));
         if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
            varBindList, errorStatus, errorIndex) then
            raise EConnListRefreshError.Create(m_err_query);
         inc(ConnIndex);
      end;
      if varBindList.list.value.asnType = ASN_NULL then
```

```
      raise EConnListRefreshError.Create(m_err_queryend);
   { читання значень локальних відео портів TCP/IP }
   ConnIndex := ListTail;

   while (varBindList.list.value.asnType = ASN_INTEGER) do
   begin
      Connections[ConnIndex].FLocalPort := varBindList.list.value.counter;
      if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
         varBindList, errorStatus, errorIndex) then
         raise EConnListRefreshError.Create(m_err_query);
      inc(ConnIndex);
   end;

   if varBindList.list.value.asnType = ASN_NULL then
      raise EConnListRefreshError.Create(m_err_queryend);
   { читання значень віддалених відео портів TCP/IP }
   ConnIndex := ListTail;
   while (varBindList.list.value.asnType = ASN_RFC1155_IPADDRESS) do
   begin
      move(varBindList.list.value.address.stream^,
         Connections[ConnIndex].FRemoteIP,
         sizeof(TInAddr));
      if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
         varBindList, errorStatus, errorIndex) then
         raise EConnListRefreshError.Create(m_err_query);
      inc(ConnIndex);
   end;

   if varBindList.list.value.asnType = ASN_NULL then
      raise EConnListRefreshError.Create(m_err_queryend);
  { читання значень віддалених відео портів TCP/IP }
   ConnIndex := ListTail;

   while (varBindList.list.value.asnType = ASN_INTEGER) do
   begin
      Connections[ConnIndex].FRemotePort := varBindList.list.value.counter;
      if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
         varBindList, errorStatus, errorIndex) then
         raise EConnListRefreshError.Create(m_err_query);
      inc(ConnIndex);
   end;
end;

procedure ReadUdpTable;
 var // Об'ява змінних
   varBind     :TRFC1157VarBind;
   varBindList :TRFC1157VarBindList;
   errorStatus :TAsnInteger;
   errorIndex  :TAsnInteger;
   ConnIndex   :Integer;
   Info        :TIpConnInfo;
```

```
      ListTail    :Integer;
   begin
      fillchar( varBindList, SizeOf(varBindList), 0);
      varBindList.List := @varBind;
      varBindList.len := 1;
      fillchar(varBind, SizeOf(varBind), 0);
      varBind.Name.idLength := mibLen;

      varBind.name.ids := @mib_udpTable;
      ListTail := FConnections.Count;
      if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
         varBindList, errorStatus, errorIndex) then
         raise EConnListRefreshError.Create(m_err_query);
      if varBindList.list.value.asnType = ASN_NULL then
         exit;
      { читання значень локальних відео портів TCP/IP }
      while (varBindList.list.value.asnType = ASN_RFC1155_IPADDRESS) do

      begin
         Info := TIpConnInfo.Create('UDP');
         move(varBindList.list.value.address.stream^,
            Info.FLocalIP,
            sizeof(TInAddr));
         FConnections.Add(Info);
         if not TMobileQueryProc(FMobileQueryProc)(ASN_RFC1157_GETNEXTREQUEST,
            varBindList, errorStatus, errorIndex) then
            raise EConnListRefreshError.Create(m_err_query);
      end;
      if varBindList.list.value.asnType = ASN_NULL then
         raise EConnListRefreshError.Create(m_err_queryend);
      { отримання даних }
      ConnIndex := ListTail;
      while (varBindList.list.value.asnType = ASN_INTEGER) do
      begin
         Connections[ConnIndex].FLocalPort := varBindList.list.value.counter;
         if not TMobileQueryProc(FMobileQueryProc)
                (ASN_RFC1157_GETNEXTREQUEST, varBindList, errorStatus,
                errorIndex)
            then raise EConnListRefreshError.Create(m_err_query);
         inc(ConnIndex);

      end;
  end;

begin
   DoLock;
   try
      if FStatus <> connlist_ready then
         raise EConnListError.Create(m_err_inv_state);
      SetStatus( connlist_refreshing );
      Clear;
```

```
      if tcp_ip in FProtocols then
         ReadTcpTable;
      if udp_ip in FProtocols then

         ReadUdpTable;
   finally
      DoUnlock;
      SetStatus(connlist_ready);

   end;
end;

end.
```

ФАЙЛ АВТОРСЬКОГО ПРАВА

```
unit About; // Назва модулю
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Виконав: Чернат Єгор Сергійович, КБ-20
Керівник: Смірнов О.А.
}

interface
{Інтерфейсна частина, тут проходе опис класів та типів}

Uses
// Підключення бібліотек
  WinTypes, WinProcs, Classes, Graphics, Forms, Controls,
  StdCtrls, ExtCtrls, Buttons, DsgnIntf, MgCommon, ShellAPI;

const
  mgcrHand = 5;

type
  TMgAboutBoxProperty = class( TPropertyEditor ) // Об'ява класу
  public
    procedure Edit; override;
    function GetAttributes : TPropertyAttributes; override;
    function GetValue : string; override;
  end;



  TMgAboutEditDlg = class( TForm )
 // Об'ява класу
    BtnOk: TButton;
    Panel1 : TPanel;
    LblCopyright: TLabel;
    BtnPaletteBmp: TSpeedButton;
    LblDescription: TLabel;
    LblCompany: TLabel;
    Bevel1: TBevel;
    LblComponentName: TLabel;
    lblURL: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure lblURLClick(Sender: TObject);
    procedure lblURLMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lblURLMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
// секція бачення
    FAboutInfo : TMgAboutInfo;
    procedure SetAboutInfo( Value : TMgAboutInfo );
  public
```

```
    ComponentName : string;
    property AboutInfo : TMgAboutInfo read FAboutInfo write SetAboutInfo;
  end;

implementation
{Секція де проходить реалізація того що описано в інтерфейс ній частині}

{$R MgCursor.res}
{$R *.DFM}

Uses
// Підключення бібліотек
  SysUtils;

{ Реалізація класу TRzAboutBoxProperty     }

function TMgAboutBoxProperty.GetAttributes : TPropertyAttributes;
// Реалізація функції
begin
  Result := [ paDialog, paReadOnly ];
end;

function TMgAboutBoxProperty.GetValue : string; // Реалізація функції
begin
  Result := 'Повернення';
end;

procedure TMgAboutBoxProperty.Edit; // Реалізація процедури
var // Об’ява змінних
  ComponentRef : TComponent;
  Dialog : TMgAboutEditDlg;
begin
  Dialog := TMgAboutEditDlg.Create( Application );
  try
    ComponentRef := GetComponent( 0 );
{ Отримання назви класу з компоненту }
    Dialog.ComponentName := ComponentRef.ClassName;
    Dialog.AboutInfo := TMgAboutInfo( GetOrdValue );
    Dialog.ShowModal;
  finally
    Dialog.Free;
  end;
end;

{Реалізація класу TMgAboutEditDlg}

procedure TMgAboutEditDlg.SetAboutInfo( Value : TMgAboutInfo );
// Реалізація процедури
var // Об’ява змінних
  BmpName : array[ 0..128 ] of Char;
begin
```

```
  if ComponentName[ 1 ] = 'T' then
    LblComponentName.Caption := Copy( ComponentName, 2, 255 )
  else
    LblComponentName.Caption := ComponentName;

  LblCopyright.Caption := 'Copyright © '+ Value.CopyrightDate;
  LblCompany.Caption := Value.Company;
  LblDescription.Caption := Value.Description;
  LblURL.Caption := Value.URL;
  { Завантаження зображення }
  StrPCopy( BmpName, UpperCase( ComponentName ) );
  BtnPaletteBmp.Glyph.Handle := LoadBitmap( HInstance, BmpName );
end;

procedure TMgAboutEditDlg.FormCreate(Sender: TObject); // Реалізація процедури
begin
   Screen.Cursors[ mgcrHand ] := LoadCursor( HInstance, PChar( 'MGHAND' ) );
end;

procedure TMgAboutEditDlg.lblURLClick(Sender: TObject); // Реалізація процедури
var // Об’ява змінних
  tempStr : array[0..255] of Char;
begin
  ShellExecute( Application.Handle, 'open', StrPCopy(tempStr, lblURL.Caption),
                nil, nil, SW_NORMAL );
end;

procedure TMgAboutEditDlg.lblURLMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
// Реалізація процедури
begin
   if Button = mbLeft then begin
      Font.Color := clPurple;
   end;
end;

procedure TMgAboutEditDlg.lblURLMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
// Реалізація процедури
begin
  if Button = mbLeft then Font.Color := clBlue;
end;

end.
```

ФАЙЛ БІБЛІОТЕКИ ВІДЕО VIDEO_PACK

```
unit Video_Pack; // Бібліотека відео
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Виконав: Чернат Єгор Сергійович, КБ-20
Керівник: Смірнов О.А.
}

interface // Інтерфейсна частина

uses // Які бібліотекі використовуємо у  модулі
  Windows, Classes, SysUtils, Messages, Graphics, Forms, Controls, ActiveX,
  DirectShow9, DirectDraw, DSUtil, ComCtrls, MMSystem, Math, Consts, ExtCtrls,
  MultiMon, Dialogs, Registry, SyncObjs, Direct3D9, WMF9;

Const // Константи
  WM_GRAPHNOTIFY = WM_APP + 1;

  WM_CAPTURE_BITMAP = WM_APP + 2;
type
// Об’ява типів
  TVideoMode = ( // Тип режиму
    vmNormal,
    vmVMR
  );

  TGraphMode = ( // Якості
    gmNormal,
    gmCapture,
    gmCAM
  );


{$IFDEF VER140}
  TVMRRenderDevice = (
    rdOverlay = 1,
    rdVidMem  = 2,
    rdSysMem  = 4
  );
{$ELSE}
  TVMRRenderDevice = Integer;
  const
    rdOverlay = 1;
    rdVidMem  = 2;
    rdSysMem  = 4;
type
{$ENDIF}

  {@exclude}
  TGraphState = ( // Поточне дія
```

```
    gsUninitialized,
    gsStopped,
    gsPaused,
    gsPlaying
  );

  { Структура відео потоку }

  // Абсолютна позиція у потоці
  TSeekingCap = (
    CanSeekAbsolute,
    CanSeekForwards,
    CanSeekBackwards,
    CanGetCurrentPos,
    CanGetStopPos,
    CanGetDuration,
    CanPlayBackwards,
    CanDoSegments,
    IMediaSeeking.SetPositions).Source);
  TSeekingCaps = set of TSeekingCap;

  TVMRPreference = (
    vpForceOffscreen,
    vpForceOverlays,
    vpForceMixer,
    vpDoNotRenderColorKeyAndBorder,
    vpRestrictToInitialMonitor,
    vpPreferAGPMemWhenMixing
  );

  PVMRPreferences = ^TVMRPreferences;
  TVMRPreferences = set of TVMRPreference;

TOnDSEvent=procedure(sender: TComponent; Event, Param1, Param2: Integer) of
                    object;
TOnGraphBufferingData=procedure(sender: TObject; Buffering: boolean) of object ;
TOnGraphComplete=procedure(sender: TObject; Result: HRESULT; Renderer:
IBaseFilter) of object;
  TOnGraphDeviceLost           = procedure(sender: TObject; Device: IUnknown;
                                  Removed: Boolean) of object ;
  TOnGraphEndOfSegment         = procedure(sender: TObject; StreamTime:
TReferenceTime; NumSegment: Cardinal) of object ;
  TOnDSResult                  = procedure(sender: TObject; Result: HRESULT) of
object ;
  TOnGraphFullscreenLost       = procedure(sender: TObject; Renderer:
IBaseFilter) of object ;
  TOnGraphOleEvent             = procedure(sender: TObject; String1, String2:
WideString) of object ;
  TOnGraphOpeningFile          = procedure(sender: TObject; opening: boolean) of
                                object ;
  TOnGraphSNDDevError          = procedure(sender: TObject; OccurWhen:
TSndDevErr; ErrorCode: LongWord) of object ;
  TOnGraphStreamControl        = procedure(sender: TObject; PinSender: IPin;
                                  Cookie: LongWord) of object ;
```

```
  TOnGraphStreamError          = procedure(sender: TObject; Operation: HRESULT;
                                 Value: LongWord) of object ;
  TOnGraphVideoSizeChanged     = procedure(sender: TObject; Width, height: word)
                                 of object ;
  TOnGraphTimeCodeAvailable= procedure(sender: TObject; From: IBaseFilter;
                                 DeviceID: LongWord) of object ;
  TOnGraphEXTDeviceModeChange  = procedure(sender: TObject; NewMode, DeviceID:
                                 LongWord) of object ;
  TOnGraphVMRRenderDevice= procedure(sender: TObject; RenderDevice:
TVMRRenderDevice) of object;

TOnCAMDataStreamChange      = procedure(sender: TObject; stream, lcid: Integer;
                              Lang: string) of object;
  TOnCAMCurrentTime         = procedure(sender: TObject; Hours,
                              minutes,seconds,frames,frate : Integer) of object;
  TOnCAMTitleChange = procedure(sender: TObject; title: Integer) of object;
TOnCAMChapterStart=procedure(sender: TObject; chapter: Integer) of object;
TOnCAMValidUOPSChange = procedure(sender: TObject; UOPS: Integer) of object;
TOnCAMChange = procedure(sender: TObject; total,current: Integer) of object;
TOnCAMStillOn = procedure(sender: TObject; NoButtonAvailable: boolean; seconds:
                          Integer) of object;
TOnCAMSubpictureStreamChange = procedure(sender: TObject; SubNum, lcid: Integer;
Lang: string) of object;
TOnCAMPlaybackRateChange = procedure(sender: TObject; rate:  single) of object;
TOnCAMParentalLevelChange= procedure(sender: TObject; level: Integer) of object;
TOnCAMAnglesAvailable=procedure(sender: TObject; available: boolean) of object;
TOnCAMButtonAutoActivated=procedure(sender: TObject; Button: Cardinal) of
object;
TOnCAMCMD=procedure(sender: TObject; CmdID: Cardinal) of object;
TOnCAMCurrentHMSFTime = procedure(sender: TObject; HMSFTimeCode:
                            TCAMHMSFTimeCode; TimeCode: TCAMTimeCode) of object;
TOnCAMKaraokeMode=procedure(sender: TObject; Played: boolean) of object;
TOnBuffer = procedure(sender: TObject; SampleTime: Double; pBuffer: Pointer;
BufferLen: longint) of object ;

//  Структура TFilterOperation

  TFilterOperation = (
    foAdding,
    foAdded,
    foRemoving,
    foRemoved,
    foRefresh
  );

  IFilter = interface // Об’ява інтерфейсу
  ['{887F94DA-29E9-44C6-B48E-1FBF0FB59878}']
    function GetFilter: IBaseFilter;
    function GetName: string;
    procedure NotifyFilter(operation: TFilterOperation; Param: integer = 0);
  end;

  TControlEvent = (
    cePlay,
    cePause,
    ceStop,
```

```
    ceFileRendering,
    ceFileRendered,
    ceCAMRendering,
    ceCAMRendered,
    ceActive
  );

  IEvent = interface // Інтерфейс
  ['{6C0DCD7B-1A98-44EF-A6D5-E23CBC24E620}']
    procedure GraphEvent(Event, Param1, Param2: integer);
    procedure ControlEvent(Event: TControlEvent; Param: integer = 0);
  end;

//  Структура TFilterGraph
  TFilterGraph = class(TComponent) // Оголошення класу
  private
    FActive     : boolean;
    FAutoCreate : boolean;
    FHandle     : THandle; // основний вказівник
    FMode       : TGraphMode;

    FFilters: TInterfaceList;
    FGraphEvents: TInterfaceList;

    // налаштування
    FFilterGraph  : IGraphBuilder;
    FCaptureGraph : ICaptureGraphBuilder2;
    FCAMGraph     : ICamGraphBuilder;
    FMediaEventEx : IMediaEventEx;
    FGraphEdit    : boolean;
    FGraphEditID  : Integer;

    // Файл Log
    FLogFileName: String;
    FLogFile: TFileStream;
    FOnActivate: TNotifyEvent;
    // All Events Code
    FOnDSEvent : TOnDSEvent;
    // Generic Graph Events
    FOnGraphBufferingData           : TOnGraphBufferingData;
    FOnGraphClockChanged            : TNotifyEvent;
    FOnGraphComplete                : TOnGraphComplete;
    FOnGraphDeviceLost              : TOnGraphDeviceLost;
    FOnGraphEndOfSegment            : TOnGraphEndOfSegment;
    FOnGraphErrorStillPlaying       : TOnDSResult;
    FOnGraphErrorAbort              : TOnDSResult;
    FOnGraphFullscreenLost          : TOnGraphFullscreenLost;
    FOnGraphChanged                 : TNotifyEvent;
    FOnGraphOleEvent                : TOnGraphOleEvent;
    FOnGraphOpeningFile             : TOnGraphOpeningFile;
    FOnGraphPaletteChanged          : TNotifyEvent;
```

```
FOnGraphPaused                   : TOnDSResult;
FOnGraphQualityChange            : TNotifyEvent;
FOnGraphSNDDevInError            : TOnGraphSNDDevError;
FOnGraphSNDDevOutError           : TOnGraphSNDDevError;
FOnGraphStepComplete             : TNotifyEvent;
FOnGraphStreamControlStarted     : TOnGraphStreamControl;
FOnGraphStreamControlStopped     : TOnGraphStreamControl;
FOnGraphStreamErrorStillPlaying : TOnGraphStreamError;
FOnGraphStreamErrorStopped       : TOnGraphStreamError;
FOnGraphUserAbort                : TNotifyEvent;
FOnGraphVideoSizeChanged         : TOnGraphVideoSizeChanged;
FOnGraphTimeCodeAvailable        : TOnGraphTimeCodeAvailable;
FOnGraphEXTDeviceModeChange      : TOnGraphEXTDeviceModeChange;
FOnGraphClockUnset               : TNotifyEvent;
FOnGraphVMRRenderDevice          : TOnGraphVMRRenderDevice;

FOnCAMDataStreamChange       : TOnCAMDataStreamChange;
FOnCAMCurrentTime             : TOnCAMCurrentTime;
FOnCAMTitleChange             : TOnCAMTitleChange;
FOnCAMChapterStart            : TOnCAMChapterStart;
FOnCAMAngleChange             : TOnCAMChange;
FOnCAMValidUOPSChange         : TOnCAMValidUOPSChange;
FOnCAMButtonChange            : TOnCAMChange;
FOnCAMChapterAutoStop         : TNotifyEvent;
FOnCAMStillOn                 : TOnCAMStillOn;
FOnCAMStillOff                : TNotifyEvent;
FOnCAMSubpictureStreamChange  : TOnCAMSubpictureStreamChange;
FOnCAMNoFP_PGC                : TNotifyEvent;
FOnCAMPlaybackRateChange      : TOnCAMPlaybackRateChange;
FOnCAMParentalLevelChange     : TOnCAMParentalLevelChange;
FOnCAMPlaybackStopped         : TNotifyEvent;
FOnCAMAnglesAvailable         : TOnCAMAnglesAvailable;
FOnCAMPlayPeriodAutoStop      : TNotifyEvent;
FOnCAMButtonAutoActivated     : TOnCAMButtonAutoActivated;
FOnCAMCMDStart                : TOnCAMCMD;
FOnCAMCMDEnd                  : TOnCAMCMD;
FOnCAMDiscEjected             : TNotifyEvent;
FOnCAMDiscInserted            : TNotifyEvent;
FOnCAMCurrentHMSFTime         : TOnCAMCurrentHMSFTime;
FOnCAMKaraokeMode             : TOnCAMKaraokeMode;
FOnCAMWarningInvalidCAM1_0Disc  : TNotifyEvent;//=1,
FOnCAMWarningFormatNotSupported : TNotifyEvent;//=2,
FOnCAMWarningIllegalNavCommand  : TNotifyEvent;//=3
FOnCAMWarningOpen               : TNotifyEvent;//=4
FOnCAMWarningSeek               : TNotifyEvent;//=5
FOnCAMWarningRead               : TNotifyEvent;//=6
FOnCAMDomainFirstPlay         : TNotifyEvent;
FOnCAMDomainVideoManagerMenu  : TNotifyEvent;
FOnCAMDomainVideoTitleSetMenu : TNotifyEvent;
FOnCAMDomainTitle             : TNotifyEvent;
FOnCAMDomainStop              : TNotifyEvent;
```

```
    FOnCAMErrorUnexpected                          : TNotifyEvent;
    FOnCAMErrorCopyProtectFail                     : TNotifyEvent;
    FOnCAMErrorInvalidCAM1_0Disc                   : TNotifyEvent;
    FOnCAMErrorInvalidDiscRegion                   : TNotifyEvent;
    FOnCAMErrorLowParentalLevel                    : TNotifyEvent;
    FOnCAMErrorMacrovisionFail                     : TNotifyEvent;
    FOnCAMErrorIncompatibleSystemAndDecoderRegions : TNotifyEvent;
    FOnCAMErrorIncompatibleDiscAndDecoderRegions   : TNotifyEvent;
    procedure HandleEvents;
    procedure WndProc(var Msg: TMessage);
    procedure SetActive(Activate: boolean);
    procedure SetGraphMode(Mode: TGraphMode);
    procedure SetGraphEdit(enable: boolean);
    procedure ClearOwnFilters;
    procedure AddOwnFilters;
    procedure GraphEvents(Event, Param1, Param2: integer);
    procedure ControlEvents(Event: TControlEvent; Param: integer = 0);
    procedure SetLogFile(FileName: String);
    function GetState: TGraphState;
    function GetVolume: integer;
    procedure SetVolume(Volume: Integer);
    function GetBalance: integer;
    procedure SetBalance(Balance: integer);
    function GetSeekCaps: TSeekingCaps;
    procedure SetRate(Rate: double);
    function GetRate: double;
    function GetDuration: integer;
  protected
    procedure DoEvent(Event, Param1, Param2: Integer); virtual;
    procedure InsertFilter(AFilter: IFilter);
    procedure RemoveFilter(AFilter: IFilter);
    procedure InsertEventNotifier(AEvent: IEvent);
    procedure RemoveEventNotifier(AEvent: IEvent);
  public //зона бачення змінних та типів даних
    property Duration: Integer read GetDuration; // завдання властивості типу
    property Rate: Double read GetRate write SetRate;
    property SeekCapabilities: TSeekingCaps read GetSeekCaps;
    property Balance: integer read GetBalance write SetBalance;
    property Volume: integer read GetVolume write SetVolume;
    property State: TGraphState read GetState;
    constructor Create(AOwner: TComponent); override; //Конструктор
    destructor Destroy; override; //Деструктор
    procedure Loaded; override;
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
    function Play: boolean;
    function Pause: boolean;
    function Stop: boolean;
    procedure DisconnectFilters;
    procedure ClearGraph;
    function RenderFile(FileName: WideString): HRESULT;
```

```
    function RenderFileEx(FileName: WideString): HRESULT;
    function RenderCAM(out status: TAMCAMRenderStatus;
    FileName: WideString = ''; Mode: Integer = AM_CAM_HWDEC_PREFER): HRESULT;

    procedure CAMSaveBookmark(BookMarkFile: WideString);
    procedure CAMRestoreBookmark(BookMarkFile: WideString);
  published //зона бачення змінних
    property LogFile: String read FLogFileName write SetLogFile;
    property Active: boolean read FActive write SetActive default False;
    property AutoCreate: boolean read FAutoCreate write FAutoCreate default
False;
    property Mode: TGraphMode read FMode write SetGraphMode default gmNormal;
    property GraphEdit: boolean read FGraphEdit write SetGraphEdit;

// ---------------------------------------------------------------------------
// Події
// ---------------------------------------------------------------------------

    property OnActivate: TNotifyEvent read FOnActivate write FOnActivate;
    property OnDSEvent: TOnDSEvent read FOnDSEvent write FOnDSEvent;

    property OnGraphBufferingData: TOnGraphBufferingData read
             FOnGraphBufferingData write FOnGraphBufferingData;
    property OnGraphClockChanged: TNotifyEvent read FOnGraphClockChanged write
FOnGraphClockChanged;
    property OnGraphComplete: TOnGraphComplete read FOnGraphComplete write
FOnGraphComplete;

    property OnGraphDeviceLost: TOnGraphDeviceLost read FOnGraphDeviceLost write
FOnGraphDeviceLost;

    property OnGraphEndOfSegment: TOnGraphEndOfSegment read FOnGraphEndOfSegment
write FOnGraphEndOfSegment;
    property OnGraphErrorStillPlaying: TOnDSResult read
FOnGraphErrorStillPlaying write FOnGraphErrorStillPlaying;

    property OnGraphErrorAbort: TOnDSResult read FOnGraphErrorAbort write
FOnGraphErrorAbort;

    property OnGraphFullscreenLost: TOnGraphFullscreenLost read
FOnGraphFullscreenLost write FOnGraphFullscreenLost;

    property OnGraphChanged: TNotifyEvent read FOnGraphChanged write
FOnGraphChanged;

    property OnGraphOleEvent: TOnGraphOleEvent read FOnGraphOleEvent write
FOnGraphOleEvent;

    property OnGraphOpeningFile: TOnGraphOpeningFile read FOnGraphOpeningFile
write FOnGraphOpeningFile;

    property OnGraphPaletteChanged: TNotifyEvent read FOnGraphPaletteChanged
write FOnGraphPaletteChanged;
```

```
    property OnGraphPaused: TOnDSResult read FOnGraphPaused write
FOnGraphPaused;


    property OnGraphQualityChange: TNotifyEvent read FOnGraphQualityChange write
FOnGraphQualityChange;
    property OnGraphSNDDevInError: TOnGraphSNDDevError read
FOnGraphSNDDevInError write FOnGraphSNDDevInError;
    property OnGraphSNDDevOutError: TOnGraphSNDDevError read
FOnGraphSNDDevOutError write FOnGraphSNDDevOutError;
    property OnGraphStepComplete: TNotifyEvent read FOnGraphStepComplete write
FOnGraphStepComplete;
    property OnGraphStreamControlStarted: TOnGraphStreamControl read
FOnGraphStreamControlStarted  write FOnGraphStreamControlStarted;
    property OnGraphStreamControlStopped: TOnGraphStreamControl read
FOnGraphStreamControlStopped write FOnGraphStreamControlStopped;
    property OnGraphStreamErrorStillPlaying : TOnGraphStreamError read
FOnGraphStreamErrorStillPlaying write FOnGraphStreamErrorStillPlaying;


    property OnGraphStreamErrorStopped: TOnGraphStreamError read
FOnGraphStreamErrorStopped write FOnGraphStreamErrorStopped;
    property OnGraphUserAbort: TNotifyEvent read FOnGraphUserAbort write
FOnGraphUserAbort;
    property OnGraphVideoSizeChanged: TOnGraphVideoSizeChanged read
FOnGraphVideoSizeChanged write FOnGraphVideoSizeChanged;
    property OnGraphTimeCodeAvailable: TOnGraphTimeCodeAvailable read
FOnGraphTimeCodeAvailable write FOnGraphTimeCodeAvailable;
    property OnGraphEXTDeviceModeChange: TOnGraphEXTDeviceModeChange read
FOnGraphEXTDeviceModeChange write FOnGraphEXTDeviceModeChange;
    property OnGraphClockUnset: TNotifyEvent read FOnGraphClockUnset write
FOnGraphClockUnset;
    property OnGraphVMRRenderDevice: TOnGraphVMRRenderDevice read
FOnGraphVMRRenderDevice write FOnGraphVMRRenderDevice;
    property OnCAMDataStreamChange: TOnCAMDataStreamChange read
FOnCAMDataStreamChange write FOnCAMDataStreamChange;
    property OnCAMCurrentTime: TOnCAMCurrentTime read FOnCAMCurrentTime write
                               FOnCAMCurrentTime;
    property OnCAMTitleChange: TOnCAMTitleChange read FOnCAMTitleChange write
FOnCAMTitleChange;
    property OnCAMChapterStart: TOnCAMChapterStart read FOnCAMChapterStart write
FOnCAMChapterStart;
    property OnCAMAngleChange: TOnCAMChange read FOnCAMAngleChange write
FOnCAMAngleChange;
    property OnCAMValidUOPSChange: TOnCAMValidUOPSChange read
FOnCAMValidUOPSChange write FOnCAMValidUOPSChange;
    property OnCAMButtonChange: TOnCAMChange read FOnCAMButtonChange write
FOnCAMButtonChange;
    property OnCAMChapterAutoStop: TNotifyEvent read FOnCAMChapterAutoStop write
FOnCAMChapterAutoStop;
    property OnCAMStillOn: TOnCAMStillOn read FOnCAMStillOn write FOnCAMStillOn;
    property OnCAMStillOff: TNotifyEvent read FOnCAMStillOff write
FOnCAMStillOff;
    property OnCAMSubpictureStreamChange: TOnCAMSubpictureStreamChange read
FOnCAMSubpictureStreamChange write FOnCAMSubpictureStreamChange;
    property OnCAMNoFP_PGC: TNotifyEvent read FOnCAMNoFP_PGC write
FOnCAMNoFP_PGC;
    property OnCAMPlaybackRateChange: TOnCAMPlaybackRateChange read
FOnCAMPlaybackRateChange write FOnCAMPlaybackRateChange;
```

```
    property OnCAMParentalLevelChange: TOnCAMParentalLevelChange read
FOnCAMParentalLevelChange write FOnCAMParentalLevelChange;

    property OnCAMPlaybackStopped: TNotifyEvent read FOnCAMPlaybackStopped write
FOnCAMPlaybackStopped;

    property OnCAMAnglesAvailable: TOnCAMAnglesAvailable read
FOnCAMAnglesAvailable write FOnCAMAnglesAvailable;

    property OnCAMPlayPeriodAutoStop: TNotifyEvent read FOnCAMPlayPeriodAutoStop
write FOnCAMPlayPeriodAutoStop;

    property OnCAMButtonAutoActivated: TOnCAMButtonAutoActivated read
FOnCAMButtonAutoActivated write FOnCAMButtonAutoActivated;

    property OnCAMCMDStart: TOnCAMCMD read FOnCAMCMDStart Write FOnCAMCMDStart;

    property OnCAMCMDEnd: TOnCAMCMD read FOnCAMCMDEnd Write FOnCAMCMDEnd;

    property OnCAMDiscEjected: TNotifyEvent read FOnCAMDiscEjected Write
FOnCAMDiscEjected;

    property OnCAMDiscInserted: TNotifyEvent read FOnCAMDiscInserted write
FOnCAMDiscInserted;

    property OnCAMCurrentHMSFTime: TOnCAMCurrentHMSFTime read
FOnCAMCurrentHMSFTime write FOnCAMCurrentHMSFTime;

    property OnCAMKaraokeMode: TOnCAMKaraokeMode read FOnCAMKaraokeMode write
FOnCAMKaraokeMode;

    property OnCAMDomainFirstPlay: TNotifyEvent read FOnCAMDomainFirstPlay write
FOnCAMDomainFirstPlay;

    property OnCAMDomainVideoManagerMenu: TNotifyEvent read
FOnCAMDomainVideoManagerMenu write FOnCAMDomainVideoManagerMenu;

    property OnCAMDomainVideoTitleSetMenu: TNotifyEvent read
FOnCAMDomainVideoTitleSetMenu write FOnCAMDomainVideoTitleSetMenu;

    property OnCAMDomainTitle: TNotifyEvent read FOnCAMDomainTitle write
FOnCAMDomainTitle;

    property OnCAMDomainStop: TNotifyEvent read FOnCAMDomainStop write
FOnCAMDomainStop;

    property OnCAMErrorUnexpected: TNotifyEvent read FOnCAMErrorUnexpected write
FOnCAMErrorUnexpected;

    property OnCAMErrorInvalidCAM1_0Disc: TNotifyEvent read
FOnCAMErrorInvalidCAM1_0Disc write FOnCAMErrorInvalidCAM1_0Disc;


    property OnCAMErrorInvalidDiscRegion: TNotifyEvent read
FOnCAMErrorInvalidDiscRegion write FOnCAMErrorInvalidDiscRegion;

    property OnCAMErrorLowParentalLevel: TNotifyEvent read
FOnCAMErrorLowParentalLevel write FOnCAMErrorLowParentalLevel;


    property OnCAMErrorMacrovisionFail: TNotifyEvent read
FOnCAMErrorMacrovisionFail write FOnCAMErrorMacrovisionFail;

property OnCAMErrorIncompatibleSystemAndDecoderRegions: TNotifyEvent read
FOnCAMErrorIncompatibleSystemAndDecoderRegions write
FOnCAMErrorIncompatibleSystemAndDecoderRegions;

property OnCAMErrorIncompatibleDiscAndDecoderRegions: TNotifyEvent read
FOnCAMErrorIncompatibleDiscAndDecoderRegions  write
FOnCAMErrorIncompatibleDiscAndDecoderRegions;

    property OnCAMWarningInvalidCAM1_0Disc: TNotifyEvent read
FOnCAMWarningInvalidCAM1_0Disc write FOnCAMWarningInvalidCAM1_0Disc;

    property OnCAMWarningFormatNotSupported : TNotifyEvent read
FOnCAMWarningFormatNotSupported write FOnCAMWarningFormatNotSupported;

    property OnCAMWarningIllegalNavCommand  : TNotifyEvent read
FOnCAMWarningIllegalNavCommand  write FOnCAMWarningIllegalNavCommand;

    property OnCAMWarningOpen: TNotifyEvent read FOnCAMWarningOpen write
FOnCAMWarningOpen;

    property OnCAMWarningSeek: TNotifyEvent read FOnCAMWarningSeek write
FOnCAMWarningSeek;
```

```
    property OnCAMWarningRead: TNotifyEvent read FOnCAMWarningRead write
FOnCAMWarningRead;
  end;

//   Структура TVMROptions

  TVideoWindow = class;
// Оголошення класу

  TVMRVideoMode = (
    vmrWindowed,
    vmrWindowless,
    vmrRenderless
  );

  TVMROptions = class(TPersistent)
// Оголошення класу
  private
    FOwner: TVideoWindow;
    FStreams: cardinal;
    FPreferences: TVMRPreferences;
    FMode: TVMRVideoMode;
    FKeepAspectRatio: boolean;
    procedure SetStreams(Streams: cardinal);
    procedure SetPreferences(Preferences: TVMRPreferences);
    procedure SetMode(AMode: TVMRVideoMode);
    procedure SetKeepAspectRatio(Keep: boolean);
  public //зона бачення змінних та типів даних
    constructor Create(AOwner: TVideoWindow); //Конструктор
  published //зона бачення змінних та типів даних
    property Mode: TVMRVideoMode read FMode write SetMode;
    // завдання властивості типу
    property Streams: Cardinal read FStreams write SetStreams default 4;
    property Preferences: TVMRPreferences read FPreferences write SetPreferences
default [vpForceMixer];
    property KeepAspectRatio: boolean read FKeepAspectRatio write
SetKeepAspectRatio default True;
  end;

// ****************************************************************************
//   структура TVideoWindow
// ****************************************************************************

  TAbstractAllocator = class(TInterfacedObject) // Оголошення класу
    constructor Create(out hr: HResult; wnd: THandle; d3d: IDirect3D9 = nil;
d3dd: IDirect3DDevice9 = nil); virtual; abstract;
  end;
  TAbstractAllocatorClass = class of TAbstractAllocator; // Оголошення класу
  TVideoWindow = class(TCustomControl, IFilter, IEvent) // Оголошення класу
  private
    FMode          : TVideoMode;
    FVMROptions    : TVMROptions;
```

```
    FBaseFilter    : IBaseFilter;
    FVideoWindow   : IVideoWindow;
    FWindowLess    : IVMRWindowlessControl9;
    FFullScreen    : boolean;
    FFilterGraph   : TFilterGraph;
    FWindowStyle   : LongWord;
    FWindowStyleEx : LongWord;
    FTopMost       : boolean;
    FIsFullScreen  : boolean;
    FOnPaint       : TNotifyEvent;
    FKeepAspectRatio: boolean;
    FAllocatorClass: TAbstractAllocatorClass;
    FCurrentAllocator: TAbstractAllocator;
    FRenderLessUserID: Cardinal;
    procedure SetVideoMode(AMode: TVideoMode);
    procedure SetFilterGraph(AFilterGraph: TFilterGraph);
    procedure SetFullScreen(Value: boolean);
    procedure NotifyFilter(operation: TFilterOperation; Param: integer = 0);
    procedure GraphEvent(Event, Param1, Param2: integer);
    function GetName: string;
    function GetVideoHandle: THandle;
    procedure ControlEvent(Event: TControlEvent; Param: integer = 0);
    procedure SetTopMost(TopMost: boolean);
    function GetVisible: boolean;
    procedure SetVisible(Vis: boolean);
  protected
    {@exclude}
    procedure Loaded; override;
    {@exclude}
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
    {@exclude}
    procedure Resize; override;
    {@exclude}
    procedure ConstrainedResize(var MinWidth, MinHeight, MaxWidth, MaxHeight:
Integer); override;
    {@exclude}
    function GetFilter: IBaseFilter;
    {@exclude}
    procedure WndProc(var Message: TMessage); override;
    {@exclude}
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    {@exclude}
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    {@exclude}
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    {@exclude}
    procedure Paint; override;
  public //зона бачення змінних та типів даних
    {@exclude}
```

```
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
    constructor Create(AOwner: TComponent);override; //Конструктор
    destructor Destroy; override; //Деструктор
    class function CheckVMR: boolean;
    function VMRGetBitmap(Stream: TStream): boolean;
    function CheckInputPinsConnected: boolean;
    procedure SetAllocator(Allocator: TAbstractAllocatorClass; UserID:
Cardinal);
  published //зона бачення змінних
    property OnPaint: TNotifyEvent read FOnPaint write FOnPaint;
    // завдання властивості типу
    property FullScreenTopMost: boolean read FTopMost write SetTopMost
             default false;
    property Mode: TVideoMode read FMode write SetVideoMode default vmNormal;
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;
    property VideoHandle: THandle read GetVideoHandle;
    property VMROptions: TVMROptions read FVMROptions write FVMROptions;
    property FullScreen: boolean read FFullScreen write SetFullScreen
                         default false;
    property Color;
    property Visible: boolean read GetVisible write SetVisible default True;
    property ShowHint;
    property Anchors;
    property Canvas;
    property PopupMenu;
    property Align;
    property TabStop default True;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnCanResize;
    property OnClick;
    property OnConstrainedResize;
    property OnDblClick;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnMouseWheel;
    property OnMouseWheelDown;
    property OnMouseWheelUp;
    property OnResize;
  end;

  TSampleGrabber = class; // Оголошення класу

  TSampleGrabber = class(TComponent, IFilter,
                         ISampleGrabberCB) // Оголошення класу
  private
    FOnBuffer: TOnBuffer;
```

```
    FBaseFilter: IBaseFilter;
    FFilterGraph : TFilterGraph;
    FMediaType: TMediaType;
    BMPInfo : PBitmapInfo;
    FCriticalSection: TCriticalSection;
    function GetFilter: IBaseFilter;
    function GetName: string;
    procedure NotifyFilter(operation: TFilterOperation; Param: integer = 0);
    procedure SetFilterGraph(AFilterGraph: TFilterGraph);
    function  SampleCB(SampleTime: Double; pSample: IMediaSample): HResult;
                       stdcall;
    function  BufferCB(SampleTime: Double; pBuffer: PByte; BufferLen: longint):
                       HResult; stdcall;
  protected
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
  public //зона бачення змінних та типів даних
    SampleGrabber: ISampleGrabber;
    InPutPin  : IPin;
    OutPutPin : IPin;
    constructor Create(AOwner: TComponent); override; //Конструктор
    destructor Destroy; override; //Деструктор
    procedure UpdateMediaType;
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;


    procedure SetBMPCompatible(Source: PAMMediaType; SetDefault: cardinal);
    function GetBitmap(Bitmap: TBitmap; Buffer: Pointer; BufferLen: Integer):
boolean; overload;
    function GetBitmap(Bitmap: TBitmap): boolean; overload;
    class function CheckFilter: boolean;
  published //зона бачення змінних
    property OnBuffer: TOnBuffer read FOnBuffer write FOnBuffer;
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;
     // завдання властивості типу
    property MediaType: TMediaType read FMediaType write FMediaType;
  end;
//  структура TFilter
  TFilter = class(TComponent, IFilter) // Оголошення класу
  private
    FFilterGraph : TFilterGraph;
    FBaseFilter: TBaseFilter;
    FFilter: IBaseFilter;
    function GetFilter: IBaseFilter;
    function GetName: string;
    procedure NotifyFilter(operation: TFilterOperation; Param: integer = 0);
    procedure SetFilterGraph(AFilterGraph: TFilterGraph);
  protected
    {@exclude}
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
  public
//зона бачення змінних та типів даних
```

```
    { Конструктор }
    constructor Create(AOwner: TComponent); override;
    { Деструктор method. }
    destructor Destroy; override; //Деструктор
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
  published //зона бачення змінних та типів даних
    property BaseFilter: TBaseFilter read FBaseFilter write FBaseFilter;
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;
  end;

//  структура TASFWriter

  TASFWriter = class(TComponent, IFilter) // Оголошення класу
  private
    FFilterGraph : TFilterGraph;
    FFilter      : IBaseFilter;
    FPort        : Cardinal;
    FMaxUsers    : Cardinal;
    FProfile     : TWMPofiles8;
    FFileName    : WideString;
    FAutoIndex   : boolean;
    FMultiPass   : boolean;
    FDontCompress: boolean;
    function  GetProfile: TWMPofiles8;
    procedure SetProfile(profile: TWMPofiles8);
    function  GetFileName: String;
    procedure SetFileName(FileName: String);
    function  GetFilter: IBaseFilter;
    function  GetName: string;
    procedure NotifyFilter(operation: TFilterOperation; Param: integer = 0);
    procedure SetFilterGraph(AFilterGraph: TFilterGraph);
  protected
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
  public //зона бачення змінних та типів даних
    WriterAdvanced2      : IWMWriterAdvanced2;
    { NetWork streaming configuration. }
    WriterNetworkSink    : IWMWriterNetworkSink;
    { Вхідні дані }
    DataInput           : IPin;
    VideoInput           : IPin;
    DataStreamConfig    : IAMStreamConfig;
    VideoStreamConfig    : IAMStreamConfig;
    constructor Create(AOwner: TComponent); override; //Конструктор
    destructor Destroy; override; //Деструктор
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
  published //зона бачення змінних
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;
    property Profile: TWMPofiles8 read GetProfile write SetProfile;
    property FileName: String read GetFileName write SetFileName;
```

```
    // завдання властивості типу
    property Port: DWORD read FPort write FPort;
    property MaxUsers: DWORD read FMaxUsers write FMaxUsers;
    property AutoIndex   : boolean read FAutoIndex write FAutoIndex default
True;
    property MultiPass   : boolean read FMultiPass write FMultiPass default
False;
    property DontCompress: boolean read FDontCompress write FDontCompress
default False;


  end;


//  структура TDSTrackBar


  TTimerEvent = procedure(sender: TObject; CurrentPos, StopPos: Cardinal) of
                          object ;


  TDSTrackBar = class(TTrackBar, IEvent) // Оголошення класу
  private
    FFilterGraph: TFilterGraph;
    FMediaSeeking: IMediaSeeking;
    FWindowHandle: HWND;
    FInterval: Cardinal;
    FOnTimer: TTimerEvent;
    FEnabled: Boolean;
    FMouseDown: boolean;
    procedure UpdateTimer;
    procedure SetTimerEnabled(Value: Boolean);
    procedure SetInterval(Value: Cardinal);
    procedure SetOnTimer(Value: TTimerEvent);
    procedure SetFilterGraph(AFilterGraph: TFilterGraph);
    procedure GraphEvent(Event, Param1, Param2: integer);
    procedure ControlEvent(Event: TControlEvent; Param: integer = 0);
    procedure TimerWndProc(var Msg: TMessage);
    property TimerEnabled: Boolean read FEnabled write SetTimerEnabled;
  protected
    {@exclude}
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
    {@exclude}
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState;
      X, Y: Integer); override;
    {@exclude}
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
      X, Y: Integer); override;
    {@exclude}
    procedure Timer; dynamic;
  public //зона бачення змінних та типів даних
    { Конструктор }
    constructor Create(AOwner: TComponent); override;
    { Деструктор }
    destructor Destroy; override;
```

```
//Деструктор
  published //зона бачення змінних
    property FilterGraph: TFilterGraph read FFilterGraph Write SetFilterGraph;
    property TimerInterval: Cardinal read FInterval write
                            SetInterval default 1000;
    property OnTimer: TTimerEvent read FOnTimer write SetOnTimer;
  end;

  { @exclude }
  TDSVideoWindowEx2 = class; // Оголошення класу

//  структура TColorControl

  TColorControl = class(TPersistent) // Оголошення класу
  private
    FBrightness : Integer;
    FContrast   : Integer;
    FHue        : Integer;
    FSaturation : Integer;
    FSharpness  : Integer;
    FGamma      : Integer;
    FUtilColor  : Boolean;
    FDefault    : TDDColorControl;
  protected
    { Protected секція }
    { @exclude }
    FOwner : TDSVideoWindowEx2;
    { @exclude }
    Procedure SetBrightness(Value : Integer);
    { @exclude }
    Procedure SetContrast(Value : Integer);
    { @exclude }
    procedure SetHue(Value : Integer);
    { @exclude }
    procedure SetSaturation(Value : Integer);
    { @exclude }
    procedure SetSharpness(Value : Integer);
    { @exclude }
    procedure SetGamma(Value : Integer);
    { @exclude }
    procedure SetUtilColor(Value : Boolean);
    { @exclude }
    function GetBrightness : Integer;
    function GetContrast : Integer;
    function GetHue : Integer;
    function GetSaturation : Integer;
    function GetSharpness : Integer;
    function GetGamma : Integer;
    { @exclude }
    function GetUtilColor : Boolean;
    { @exclude }
```

```
    Procedure ReadDefault;
    { @exclude }
    procedure UpdateColorControls;
    { @exclude }
    procedure GetColorControls;
  public //зона бачення змінних та типів даних
    { Public секція }
    constructor Create(AOwner: TDSVideoWindowEx2);  virtual; //Конструктор
    procedure RestoreDefault;
  Published //зона бачення змінних
    property Brightness : Integer read GetBrightness write SetBrightness;
    // завдання властивості типу
    property Contrast : Integer read GetContrast write SetContrast;
    property Hue : Integer read GetHue write SetHue;

    property Saturation : Integer read GetSaturation write SetSaturation;

    property Sharpness : Integer read GetSharpness write SetSharpness;

    property Gamma : Integer read GetGamma write SetGamma;

    property ColorEnable : Boolean read GetUtilColor write SetUtilColor;
  end;

//  структура TDSVideoWindowEx2Caps
  TDSVideoWindowEx2Caps = class(TPersistent) // Оголошення класу
  protected
    { Protected секція }
    Owner : TDSVideoWindowEx2;
    function GetCanOverlay : Boolean;
    function GetCanControlBrigtness : Boolean;
    function GetCanControlContrast : Boolean;
    function GetCanControlHue : Boolean;
    function GetCanControlSaturation : Boolean;
    function GetCanControlSharpness : Boolean;
    function GetCanControlGamma : Boolean;
    function GetCanControlUtilizedColor : Boolean;
  public
    { Public секція }
    { @exclude }
    constructor Create(AOwner: TDSVideoWindowEx2);  virtual; //Конструктор
  published //зона бачення змінних
    Property CanOverlayGraphic : Boolean read GetCanOverlay;
    Property CanControlBrigtness : Boolean read GetCanControlBrigtness;
    Property CanControlContrast : Boolean read GetCanControlContrast;
    Property CanControlHue : Boolean read GetCanControlHue;

    Property CanControlSaturation : Boolean read GetCanControlSaturation;
    Property CanControlSharpness : Boolean read GetCanControlSharpness;
    Property CanControlGamma : Boolean read GetCanControlGamma;
    Property CanControlColorEnabled : Boolean read GetCanControlUtilizedColor;
```

```
  end;

//  структура TOverlayCallback

  TOverlayCallback = class(TInterfacedObject, IDDrawExclModeVideoCallBack)
                           AOwner : TObject; // Оголошення класу
    constructor Create(Owner : TObject); virtual; //Конструктор
    function OnUpdateOverlay(bBefore: BOOL; dwFlags: DWORD; bOldVisible: BOOL;
             var prcOldSrc, prcOldDest: TRECT; bNewVisible: BOOL; var prcNewSrc,
             prcNewDest: TRECT): HRESULT; stdcall;
    function OnUpdateColorKey(var pKey: TCOLORKEY; dwColor: DWORD): HRESULT;
                              stdcall;
    function OnUpdateSize(dwWidth, dwHeight, dwARWidth, dwARHeight: DWORD):
                          HRESULT; stdcall;
  end;

//  структура TDSVideoWindowEx2

  TRatioModes  = (rmStretched, rmLetterBox, rmCrop);

  TOverlayVisibleEvent = procedure (Sender: TObject; Visible : Boolean) of
object;

  { @exclude }
  TCursorVisibleEvent = procedure (Sender: TObject; Visible : Boolean) of
object;
  TDSVideoWindowEx2 = class(TCustomControl, IFilter, IEvent) // Оголошення класу
  private
    FVideoWindow       : IVideoWindow;
    FFilterGraph       : TFilterGraph;
    FBaseFilter        : IBaseFilter;
    FOverlayMixer      : IBaseFilter;
    FVideoRenderer     : IBaseFilter;
    FDDXM              : IDDrawExclModeVideo;
    FFullScreen        : Boolean;
    FTopMost           : Boolean;
    FColorKey          : TColor;
    FWindowStyle       : LongWord;
    FWindowStyleEx     : LongWord;
    FVideoRect         : TRect;
    FOnPaint           : TNotifyEvent;
    FOnColorKey        : TNotifyEvent;
    FOnCursorVisible   : TCursorVisibleEvent;
    FOnOverlay         : TOverlayVisibleEvent;
    FColorControl      : TColorControl;
    FCaps              : TDSVideoWindowEx2Caps;
    FZoom              : Integer;
    FAspectMode        : TRatioModes;
    FNoScreenSaver     : Boolean;
    FIdleCursor        : Integer;
    FMonitor           : TMonitor;
    FFullscreenControl : TForm;
```

```
    GraphWasUpdatet    : Boolean;
    FOldParent         : TWinControl;
    OverlayCallback    : TOverlayCallback;
    GraphBuildOK       : Boolean;
    FVideoWindowHandle : HWND;
    LMousePos          : TPoint;
    LCursorMov         : DWord;
    RememberCursor     : TCursor;
    IsHidden           : Bool;
    FOverlayVisible    : Boolean;
    OldDesktopColor    : Longint;
    OldDesktopPic      : String;
    FDesktopPlay       : Boolean;
    procedure NotifyFilter(operation: TFilterOperation; Param: integer = 0);
    procedure GraphEvent(Event, Param1, Param2: integer);
    function GetName: string;
    procedure ControlEvent(Event: TControlEvent; Param: integer = 0);
    procedure SetFilterGraph(AFilterGraph: TFilterGraph);
    procedure SetTopMost(TopMost: boolean);
    procedure SetZoom(Value : Integer);
    function UpdateGraph : HResult;
    function GetVideoInfo : HResult;
    procedure SetAspectMode(Value : TRatioModes);
    procedure FullScreenCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure SetVideoZOrder;
  protected
    {@exclude}
    function GetFilter: IBaseFilter;
    {@exclude}
    procedure resize; override;
    {@exclude}
    procedure Loaded; override;
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
    {@exclude}
    procedure WndProc(var Message: TMessage); override;
    procedure Paint; override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure MyIdleHandler(Sender: TObject; var Done: Boolean);
    procedure RefreshVideoWindow;
  public //зона бачення змінних та типів даних
    constructor Create(AOwner: TComponent); override; //Конструктор
    destructor Destroy; override; //Деструктор
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
    procedure ClearBack;
    procedure StartDesktopPlayback; overload;
    procedure StartDesktopPlayBack(OnMonitor : TMonitor); overload;
```

```
    procedure NormalPlayback;
    procedure StartFullScreen; overload;
    procedure StartFullScreen(OnMonitor : TMonitor); overload;
    property FullScreen: boolean read FFullScreen;
    property DesktopPlayback : Boolean Read FDesktopPlay;
    property Canvas;
    property ColorKey : TColor read FColorKey;
    property Capabilities : TDSVideoWindowEx2Caps read FCaps;
    property OverlayVisible : Boolean read FOverlayVisible;
published //зона бачення змінних
    property AspectRatio : TRatioModes read FAspectMode write SetAspectMode;
    property AutoHideCursor : Integer read FIdleCursor write FIdleCursor;
    // завдання властивості типу
    property DigitalZoom : Integer read FZoom write SetZoom;
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;
    property FullScreenTopMost: boolean read FTopMost write SetTopMost default
                                false;
property OnColorKeyChanged: TNotifyEvent read FOnColorKey write FOnColorKey;
property ColorControl: TColorControl read FColorControl write FColorControl;

property NoScreenSaver: Boolean read FNoScreenSaver write FNoScreenSaver;

property OnOverlayVisible: TOverlayVisibleEvent read FOnOverlay write
                           FOnOverlay;

property OnPaint : TNotifyevent read FOnPaint Write FOnPaint;
property OnCursorShowHide : TCursorVisibleEvent read FOnCursorVisible write
                            FOnCursorVisible;
    property Color;
    property Visible;
    property ShowHint;
    property Anchors;
    property PopupMenu;
    property Align;
    property TabStop default True;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnCanResize;
    property OnClick;
    property OnConstrainedResize;
    property OnDblClick;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnMouseWheel;
    property OnMouseWheelDown;
    property OnMouseWheelUp;
    property OnResize;
  end;
```

```
// Оголошення класу TVMRBitmap
type // Об’ява власних типів та структур

  TVMRBitmapOption=(
    vmrbDisable,
    vmrbSrcColorKey,
    vmrbSrcRect);

  TVMRBitmapOptions = set of TVMRBitmapOption;

  TVMRBitmap = class // Класи
  private
    FVideoWindow: TVideoWindow;
    FCanvas: TCanvas;
    FVMRALPHABITMAP: TVMR9ALPHABITMAP;
    FOptions: TVMRBitmapOptions;
    FBMPOld: HBITMAP;
    procedure SetOptions(Options: TVMRBitmapOptions);
    procedure ResetBitmap;
    procedure SetAlpha(const Value: Single);
    procedure SetColorKey(const Value: COLORREF);
    procedure SetDest(const Value: TVMR9NormalizedRect);
    procedure SetDestBottom(const Value: Single);
    procedure SetDestLeft(const Value: Single);
    procedure SetDestRight(const Value: Single);
    procedure SetDestTop(const Value: Single);
    procedure SetSource(const Value: TRect);
    function GetAlpha: Single;
    function GetColorKey: COLORREF;
    function GetDest: TVMR9NormalizedRect;
    function GetDestBottom: Single;
    function GetDestLeft: Single;
    function GetDestRight: Single;
    function GetDestTop: Single;
    function GetSource: TRect;
  public //зона бачення змінних та типів даних
    constructor Create(VideoWindow: TVideoWindow); //Конструктор
    destructor Destroy; override; //Деструктор
    procedure LoadBitmap(Bitmap: TBitmap);
    procedure LoadEmptyBitmap(Width, Height: Integer; PixelFormat: TPixelFormat;
              Color: TColor);
    procedure Draw;
    procedure DrawTo(Left, Top, Right, Bottom, Alpha: Single; doUpdate: boolean
              = false);
    procedure Update;
    property Canvas: TCanvas read FCanvas write FCanvas;
    property Alpha: Single read GetAlpha write SetAlpha;
    property Source: TRect read GetSource write SetSource;
    property DestLeft   : Single read GetDestLeft write SetDestLeft;
    property DestTop    : Single read GetDestTop write SetDestTop;
    property DestRight  : Single read GetDestRight write SetDestRight;
    property DestBottom : Single read GetDestBottom write SetDestBottom;
```

```
    property Dest: TVMR9NormalizedRect read GetDest write SetDest;
    // завдання властивості типу COLORREF
    property ColorKey: COLORREF read GetColorKey write SetColorKey;
    property Options: TVMRBitmapOptions read FOptions write SetOptions;
  end;
End.
```

ФАЙЛ ГОЛОВНОГО ВІКНА

```
unit Video_RTSP; // Назва модулю
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпеки та програмного забезпечення
Виконав: Чернат Єгор Сергійович, КБ-20
Керівник: Смірнов О.А.
}

Interface
{Інтерфейсна частина, тут проходе опис класів та типів}

uses Classes, SysUtils; // Підключення бібліотек

type

TVideo_RTSPType = class(TComponent) // Об’ява класу
private // секція бачення
    FValues: TStringList;
    FVideo_RTSPCap: string;
    FUserAgent: string;
    procedure   ReadUserAgentInfo;
    procedure   SetVideo_RTSPCap(const Value: string);
    procedure   SetUserAgent(const Value: string);
    function    GetCount: integer;
    function    GetKey(Index: integer): string;
    function    GetValue(const Index: variant): variant;
    function    GetVideo_RTSP: string;
    function    GetVersion: string;
    function    GetMajorVer: integer;
    function    GetMinorVer: integer;
    function    GetFrames: boolean;
    function    GetTables: boolean;
    function    GetCookies: boolean;
    function    GetBackgroundSounds: boolean;
    function    GetVBScript: boolean;
    function    GetJavaScript: boolean;
    function    GetJavaApplets: boolean;
    function    GetPlatform: string;
    function    GetActiveXControls: boolean;
    function    GetBeta: boolean;
public // секція бачення
    constructor Create(AOwner: TComponent); override;
    destructor  Destroy; override; // Деструктор
    property    Video_RTSPCap: string read FVideo_RTSPCap write
SetVideo_RTSPCap;
    property    UserAgent: string read FUserAgent write SetUserAgent;
    property    Video_RTSP: string read GetVideo_RTSP;
    property    Version: string read GetVersion;
    property    MajorVer: integer read GetMajorVer;
    property    MinorVer: integer read GetMinorVer;
```

```
    property    Frames: boolean read GetFrames;
    property    Tables: boolean read GetTables;
    property    Cookies: boolean read GetCookies;
    property    BackgroundSounds: boolean read GetBackgroundSounds;
    property    JavaScript: boolean read GetJavaScript;
    property    JavaApplets: boolean read GetJavaApplets;
    property    Platform: string read GetPlatform;
    property    ActiveXControls: boolean read GetActiveXControls;
    property    Beta: boolean read GetBeta;
    property    Count: integer read GetCount;
    property    Keys[Index: integer]: string read GetKey;
    property    Values[const Index: variant]: variant read GetValue; default;
end;

procedure   Register;

implementation
{Секція де проходить реалізація того що описано в інтерфейс ній частині}

uses Windows, IniFiles; // Підключення бібліотек

constructor TVideo_RTSPType.Create(AOwner: TComponent); //Конструктор
begin
    inherited create(AOwner);
    FValues := TStringList.create;
    SetVideo_RTSPCap('BROWSCAP.INI');
end;

destructor  TVideo_RTSPType.Destroy; // Деструктор
begin
    FValues.free;
    inherited destroy;
end;

function IsWindowsNT : Boolean; // Реалізація функції
begin
    result := (Win32Platform = VER_PLATFORM_WIN32_NT);
end;

procedure   RaiseWin32Error; // Реалізація процедури
begin
    raise exception.create(SysErrorMessage(GetLastError));
end;

var IniLock: TRTLCriticalSection; // Об’ява змінних

procedure   TVideo_RTSPType.ReadUserAgentInfo;
// Реалізація процедури
var // Об’ява змінних
    ini: TIniFile;
```

```
    function matchWithWildcard(const s: string): boolean; // Реалізація функції
    var // Об’ява змінних
        p: integer;
        rightPartLen: integer;
    begin
        result := false;
        p := pos('*', s);
        // отримуємо строку
        if (p = 0) then
            exit;
        if (copy(s, 1, p - 1) <> copy(FUserAgent, 1, p - 1)) then
            exit;
        rightPartLen := length(s) - p;
        if (copy(s, p + 1, rightPartLen) <> copy(FUserAgent, length(FUserAgent)
- rightPartLen + 1, rightPartLen)) then
            exit;
        result := true;
    end;

    function determineSection: string; // Реалізація функції
    const
        defaultVideo_RTSPSection = 'Default Video_RTSP Capability Settings';
    var // Об’ява змінних
        sections: TStringList;
        i: integer;
    begin
        sections := TStringList.create;
        try
            ini.readSections(sections);
            // спроба отримання даних
            if (sections.indexOf(FUserAgent) <> -1) then begin
                result := FUserAgent;
                exit;
            end;
            // пошук
            for i := 0 to (sections.count - 1) do begin
                if matchWithWildcard(sections[i]) then begin
                    result := sections[i];
                    exit;
                end;
            end;
            // нічого не знайдено, встановлення значень по замовчанню
            if (sections.indexOf(defaultVideo_RTSPSection) <> -1) then begin
                result := defaultVideo_RTSPSection;
                exit;
            end;
            // похибка
            result := '';
        finally
            sections.free;
        end;
```

```
    end;

    procedure   readSection(const section: string); // Реалізація процедури
    var // Об’ява змінних
        parentSection: string;
        sectionValues: TStringList;
        i: integer;
        key: string;
    begin
        parentSection := ini.readString(section, 'parent', '');
        if ((parentSection <> '') and (parentSection <> section)) then
            readSection(parentSection);
        sectionValues := TStringList.create;
        try
            ini.readSectionValues(section, sectionValues);
            for i := 0 to (sectionValues.count - 1) do begin
                key := sectionValues.names[i];
                FValues.values[key] := sectionValues.values[key];
            end;
        finally
            sectionValues.free;
        end;
    end;

var // Об’ява змінних
    section: string;
begin
    FValues.clear;
    if (FUserAgent = '') then
        exit;

    ini := TIniFile.create(FVideo_RTSPCap);
    try
        EnterCriticalSection(IniLock);
        try
            section := determineSection;
            if (section <> '') then
                readSection(section);
        finally
            LeaveCriticalSection(IniLock);
        end;
    finally
        ini.free;
    end;
end;

procedure   TVideo_RTSPType.SetVideo_RTSPCap(const Value: string);
// Реалізація процедури
begin
    if (value <> FVideo_RTSPCap) then begin
        FVideo_RTSPCap := value;
```

```
        if (FUserAgent <> '') then
            ReadUserAgentInfo;
    end;
end;

procedure   TVideo_RTSPType.SetUserAgent(const Value: string);
// Реалізація процедури
begin
    if (value <> FUserAgent) then begin
        FUserAgent := value;
        ReadUserAgentInfo;
    end;
end;

function    TVideo_RTSPType.GetCount: integer; // Реалізація функції
begin
    result := FValues.count;
end;

type TVarKind = (vkStr, vkInt);

function    getVarKind(const v: variant): TVarKind; // Реалізація функції
const
    varIntKindMask = varSmallint or varInteger or varByte;
    varStrKindMask = varOleStr or varString;
begin
    if ((varType(v) and varTypeMask and varIntKindMask) <> 0) then
        result := vkInt
    else if ((varType(v) and varTypeMask and varStrKindMask) <> 0) then
        result := vkStr
    else
        raise exception.create('Expecting an integer or a string');
end;

function    browsCapStrToVar(const s: string): variant; // Реалізація функції
begin
    if (s = '') then
        result := ''
    else if (s[1] = '#') then begin
        try
            result := strToInt(copy(s, 2, maxInt));
        except
            on EConvertError do
                result := s;
        end;
    end else if (compareText(s, 'true') = 0) then
        result := true
    else  if (compareText(s, 'false') = 0) then
        result := false
    else
        result := s;
```

```
end;

function TVideo_RTSPType.GetValue(const Index: variant): variant;
// Реалізація функції
var // Об’ява змінних
    key: string;
begin
    case getVarKind(index) of
        vkInt: key := getKey(index);
        vkStr: key := string(index);
    end;
    if (FValues.IndexOfName(key) = -1) then
        result := null
    else
        result := browsCapStrToVar(FValues.values[key]);
end;

function TVideo_RTSPType.GetKey(Index: integer): string; // Реалізація функції
begin
    result := FValues.Names[index];
end;

function    TVideo_RTSPType.GetVideo_RTSP: string; // Реалізація функції
begin
    result := VarToStr(values['Video_RTSP']);
end;

function    TVideo_RTSPType.GetVersion: string; // Реалізація функції
begin
    result := VarToStr(values['Version']);
end;

function    SafeVarToInt(const v: variant): integer; // Реалізація функції
begin
    if VarIsNull(v) then
        result := 0
    else
        result := integer(v);
end;

function    SafeVarToBoolean(const v: variant): boolean; // Реалізація функції
begin
    if VarIsNull(v) then
        result := false
    else
        result := boolean(v);
end;

function    TVideo_RTSPType.GetMajorVer: integer; // Реалізація функції
begin
    result := SafeVarToInt(values['MajorVer']);
```

```
end;

function    TVideo_RTSPType.GetMinorVer: integer; // Реалізація функції
begin
    result := SafeVarToInt(values['MinorVer']);
end;

function    TVideo_RTSPType.GetFrames: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['Frames']);
end;

function    TVideo_RTSPType.GetTables: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['Tables']);
end;

function    TVideo_RTSPType.GetCookies: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['Cookies']);
end;

function    TVideo_RTSPType.GetBackgroundSounds: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['BackgroundSounds']);
end;

function    TVideo_RTSPType.GetVBScript: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['VBScript']);
end;

function    TVideo_RTSPType.GetJavaScript: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['JavaScript']);
end;

function    TVideo_RTSPType.GetJavaApplets: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['JavaApplets']);
end;

function    TVideo_RTSPType.GetPlatform: string; // Реалізація функції
begin
    result := VarToStr(values['Platform']);
end;

function    TVideo_RTSPType.GetActiveXControls: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['ActiveXControls']);
end;
```

```
function    TVideo_RTSPType.GetBeta: boolean; // Реалізація функції
begin
    result := SafeVarToBoolean(values['Beta']);
end;


initialization
begin
    InitializeCriticalSection(IniLock);
end;

finalization
begin
    DeleteCriticalSection(IniLock);
end;

end.
```