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 для обробки даних відеонагляду з використанням БД та технології KVM”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Кіріченко Т.М.
«____» _____ 2024 р.

Керівник проекту
доктор технічних 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

Кіріченко Тетяні Миколаївні
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM*

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 для обробки даних відеонагляду з використанням БД та технології KVM*

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
_____ Кіріченко Т.М.
(прізвище та ініціали)

АНОТАЦІЯ

Кіріченко Т.М. Програмне забезпечення системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM. 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 для обробки даних відеонагляду з використанням БД та технології KVM.

Метою розробки є програмне забезпечення системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

Результат роботи – програмна реалізація системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

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 10.4.1.

Ключові слова: кібербезпека, відеонагляд, KVM

ABSTRACT

Kirichenko T.M. Cyber security system software for video surveillance data processing using DB and KVM technology. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for a cyber security system for processing video surveillance data using a database and KVM technology.

The goal of development is cyber security system software for video surveillance data processing using a database and KVM technology.

The result of the work is the software implementation of a cyber security system for video surveillance data processing using a database and KVM technology.

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 10.4.1 environment.

Keywords: cyber security, video surveillance, KVM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АРП	–	автоматичне регулювання посилення
АТМ	–	автоматичний телевізійний моніторинг
ЕПТ	–	електронно-променева трубка
ІБ	–	інформаційна безпека
ІС	–	інформаційна система
НСД	–	несанкціонований доступ
ПЗ	–	програмне забезпечення
ПЗЗ	–	прилади із зарядовим зв'язком
СКД	–	системи контролю даних
СЦК	–	система централізованого керування
ТВЛ	–	телевізійні лінії
IP	–	Internet Protocol. Протокол міжмережної взаємодії
PIN-код	–	персональний ідентифікаційний код
TCP	–	Transmission Control Protocol. Протокол керування передачею (даних) Основний транспортний протокол у стеці протоколів TCP/IP. Встановлює зв'язок між двома ПК й організує обмін даними в дуплексному режимі
TCP/IP	–	Transmission Control Protocol/Internet Protocol. Стек протоколів, що забезпечують організацію зв'язку між комп'ютерами в мережі Інтернет

ВСТУП

Актуальність теми. В умовах війни, рівень криміналітету України значно виріс, що саме по собі говорить про необхідність убезпечити своє житло або офіс всіма можливими для цього засобами, у тому числі й відеоспостереженням. Системи відеоспостереження дійсно є ефективним методом захисту від незваних гостей. Комп'ютерна система відеоспостереження в будинку, квартирі або офісі надасть безліч переваг. Наприклад, огляд прилеглої території дозволить побачити, хто до вас прийшов. Якщо відеодомофон охоплює тільки тог, хто безпосередньо перед ним, то камери відеоспостереження допоможуть побачити інше.

У випадку проникнення в будинок або офіс під час відсутності хазяїна, записи з камер відеоспостереження, зафіксовані на відеореєстратор, пришвидшать упіймання злочинця правоохоронними органами. Досить часто, з метою психологічного ефекту, по периметру встановлюють муляжі камер, але дане лякання може подіяти тільки на недосвідчених ведмежатників. Іноді, відеореєстратори встановлюють у сейфах або важкодоступних місцях, щоб зберегти запис від тих, кому це невигідно. Вся інформація зберігається на жорстких дисках. Відеоспостереження в будинку або квартирі забезпечує не тільки додаткову охорону під час відсутності хазяїна, але й додаткову впевненість у домашньому персоналі, при наявності таких. Наприклад, якщо ви найняли няньку для дитини, то за допомогою системи відеоспостереження через інтернет, ви завжди зможете в реальному часі подивитися, що зараз відбувається будинку і як, найнята на роботу людина виконує свої обов'язки.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

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

– Огляд існуючих систем для обробки даних відеонагляду з використанням БД та технології KVM.

– Дослідження системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

– Програмна реалізація системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для обробки даних відеонагляду з використанням БД та технології KVM.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM, 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 відеонагляду з використанням технології KVM.

KVM – це абревіатура від словосполучення «Keyboard, Video Monitor, Mouse» (клавіатура, відео монітор, миша).

Фізичні розміри KVM-пристроїв – від розміру долоні до розміру ноутбука.

Основні завдання, які вирішує KVM перемикач:

– забезпечити користувачеві доступ і контроль над декількома пристроями (комп'ютерами, серверами) з однієї KVM-консолі;

– забезпечити декільком користувачам, що використовують KVM-консоль, доступ і контроль над декількома пристроям.

Область використання:

– охоронні агентства, служби безпеки: одержання оперативної інформації з декількох відеокамер спостереження;

– рекламні компанії, що роблять послуги зі створення й розміщення зовнішньої медійної реклами;

– фінансові й товарні біржі, банківські структури. Зручність для роботи брокерів з декількома вікнами з одного пристрою, швидке перемикання між торговельними площадками в різних країнах миру, додатками;

– роздрібні мережі торговельних точок;

– для домашнього користування, наприклад, для мережних комп'ютерних ігор при наявності декількох гравців – побачити, що відбувається на екрані суперника.

– медичні установи. Швидкий і оперативний доступ до даних, що перебуває на різних комп'ютерах, особливо в критичних ситуаціях;

– урядові заклади;

– військові установи;

– авіація й космічна промисловість. Одержання інформації про поточний стан приладів з офісу аеропорту із приладових панелей декількох літаків, космічних кораблів, і віддалене керування;

– аптечні мережі;

– І інші компанії малого, середнього й великого бізнесу, де є необхідність у віддаленому керуванні сервером (-ами), комп'ютером (-ами) з іншої кімнати, офісу, міста або країни.

Переваги використання KVM-пристроїв:

– Економія витрат – при використанні KVM-перемикачів немає необхідності здобувати окремі клавіатури, монітори, мишки для кожного керованого пристрою. Також, за рахунок застосування KVM-пристроїв звільняється простір, знижуються витрати на електроенергію й вентиляційні системи.

– Безперервність роботи – KVM-перемикачі забезпечують доступ до серверів і інших керованих пристроїв не через корпоративну мережу. Із цієї причини навіть у випадку неполадок у мережі, адміністратор може управляти роботою цих пристроїв. KVM-пристрої з функцією віддаленого керування живленням, крім того, дозволяють робити перезавантаження керованої системи віддалено.

– Безпека – при віддаленому доступі гарантується безпека передачі інформації за рахунок спеціально розроблених технологій автентифікації, кодування й авторизації.

– Можливість роботи дистанційно – IT-фахівці організації можуть управляти серверами й швидко усувати будь-які неполадки, де б не перебували керовані пристрої. Крім того, це дозволяє знизити відрядні витрати.

– Незалежність від апаратних платформ і операційних систем – KVM-перемикачі сумісні з абсолютно різними ОС. Пристрої не залежать ні від апаратної платформи, ні від бази даних, ні від операційної системи й марки компанії-виробника пристроїв. Перемикачі призначені для роботи з

устаткуванням будь-якого виробника (наприклад, HP, IBM®, Sun®) з будь-який ОС (наприклад, UNIX®, Windows® і т.п.).

1.2 Область застосування

Кожна людина, будь то власник заміського будинку або бізнесу, рано або пізно замислюється про охорону свого майна. Прагнення убезпечити себе, контролювати те, що відбувається, вже не примха, а необхідність. Щоб допомогти людині зробити своє життя безпечніше, розроблені системи відеоспостереження. Стандартні завдання, що стоять перед відеоспостереженням на будь-якому об'єкті, будь те великий, малий бізнес або приватний будинок, схожі. Виділимо основні:

1. Поточне спостереження.

2. Робота з архівом відеозаписів.

3. Дистанційний перегляд поточне зображення й архіву.

4. Запис відеозображення по детекторі руху, а також при спрацьовуванні охоронних датчиків або втраті сигналу.

На великому об'єкті до стандартного додаються наступні завдання:

1. Інтеграція із системою охоронної й пожежної сигналізації.

2. Інтеграція з апаратно-програмним комплексом системи контролю й керування доступом (СКУД).

3. Масштабованість і модернізація системи відеоспостереження при необхідності.

4. Поточне спостереження й керування всією системою з однієї крапки, у тому числі організація відеоспостереження через Інтернет.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM, 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

Найпоширеніші на ринку безпеки наступні системи відеоспостереження:

1. Система відеоспостереження, побудована на базі відеореєстратора.
2. Система відеоспостереження, побудована на базі персонального комп'ютера й плат відеозахвата.
3. Система відеоспостереження, побудована на базі персонального комп'ютера й IP-камер.

Система відеоспостереження на базі відеореєстратора

Апаратна частина

Відеореєстратор являє собою закінчений пристрій по виду ідентичний звичайному DVD програвачу. По внутрішньому пристрої відеореєстратор схожий на персональний комп'ютер. Він має материнську плату, процесор, оперативну пам'ять, жорсткий диск, рознімання відеовходів і відеовиходів, USB порт, порт керування PTZ камерою. Все це реалізовано на одній платі й не має можливості якої-небудь модернізації. Багато відеореєстраторів мають тривожні входи й виходи для підключення охоронних датчиків (датчики обсягу, витоку газу, води) і зовнішніх що сповіщають і виконавчих пристроїв, наприклад, сирени, електромеханічних замків, оповіщувачів по GSM каналам, пульта охорони.

Звичайно відеореєстратори мають 4, 8 або 16 відеовходів для підключення камер відеоспостереження. При цьому вам зовсім не обов'язково використовувати всі відеовходи відразу, деякі можуть залишатися в резерві. Чим більше кількість відеовходів, тим вище ціна пристрою, причому подвоєння ціни при подвоєнні кількості відеовходів не відбувається.

Використовувані в даній системі відеоспостереження камери є аналоговими. Даний тип камер перетворить відеосигнал у формат, що може бути отриманий телевізійним або іншим приймачем, наприклад охоронним монітором. У той же час система відеоспостереження є цифровий, тому що отримані дані зберігаються на цифровий носій.

Продуктивність системи відеоспостереження практично повністю залежить від продуктивності процесора. Показником продуктивності процесора є тактова частота й архітектура. Частота процесора, застосовуваного у відеореєстраторі, звичайно коливається між 500 Мгц і 1000 Мгц, що обумовлено складністю відводу тепла. У даній системі всі ресурси спрямовані на обробку відеосигналу й не використовуються сторонніми програмами, тому такий, на перший погляд не високої частоти процесора, цілком достатньо.

Відеореєстратор підключається до звичайного телевізора через відеовхід VIDEO-IN, так само його називають «тюльпан» або RCA, або комп'ютерному монітору через VGA.

Програмна частина

Як операційна система у відеореєстраторах використовуються різні клони системи Linux, часто розроблені спеціально для даного пристрою.

Фахівці однозначно визначають операційну систему Linux, як найбільш стабільну при довгостроковій безперервній роботі й стійку до зовнішніх впливів – віруси, піратські зломи й т.п.

Відеореєстратори дають користувачеві обмежений доступ до програмної частини, наприклад, ви не можете встановити нові програми або драйвера, видалити старі – тільки виконання тих команд, які передбачені в меню, як при роботі з будь-яким DVD-плеєром або телевізором. Це є запорукою безперебійної роботи й більше високої надійності системи.

Керування системою

Управляється відеореєстратор за допомогою графічного меню, відображуваного на екрані телевізора, до якого підключена система. Робота в

меню виробляється кнопками лицьової панелі відеореєстратора й пультом ДУ й повністю аналогічна керуванню телевізором або DVD плеєром. Багато відеореєстраторів так само мають можливість керування звичної всім комп'ютерною мишею.

Інтеграція з іншими охоронними системами

Відеореєстратор має обмежені здатності інтеграції з іншими охоронними системами: охоронною й пожежною сигналізацією, системами керування доступом і т.п. Це обумовлено «твердістю» їх апаратної й програмної частини. Відеореєстратори мають стандартні тривожні входи. Кількість тривожних входів звичайно збігається з кількістю відеовходів.

До тривожного входу можна підключити кожної з охоронних датчиків, наявних на ринку, тому що принцип роботи у всіх однаковий – вони або розмикають ланцюг при спрацьовуванні й називаються нормально-замкнутими або замикають її – нормально-розімкнуті. Будь-який датчик підключається до відеореєстратора звичайним двожильним кабелем, що називають «локшиною». Так само ви маєте можливість підключити до одного тривожного входу кілька охоронних датчиків, з'єднаних їх між собою. Така послідовно з'єднана між собою ланцюг з датчиків називається «контуром». Якщо у вас уже встановлена сигналізація, ви можете підключити тривожні входи відеореєстратора до її тривожних входів тої ж «локшиною».

При спрацьовуванні охоронного датчика відеореєстратор може виконати наступні дії: почати запис відео з обраних вами камер, подати звуковий сигнал через убудований динамік, подати сигнал тривоги на зовнішній пристрій. Зовнішнім пристроєм може бути: сирена, оповіщувач по GSM каналу, що відправить вам СМС на стільниковий телефон, пульт охорони. При цьому на один тривожний вихід можна підключити всі ці пристрої одночасно.

З однієї сторони можливості відеореєстратора по інтеграції з іншими охоронними системами обмежені, але з іншої сторони дозволяють вирішити більшість типових завдань.

Масштабованість і модернізація

Цифрова система відеоспостереження, побудована на базі відеореєстратора абсолютно не масштабована. Ні можливості збільшити число відеовходів або тривожних входів, додати ще один USB порт або встановити мережну карту. Відеореєстратор є закінченим, «жорстко зібраним» пристроєм із чітко визначеним набором функціональних можливостей.

Масштабованість системи відеоспостереження можлива тільки при підключенні декількох відеореєстраторів до локальної мережі й наявності спеціального програмного забезпечення, що дозволяє робити їхній перегляд в одному вікні на віддаленому комп'ютері.

Система відеоспостереження на базі персонального комп'ютера й плат відеозахвату

Апаратна частина

Система відеоспостереження на базі персонального комп'ютера й плат відеозахвату складається зі звичного всім персонального комп'ютера, зі спеціальною платою захвата відеозображення. Кожна така плата, найчастіше, має вузьку спеціалізацію. Тобто для можливості використання тривожних входів і виходів або PTZ камери буде потрібно покупка додаткової плати, що також буде необхідно інсталювати в комп'ютер. Але кожна додаткова інсталяція буде знижувати як надійність, так і взагалі працездатність системи в цілому у зв'язку з можливістю виникнення конфліктів пристроїв. Чим більше додаткових плат, тим скоріше вони почнуть конфліктувати.

Плати відеозахвата, аналогічно відеореєстраторам, звичайно мають 4, 8 або 16 відеовходів. Використовувані тут камери також є аналоговими, а система відеоспостереження – цифровий.

Тактова частота сучасного комп'ютера становить порядку 3 Ггц, що перевищує тактову частоту відеореєстратора в 3 рази. Доречно буде відзначити, що сучасна архітектура процесора, що застосовує сучасну оперативну пам'ять, дає ще більш ніж дворазовий приріст продуктивності. Таким чином,

продуктивність комп'ютера вище продуктивності відеореєстратора більш ніж в 6 разів. Говорити про необхідність такого потенціалу для приватного домоволодіння й малого бізнесу складно, зате вартість такої системи буде неухильно рости разом із продуктивністю й можливістю розширення. І може відрізнятися від системи на базі відеореєстратора в 2-3 рази й більше. Оцінити цю величину точно не представляється можливим, тому що в системах відеоспостереження на базі персонального комп'ютера й плат відеозахвата істотний внесок у загальну вартість вносить ПО, ціна якого іноді дорівнює вартості системи.

Для перегляду зображення, як і будь-який звичайний комп'ютер, система відеоспостереження на базі ПК і плат захвата підключається до монітора або телевізора. Інтерфейс підключення залежить від відеокарти або материнської плати персонального комп'ютера.

Програмна частина

Операційна система Windows – найбільш популярне рішення для систем відеоспостереження на базі ПК. Великий набір функціональних можливостей посуває користувачів систем відеоспостереження на їхнє використання, а загальна комп'ютерна грамотність по використанню саме Windows створила армію псевдо-професіоналів, що вміють установити комп'ютерну гру, але не усвідомлюють можливі наслідку цієї дії для сучасної системи безпеки. Сучасні програми підмінюють деякі системні файли, ускладнюючи або унеможливлюючи роботу системи відеоспостереження. Забираючи всі ресурси процесора, вони не дають нормально функціонувати споконвічно досить вимогливої до ресурсів системі. І навіть якщо користувачі не грають на комп'ютері, вони здатні декількома некваліфікованими діями повністю порушити роботу системи. Навіть у тому випадку, якщо система відеоспостереження підмінює Windows Explorer і завантажується до завантаження оболонки, охоронці прекрасно знають, як перемкнутися між завданнями (Ctrl-Alt-Del) і завантажити іншу програму. При цьому для надійного захисту від зовнішніх впливів (різних вірусів і атак) в

операційній системі Windows потрібен комплексний підхід і покупка дорогого спеціалізованого програмного забезпечення, що не завжди доступно звичайному користувачеві.

З вищесказаного можна зробити вивід, що для стабільної й зручної роботи системи відеоспостереження на базі ПК потрібний окремий комп'ютер, що буде надійно захищений. Є думка, що використання робочого комп'ютера як знизить вартість системи, так і не вплине на стабільність роботи. Ця думка помилкова. Технічне обслуговування системи відеоспостереження вимагає використання послуг інсталяторів через складність установки й настроювання програмного забезпечення. А постійні збої в роботі при використанні робочого комп'ютера, швидше за все, приведуть до покупки окремого ПК.

Керування системою

Інтерфейс користувача в комп'ютерній системі являє собою звичне для більшості користувачів рішення на базі операційної системи Windows. Після включення комп'ютера буде потрібно запустити спеціалізовану програму, у діалоговому вікні якої й буде відбуватися вся робота. Керування обмежується клавіатурою й мишею, що цілком звично й зручно звичайному користувачеві. З іншого боку, простота доступу може зіграти злий жарт як при спробі доступу до системи зловмисника, так і звичайної дитини.

Інтеграція з іншими охоронними системами.

Велика кількість додаткового програмного забезпечення для комп'ютерних систем відеоспостереження дозволяє рішенням на пазу ПК мати підвищену гнучкість у порівнянні з рішеннями на базі відеореєстраторів. Це викликано тим, що, використовуючи розповсюджену платформу операційної системи Windows, значно простіше створити велику кількість підсистем з різними функціональними можливостями, а також інтегрувати розробки інших виробників як програмні, так і апаратні. Комп'ютерна система відеоспостереження здатна інтегруватися з касовим робочим місцем у магазині роздрібної торгівлі, працювати разом із системою контролю доступу,

розпізнавати автомобільні номери й особи людей, працювати разом із системами охоронно-пожежної сигналізації. Щоб одержати всі вищевикладені можливості необхідно витратити багато часу й сил, а, що найважливіших грошей. Необхідно відзначити, що витрати на можливого функціонала, як і на будь-які інші програмні продукти, часом, не відповідають очікуваним результатам.

Масштабованість і модернізація

Масштабованість даного рішення не викликає питань. Наприклад, для збільшення кількості камер, що підключаються, або для підключення охоронних датчиків необхідно придбати й установити в комп'ютер ще одну плату з відповідними функціональними можливостями. З іншого боку, можливо, у цьому випадку прийдеться міняти програмне забезпечення або, як мінімум, його перенастроювати. А, тому що самостійне настроювання практично неможливе через її складність, виникне необхідність використання послуг інсталятора, що в черговий раз приведе до додаткових грошових витрат.

IP відеоспостереження

Система відеоспостереження, побудована на базі персонального комп'ютера й IP-камер багато в чому аналогічна попередній. Центром керування системою також є персональний комп'ютер.

Основною відмінністю від попередньої є спосіб передачі сигналу від камери до ПК і відсутність плат захвата відеозображення. Сигнал передається по мережному кабелі. Це зручно коли структура мережі добре організована, а робота стабільна. Замість плат тут використовується мережне встаткування (маршрутизатори, роутери, мости) і програмне забезпечення, що вимагає ще більш високої кваліфікації інсталяторів, із глибоким знанням мережних технологій. А, як відомо, праця професіоналів коштує дуже дорого.

Камера для системи IP відеоспостереження як мінімум в 3 рази дорожче звичайної. Якщо гнатися за дозволом 1-3 МП, то для цього вже потрібні гігабітні мережі (а не 100-мегабітні, як скрізь), що піднімає вартість у порівнянні з відеореєстраторами на порядок. И в 2-3 рази дорожче системи на базі ПК і плата

відеозахвата. Одна камера на 3 МП може коштувати як повний комплект із 4 камер і відеореєстратора.

Керування системою здійснюється в діалоговому вікні програми в операційному середовищі Windows з усіма її мінусами. Необхідно відзначити, що версія операційної системи тут скоріше буде не домашня, а мережна.

По інтеграції й масштабованості дана система обходить всіх конкурентів, але не потрібно забувати, що й витрати на це будуть самими більшими в порівнянні з конкурентами.

Огляд безкоштовного програмного забезпечення для систем відеоспостереження

Універсальне ПЗ для централізованого керування засобами IP-відеоспостереження, розраховане на роботу з камерами різних виробників, як правило, коштує чималих грошей. Але в той же час існують і безкоштовні програмні продукти, хоча вони, звичайно ж, мають певні обмеження.

Більшість розроблювачів, особливо із числа тих, які пропонують ПЗ для корпоративної сфери, щиро сподіваються, що спробувавши зручність і якість тестового зразка, покупець заплатить гарну ціну за повну версію. У теж час у багатьох випадках пробної версії може цілком вистачити для того, щоб повністю вирішити завдання потенційного клієнта. Декілька безкоштовних, але досить функціональних програмних розробок пропонується й у сфері IP-відеоспостереження.

Коли мова йде про завдання централізованого керування IP-відеокамерами, існують два основних варіанти рішення – використання фірмового ПЗ, що поставляється виробником камер, і застосування продуктів незалежних розроблювачів. У першому випадку програмне забезпечення поставляється у вигляді безкоштовного додатка до встаткування, а в другому – майже завжди є додатковим джерелом витрат. Здавалося б, хто буде купувати стороннє ПЗ, якщо є фірмові розробки? Але на практиці в багатьох випадках «незалежне» програмне забезпечення має більш широкі функціональні

можливості, ніж «рідні» продукти. До того ж фірмове ПЗ не дозволяє інтегрувати в мережу IP CCTV камери інших виробників, на відміну від програм спеціалізованих розроблювачів.

Справедливості заради варто відзначити, що прив'язка до власного встаткування відбувається не стільки зі шкідливості й небажання допустити камери інших виробників на проекти, скільки через технологічні особливості розвитку IP CCTV. Справа в тому, що сьогодні поки немає єдиних стандартів функціонування, яким би відповідали всі IP-камери. Виробіток загальних технологічних «правил гри» усе ще триває. Два найбільших альянси – ONVIF і PSIA – розвиваються паралельно.

Тому виробникові простіше й дешевше реалізувати повну підтримку власних стандартів устаткування, чим намагатися розробити універсальний програмний продукт. В умовах постійних цінових воєн, коли ринок активно завойовують недорогі IP-камери із КНР і Китайської Республіки (Тайваню), додаткові вкладення в розробку ПЗ природно здорожують і без того недешеві камери відеоспостереження європейських і американських виробників. При цьому розроблювачі спеціалізованого ПЗ всі сили направляють на те, щоб створювати як можна більше універсальні, гнучкі й функціональні продукти.

Що стосується безкоштовного ПЗ, то воно, як правило, являє собою «урізану» версію основної програми. Обмеження звичайно бувають по кількості камер, що підключаються, глибині зберігання відеоархіву й часу використання (до тридцяти днів). «Пробні» системи можуть також не підтримувати можливість подальшого нарощування й модернізації (у випадку покупки повної версії тестовий варіант не можна буде інтегрувати в загальну систему керування). Однак якщо для великої компанії тестової системи буде явно недостатньо, те невелика організація цілком може обійтися можливостями безкоштовного продукту.

Windows, Linux, Mac OS

Програмні системи для централізованого відеоспостереження існують для різних платформ – Windows, Linux, Mac OS, причому для кожної з них пропонується як мінімум один умовно безкоштовний продукт (табл. 2.1).

Таблиця 2.1 – Основні характеристики безкоштовного універсального ПЗ для центрального відеоспостереження

Компанія	Milestone	BenSoftwate	ZoneMinder	Axxon Soft	ISS
Назва ПЗ	XProject Go	SecuritySpy	ZoneMinder	Smart Start	SecurOS Lite
Обмеження по кількості камер	8	-	-	16	4
Обмеження по кількості серверів	1	-	-	1	1
Обмеження відеоархіву	5 днів	-	-	1 ТБ	-
Підтримка H.264	+	-	-	+	+
Детектор руху	+	+	+	+	+
Аналітичні функції	-	+	+	+	-
Інтелектуальний пошук в архіві	-	+	-	+	-
Операційна система	Windows	Mac OS X	Linux	Windows	Windows
Обмеження за часом використання	-	30 днів	-	-	-

Відзначимо, що рекламні версії пропонують не тільки маловідомі компанії, але й світові лідери сегмента. Один з них – датська компанія Milestone – також пропонує XProject Go (рисунок 2.1) – безкоштовну версію свого флагманського продукту. Можливості цієї версії досить скромні. Система підтримує до восьми IP-камер, один сервер, усього лише п'ять днів зберігання відеоархіву, а також формат відеозапису avi. Перший платний варіант ПЗ Milestone здатний підтримувати 26 камер, п'ять користувачів, необмежений час зберігання даних, формати avi і jpg, а також віддалену роботу через веб-клієнт. Для порівняння: найбільш потужний варіант – XProject Corporate – здатний підтримувати необмежену кількість камер, користувачів і час зберігання відеоархіву, має розвинені аналітичні функції, можливістю інтеграції ПЗ сторонніх виробників і т.д.

Рисунок 2.1 – Вікно керування ПЗ Milestone XProject Smart Client

У той же час XProject Go, як і всі розробки Milestone, підтримує біля дев'ятисот моделей камер більш ніж вісімдесятьох різних виробників (повний список можна побачити на сайті компанії) незалежно від того, у який альянс вони входять – ONVIF або PSIA. Також є можливість використання відеокамер, що підключаються по USB. Крім того, до системи можна приєднати також аналогові камери за допомогою спеціального IP-Декодера; XProject Go буде працювати також і з ними. Незважаючи на те що пробна версія продукту є безкоштовної, у ній підтримуються сучасні формати стиску відео: MPEG4 ASP, MxPEG, H.264. До вбудованих «інтелектуальних» функцій можна віднести програмний детектор руху, що працює незалежно від камери.

Як сервер можна використовувати звичайний ПК із процесором 2,4 ГГц і обсягом оперативної пам'яті 1-2 ГБ. Якщо планується використовувати продукт не більше місяця, то можна навіть не реєструватися на сайті. Процедура обов'язкова тільки у випадку застосування ПЗ понад тридцять днів. Інтерфейс програми підтримує двадцять мов, у тому числі росіянин. Однак XProject Go працює тільки з різними варіантами ОС Windows (XP, Vista, 7). Для роботи з іншими операційними системами існують інші продукти. Одним з них є умовно безкоштовна версія ПЗ для централізованого відеоспостереження, що пропонує компанія BenSoftware. Її розробка зветься SecuritySpy (рисунок 2) і розроблена вона для комп'ютерів Macintosh. Діапазон підтримуваних ОС – від Mac OS X 10.4.11 до 10.7.

Пробну версію можна використовувати протягом місяця, після чого вартість ліцензії складе від тридцяти до п'ятисот британських фунтів (залежно від кількості підключених камер).

Система підтримує необмежене число відеокамер (аналогових, IP, USB), серверів і обсяг архіву. Використовуються кодеки MJPEG і MPEG4. Є убудований веб-сервер, функції детектування руху, оповіщення відповідальних осіб по e-mail, автоматичного завантаження даних на зазначений FTP-сервер. Цікава функція – буфер запису, у який автоматично зберігається відео, зняте за

деякий, досить обмежений час, скажемо, за останні десять секунд. Це зроблено для того, щоб у випадку спрацьовування детектора руху можна було побачити не тільки тривожну подію, але й те, що їй передувало. В SecuritySpy є також можливості пошуку подій в архіві й автоматичній його оптимізації (яка зводиться головним чином до видалення застарілих даних). Підтримуються безліч камер, у тому числі роботизованих, таких виробників, як Axis, JVC, Panasonic, Pixord, D-Link і багатьох інших.

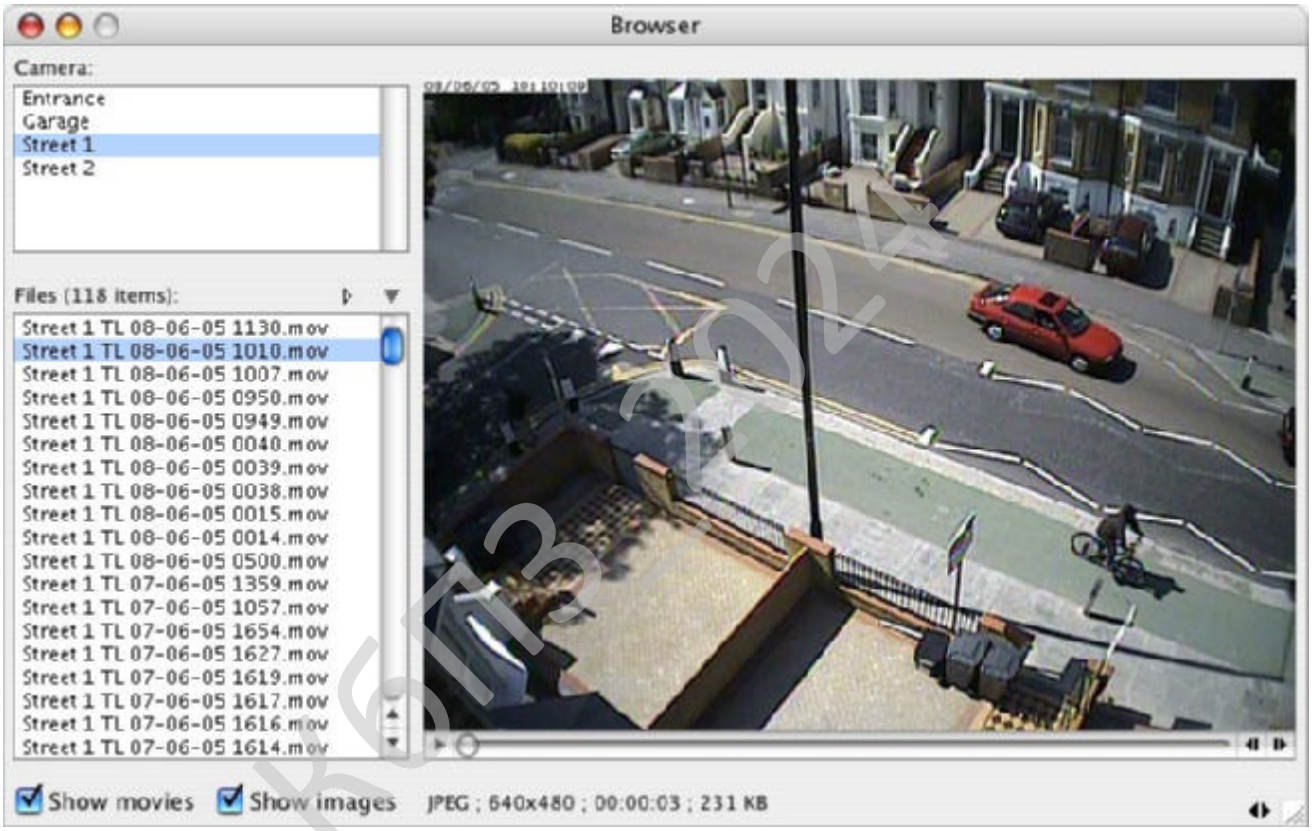

Рисунок 2.2 – Програма BenSoftware SecuritySpy призначена для комп'ютерів Macintosh

Операційна система Linux теж не залишилася осторонь. Щоб безкоштовно управляти IP-камерами в цьому середовищі, можна використовувати програму ZoneMinder (рисунок 2.3). Величезною перевагою даного продукту є його безкоштовність, оскільки весь проект існує на добровільні пожертвування. Тут

немає обмежень на число відеокамер, користувачів або розмір архіву – все залежить тільки від можливостей ПК або сервера, що обробляє й зберігає відеопотоки. У той же час тут, як і у двох вищезгаданих системах, є підтримка IP-USB- і аналогових камер різних виробників, детектор руху (з функцією автоматичного спрацьовування запису й відправлення повідомлення по e-mail або SMS), веб-клієнт, багатомовна підтримка й т.д. Оскільки програма ZoneMinder задумана як вільна платформа (побудована на базі C++, Perl і PHP), вона здатна інтегруватися із програмними розробками сторонніх виробників. Також передбачена можливість завантаження відео на FTP-сервер. Це програмне рішення позиціонується як безкоштовна альтернатива не тільки для невеликих компаній і кінцевих користувачів (наприклад, у випадку спостереження за периметром частки будинку), але й для великих організацій.

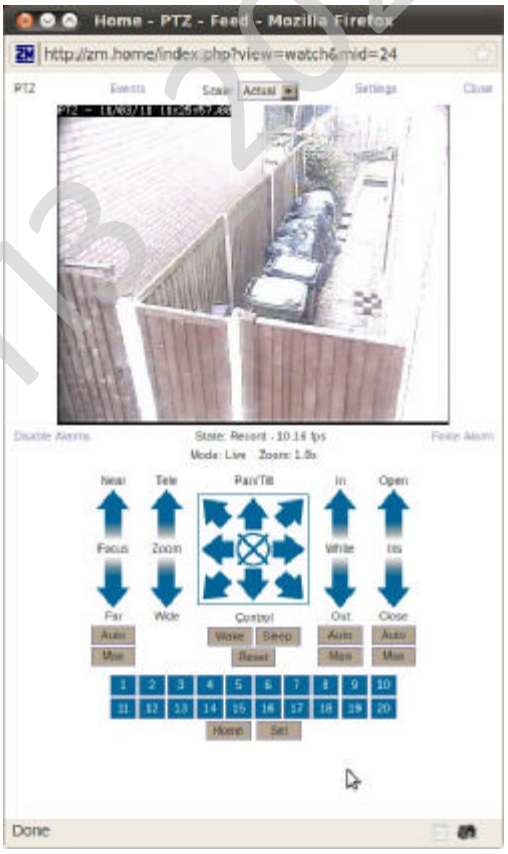

Рисунок 2.3 – Інтерфейс контролю над PTZ-камерою за допомогою ПЗ ZoneMinder

Як бачимо, універсальне й при цьому безкоштовне ПЗ для централізованого відеоспостереження існує, і одержати його не становить праці. При цьому можна підібрати систему, що володіє дуже широкими можливостями й до того ж під будь-яку популярну ОС. Можливості таких продуктів досить широкі, незважаючи на тестовий статус, і в ряді випадків невелика компанія може обійтися цим «безкоштовним сиром», не здобуваючи повну версію продукту. У той же час великий замовник цілком здатний оцінити переваги й недоліки основних функцій програмної розробки, перш ніж ухвалювати рішення щодо покупці комерційної версії ПЗ.

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 для обробки даних відеонагляду з використанням БД та технології KVM.

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

KVM устаткування

Абревіатура KVM походить від англійські слова Keyboard Video Mouse, що дослівно означає клавіатура, відео, миша. Іноді абревіатура одержує додаткові букви, наприклад, KVMA – «A» означає аудіо.

Всі пристрої сімейства KVM можна розділити на 4 групи:

– KVM extender або KVM подовжувачі – пристрої призначені для віддаленого керування встаткуванням на відстані. З використанням цих пристроїв організація доступу до комп'ютера здійснюється на відстані від декількох десятків метрів до декількох десятків кілометрів. KVM over IP подовжувачі дозволяють організувати віддалене керування комп'ютером через Інтернет або локальну мережу.

– KVM switch або KVM перемикачі – пристрої для оптимізації керування групою комп'ютерів або серверів з одного або декількох робочих місць. Можливо спільне використання з KVM подовжувачами, що дозволить із віддаленого робочого місця (консолі) управляти або адмініструвати групу комп'ютерів. Серед KVM перемикачів варто виділити групу безпечних KVM перемикачів, застосовуваних у системах з підвищеним рівнем безпеки.

– Перемикач клавіатури й миші або перемикач USB забезпечить керування групою комп'ютерів з однієї клавіатури й миші, однак, при цьому всі відеосигнали будуть передаватися на монітори, так щоб користувач завжди бачив зображення із всіх комп'ютерів.

– Матричний KVM перемикач являє собою пристрій керування, що забезпечує, групою комп'ютерів з великої кількості робочих консолей. У такий

KVM системі кожний з користувачів одержує можливість підключатися до кожного із присутніх у системі комп'ютерів.

KVM перемикачі

KVM перемикач (скорочення від англ. kvm switch (keyboard video mouse switcher) – перемикач клавіатури, відео, миші) – пристрій, призначений для перемикання одного або групи пристроїв уведення-виводу між комп'ютерами або іншими джерелами інтерфейсних сигналів.

Споконвічно KVM перемикачі були громіздкими пристроями і їх можна було називати суворим словом “комутатор” через великі розміри. KVM перемикачі застосовувалися в основному в сфері промисловості й у лабораторіях і несли в собі примітивне призначення – перемикання клавіатури, миші й відеомонітора. Перемикання було механічним, а не електронним як зараз, здійснювалося спеціальною ручкою або величезними кнопками, на які потрібно було сильно надавлювати.

Ті перші моделі KVM перемикачів комутирували тільки роз’єми відео VGA (DSUB), миші через COM (DE-9) порт і клавіатуру AT (DIN). Пізніше можна було зустріти механічні KVM перемикачі з роз’ємами VGA (DSUB) і PS/2.

Зараз такий KVM перемикач (kvm switch) можна було б віднести до настільних KVM перемикачів. Настільні KVM перемикачі – самі продавані KVM пристрої, тому що вони самі незамінні, і для ряду завдань вартість настільного KVM може бути менше вартості комплекту миші й клавіатури.

З підвищенням рівня технологічності всіх сфер діяльності й повсюдного застосування як настільних, так і серверних комп'ютерних рішень, з'явилася необхідність у спеціальних пристроях, що дозволяють віддаленно управляти декількома робочими станціями, використовуючи різні інтерфейси (USB, PS/2, CAT5, IP). Розроблені для рішення цих завдань пристрої стали називати KVM перемикачами – абревіатура від слів Keyboard, Video monitor, Mouse.

Первісне перемикання здійснювалося на напівавтоматичному рівні, коли після переходу від одного комп'ютера до іншому потрібно було натискати спеціальну механічну кнопку на перемикачі, що лише незначно відрізнялося від виконання всіх операції вручну. Для прискорення процесу була розроблена автоматична система, названа «Enumerated USB switching». Ця технологія заснована на нумерації кожного USB з'єднання. У момент підключення створюється прямий канал між комп'ютером і KVM консоллю, що дає великий плюс для більшості USB пристроїв, тому що підтримується висока швидкість з'єднання. Недолік цієї системи ставати очевидним, коли починається робота й звичні сполучення «гарячих клавіш», що роблять роботу більше комфортною, не можуть допомогти при перемиканні між комп'ютерами. Ненадійність операцій з деякими USB пристроями й операційними системами так само характерна для технологічних рішень заснованих на нумерації.

Наступним кроком в еволюції KVM перемикачів повинно було стати підвищення комфортності й надійності. Рішення завдання було знайдено з розробкою технології “USB Emulation”, що дозволила постійно емулювати сигнал USB пристрою на комп'ютер, змушуючи його «думати», що миша й клавіатура постійно підключені. Однак, вирішивши деякі проблеми попередньої технології, “USB Emulation” мала й свої недоліки, основним з яких стала обмеженість. Більшість виробників налаштовували свої KVM перемикачі на емуляцію тільки певних моделей мишок і клавіатур, що унеможливлювало використання більше складних моделей, що надають додаткові можливості сучасних USB пристроїв.

Знаючи про недоліки цих систем, компанія Adder приклала значні зусилля на їхнє рішення. Результатом стала революційна технологія «True USB emulation». True Emulation дозволяє KVM перемикачу «завантажити» інформацію про підключену клавіатуру й мишку й згодом транслювати на комп'ютери їхні віртуальні образи. Розроблений «Emulation Engine» не тільки емулює, але інтерпретує сигнали USB інтерфейсу, що дає можливість перемикання «гарячими клавішами» клавіатури або третьою (scroll) кнопкою мишки – це зручність для

локальних рішень і найважливіший функціонал для рішень віддаленого керування по CAT5 KVM або IP KVM.

True Emulation дає велику перевагу при використанні миші й клавіатури, однак, інші пристрої не мають потреби в емулюванні. Тому на пристроях збереглися й нумеровані ланцюги підключення USB пристроїв. Таким чином робота цих пристроїв здійснюється на максимальних швидкостях. Нумеровані ланцюги доповнюватися USB концентраторами (hubs), які дозволяють стабілізувати роботу каналів, що очікують, і перемикати додаткові пристрої незалежно друг від друга.

Завдяки напруженій роботі інженерів компанії Adder, недавно був представлений перший DVI KVM перемикач, що повністю підтримує описану новинку – Adder_View8 Pro (8-мі портовий USB KVN перемикач).

Основні особливості які відрізняють настільний KVM перемикач:

– невеликий розмір;

– мала кількість комп'ютерів, що приєднуються, (до 8);

– керування кнопками на передній панелі KVM перемикача;

– порти для підключення комп'ютера – стандартні інтерфейси;

– презентабельний зовнішній вигляд.

Головні відмінності KVM перемикачів для серверних кімнат від настільних:

– KVM перемикачі для серверних кімнат використовуються в основному для віддаленого керування;

– всі KVM перемикачі для серверних кімнат мають пропрієтарні кабелі для підключення до серверів;

– KVM перемикачі для серверних кімнат мають більші розміри й в основному розташовуються в стійці.

Сучасні настільні KVM перемикачі здатні комутирувати різні відео інтерфейси й інтерфейси керування, можуть кріпитися в стійковий простір, мають велику кількість спеціалізованих функцій і ергономічно вписуються в

робоче місце оператора, а головне перемикаються вже електронно й мають у своїй начинці сучасні технології, що дозволяють більш комфортно працювати.

Активні й пасивні KVM перемикачі

На українському й міжнародному ринку велика кількість KVM перемикачів працюють без блока живлення, живлення одержують від інтерфейсів клавіатури й миші, такі KVM перемикачі називаються пасивними – їх дуже великий недолік – це нестабільність роботи з певними моделями клавіатур і мишей, тому що при підключенні в KVM перемикача просто не вистачає потужності – щоб здійснити роботу. Активні KVM перемикачі завжди забезпечуються живленням, але й у них виникають проблеми з роботою нестандартних клавіатур і мишей, причина цьому – використання застарілих технологій перемикання.

Комутація відео інтерфейсів

KVM перемикачі можна розділити по типі сигналу, що комутирується відео. Сучасні KVM перемикачі здатні комутирувати наступні відео інтерфейси VGA, DVI Single Link, DVI Dual Link, Display Port.

KVM перемикачі з інтерфейсом VGA

Відео інтерфейс VGA здатний пропускати аналоговий відео сигнал і є найпоширенішим, так і KVM перемикачі з інтерфейсом VGA є в будь-якого виробника, стандартні максимальні розрішення з інтерфейсом VGA: 1600x1200 і 1920x1080. Тому при виборі KVM перемикачів з інтерфейсом VGA уточните необхідне вам розрішення. Як правило, цей вид KVM перемикачів є найпоширенішим і застосовується у всіх сферах діяльності IT фахівців, у зв'язку з дуже великою популярністю виробники бюджетних KVM пристроїв випустили окремі лінійки “кишенькових” KVM перемикачів, які можна брати із собою на зустрічі, у відрядження й тд.

KVM перемикачі з інтерфейсом DVI single link

KVM перемикачі з інтерфейсом DVI single link випускаються вже біля 4-х років і посіли друге місце по популярності в користувачів. Інтерфейс Single link

(одинарний режим) DVI використовує чотири кручених пари проводів (червоний, зелений, синій, і clock), що забезпечують можливість передавати 24 біта на піксель. З ним може бути досягнутий максимальне можливе розрішення при 1920x1200 60 Гц, при 1920x1080 – 75Гц.

При покупці KVM перемикачів з інтерфейсом DVI варто уважно вивчити роз'єми на відеокарті комп'ютера, тому що інтерфейс DVI може пропускати тільки цифровий сигнал (DVI-D), а може й цифровий і аналоговий (DVI-A), небагато про самий інтерфейс DVI:

– DVI-A – тільки аналогова передача.
– DVI-I – аналогова й цифрова передача.
– DVI-D – тільки цифрова передача.

Відеокарти з DVI-A не підтримують стандартні монітори з DVI-D.

DVI-I – відеокарту можна підключити до DVI-D-монітора (кабелем із двома коннекторами DVI-D – тато).

KVM перемикачі з інтерфейсом DVI Dual Link

Dual Link (подвійний режим) DVI подвоює пропускну здатність і дозволяє одержувати розрішення екрана 2560x1600 і 2048x1536. Тому для самих великих LCD моніторів з більшим розрішенням, таких, як 30" і 32" моделі, обов'язково потрібна відеокарта із двоканальним DVI Dual-Link виходом.

KVM перемикачі з інтерфейсом DVI Dual Link стали користуватися популярністю в 2010 році, вони здатні пропускати через себе цифровий сигнал DVI з розрішенням до 2560x1600 і вище. Такі KVM перемикачі найчастіше використовуються при професійній обробці відео – у телестудіях, у студіях постпродакшн і диспетчерських, де потрібне висока якість зображення.

KVM перемикачі з інтерфейсом Display Port

KVM перемикачі з інтерфейсом Display Port – новітні пристрої KVM, застосовуються поки рідко, у зв'язку з поки ще не високою популярністю моніторів з інтерфейсом Display Port, використовуються найчастіше в сфері дизайну й професійної обробки відео.

Інтерфейс DisplayPort підтримує HDCP версії 1.3 і має пропускну здатність удвічі більшу, ніж Dual-Link DVI, низька напруга живлення й низьку чутливість до сторонніх наведень. Технологія, реалізована в DisplayPort, дозволяє передавати одночасно як графічні, так і аудіо сигнали. DisplayPort 1.2 має максимальну швидкість передачі даних 21,6 Гбіт/с на відстані до 3 метрів, що більше ніж HDMI Type B (2x10,2 Гбіт/с). Також підтримує кілька незалежних потоків, пропускна здатність допоміжного каналу в стандарті збільшена з 1 до 720 Мбіт/с.

Таким чином, через інтерфейс DisplayPort 1.2 можна підключити до двох моніторів, що відтворюють картинку розміром 2560x1600 точок із частотою 60 Гц, або до чотирьох моніторів з розрішенням 1920x1200 точок. При використанні одиночного монітора підтримуване розрішення зростає до 3840x2400 точок із частотою 60 Гц, монітор з підтримкою частоти відновлення 120 Гц підтримується при розрішеннях до 2560x1600 точок. Це дозволяє стандарту DisplayPort 1.2 працювати з технологіями побудови стереоскопічного зображення (3D відео).

Інтерфейси керування комп'ютерами, підключеними до KVM перемикача

Якщо звернутися до історії, то споконвічно KVM перемикачі вміли працювати тільки з інтерфейсом PS/2, але в міру популяризації й розробки нових інтерфейсів керування виробники KVM устаткування додавали нововведення у свої прилади. Ми будемо розглядати наступні інтерфейси: PS/2, USB1.1, USB2.0, RS232, Audio.

KVM перемикачі з інтерфейсом PS/2

Дотепер користуються попитом KVM перемикачі. Даний інтерфейс може використовуватися або старими комп'ютерами й клавіатурами, або професійними пристроями уведення, такими як спеціалізовані клавіатури й трекболи – застосовуються в аеронавігації, телемовленні й постпродакшн. KVM перемикачі з інтерфейсом PS/2 звичайно використовуються рідше, ніж раніше, але з виробництва їх знімати ніхто поки не збирається.

KVM перемикачі з інтерфейсом USB

Найбільш популярні KVM перемикачі, які застосовуються у всіх галузях і IT рішеннях. За допомогою інтерфейсу USB на KVM перемикачі ви можете комутирувати практично будь-які периферійний пристрої, але тут варто звернути особливу увагу на призначення цього інтерфейсу. Як правило на KVM перемикачах для підключення клавіатури й миші й інших периферійних пристроїв використовуються спеціально призначені USB порти. Якщо KVM перемикач підтримує й клавіатуру, мишу й інші периферійні пристрої, то в нього як правило 4 USB порта.

KVM перемикачі з інтерфейсом USB 2.0

Це KVM пристрій останнього покоління (після 2009 року), які вже мають окремі порти для підключення периферійних пристроїв і миші із клавіатурою, плюс такі перемикачі, як правило, постачені спеціальними технологіями, які забезпечують більше чітку синхронізацію між комп'ютером і USB пристроями, наприклад технологія компанії ADDER True Emulation USB, що ми розглянемо пізніше. Інтерфейси USB 2.0 в KVM перемикачі пропонують більш широкі можливості вибору USB пристроїв, наприклад такі як джойстики, USB контролери, зовнішні накопичувачі даних та інші пристрої.

KVM перемикачі з підтримкою RS232

RS-232 (англ. Recommended Standard 232) – у телекомунікаціях, стандарт послідовної асинхронної передачі двійкових даних між терміналом (англ. Data Terminal Equipment, DTE) і комунікаційним пристроєм (англ. Data Communications Equipment, DCE).

Даний інтерфейс використовується в KVM перемикачах для керування спеціалізованими пристроями, наприклад, пристроями віддаленого керування живленням або спеціалізованого промислового або медичного встаткування.

KVM перемикачі з підтримкою аудіо

Такі KVM перемикачі називають ще KVMA Switch, вони дозволяють комутирувати як правило небалансовий стерео аудіо сигнал, з роз’ємом mini jack

(3,5 мм) – основним приймачем такого сигналу служать звичайні комп'ютерні колонки. Звук комутирують через KVM перемикач у рішеннях, де для певних позаштатних ситуацій є набір звукових оповіщень, або просто потрібно чути звук операторові.

Проблеми сучасних KVM перемикачів і рішення

У даному розділі ми розповімо, з якими проблемами ви можете зштовхнутися при використанні KVM перемикачів і як їх уникнути. Саме головне при виборі KVM перемикача – визначитися для яких завдань ви будете його використовувати. Визначимо основні проблеми:

– "Замиленість" картинки при використанні KVM перемикача – Зображення може “милитися” з кількох причин. По-перше, якщо джерело сигналу підключене до KVM перемикача через KVM подовжувач, то проблема в тому, що в KVM подовжувача відсутня функція De-Skew, що коректує затримку при передачі кольорів по кручений парі, у такій ситуації виходом може стати використання іншого KVM подовжувача або заміна UTP на кручену пару NO-Skew. Якщо KVM перемикач підключений прямо або проблема не вирішилася заміною крученої пари, те це недолік самого KVM перемикача, що не підтримує роботу з високими розрішенями (як правило така проблема виникає починаючи від дозволу понад 1600x1200).

– “Сніг” (мерехтливі різними кольорами пікселі в різних частинам екрана) – При підключенні через KVM перемикач – швидше за все ви використовуєте неякісний відео кабель із поганим екрануванням або відсутніми “пінами”. У такій ситуації потрібно замінити кабель. Якщо ж кабель більше стандартної довжини (2-5 метрів), спробуйте замінити його KVM подовжувачем. Зверніть увагу, що дешеві KVM подовжувачі по кручений парі можуть викликати аналогічну проблему, якщо KVM подовжувач по кручений парі не допоміг, то вам підійде тільки оптоволоконний KVM подовжувач.

– При перемиканні з комп'ютера на комп'ютер, використовуючи KVM перемикач, відео розрішення визначається не правильно або екран залишається

чорним – у цьому випадку проблема в тому, що KVM перемикач не може запам'ятати й передати параметр EDID. EDID (Extended Display Identification Data) – відомості про розрішення і стандарти підтримувані монітором, представлені у вигляді таблиці. Тому при покупці KVM перемикача переконаєтеся, що він підтримує функцію DDC (Display Data Channel) – канал який забезпечує спілкування відеокарти й монітора для забезпечення оптимальної передачі даних і кращої якості картинки.

– При використанні KVM перемикача, під час перемикання клавіатура й миша ініціалізуються дуже повільно – ви використовуєте KVM перемикач, процесор якого не підтримує функцію емуляції USB. При виборі KVM перемикача обов'язково дізнавайтеся, чи підтримує пристрій емуляцію USB.

– При використанні KVM перемикача, при перемиканні не працюють допоміжні мультимедійні клавіші на клавіатурі – дана проблема аналогічна вищеописаній – в KVM перемикачі може використовуватися емуляція, але виконується вона не інтелектуально, а за допомогою нумерації. Використовуйте KVM перемикач із функцією TRUE USB Emulation.

– Не працює клавіатура або миша при перемиканні – дана проблема проявляється при використанні KVM перемикачів з пасивним живленням (пристрій живиться від портів клавіатури й миші), якщо в KVM перемикача є порт живлення, то вам досить підключити живлення, якщо порту живлення ні, те прийде скористатися більше простою клавіатурою або поміняти KVM перемикач.

– При наборі тексту з використанням KVM перемикача "залипає" одна із клавіш. Якщо заміна клавіатури не допомогла, то проблема в процесорі KVM перемикача або в прошиванні, спробуйте обновити прошивання, якщо проблема після заміни прошивання не вирішилася, то прийдеться замінити KVM перемикач.

– При використанні KVM перемикача не працює USB-hub, розташований на клавіатурі – проблема викликана або тим, що KVM перемикач підтримує тільки стандартні клавіатури й миші – без використання інших периферійних

пристроїв, або в KVM перемикача відсутня функція емуляції USB, через яку не правильно працюють мультимедійні клавіатури.

– При використанні KVM перемикача не виходить виставити частоту відновлення екрана понад 60 Гц – проблема в самому KVM перемикачі, він просто не підтримує високу частоту розгорнення екрана. Спробуйте знизити розрішення відео.

Сучасні технології KVM перемикачів

Нижчеописані технології не тільки дозволять виконати завдання, але й звільнять вас від дрібних проблем, які ви маєте, із за використання неякісного устаткування:

– True USB Emulation.

– Screen Freeze.

– Free Flow.

– Trader Switch – Free Flow.

– SNMP Monitoring+.

– DDC.

Додаткові функції й особливості KVM перемикачів

Крім технологій і підтримки усіх можливих інтерфейсів вирішальним фактором до покупки певного пристрою можна вважати дрібну додаткову особливість, у цьому розділі ми опишемо самі популярні особливості, які можуть надають різні виробники:

– Кріплення в стійку – більшість настільних KVM перемикачів має можливість кріпитися в стійку, для цього в комплекті або опціонально є спеціальні кріплення.

– Керування живленням через KVM перемикач – керування живленням здійснюється через порт RS232 на KVM перемикачі.

– Матеріал корпуса й колір корпуса – ряд виробників KVM устаткування дуже піклуватися про зовнішній вигляд пристроїв, наприклад у компанії Adder всі нові серії KVM перемикачів виконані в металевому корпусі чорного кольору.

А у виробника KVM устаткування G&D корпус всіх пристроїв виконаний з анодируваного сплаву алюмінію сріблистого кольору, корпус шумопоглинальний.

– Малі габарити пристрою – зараз більшість виробників прагнути зменшити габарити KVM перемикачів, завдяки цьому вони легко містяться навіть на маленьких столах.

– Можливість каскадування – ряд виробників, наприклад компанія Adder використовує у своїх KVM перемикачах додаткову опцію каскадування, що дозволяє або збільшити число портів KVM перемикача, або за допомогою каскадування можна одномоніторний KVM перемикач зробити двох або трьох моніторним. Як правило каскадування здійснюється через порт RS232.

– Керування через IP – у настільних KVM перемикачах ця функція рідкість, таку функцію додала компанія G&D у свої настільні KVM перемикачі.

– Порти USB на передній панелі – багато виробників намагаються підвищити ергономічність і зручність людей працюючих за KVM перемикачами й USB порти для периферійних пристроїв розміщає на передній панелі, наприклад так робить компанія G&D

– Присутність кнопок для перемикань кожного комп'ютера – це рідке, але корисне додавання. Наприклад якщо до KVM перемикача можна підключити 4 комп'ютери, то компанія G&D поміщає на передню панель 4 кнопки, кожна з яких дозволить перемкнутися на потрібний комп'ютер.

Багатомоніторні KVM перемикачі

KVM перемикачі багатомоніторні виглядають і працюють майже також, як звичайні настільні KVM перемикачі, але замість одного роз’єми для відео на кожний порт комп'ютера, у них може бути від 2 до 4. Система перемикання нічим не відрізняється від одномоніторного KVM перемикача, тільки перемикається відразу не один монітор, а всі 4.

На жаль, не всі виробники мають бажання випускати різноманітні моделі многомоніторних KVM перемикачів. Самий широкий вибір таких пристроїв на

сьогоднішній день надають компанія Adder і G&D. В обох виробників вибір дуже великий.

KVM консоль

KVM консоль – пристрій, що поєднує в собі клавіатуру, пристрій маніпуляції (миша), і LCD монітор, і що має здатність монтажу в стійку, призначена для комутації одного комплекту пристроїв уведення-виводу між декількома або одним комп'ютерами.

Де раніше усього стали застосуються KVM консолі й у якій сфері вони найчастіше застосовуються зараз? Звичайно серверні кімнати! Це можуть бути, як величезні центри обробки даних, так і невеликі серверні кімнати або просто стійка з IT устаткуванням. А такі стійки й великі й не невеликі серверні кімнати є у будь-якій організації й інтегруються в будь-які проекти, і в будь-які компанії, наприклад:

- Хостинг провайдери.
- Оператори зв'язку.
- Компанії Телекоми.
- Диспетчерські.
- Ситуаційні центри.
- Банки й фінансові організації.
- Телебачення.
- Постпродакшн – професійна обробка відео й аудіо матеріалів.
- Аеронавігація.
- Підприємства ПЕК.
- Офіси невеликих компаній.

Завдання для KVM консолі

Які завдання вирішуються за допомогою KVM консолей? Питання, на який можна відповісти розгорнуто:

- Локальне й віддалене керування парком комп'ютерів або серверів.
- Економія серверного простору.

– Економія на засобах уведення й відображення (моніторів, клавіатур і мишей).

– Ергономіка серверної кімнати.

– Економія електроенергії.

Відмінні риси KVM консолей

KVM консолі можна розділити на кілька груп, що дозволить більш докладно й зрозуміло зрозуміти принципові відмінності пристроїв і їхньої особливості.

1. Кількість портів для підключення серверів. KVM консолі однопортові застосовуються в великих серверних кімнатах, або у випадку, коли серверна стійка вже укомплектована KVM перемикачем. Дана модель KVM консолі не має убудованого KVM перемикача, а на задній панелі є роз'єми PS/2, USB,VGA і аудіо для підключення до KVM перемикача. Багатопортові KVM консолі вже мають убудований KVM перемикач, але залежно від виробника можуть синхронізуватися з іншими KVM перемикачами, якщо потрібне розширення парку серверів. Багатопортові консолі комплектуються 8-мі й 16-ти портовими KVM перемикачами.

2. Діагональ екрана. KVM консолі можуть бути з різною діагоналлю LCD екрана й відрізнятися розрішенням екрана. На сьогоднішній день найпоширеніші діагоналі екрана, це 15,17,19 дюймів. Але на ринку зустрічаються моделі й з більшою діагоналлю екрана (наприклад 22 дюйма), але поставляються такі KVM консолі тільки під замовлення.

3. Спосіб керування KVM консоллю. За способом керування KVM консолі підрозділяються на пристрої з локальним керуванням і керуванням по IP. Локальне керування має на увазі керування KVM консоллю безпосередньо, тобто через екран шуканого пристрою. Керування по IP має на увазі підключення KVM консолі до Інтернету або локальної мережі, що дозволяє операторові за допомогою спеціальної програми (наприклад VNC) через комп'ютер з будь-якою

операційною системою, ноутбук або навіть телефон і планшет віддаленно управляти комп'ютерами, підключеними до KVM консолі.

4. Спосіб експлуатації – трансформування. Dual Rail – монітор і клавіатура можуть виїжджати зі стоїки окремо. Single Rail – монітор і клавіатура можуть виїжджати зі стойки тільки спільно

5. Спосіб підключення до серверів. Залежно від виробника існують кілька способів підключення серверів до KVM консолі:

– пропрієтарні кабелі, спеціалізовані розроблені інтерфейси, найчастіше SPHD (клавіатура, відео, миша);

– підключення по CATx, спеціалізовані адаптери, що мають із однієї сторони роз’єми VGA, DVI, USB і ін., а з іншої роз’єми RJ45 – підключаються до сервера інтерфейсними роз’ємами, а до KVM перемикача роз’ємами кручена пара.

Можливості консолей компанії Adder Technologies

Сьогодні на українському ринку надана безліч брендів, які надають свої рішення KVM консолей на українському ринку, один із провідних виробників рішень по KVM устаткуванню компанія Adder надає для нас унікальні пристрої для організації керування серверами, а саме KVM консолей. От деякі відмінні риси KVM консолей компанії Adder:

– підтримка сторонніх виробників KVM перемикачів;

– підтримка USB, PS/2 засобів керування;

– підтримка підключення серверів і комп'ютерів через інтерфейс DisplayPort;

– KVM консолі з одночасним доступом для локального й віддаленого керування групою до 4-х користувачів;

– контроль пристроїв, що забезпечують віддалене керування живленням;

– футуристичний дизайн (стійка з такий KVM консоллю може стояти навіть у переговірній кімнаті й буде презентабельно виглядати на тлі іншого встаткування);

– можливість каскадування з KVM перемикачами серії AVX1000 і AVX5000;

– можливість керування через віддалений KVM модуль.

3.2 Розробка структурної схеми

Необхідно побудувати систему відеоспостереження, що включає: центральний сервер (ПК) з монітором для охоронця й кілька відеокамер, розташованих по всій території об'єкта. Необхідно вести самостійне спостереження за об'єктами.

Для цього кабінети керівництва необхідно оснастити комплектами (монітор і миша), підключеними до центрального ПК.

Були поставлені наступні завдання:

– До центрального ПК підключити 6 комплектів устаткування (монітор і миша), установлених у різних кімнатах офісу компанії.

– Далекість кожного комплекту від центрального ПК не менш 70 м.

– Можливість перемикання керування центральним ПК між всіма комплектами.

Для того, щоб збільшити число використовуваних моніторів і пристроїв уведення (мишок/клавіатур), необхідних для керування системою відеоспостереження, пропонується використовувати KVM-устаткування. Абревіатуру «KVM» можна розшифрувати як: «клавіатура» («Keyboard»), «монітор» («Video monitor»), «миша» («Mouse»).

Запропоноване рішення реалізується на обладнанні англійської компанії Adder. Компанія Adder – ведучий розроблювач і постачальник рішень для локального, віддаленого й глобального контролю над комп'ютерами, випускає весь спектр високоякісного багатофункціонального KVM-обладнання.

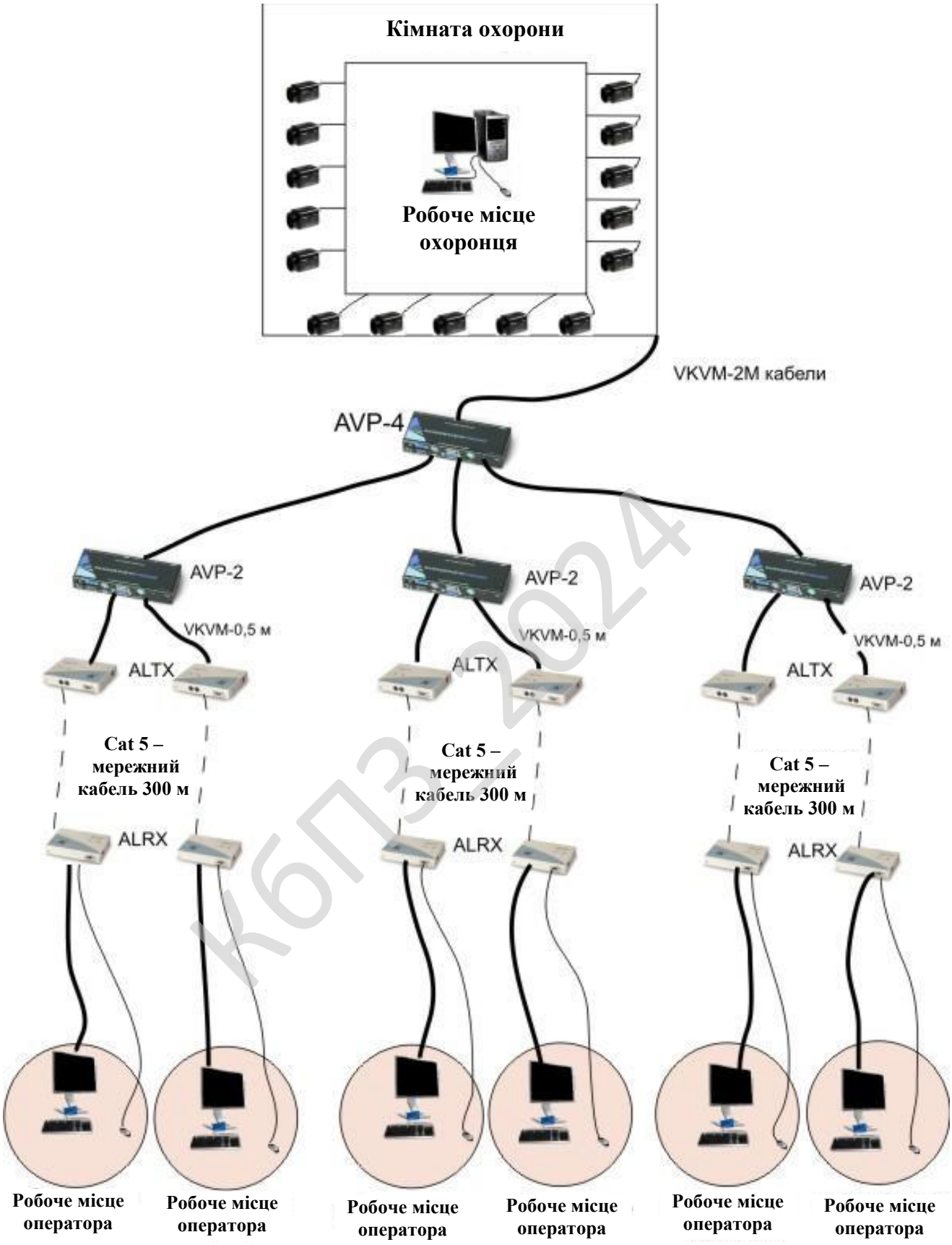

Рисунок 3.1 – Структурна схема системи

До центрального ПК за допомогою інтерфейсного кабелю Adder VKVM-2M, підключається Adder_View AVP4 – KVM перемикач. Він являє собою цифровий пристрій, комутуючий відеосигнал монітора й сигнали клавіатури й миші від центрального комп'ютера на кілька комплектів Клавіатура/Монітор/Миша (Keyboard/Video/Mouse.) KVM перемикач Adder_View AVP4, складається із двох основних пристроїв:

– відео-перемикач, що міняє напрямок аналогових відеоімпульсів між моніторами й комп'ютером спільного використання;

– мікропроцесорна система, що передає й приймає сигнали із клавіатури й миші й дає можливість управляти комп'ютером з кожного з робочих місць по черзі перемикаючись між ними).

При цьому число комплектів Клавіатура/Монітор/Миша визначається тільки можливостями й кількістю KVM-Перемикачів. Немає необхідності мати спеціальне програмне забезпечення, і відсутні традиційні громіздкі процедури підключення.

KVM перемикач Adder_View AVP4 здатний віддаленно підключити до центрального комп'ютера до 4 комплектів Клавіатура/Монітор/Миша або інше встаткування.

Для того, щоб розгалузити систему до Adder_View AVP4 за допомогою інтерфейсних кабелів Adder VKVM – 0,5M підключаються 3 KVM перемикачі Adder_View AVP2, кожний з яких дозволяє віддаленно підключити до 2 робочих місць.

До них, у свою чергу, підключаються екстендери (подовжувачі) AdderLink ALXT, які допомагають одержати високу якість зображення й звуку на відстані до 300 м.

Екстендери Adder, забезпечують передачу відеосигналу з високим дозволом, і можливість гнучкого розподілу пристроїв, керування, контролю й взаємодії із пристроями відображення.

Дані екстендери гарантують якісне з'єднання з необхідним устаткуванням і розроблені спеціально для підприємств, що мають розосередження встаткування, місце розташування якого може змінюватися залежно від вимог клієнта.

При використанні екстендерів Adder AdderLink немає необхідності використовувати оптоволоконні кабелі й розташовувати центральний процесор поруч із кожним екраном.

Екстендер ALXT, що є приймачем цифрових сигналів зв'язується з ALXR (передавач цифрових сигналів), за допомогою мережного кабелю CAT5 (кабель типу "кручена пари" категорії 5). Його відмінною рисою, є те, що він дозволяє підключити центральний комп'ютер до іншого встаткування на відстані до 300 м.

До екстендерів ALXR за допомогою кабелів Adder VKVM – 0,5M підключаються 6 комплектів Клавіатура/Монітор/Миша, необхідних замовникові. Це дуже зручно, оскільки потрібно тільки один кабель для кожного підключення до кожного робочого місця.

Завдяки запропонованому рішенню на базі встаткування Adder розроблена система одержала наступні переваги:

– Можливість захищеного й надійного доступу до центрального ПК із кожного з 6 створених робочих місць.

– Можливістю перемикання відеосигналу як з різних камер на різні робочі місця.

– Кожне робоче місце може бути віддалене від центрального ПК на відстань до 300 м.

– Висока якість переданого відео й звуку.

– Немає необхідності мати спеціальне програмне забезпечення, і відсутні традиційні громіздкі процедури підключення.

– Значна економія засобів на додаткових ПК, електроенергії, офісному просторі, а також вартості експлуатації й технічної підтримки системи.

Запропоноване рішення можна використовувати в наступних областях:

– Для установки систем безпеки, систем відеозображення.

– Системні адміністратори.

– У промисловості, при моніторингу й контролі функціонування технологічних ліній, виробничих процесів і взаємодії між ними.

– У науково-дослідних підрозділах, технологічних іспитових лабораторіях.

– У малих офісах і будинках.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Найпершим блоком у схемі є кінцевий об'єкт (сектор) за яким власне і ведеться відеоспостереження. Далі за допомогою відеокамер спостереження фірми DLink за розробленою схемою (рисунок 3.1) і застосуванні локального обладнання фірми Dlink (KVM) сигнал надходить у кімнату операторів де обробляється на сервері за допомогою розробленого ПЗ.

Спочатку проходить обробка KVM даних з послідуючим аналізом. Всі дані зберігаються у файловому архіві у форматі *.avi. Кожну добу інформація з камер спостереження формується у відповідну папку (шаблон папки ЧИСЛО_МІСЯЦЬ_РІК). Для зміни налаштувань чи перегляду документації існують відповідні вікна.

Також для організації маштабуємості системи й контролю дій над співробітниками сигнал також через мережне обладнання фірми Dlink (KVM) надходить до керівництва. Розробляючи апаратну й програмну частину бакалаврського проектування враховувався ціновий фактор, а також особливості застосування камер в умовах України, а саме – застосовувався ударостійкий корпус, були обрані камери з розширеними температурними діапазонами роботи (-40 С до +55 С) і великим кутом огляду камери.

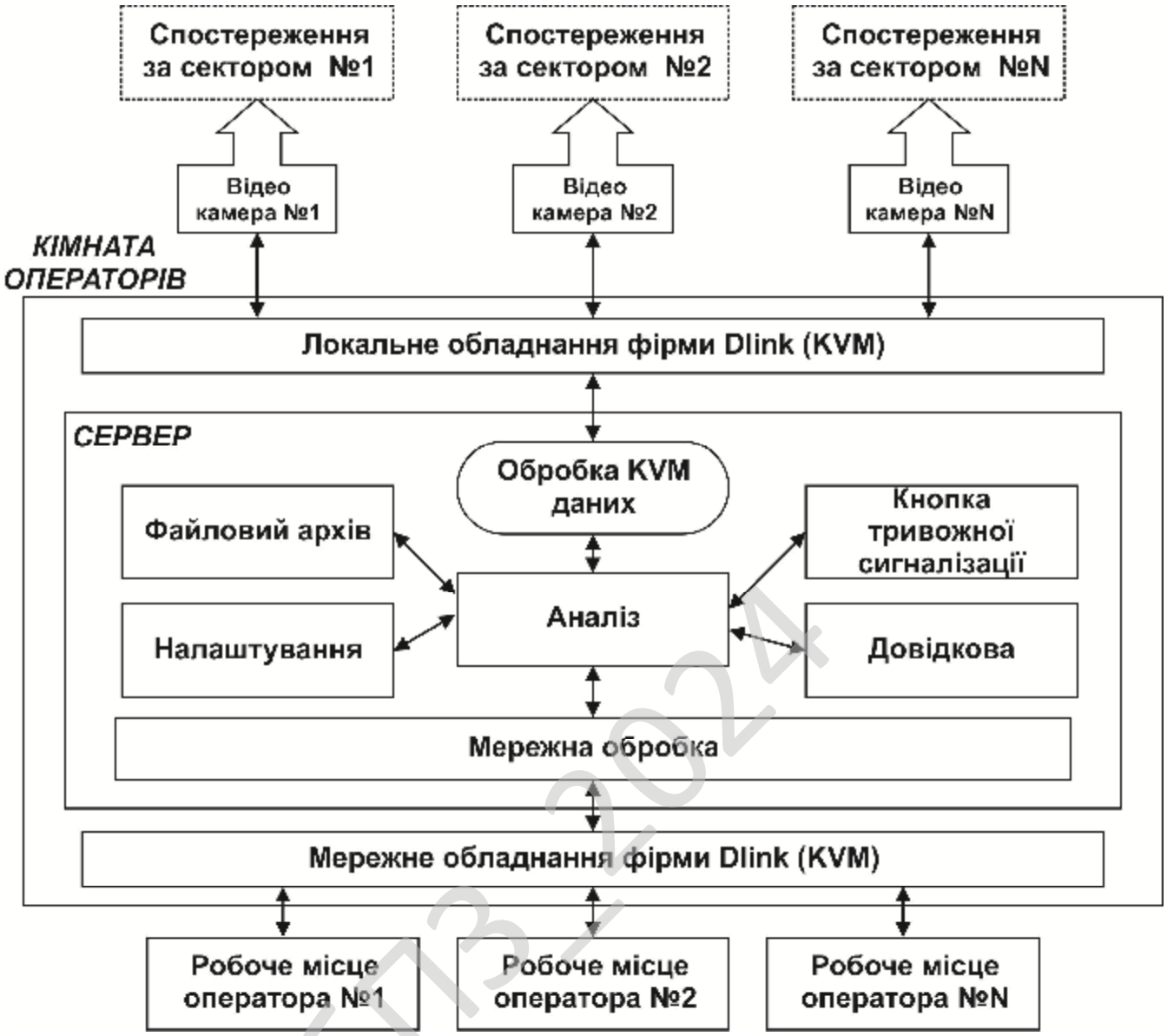

Рисунок 3.2 – Функціональна схема системи

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3. Після початку роботи розробленого ПЗ через інтерфейс ПЗ та обробник помилок ми потрапляємо до

блоку обробка KVM даних. Звідси можемо переглянути авторське право та через моніторинг даних потрапити до блоків налаштування ПЗ, довідкової системи, кнопки тривожної сигналізації, файлового архіву через локальний захист. Крім цього через модуль обробка мережних запитів проводиться прийом/передача пакетів з віддалених робочих місць.

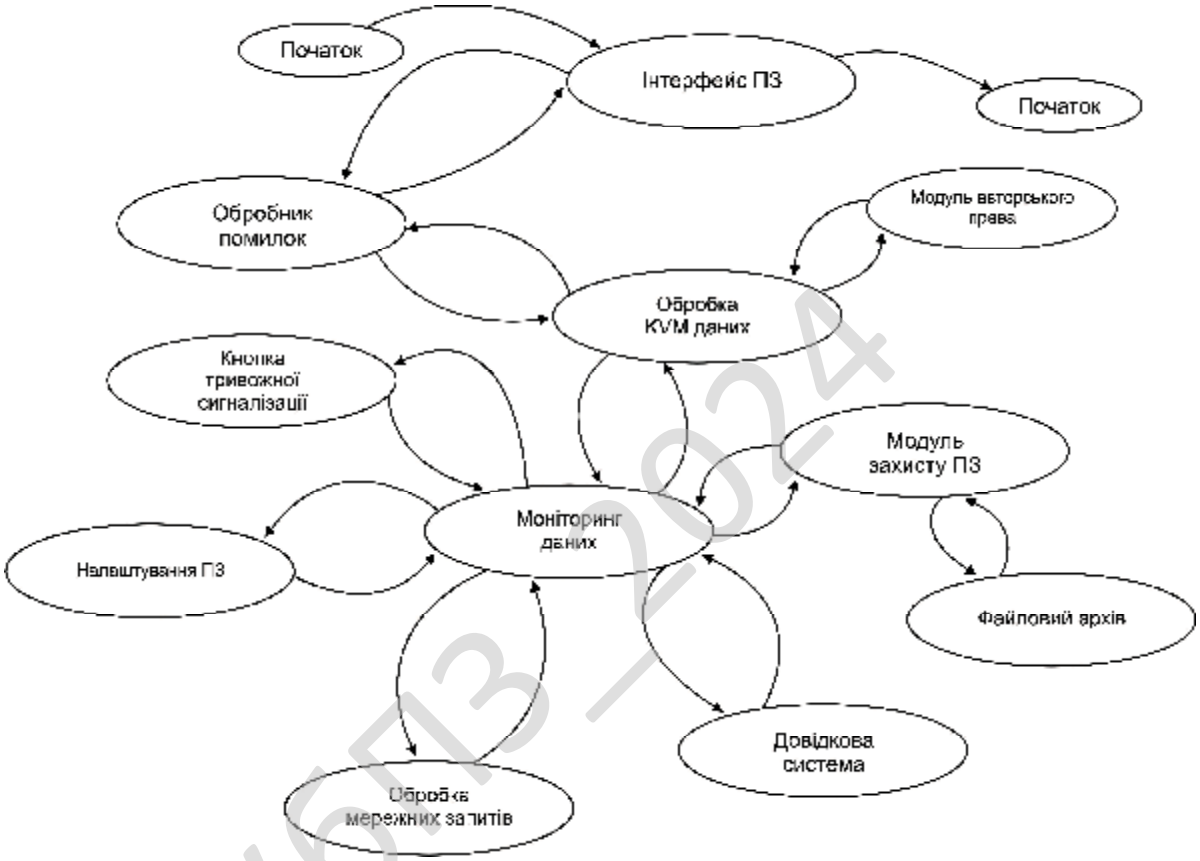

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена основна блок-схема програми. Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми. При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захвата відеопотоку зображення, модулю обробки помилок програми і основному модулю.

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

Перед розглядом подробиць схеми роботи програми розглянемо виконані основні напрацювання.

Розроблені класи:

Клас TAbstract_Allocator

Оголошення класу:

```
Type Abstract_Allocator = class()
```

Опис: абстрактне виділення покажчика для захоплення відео потоку.

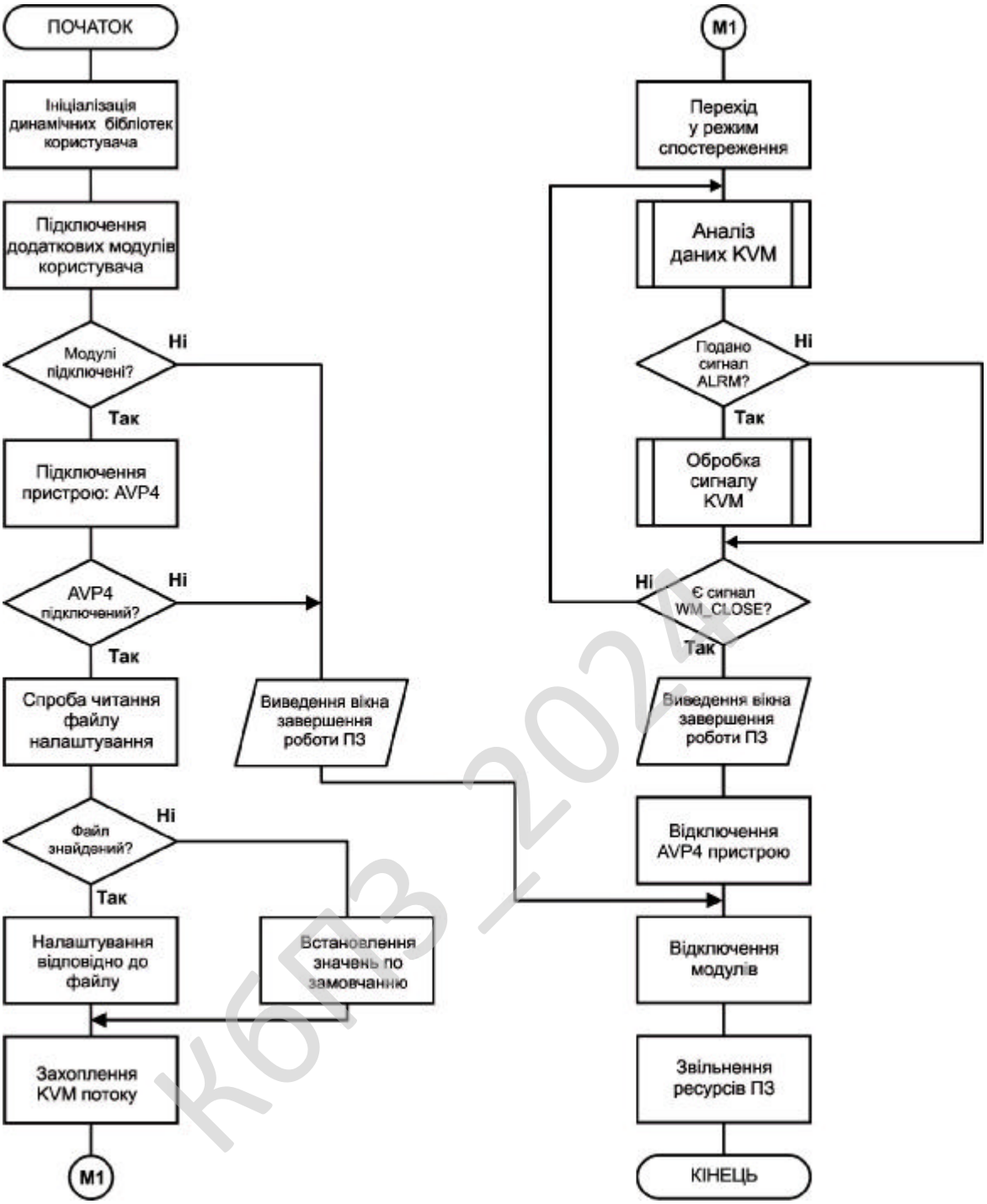

Рисунок 4.1 – Блок схема основної програми

Клас TSample_Grabber

Оголошення класу:

```
type TSample_Grabber = class(A,B,C)
```

Опис:

Клас дозволяє створювати знімки екрану і знімки аудіо ділянок.

Клас TVideo_Window

Оголошення класу:

```
type TVideo_Window = class(A,B,C)
```

Опис:

Управління висновком потокового відео на екран додатку.

Клас TVMR_Bitmap

Оголошення класу:

```
type TVMR_Bitmap = class
```

Опис:

Коректування вида матриці формату BMP.

Клас TASF_Writer

Оголошення класу:

```
Type TASF_Writer = class(A,B)
```

Опис:

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

Клас TColor_Control

Оголошення класу:

```
type TColor_Control = 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 Track_Bar

Оголошення класу:

```
type TTrackBar = class(A,B)
```

Опис:

Панель коректування відео потоку інформації.

Клас TVideo_WindowEx2

Оголошення класу:

```
type TVideo_WindowEx2 = class(A,B,C)
```

Опис**:**

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

Клас TVideo_WindowEx2Caps

Оголошення класу:

```
type TVideo_WindowEx2Caps = class()
```

Опис:

Перевірка на сумісність висновку потокової інформації.

Клас TFilterA

Оголошення класу:

```
type TFilterA = class(A,B)
```

Опис:

Клас пропонує можливість створення і підключення до програми додаткових фільтрів.

Клас TFilter_Graph

Оголошення класу:

```
type TFilter_Graph = class()
```

Опис:

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

Типи даних, що використовуються:

– PVMR_Preferences Тип-покажчик на TVMRPreferences.

– TAbstract_AllocatorClass Тип даних абстрактного виділення для класу.

– TGraph_Mode Тип даних для вибору графічного режиму.

– TSeeking_Cap Конкретизує здібності пошуку медіа потоку.

– TSeeking_Caps Конкретизує здібності пошуку медіа потоків.

– TVideo_Mode Тип відео установки для класу TVideoWindow.

– TBitmap_Option Тип даних настройок захоплення кадру

Основні константи, що часто використовуються:

– 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.

При детальному розгляді рисунку 4.1 програма розбита на декілька важливих блоків, таких як:

– Блок ініціалізації динамічних бібліотек користувача.

– Блок підключення додаткових модулів – безкоштовні модулі для взаємодії з апаратурою Dlink, та обробки сигналу KVM.

– Блок читання файлів налаштування і управління ПЗ.

– Блок захоплення KVM потоку.

– Блок очікування дій користувача.

– Блок аналізу даних KVM.

– Блок обробки сигналу KVM.

Два останні блоки є двома підпрограмами відображеними на рисунку 4.2. Підпрограма Блок аналізу даних KVM описує алгоритм перетворення сигналу на потоковий та аналізу потокового відео операторами служби відеонагляду.

Саме в цьому блоці проводиться отримання апаратної інформації з AVP4, перетворення потокової інформації, аналіз кадру відео інформації та встановлення об'єктної чутливості, обробка кадру відео інформації та проведення запиту.

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 висновку інформації на екран. Як відомо для перегляду відеопотоку необхідно не менше 24 кадрів в секунду.

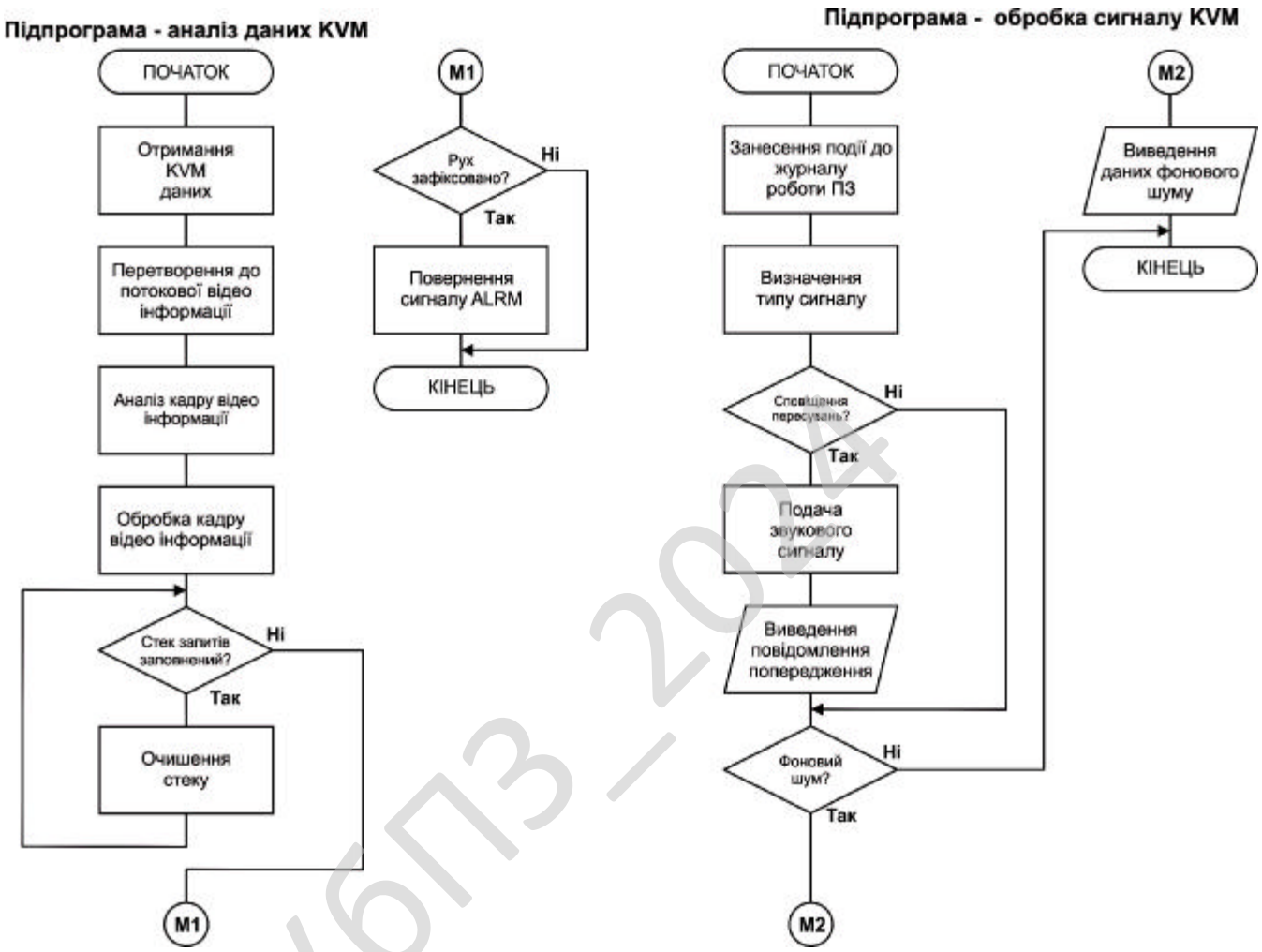

Рисунок 4.2 – Блок схема підпрограм

При застосуванні у бакалаврському проекті багатокрокових алгоритмів обробки кадру для створення відеодатчиків сигналізацій відбувається виведення менше 24 кадрів в секунду, незалежно від потужності персонального комп'ютера, що приводить до величезних проблем при експлуатації програми (поява слайдшоу).

Після зменшення чутливості (залежно від налаштувань) програма дозволяє провести точний розрахунок який дає результат про зміни у відео кадрі.

При поверненні сигналу з підпрограми аналізу даних KVM проходить обробка тривоги яка обробляється у другій підпрограмі обробки сигналу KVM яку можна побачити на рисунку 4.2.

Підпрограма обробки сигналу KVM реалізує процес обробки прапора тривоги (ALRM) зміни зображення, що означає зміни на картинці відеокамери (відеосигнал змінений або спотворений). В першу чергу відбувається запис в журнал роботи програми про зміну з точною вказівкою часу і збереження кадрів результату. Далі залежно від дій операторів походить сповіщення дій – незаконних чи законних.

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму Md5. Він отримує на вході повідомлення довільної довжини і створює на виході дайджест повідомлення довжиною 128 біт. Алгоритм складається з наступних кроків:

1. Додавання недостаючих біт. Повідомлення доповнюється так, щоб його довжина стала рівна 448 по модулю 512 (довжина 448 mod 512). Це означає, що довжина доданого повідомлення на 64 біта менше, ніж число, кратне 512. Додавання проводиться завжди, навіть якщо повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 біт, воно доповнюється 512 бітами до 960 біт. Таким чином, число біт, що додаються, знаходиться в діапазоні від 1 до 512.

Додавання складається з одиниці, за якою слідує необхідна кількість нулів.

2. Додавання довжини. 64-бітове представлення довжини початкового (до додавання) повідомлення в бітах приєднується до результату першого кроку. Якщо первинна довжина більша, ніж 264, то використовуються тільки останні 64 біта. Таким чином, поле містить довжину початкового повідомлення по модулю 264.

В результаті перших двох кроків створюється повідомлення, довжина якого кратна 512 бітам. Це розширене повідомлення представляється як послідовність 512-бітових блоків Y_0, Y_1,..., Y_{l-1}, при цьому загальна довжина розширеного повідомлення рівна L*512 бітам. Таким чином, довжина отриманого розширеного повідомлення кратна шістнадцяти 32-бітовим словам.

3. Ініціалізація MD-буфера. У алгоритмі Md5 використовується 128-бітовий буфер для зберігання проміжних і остаточних результатів хеш-функції. Буфер може бути представлений як чотири 32-бітові регістри (*A*, *B*, *C*, *D*). Ці регістри ініціалізувалися наступними шістнадцятковими числами:

A = 01234567

B = 89abcdef

C = Fedcba98

D = 76543210

4. Обробка послідовності 512-бітових (16-словних) блоків. Основою алгоритму Md5 є модуль, що складається з чотирьох циклічних обробок, позначений як Hmd5. Чотири цикли мають схожу структуру, але кожен цикл використовує свою елементарну логічну функцію, *ff*, що позначається, *fg*, *fh* і *fi* відповідно.

Кожен цикл приймає на вхід поточний 512-бітовий блок Y_q, що обробляється в даний момент, і 128-бітове значення буфера ABCD, яке є проміжним значенням дайджесту, і змінює вміст цього буфера. Кожен цикл також використовує четвертую частина 64-елементної таблиці *T*[1 . 64], побудованої на основі функції sin. *i*-ий елемент *T*, *T*[i], що позначається, має значення, рівне цілій частині від 232 * abs (sin (*i*)), і задане в радіанах. Оскільки abs (sin (*i*)) є числом між 0 і 1, кожен елемент *T* є цілим, яке може бути представлене 32 бітами. Таблиця забезпечує “випадковий” набір 32-бітових значень, які повинні ліквідовувати будь-яку регулярність у вхідних даних.

Для отримання *Mdq*+1 вихід чотирьох циклів складається по модулю 232 з *Mdq*. Складання виконується незалежно для кожного з чотирьох слів в буфері.

5) Вихід Md5. Після обробки всіх *L* 512-бітових блоків виходом *L*-ої стадії є 128-бітовий дайджест повідомлення.

Детальніше логіку кожного з чотирьох циклів виконання одного 512-бітового блоку розглянуто нижче. Кожен цикл складається з 16 кроків, що оперують з буфером *ABCD*.

$$A \quad B + \text{Clss}\,(A + f\,(B, C, D) + X\,[k] + T\,[i]),$$

A, *B*, *C*, *D* – чотири слова буфера; після виконання кожного окремого кроку відбувається циклічне зрушення вліво на одне слово.

f – одна з елементарних функцій *ff*, *fg*, *fh*, *fi*.

CLSs – циклічне зрушення вліво на s біт 32-бітового аргументу.

X [*k*] – *M* [*q* * 16 + *k*] – кодує 32-бітове слово в *q*-ому 512 блоці повідомлення.

T [*i*] – *i*-е 32-бітове слово в матриці T.

+ – складання по модулю 232.

На кожному з чотирьох циклів алгоритму використовується одна з чотирьох елементарних логічних функцій. Кожна елементарна функція отримує три 32-бітові слова на вході і на виході створює одне 32-бітове слово. Кожна функція є безліччю побітових логічних операцій, тобто *n*-ий біт виходу є функцією від n-ого біта трьох входів. Елементарні функції наступні:

$$ff = (B \,\&\, C) \quad (\text{not } B \,\&\, D)$$

$$fg = (B \,\&\, D)\ \text{V}\ (C \,\&\, \text{not } D)$$

$$fh = B \quad C \quad D$$

$$fi = C \quad (B \,\&\, \text{not } D)$$

Масив з 32-бітових слів *X* [0..15] містить значення поточного 512-бітового вхідного блоку, який обробляється зараз. Кожен цикл виконується 16 разів, а оскільки кожен блок вхідного повідомлення обробляється в чотирьох циклах, то кожен блок вхідного повідомлення обробляється по схемі 64 рази. Якщо представити вхідний 512-бітовий блок у вигляді шістнадцяти 32-бітових слів, то кожне вхідне 32-бітове слово використовується чотири рази, по одному разу в

кожному циклі, і кожен елемент таблиці T, що складається з 64 32-бітових слів, використовується тільки один раз. Після кожного кроку циклу відбувається циклічне зрушення вліво чотирьох слів *A*, *B*, *C* і *D*. На кожному кроці змінюється тільки одне з чотирьох слів буфера *ABCD*. Отже, кожне слово буфера змінюється 16 разів, і потім 17-й раз в кінці для отримання остаточного виходу даного блоку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно ПЗ, у режимі чекання сигналу з обладнання Dlink. Якщо дані надходять на екран виводяться відео дані, як це зображено на рисунку 5.2.

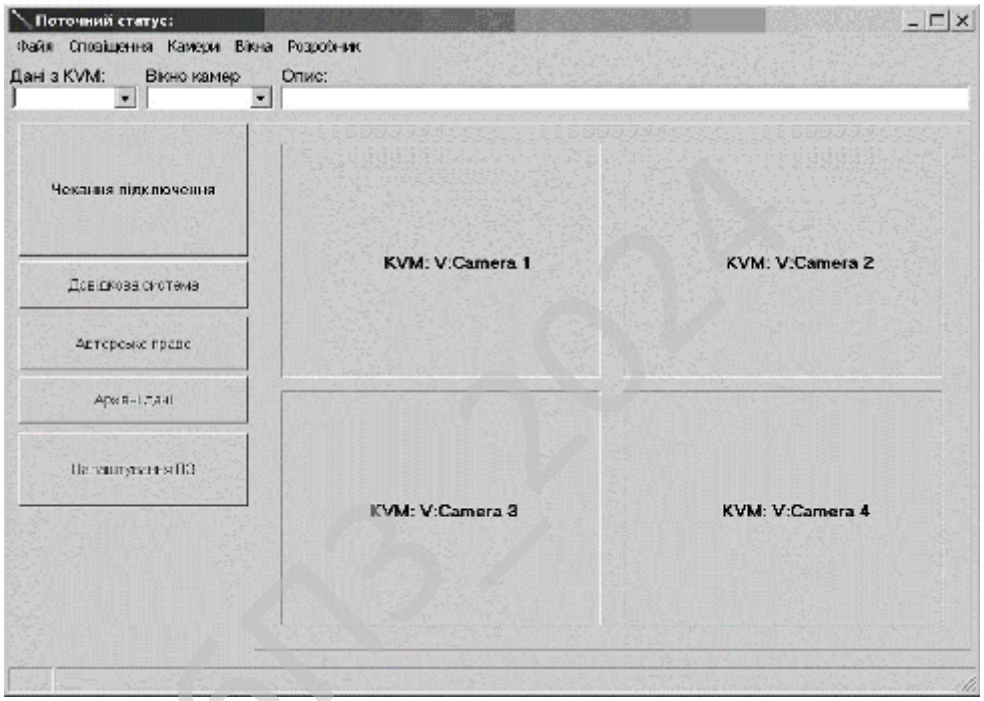

Рисунок 5.1 – Головне вікно ПЗ, чекання сигналу з обладнання Dlink

Розглянемо основне вікно розробленої програми. Воно складається з навігаційних та функціональних клавіш лівої частини вікна:

– Чекання підключення.

– Довідкова система.

– Авторське право.

– Архівні дані.

– Налаштування ПЗ.

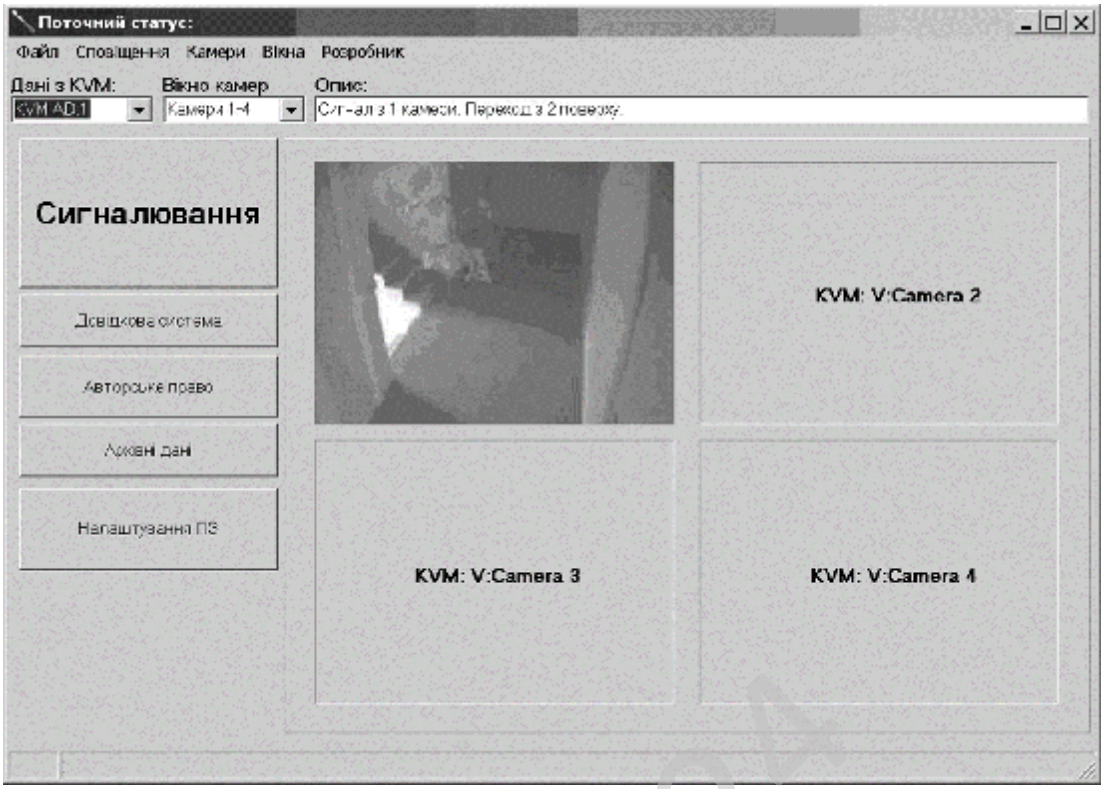

Рисунок 5.2 – Головне вікно ПЗ, отримання відеосигналу

На верхній частині вікна можна побачити навігаційне меню, яке складається з:

– Файл.

– Сповіщення.

– Камери.

– Вікна.

– Розробник.

У верхній частині вікна знаходяться перемикачі основного сигналу KVM:

– Дані з KVM.

– Вікно камер.

– Опис.

Розроблене програмне забезпечення, рекомендується для впровадження в банківських установах, організаціях, фірмах що займаються фінансовою діяльністю та потребують у відео спостереженні.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем.

Кожен, хто в достатньому обсязі володіє операційним середовищем Windows 10/11 без особливих складностей освоїть і цю програму, оскільки її інтерфейс повністю розроблений під дане операційне середовище.

Для налаштування системи на оптимальні умови роботи необхідно виконати наступні кроки:

1. Придбати пристрій для спостереження.
2. Придбати цифровий відео регістратор.
3. Придбати мережне устаткування.
4. Придбати пристрій для обробки аналогового зображення.
5. Встановити драйвер, що поставляється з обладнанням.
6. Встановити DirectX.
7. Встановити розроблене ПЗ.
8. Запустити ПЗ.

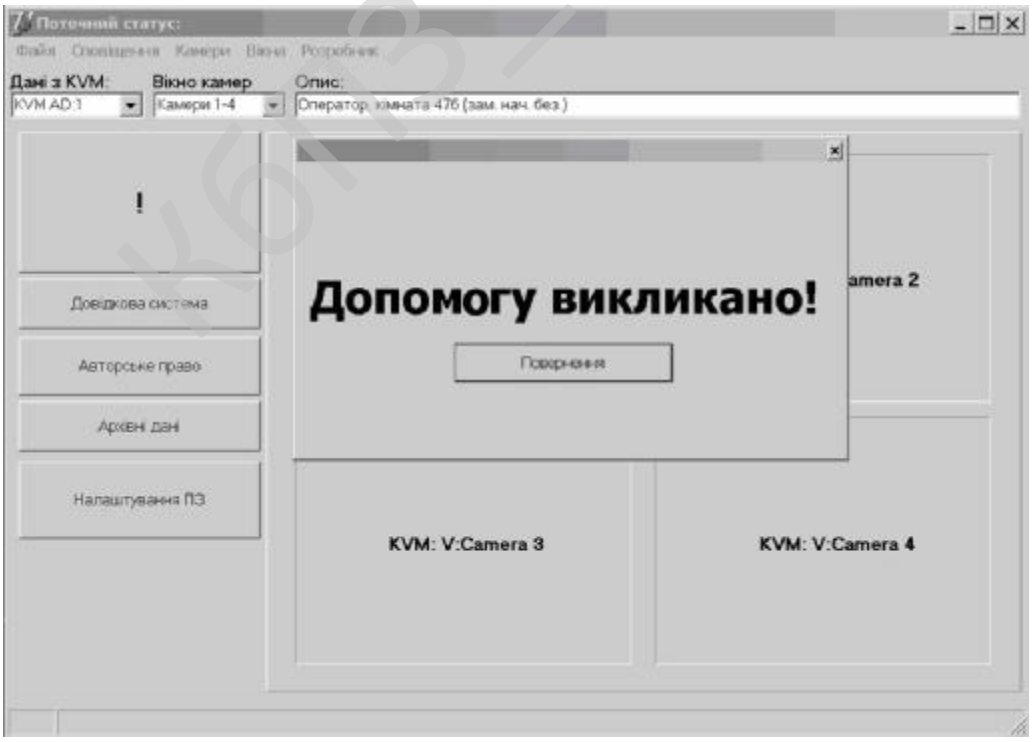

Рисунок 5.3 – Головне вікно ПЗ, подача сигналу

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма. На рисунку 5.4 зображено авторські дані розробленого програмного забезпечення.

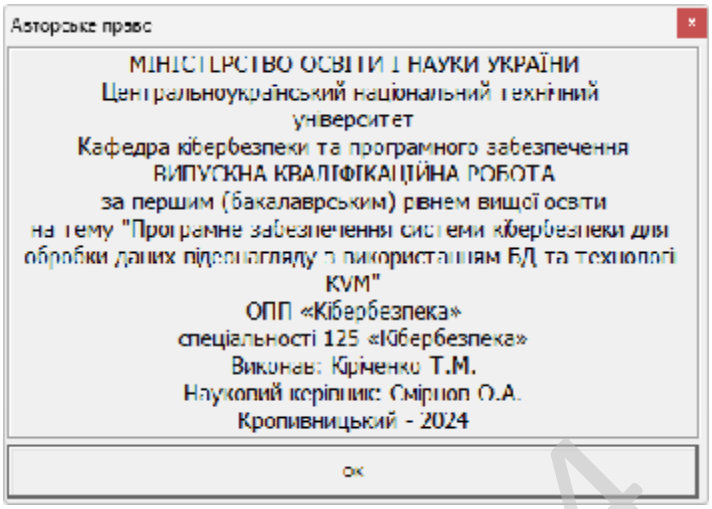

Рисунок 5.4 – Авторське право

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для обробки даних відеонагляду з використанням БД та технології KVM.

– Досліджена система для обробки даних відеонагляду з використанням БД та технології KVM.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для обробки даних відеонагляду з використанням БД та технології KVM.

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 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM. 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 Md5.

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

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

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

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

5. 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.

6. 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.

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

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

9. 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.

10. 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.

11. 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.

12. 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,

13. 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.

14. 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

15. 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.

16. 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.

17. 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.

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

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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).

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для обробки даних відеонагляду з використанням БД та технології KVM.

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 для обробки даних відеонагляду з використанням БД та технології KVM.

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 для обробки даних відеонагляду з використанням БД та технології KVM;

– 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 забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 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 10.4.1.

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 – 73 аркушів.

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 4.06.2024 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 для обробки даних відеонагляду з використанням БД та технології KVM

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 40

Літера: РП

Кропивницький – 2024 року

Файл основного файлу програми Videonaglyad_KVM_Project.dpr

```
program Videonaglyad_KVM_Project;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//

uses

  Forms,
  SysUtils,
  Unit1 in 'Unit1.pas' {Form1},
  frmLOG in 'G.pas' {Form3},
  frmLOG_GRAPH in 'G_GRAPH.pas' {Form4},
  frmSETTINGS in 'SS.pas' {Form2},
  frmSPLASH in 'SPLASH.pas' {Form5},
  frmAbout in 'About.pas' {AboutBox};

{$R *.res}

Begin

    Form5:=TForm5.Create(Application);
    Form5.Show;
    Form5.Update;
try

  Application.HintPause:=200;//ms
  Application.HintHidePause:=7000;//ms
  Application.HintShortPause:=25;//ms

  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
        Application.CreateForm(TForm3, Form3);
        Application.CreateForm(TForm4, Form4);
        Application.CreateForm(TForm2, Form2);
        Application.CreateForm(TAboutBox, AboutBox);
  finally
  Form5.Free;
end;
Application.Run;

end.
```

Файл модулю роботи ПЗ frmLOG.pas

```
unit frmLOG;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//

interface

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

type
  TForm3 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Button1: TButton;
    Memo1: TMemo;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

uses Unit1;

{$R *.dfm}

procedure TForm3.Button1Click(Sender: TObject);
begin
      Form1.show;
      Form3.hide;
end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  Form3.Memo1.lines.add('Програма успішно запрацювала: '+DateToStr(now));

end;

procedure TForm3.Button2Click(Sender: TObject);
begin

  Form3.Memo1.lines.Clear;

end;

end.
```

Під час написання програми були використанні наступні стандартні бібліотеки та модулі для підтримки технології відеотехнології (DirectX):

comlite.h, errors.h, dv.h, strmif.h, mmstream.h, amstream.h,
ddstream.h, austream.h, mpconfig.h, control.h, qnetwork.h,
playlist.h, il21dec.h, amvideo.h, amaudio.h, vptype.h,
vpconfig.h, vpnotify.h, mpegtype.h, dvdevcod.h, dvdmedia.h,
bdatypes.h, activecf.h, vfwmsgs.h,(edevdefs.h, XPrtDefs.h),
aviriff.h, evcode.h, uuids.h, ksuuids.h, DXVA.h,AMVA.h,
videoacc.h, regbag.h, tuner.h, DXTrans.h, QEdit.h, mpeguids.h,
dshowasf.h, amparse.h, audevcod.h, atsmedia.h, MediaErr,
MedParam.h, mediaobj.h, dmodshow.h, dmoreg.h, DMORt.h,

dmoimpl.h, ks.h, ksproxy.h, ksmedia.h, dmksctrl.h, bdamedia.h,
BDATIF.idl, AMVPE.idl, Mixerocx.idl, Mpeg2Data.idl,
Mpeg2Structs.idl, Mpeg2Bits.h, Mpeg2Error.h, EDevCtrl.h,
sbe.idl, wmdxva.h, vmr9.idl

Файл основної форми програми Unit.pas

```
unit Unit1;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ComCtrls, ExtCtrls, StdCtrls, DSPack, DSUtil, DirectShow9,
  jpeg, Buttons;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    StatusBar1: TStatusBar;
    MainMenu1: TMainMenu;
    N1231: TMenuItem;
    Panel2: TPanel;
    Panel3: TPanel;
    VideoWindow: TVideoWindow;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    Button9: TButton;
    Button10: TButton;
    CaptureGraph: TFilterGraph;
    SaveDialog: TSaveDialog;
    StartButton: TBitBtn;
    Image1: TImage;
    Timer1: TTimer;
    StopButton: TBitBtn;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    N11: TMenuItem;
    N12: TMenuItem;
    N13: TMenuItem;
    N14: TMenuItem;
    N15: TMenuItem;
    N16: TMenuItem;
    N17: TMenuItem;
    N18: TMenuItem;
    N19: TMenuItem;
    N20: TMenuItem;
    N21: TMenuItem;
    N22: TMenuItem;
    N23: TMenuItem;
    N24: TMenuItem;
    N25: TMenuItem;
    N26: TMenuItem;
    N27: TMenuItem;
    N28: TMenuItem;
```

```
    Timer2: TTimer;
    Button11: TButton;
    Label1: TLabel;
    SampleGrabber: TSampleGrabber;
    Label2: TLabel;
    SaveDialog1: TSaveDialog;
    BitBtn1: TBitBtn;
    Image: TImage;
    procedure Button9Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure StartButtonClick(Sender: TObject);
    procedure StopButtonClick(Sender: TObject);
    procedure N8Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure N16Click(Sender: TObject);
    procedure N18Click(Sender: TObject);
    procedure N20Click(Sender: TObject);
    procedure N22Click(Sender: TObject);
    procedure N24Click(Sender: TObject);
    procedure N26Click(Sender: TObject);
    procedure N28Click(Sender: TObject);
    procedure N9Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SampleGrabberBuffer(sender: TObject; SampleTime: Double;
      pBuffer: Pointer; BufferLen: Integer);
    procedure Button11Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    procedure ShowCAPdevise;
    procedure SETKAS;
  end;

var
  Form1: TForm1;
  SETKA:integer;
  CapEnum: TSysDevEnum;
  VideoMediaTypes, AudioMediaTypes: TEnumMediaType;
  CapFile: WideString = 'c:\zaxvat.avi';
  Ok_normal:boolean;
  AR: array of array of tcolor;
  GO:boolean;

implementation

uses frmAbout, frmLOG, frmSETTINGS;

{$R *.dfm}

procedure TForm1.Button9Click(Sender: TObject);
begin
      Form1.hide;
      Form3.Memo1.lines.add('Вызов просмотра о программе');
      AboutBox.show;
end;

procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
      MessageDlg('',mtInformation,[mbOK],0);
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
      Form1.hide;
      Form3.show;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  S:string;
begin
  InputQuery('Внимание','Введите путь для сохранения видео',S);
  CapFile:=S;
  Form1.hide;
  Form2.show;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;
end;

procedure TForm1.Button10Click(Sender: TObject);
begin
  Form1.Close;
end;

procedure TForm1.ShowCAPdevise;
var
  i:integer;
begin
      Form2.VideoCapFilters.Items.Clear;
      Form2.AudioCapFilters.Items.Clear;
      CapEnum := TSysDevEnum.Create(CLSID_VideoInputDeviceCategory);
      for i := 0 to CapEnum.CountFilters - 1 do
      Form2.VideoCapFilters.Items.Add(CapEnum.Filters[i].FriendlyName);
      CapEnum.SelectGUIDCategory(CLSID_AudioInputDeviceCategory);
      for i := 0 to CapEnum.CountFilters - 1 do
      Form2.AudioCapFilters.Items.Add(CapEnum.Filters[i].FriendlyName);
      VideoMediaTypes := TEnumMediaType.Create;
      AudioMediaTypes := TEnumMediaType.Create;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var
  position: int64;
  Hour, Min, Sec, MSec: Word;
const MiliSecInOneDay = 86400000;
begin
  if CaptureGraph.Active then
  begin
    with CaptureGraph as IMediaSeeking do
      GetCurrentPosition(position);
    DecodeTime(position div 10000 / MiliSecInOneDay, Hour, Min, Sec, MSec);
    StatusBar1.panels[1].Text := Format('%d:%d:%d:%d',[Hour, Min, Sec, MSec]);
  end;
end;

procedure TForm1.StartButtonClick(Sender: TObject);
var
  multiplexer: IBaseFilter;
  Writer: IFileSinkFilter;
  PinList: TPinList;
  i: integer;
begin
```

```
  // Активізуйте графічні фільтри, на цій стадії додані початкові графічні
фільтри
  CaptureGraph.Active := true;

  // конфігурація виходу аудіо
  if Form2.AudioSourceFilter.FilterGraph <> nil then
  begin
    PinList := TPinList.Create(Form2.AudioSourceFilter as IBaseFilter);
    i := 0;
    while i < PinList.Count do
      if PinList.PinInfo[i].dir = PINDIR_OUTPUT then
        begin
          if Form2.AudioFormats.ItemIndex <> -1 then
            with (PinList.Items[i] as IAMStreamConfig) do

SetFormat(AudioMediaTypes.Items[Form2.AudioFormats.ItemIndex].AMMediaType^);
PinList.Delete(i);
        end else inc(i);
    if Form2.InputLines.ItemIndex <> -1 then
      with (PinList.Items[Form2.InputLines.ItemIndex] as IAMAudioInputMixer) do
        put_Enable(true);
    PinList.Free;
  end;

  // конфігурація виходу відео
if Form2.VideoSourceFilter.FilterGraph <> nil then
begin
      PinList := TPinList.Create(Form2.VideoSourceFilter as IBaseFilter);
      if Form2.VideoFormats.ItemIndex <> -1 then
      with (PinList.First as IAMStreamConfig) do
   SetFormat(VideoMediaTypes.Items[Form2.VideoFormats.ItemIndex].AMMediaType^);
      PinList.Free;
end;


// обробка потоку
with CaptureGraph as IcaptureGraphBuilder2 do
  begin
    // задання файлу запису
SetOutputFileName(MEDIASUBTYPE_Avi, PWideChar(CapFile), multiplexer, Writer);
    // Підключаємо перегляд відео(VideoWindow)
if Form2.VideoSourceFilter.BaseFilter.DataLength > 0 then
RenderStream(@PIN_CATEGORY_PREVIEW, nil, Form2.VideoSourceFilter as IBaseFilter,
             nil , VideoWindow as IBaseFilter);
            // Підключаємо відео поток
            if Form2.VideoSourceFilter.FilterGraph <> nil then
RenderStream(@PIN_CATEGORY_CAPTURE, nil, Form2.VideoSourceFilter as IBaseFilter,
                nil, multiplexer as IBaseFilter);

            // Підключаємо аудіо поток

            if Form2.AudioSourceFilter.FilterGraph <> nil then
            begin

RenderStream(nil, nil, Form2.AudioSourceFilter as IBaseFilter,
             nil, multiplexer as IBaseFilter);
            end;
  end;


  CaptureGraph.Play;
  StopButton.Enabled := true;
  StartButton.Enabled := false;
  Form2.AudioFormats.Enabled := false;
  Form2.AudioCapFilters.Enabled := false;
  Form2.VideoFormats.Enabled := false;
  Form2.VideoCapFilters.Enabled := false;
```

```
  Go:=false;
  Timer1.Enabled := true;
  VideoWindow.Visible:=false;
end;

procedure TForm1.StopButtonClick(Sender: TObject);
begin

  Timer1.Enabled := false;
  StopButton.Enabled := false;
  StartButton.Enabled := true;
  CaptureGraph.Stop;
  CaptureGraph.Active := False;
  Form2.AudioFormats.Enabled := true;
  Form2.AudioCapFilters.Enabled := true;
  Form2.VideoFormats.Enabled := true;
  Form2.VideoCapFilters.Enabled := true;

end;

procedure TForm1.N8Click(Sender: TObject);
begin
      Form1.Close;
end;

procedure TForm1.N4Click(Sender: TObject);
begin

      Form1.hide;
      AboutBox.show;
end;

procedure TForm1.N6Click(Sender: TObject);
begin
      MessageDlg('',mtInformation,[mbOK],0);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
  Form1.hide;
 Form2.show;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin



if Button7.Caption='Ввімкнути детектор переміщення' then
begin
  Button7.Caption:='Вимкнути детектор переміщення';
  Go:=false;
  Timer2.Enabled:=true;
end
else
begin
  Button7.Caption:='Ввімкнути детектор переміщення';
  Timer2.Enabled:=false;
end;

end;

procedure TForm1.Button8Click(Sender: TObject);
```

```
begin
  Form1.hide;
  Form2.show;

end;

procedure TForm1.N16Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;
end;

procedure TForm1.N18Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;

end;

procedure TForm1.N20Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;

end;

procedure TForm1.N22Click(Sender: TObject);
begin
Form1.hide;
Form3.show;
end;

procedure TForm1.N24Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;
end;

procedure TForm1.N26Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;
end;

procedure TForm1.N28Click(Sender: TObject);
begin
  Form1.hide;
  Form2.show;
end;

procedure TForm1.N9Click(Sender: TObject);
begin
Application.Minimize;
end;

procedure TForm1.N10Click(Sender: TObject);
begin
if Ok_normal then
  begin
    MessageDlg('Перевірка пройшла успішно',mtInformation,[mbOK],0);
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
X,Y:integer;
begin
Ok_normal:=true;
SETKA:=50;
```

```
VideoWindow.Canvas.Pen.Color:=clLime;
VideoWindow.Canvas.Pen.Width:=4;
  BL:=1; BLyYa:=1;

        for x :=1 to 320 do
          begin
                  if (x mod SETKA)=0 then
                  begin
                      inc(BL);
                  end;
          end;

            for y :=1 to 240 do
              begin
                  if (y mod SETKA)=0 then
                  begin
                      inc(BL);
                  end;
              end;
SetLength(Ar,320,240);
end;

procedure TForm1.SampleGrabberBuffer(sender: TObject; SampleTime: Double;

pBuffer: Pointer; BufferLen: Integer);
begin
    Image.Canvas.Lock;
    try
      SampleGrabber.GetBitmap(Image.Picture.Bitmap, pBuffer, BufferLen);
      SETKAS;
    finally
      Image.Canvas.Unlock;
    end;

   SampleGrabber.GetBitmap(Image2.Picture.Bitmap, pBuffer, BufferLen);

   Image2.Canvas.Rectangle(1,1,400,400);
end;

procedure TForm1.Button11Click(Sender: TObject);
begin
if SaveDialog1.Execute then
  begin
     if SaveDialog1.FileName<>'' then
       begin
         SampleGrabber.GetBitmap(Image.Picture.Bitmap);
         Image.Picture.SaveToFile(SaveDialog1.FileName);
       end;
  end;
end;

procedure TForm1.SETKAS;
var
  x,y:integer;
  G:tcolor;
begin

//X VideoWindow.ClientWidth
//Y VideoWindow.ClientHeight
//VideoWindow.Canvas.Pixels[]
//  Label1.Caption:=ColorToString(Image.Canvas.Pixels[20,20]);

//-------------------------------------------------------------------------if go
and ((Image.Picture.Width<>0) and (Image.Picture.Height<>0))then
  begin
    {
      for x :=1 to Image.Picture.Width do
          begin
            for y :=1 to Image.Picture.Height do
```

```
              begin
                  if ((y mod SETKA)=0) and ((x mod SETKA)=0) then
                  begin
                     Image.Canvas.Rectangle(x,y,x+10,y+10);

Label1.Caption:=ColorToString(VideoWindow.Canvas.Pixels[x,y]);

 /////*********************************

                     if Ar[X,Y]<>Image.Canvas.Pixels[x,y] then
                     begin

                          Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                          Form3.Memo1.lines.add('изм г на г');
                     end;

/////*********************************

                  end;

                  if ((y mod SETKA)=0) and (x = 1) then
                  begin
                     //Image.Canvas.Rectangle(x,y,x+4,y+4);
                     /////*********************************
                     if Ar[X,Y]<>Image.Canvas.Pixels[x,y] then
                     begin
                        Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                          Form3.Memo1.lines.add('изм г на 1');
                     end;
                     /////*********************************
                  end;


                  if (y = 1) and ((x mod SETKA)=0) then
                  begin
                     //Image.Canvas.Rectangle(x,y,x+4,y+4);
                     /////*********************************
                     if Ar[X,Y]<>Image.Canvas.Pixels[x,y] then
                     begin
                        Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                        Form3.Memo1.lines.add('изм 1 на г');
                     end;

/////*********************************

                  end;


                  if (y = 1) and (x = 1) then
                  begin
                     //Image.Canvas.Rectangle(x,y,x+4,y+4);
                     /////*********************************
                     if Ar[X,Y]<>Image.Canvas.Pixels[x,y] then
                     begin
                        Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                           Form3.Memo1.lines.add('изм 1 на 1');
                     end;

 /////*********************************

                  end;
              end;
          end  }
  end;
//-----------------------------------------------------------------------

if (not go)and ((Image.Picture.Width<>0) and (Image.Picture.Height<>0)) then
  begin
        for x :=1 to Image.Picture.Width do
```

```
          begin
            for y :=1 to Image.Picture.Height do
             begin
                  if ((y mod SETKA)=0) and ((x mod SETKA)=0) then
                  begin
                     Image.Canvas.Rectangle(x,y,x+10,y+10);

Label1.Caption:=ColorToString(VideoWindow.Canvas.Pixels[x,y]);
                     Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                  end;

                  if ((y mod SETKA)=0) and (x = 1) then
                  begin
                     Image.Canvas.Rectangle(x,y,x+4,y+4);
                     Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                  end;


                  if (y = 1) and ((x mod SETKA)=0) then
                  begin
                     //Image.Canvas.Rectangle(x,y,x+4,y+4);
                     Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                  end;


                  if (y = 1) and (x = 1) then
                  begin
                     //Image.Canvas.Rectangle(x,y,x+4,y+4);
                     Ar[X,Y]:=Image.Canvas.Pixels[x,y];
                  end;
             end;
          end
  end;
//---------------------------------------------------------------------------

go:=true;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Label1.Caption:=inttostr(Image.Picture.Width);
  Label2.Caption:=inttostr(Image.Picture.Height);
end;

end.
```

Файл модулю зберігання відеокадрів (переадресація) frmLOG_GRAPH.pas

```
unit frmLOG_GRAPH;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//

interface

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

type
  TForm4 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Button1: TButton;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form4: TForm4;
implementation
{$R *.dfm}

end.
```

Файл настроювання захоплення видеопотока frmSETTINGS.pas

```
unit frmSETTINGS;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, ExtCtrls, StdCtrls, DSPack, DSUtil, DirectShow9;

type
  TForm2 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Button1: TButton;
    VideoSourceFilter: TFilter;
    AudioSourceFilter: TFilter;
    AudioFormats: TListBox;
    Label4: TLabel;
    VideoFormats: TListBox;
    Label3: TLabel;
    AudioCapFilters: TListBox;
    Label2: TLabel;
    VideoCapFilters: TListBox;
    Label1: TLabel;
    InputLines: TComboBox;
    procedure Button1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure VideoCapFiltersClick(Sender: TObject);
    procedure AudioCapFiltersClick(Sender: TObject);
    procedure VideoFormatsClick(Sender: TObject);
    procedure AudioFormatsClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

uses Unit1, frmLOG;

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
  Form1.show;
  Form2.hide;
end;

procedure TForm2.FormShow(Sender: TObject);

begin
    Form1.ShowCAPdevise;

end;

procedure TForm2.VideoCapFiltersClick(Sender: TObject);
```

```
var
  PinList: TPinList;
  i: integer;

begin

  CapEnum.SelectGUIDCategory(CLSID_VideoInputDeviceCategory);
  if VideoCapFilters.ItemIndex <> -1 then
  begin

    VideoSourceFilter.BaseFilter.Moniker :=
    CapEnum.GetMoniker(VideoCapFilters.ItemIndex);

    VideoSourceFilter.FilterGraph := Form1.CaptureGraph;
    Form1.CaptureGraph.Active := true;

    PinList := TPinList.Create(VideoSourceFilter as IBaseFilter);
    VideoFormats.Clear;

    VideoMediaTypes.Assign(PinList.First);
    for i := 0 to VideoMediaTypes.Count - 1 do
    VideoFormats.Items.Add(VideoMediaTypes.MediaDescription[i]);

    Form1.CaptureGraph.Active := false;
    PinList.Free;

    Form1.StartButton.Enabled := true;
     Form3.Memo1.lines.add('Установка
     '+AudioCapFilters.Items.Strings[VideoCapFilters.ItemIndex]);
end;
end;

procedure TForm2.AudioCapFiltersClick(Sender: TObject);

var
  PinList: TPinList;
  g,i, LineIndex: integer;
  ABool: LongBool;

begin

  CapEnum.SelectGUIDCategory(CLSID_AudioInputDeviceCategory);
  if AudioCapFilters.ItemIndex <> -1 then

  begin
     AudioSourceFilter.BaseFilter.Moniker :=
     CapEnum.GetMoniker(AudioCapFilters.ItemIndex);

    AudioSourceFilter.FilterGraph := Form1.CaptureGraph;
    Form1.CaptureGraph.Active := true;

PinList := TPinList.Create(AudioSourceFilter as IBaseFilter);
    AudioFormats.Clear;
    i := 0;
    while i < PinList.Count do
      if PinList.PinInfo[i].dir = PINDIR_OUTPUT then
        begin
          AudioMediaTypes.Assign(PinList.Items[i]);
          PinList.Delete(i);
        end else inc(i);

    for i := 0 to AudioMediaTypes.Count - 1 do
    begin
      AudioFormats.Items.Add(AudioMediaTypes.MediaDescription[i]);
    end;

    Form1.CaptureGraph.Active := false;
    InputLines.Clear;
    LineIndex := -1;
```

```
    for i := 0 to PinList.Count - 1 do
    begin
      InputLines.Items.Add(PinList.PinInfo[i].achName);
      with (PinList.Items[i] as IAMAudioInputMixer) do get_Enable(ABool);
      if ABool then LineIndex := i;
    end;

    InputLines.ItemIndex := LineIndex;
    PinList.Free;
    Form1.StartButton.Enabled := true;
    g:=AudioCapFilters.ItemIndex;

    Form3.Memo1.lines.add('Установка '+AudioCapFilters.Items.Strings[g]);

  end;
end;

procedure TForm2.VideoFormatsClick(Sender: TObject);
begin
  if VideoCapFilters.ItemIndex <> -1 then
  begin

Form3.Memo1.lines.add('Видео'+VideoCapFilters.Items.Strings[VideoCapFilters.Item
Index]);

  end;
end;


procedure TForm2.AudioFormatsClick(Sender: TObject);
begin
  if AudioFormats.ItemIndex <> -1 then
  begin

Form3.Memo1.lines.add('Видео

'+AudioFormats.Items.Strings[AudioFormats.ItemIndex]);

  end;
end;

end.
```

Файл модулю підключення плагінів frmPLUG.pas

```
unit frmPLUG;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
type
  TForm_9 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
     Panel4: TPanel;
     Panel5: TPanel;
     DoT_A:TpluginsAddStrings;
     DoT_B:TpluginsAddFiles;
     DoT_C:TpluginsAddData;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form9: TForm9;

implementation

{$R *.dfm}

end.
```

Файл модулю вспиливаючого вікна програми frmSPLASH.pas

```
unit frmSPLASH;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//

interface

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

type
  TForm5 = class(TForm)
    Image1: TImage;
    ScrollBox1: TScrollBox;
  private
    { Private declarations }

  public
    { Public declarations }

  end;

var
  Form5: TForm5;

implementation

{$R *.dfm}

end.
```

Файл модулю авторських прав frmAbout.pas

```
unit frmAbout;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls;
type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  AboutBox: TAboutBox;
implementation
uses Unit1;
{$R *.dfm}
procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
      Form1.show;
      AboutBox.hide;
end;
end.
```

Файл бібліотеки захвату потоку відео DSPack.pas

```
unit DSPack;
//
// Copyright (C) Кіріченко Тетяна Миколаївна
// 2024
// Програма на бакалаврську роботу
//

Interface

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

const
  WM_GRAPHNOTIFY = WM_APP + 1;

  WM_CAPTURE_BITMAP = WM_APP + 2;
type

  TVideoMode = (
    vmNormal,
    vmVMR
  );

  TGraphMode = (
    gmNormal,
    gmCapture,
    gmDVD
  );


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

  {@exclude}
  TGraphState = (
    gsUninitialized,
    gsStopped,
    gsPaused,
    gsPlaying
  );

  { }
  TSeekingCap = (
    CanSeekAbsolute,   // Потік може переміститися у абсолютну позицію.
    CanSeekForwards,   // Потік може переміститися вперед.
    CanSeekBackwards,  // Потік може переміститися назад.
    CanGetCurrentPos,  // Потік може дати картинку з поточної позиції.
    CanGetStopPos,     // Потік може дати картинку зі стоп-кадру.
    CanGetDuration,    // Потік може повідомити тривалість запису.
    CanPlayBackwards,  // Потік може програти назад.
```

```
    CanDoSegments,     // Потік може зациклитись (see
IMediaSeeking.SetPositions).
    Source             // Reserved.
  );
  {. }
  TSeekingCaps = set of TSeekingCap;


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

  { Pointer to @link(TVMRPreferences).}
  PVMRPreferences = ^TVMRPreferences;
  { Set of @link(TVMRPreference).}
  TVMRPreferences = set of TVMRPreference;

TOnDSEvent=procedure(sender: TComponent; Event, Param1, Param2: Integer) of
object;
{@exclude}
TOnGraphBufferingData=procedure(sender: TObject; Buffering: boolean) of object ;
{@exclude}
TOnGraphComplete=procedure(sender: TObject; Result: HRESULT; Renderer:
IBaseFilter) of object;
  TOnGraphDeviceLost           = procedure(sender: TObject; Device: IUnknown;
Removed: Boolean) of object ;                    {@exclude}
  TOnGraphEndOfSegment         = procedure(sender: TObject; StreamTime:
TReferenceTime; NumSegment: Cardinal) of object ;         {@exclude}

TOnDSResult                   = procedure(sender: TObject; Result: HRESULT) of
object ;                                        {@exclude}
  TOnGraphFullscreenLost       = procedure(sender: TObject; Renderer:
IBaseFilter) of object ;                                  {@exclude}
  TOnGraphOleEvent             = procedure(sender: TObject; String1, String2:
WideString) of object ;                          {@exclude}
  TOnGraphOpeningFile          = procedure(sender: TObject; opening: boolean) of
object ;                                       {@exclude}
  TOnGraphSNDDevError          = procedure(sender: TObject; OccurWhen:
TSndDevErr; ErrorCode: LongWord) of object ;              {@exclude}
  TOnGraphStreamControl        = procedure(sender: TObject; PinSender: IPin;
Cookie: LongWord) of object ;                       {@exclude}
  TOnGraphStreamError          = procedure(sender: TObject; Operation: HRESULT;
Value: LongWord) of object ;                     {@exclude}
  TOnGraphVideoSizeChanged     = procedure(sender: TObject; Width, height: word)
of object ;                                     {@exclude}
  TOnGraphTimeCodeAvailable= procedure(sender: TObject; From: IBaseFilter;
DeviceID: LongWord) of object ;                    {@exclude}
  TOnGraphEXTDeviceModeChange  = procedure(sender: TObject; NewMode, DeviceID:
LongWord) of object ;                              {@exclude}
  TOnGraphVMRRenderDevice= procedure(sender: TObject; RenderDevice:
TVMRRenderDevice) of object;

{@exclude}
  TOnDVDAudioStreamChange      = procedure(sender: TObject; stream, lcid:
Integer; Lang: string) of object;                        {@exclude}
  TOnDVDCurrentTime            = procedure(sender: TObject; Hours,
minutes,seconds,frames,frate : Integer) of object;              {@exclude}
  TOnDVDTitleChange = procedure(sender: TObject; title: Integer) of object;
{@exclude}
TOnDVDChapterStart=procedure(sender: TObject; chapter: Integer) of object;
{@exclude}
  TOnDVDValidUOPSChange = procedure(sender: TObject; UOPS: Integer) of object;
{@exclude}
```

```
  TOnDVDChange = procedure(sender: TObject; total,current: Integer) of object;
{@exclude}
  TOnDVDStillOn = procedure(sender: TObject; NoButtonAvailable: boolean;
seconds: Integer) of object;               {@exclude}
  TOnDVDSubpictureStreamChange = procedure(sender: TObject; SubNum, lcid:
Integer; Lang: string) of object;                        {@exclude}
  TOnDVDPlaybackRateChange = procedure(sender: TObject; rate:  single) of
object;                                                  {@exclude}
  TOnDVDParentalLevelChange= procedure(sender: TObject; level: Integer) of
object; {@exclude}
  TOnDVDAnglesAvailable=procedure(sender: TObject; available: boolean) of
object; {@exclude}
  TOnDVDButtonAutoActivated=procedure(sender: TObject; Button: Cardinal) of
object; {@exclude}
  TOnDVDCMD=procedure(sender: TObject; CmdID: Cardinal) of object;
{@exclude}
  TOnDVDCurrentHMSFTime = procedure(sender: TObject; HMSFTimeCode:
TDVDHMSFTimeCode; TimeCode: TDVDTimeCode) of object;  {@exclude}
  TOnDVDKaraokeMode=procedure(sender: TObject; Played: boolean) of object;
  {@exclude}
  TOnBuffer = procedure(sender: TObject; SampleTime: Double; pBuffer: Pointer;
BufferLen: longint) of object ;

//**********************************************************************
//  IFilter
//**********************************************************************

  {@exclude}
  TFilterOperation = (
    foAdding,    //
    foAdded,     //.
    foRemoving,  //
    foRemoved,   //
    foRefresh    //
  );


  {@exclude}
  IFilter = interface
  ['{887F94DA-29E9-44C6-B48E-1FBF0FB59878}']
    function GetFilter: IBaseFilter;

    function GetName: string;

    procedure NotifyFilter(operation: TFilterOperation; Param: integer = 0);
  end;

  {@exclude}
  TControlEvent = (
    cePlay,
    cePause,
    ceStop,
    ceFileRendering,
    ceFileRendered,
    ceDVDRendering,
    ceDVDRendered,
    ceActive
  );

  {@exclude}
  IEvent = interface
  ['{6C0DCD7B-1A98-44EF-A6D5-E23CBC24E620}']
    { FilterGraph events. }
    procedure GraphEvent(Event, Param1, Param2: integer);
    { Control Events. }
    procedure ControlEvent(Event: TControlEvent; Param: integer = 0);
  end;
```

```
// ****************************************************************************
//  TFilterGraph
//****************************************************************************

  TFilterGraph = class(TComponent)
  private
    FActive     : boolean;
    FAutoCreate : boolean;
    FHandle     : THandle; // подія на картинці
    FMode       : TGraphMode;

    FFilters: TInterfaceList;
    FGraphEvents: TInterfaceList;

    // побудова
    FFilterGraph  : IGraphBuilder;
    FCaptureGraph : ICaptureGraphBuilder2;
    FDVDGraph     : IDvdGraphBuilder;

    // інтерфейс подій
    FMediaEventEx : IMediaEventEx;

    // редагування графіки
    FGraphEdit    : boolean;
    FGraphEditID  : Integer;

    // лог-файл
    FLogFileName: String;
    FLogFile: TFileStream;

    FOnActivate: TNotifyEvent;

    // Коди усіх подій
    FOnDSEvent : TOnDSEvent;
    // Головні графічні події
    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;

    FOnDVDAudioStreamChange       : TOnDVDAudioStreamChange;
    FOnDVDCurrentTime             : TOnDVDCurrentTime;
    FOnDVDTitleChange             : TOnDVDTitleChange;
    FOnDVDChapterStart            : TOnDVDChapterStart;
    FOnDVDAngleChange             : TOnDVDChange;
```

```
    FOnDVDValidUOPSChange         : TOnDVDValidUOPSChange;
    FOnDVDButtonChange            : TOnDVDChange;
    FOnDVDChapterAutoStop         : TNotifyEvent;
    FOnDVDStillOn                 : TOnDVDStillOn;
    FOnDVDStillOff                : TNotifyEvent;
    FOnDVDSubpictureStreamChange  : TOnDVDSubpictureStreamChange;
    FOnDVDNoFP_PGC                : TNotifyEvent;
    FOnDVDPlaybackRateChange      : TOnDVDPlaybackRateChange;
    FOnDVDParentalLevelChange     : TOnDVDParentalLevelChange;
    FOnDVDPlaybackStopped         : TNotifyEvent;
    FOnDVDAnglesAvailable         : TOnDVDAnglesAvailable;
    FOnDVDPlayPeriodAutoStop      : TNotifyEvent;
    FOnDVDButtonAutoActivated     : TOnDVDButtonAutoActivated;
    FOnDVDCMDStart                : TOnDVDCMD;
    FOnDVDCMDEnd                  : TOnDVDCMD;
    FOnDVDDiscEjected             : TNotifyEvent;
    FOnDVDDiscInserted            : TNotifyEvent;
    FOnDVDCurrentHMSFTime         : TOnDVDCurrentHMSFTime;
    FOnDVDKaraokeMode             : TOnDVDKaraokeMode;
    // DVD Warning
    FOnDVDWarningInvalidDVD1_0Disc  : TNotifyEvent;//=1,
    FOnDVDWarningFormatNotSupported : TNotifyEvent;//=2,
    FOnDVDWarningIllegalNavCommand  : TNotifyEvent;//=3
    FOnDVDWarningOpen               : TNotifyEvent;//=4
    FOnDVDWarningSeek               : TNotifyEvent;//=5
    FOnDVDWarningRead               : TNotifyEvent;//=6
    // DVDDomain
    FOnDVDDomainFirstPlay         : TNotifyEvent;
    FOnDVDDomainVideoManagerMenu  : TNotifyEvent;
    FOnDVDDomainVideoTitleSetMenu : TNotifyEvent;
    FOnDVDDomainTitle             : TNotifyEvent;
    FOnDVDDomainStop              : TNotifyEvent;
    // DVDError
    FOnDVDErrorUnexpected                          : TNotifyEvent;
    FOnDVDErrorCopyProtectFail                     : TNotifyEvent;
    FOnDVDErrorInvalidDVD1_0Disc                   : TNotifyEvent;
    FOnDVDErrorInvalidDiscRegion                   : TNotifyEvent;
    FOnDVDErrorLowParentalLevel                    : TNotifyEvent;
    FOnDVDErrorMacrovisionFail                     : TNotifyEvent;
    FOnDVDErrorIncompatibleSystemAndDecoderRegions : TNotifyEvent;
    FOnDVDErrorIncompatibleDiscAndDecoderRegions   : 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
    {@exclude}
    procedure DoEvent(Event, Param1, Param2: Integer); virtual;
    {@exclude}
    procedure InsertFilter(AFilter: IFilter);
    {@exclude}
    procedure RemoveFilter(AFilter: IFilter);
    {@exclude}
```

```
    procedure InsertEventNotifier(AEvent: IEvent);
    {@exclude}
    procedure RemoveEventNotifier(AEvent: IEvent);
  public
    { }
    property Duration: Integer read GetDuration;
    {. }
    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;
    { TFilterGraph constructor. }
    constructor Create(AOwner: TComponent); override;
    { TFilterGraph destructor. }
    destructor Destroy; override;
    { @exclude}
    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 RenderDVD(out status: TAMDVDRenderStatus;
      FileName: WideString = ''; Mode: Integer = AM_DVD_HWDEC_PREFER): HRESULT;
    procedure DVDSaveBookmark(BookMarkFile: WideString);
    procedure DVDRestoreBookmark(BookMarkFile: WideString);
  published

    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;

    // ------------------------------------------------------------------
    // Events
    // ------------------------------------------------------------------

    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 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 OnDVDAudioStreamChange: TOnDVDAudioStreamChange read
FOnDVDAudioStreamChange write FOnDVDAudioStreamChange;

    property OnDVDCurrentTime: TOnDVDCurrentTime read FOnDVDCurrentTime write
FOnDVDCurrentTime;

    property OnDVDTitleChange: TOnDVDTitleChange read FOnDVDTitleChange write
FOnDVDTitleChange;

property OnDVDChapterStart: TOnDVDChapterStart read FOnDVDChapterStart write
FOnDVDChapterStart;


    property OnDVDAngleChange: TOnDVDChange read FOnDVDAngleChange write
FOnDVDAngleChange;

    property OnDVDValidUOPSChange: TOnDVDValidUOPSChange read
FOnDVDValidUOPSChange write FOnDVDValidUOPSChange;

    property OnDVDButtonChange: TOnDVDChange read FOnDVDButtonChange write
FOnDVDButtonChange;

    property OnDVDChapterAutoStop: TNotifyEvent read FOnDVDChapterAutoStop write
FOnDVDChapterAutoStop;


    property OnDVDStillOn: TOnDVDStillOn read FOnDVDStillOn write FOnDVDStillOn;

    property OnDVDStillOff: TNotifyEvent read FOnDVDStillOff write
FOnDVDStillOff;


property OnDVDSubpictureStreamChange: TOnDVDSubpictureStreamChange read
FOnDVDSubpictureStreamChange write FOnDVDSubpictureStreamChange;


    property OnDVDNoFP_PGC: TNotifyEvent read FOnDVDNoFP_PGC write
FOnDVDNoFP_PGC;


    property OnDVDPlaybackRateChange: TOnDVDPlaybackRateChange read
FOnDVDPlaybackRateChange write FOnDVDPlaybackRateChange;

    property OnDVDParentalLevelChange: TOnDVDParentalLevelChange read
FOnDVDParentalLevelChange write FOnDVDParentalLevelChange;

    property OnDVDPlaybackStopped: TNotifyEvent read FOnDVDPlaybackStopped write
FOnDVDPlaybackStopped;

    property OnDVDAnglesAvailable: TOnDVDAnglesAvailable read
FOnDVDAnglesAvailable write FOnDVDAnglesAvailable;

    property OnDVDPlayPeriodAutoStop: TNotifyEvent read FOnDVDPlayPeriodAutoStop
write FOnDVDPlayPeriodAutoStop;

    property OnDVDButtonAutoActivated: TOnDVDButtonAutoActivated read
FOnDVDButtonAutoActivated write FOnDVDButtonAutoActivated;

    property OnDVDCMDStart: TOnDVDCMD read FOnDVDCMDStart Write FOnDVDCMDStart;

    property OnDVDCMDEnd: TOnDVDCMD read FOnDVDCMDEnd Write FOnDVDCMDEnd;
```

```
    property OnDVDDiscEjected: TNotifyEvent read FOnDVDDiscEjected Write
FOnDVDDiscEjected;

    property OnDVDDiscInserted: TNotifyEvent read FOnDVDDiscInserted write
FOnDVDDiscInserted;

    property OnDVDCurrentHMSFTime: TOnDVDCurrentHMSFTime read
FOnDVDCurrentHMSFTime write FOnDVDCurrentHMSFTime;

    property OnDVDKaraokeMode: TOnDVDKaraokeMode read FOnDVDKaraokeMode write
FOnDVDKaraokeMode;

    property OnDVDDomainFirstPlay: TNotifyEvent read FOnDVDDomainFirstPlay write
FOnDVDDomainFirstPlay;

    property OnDVDDomainVideoManagerMenu: TNotifyEvent read
FOnDVDDomainVideoManagerMenu write FOnDVDDomainVideoManagerMenu;

    { Displaying menus for current title set. }
    property OnDVDDomainVideoTitleSetMenu: TNotifyEvent read
FOnDVDDomainVideoTitleSetMenu write FOnDVDDomainVideoTitleSetMenu;

    { Displaying the current title. }
    property OnDVDDomainTitle: TNotifyEvent read FOnDVDDomainTitle write
FOnDVDDomainTitle;

    { The DVD Navigator is in the DVD Stop domain.}
    property OnDVDDomainStop: TNotifyEvent read FOnDVDDomainStop write
FOnDVDDomainStop;

    property OnDVDErrorUnexpected: TNotifyEvent read FOnDVDErrorUnexpected write
FOnDVDErrorUnexpected;
    property OnDVDErrorInvalidDVD1_0Disc: TNotifyEvent read
FOnDVDErrorInvalidDVD1_0Disc write FOnDVDErrorInvalidDVD1_0Disc;

    property OnDVDErrorInvalidDiscRegion: TNotifyEvent read
FOnDVDErrorInvalidDiscRegion write FOnDVDErrorInvalidDiscRegion;

    property OnDVDErrorLowParentalLevel: TNotifyEvent read
FOnDVDErrorLowParentalLevel write FOnDVDErrorLowParentalLevel;

    property OnDVDErrorMacrovisionFail: TNotifyEvent read
FOnDVDErrorMacrovisionFail write FOnDVDErrorMacrovisionFail;

    property OnDVDErrorIncompatibleSystemAndDecoderRegions: TNotifyEvent read
FOnDVDErrorIncompatibleSystemAndDecoderRegions write
FOnDVDErrorIncompatibleSystemAndDecoderRegions;

    property OnDVDErrorIncompatibleDiscAndDecoderRegions: TNotifyEvent read
FOnDVDErrorIncompatibleDiscAndDecoderRegions  write
FOnDVDErrorIncompatibleDiscAndDecoderRegions;


property OnDVDWarningInvalidDVD1_0Disc: TNotifyEvent read
FOnDVDWarningInvalidDVD1_0Disc write FOnDVDWarningInvalidDVD1_0Disc;

    property OnDVDWarningFormatNotSupported : TNotifyEvent read
FOnDVDWarningFormatNotSupported write FOnDVDWarningFormatNotSupported;

    property OnDVDWarningIllegalNavCommand  : TNotifyEvent read
FOnDVDWarningIllegalNavCommand  write FOnDVDWarningIllegalNavCommand;

    property OnDVDWarningOpen: TNotifyEvent read FOnDVDWarningOpen write
FOnDVDWarningOpen;

    property OnDVDWarningSeek: TNotifyEvent read FOnDVDWarningSeek write
FOnDVDWarningSeek;
```

```
    property OnDVDWarningRead: TNotifyEvent read FOnDVDWarningRead write
FOnDVDWarningRead;
  end;


//**********************************************************************
//   TVMROptions
//**********************************************************************

  {@exclude}
  TVideoWindow = class;

  TVMRVideoMode = (
    vmrWindowed,
    vmrWindowless,
    vmrRenderless
  );

  TVMROptions = class(TPersistent)
  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
    { Constructor method. }
    constructor Create(AOwner: TVideoWindow);
  published
    property Mode: TVMRVideoMode read FMode write SetMode;
    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;

//**********************************************************************
//   TVideoWindow
//**********************************************************************
  TAbstractAllocator = class(TInterfacedObject)
    constructor Create(out hr: HResult; wnd: THandle; d3d: IDirect3D9 = nil;
d3dd: IDirect3DDevice9 = nil); virtual; abstract;
  end;
  TAbstractAllocatorClass = class of TAbstractAllocator;

  { Manage a Video Renderer or a Video Mixer Renderer (VMR) Filter to display
    a video in your application. }
  TVideoWindow = class(TCustomControl, IFilter, IEvent)
  private
    FMode          : TVideoMode;
    FVMROptions    : TVMROptions;
    FBaseFilter    : IBaseFilter;
    FVideoWindow   : IVideoWindow; // VMR Windowed & Normal
    FWindowLess    : IVMRWindowlessControl9; // VMR Windowsless

    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
    {@exclude}
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
    { Constructor. }
    constructor Create(AOwner: TComponent);override;
    { Destructor. }
    destructor Destroy; override;
    class function CheckVMR: boolean;
    function VMRGetBitmap(Stream: TStream): boolean;
    function CheckInputPinsConnected: boolean;
    procedure SetAllocator(Allocator: TAbstractAllocatorClass; UserID:
Cardinal);
  published
     property OnPaint: TNotifyEvent read FOnPaint write FOnPaint;
    property FullScreenTopMost: boolean read FTopMost write SetTopMost default
false;
    property Mode: TVideoMode read FMode write SetVideoMode default vmNormal;
    { The @link(TFilterGraph) component }
    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;
                                  {@exclude}
    property Color;               {@exclude}
    property Visible: boolean read GetVisible write SetVisible default True;
{@exclude}
    property ShowHint;            {@exclude}
    property Anchors;             {@exclude}
```

```
    property Canvas;               {@exclude}
    property PopupMenu;            {@exclude}
    property Align;                {@exclude}
    property TabStop default True; {@exclude}
    property OnEnter;              {@exclude}
    property OnExit;               {@exclude}
    property OnKeyDown;            {@exclude}
    property OnKeyPress;           {@exclude}
    property OnKeyUp;              {@exclude}
    property OnCanResize;          {@exclude}
    property OnClick;              {@exclude}
    property OnConstrainedResize;  {@exclude}
    property OnDblClick;           {@exclude}
    property OnMouseDown;          {@exclude}
    property OnMouseMove;          {@exclude}
    property OnMouseUp;            {@exclude}
    property OnMouseWheel;         {@exclude}
    property OnMouseWheelDown;     {@exclude}
    property OnMouseWheelUp;       {@exclude}
    property OnResize;
  end;

//***************************************************************************
//  TFilterSampleGrabber declaration
//***************************************************************************

  {@exclude}
  TSampleGrabber = class;

  TSampleGrabber = class(TComponent, IFilter, ISampleGrabberCB)
  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
    {@exclude}
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
  public
    SampleGrabber: ISampleGrabber;
    InPutPin  : IPin;
    OutPutPin : IPin;
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure UpdateMediaType;
    {@exclude}
    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
```

```
    property OnBuffer: TOnBuffer read FOnBuffer write FOnBuffer;
    { The filter must connected to a TFilterGraph component.}
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;

    property MediaType: TMediaType read FMediaType write FMediaType;
  end;
//**********************************************************************
//  TFilter
//**********************************************************************

  TFilter = class(TComponent, IFilter)
  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
    { Constructor method. }
    constructor Create(AOwner: TComponent); override;
    { Destructor method. }
    destructor Destroy; override;
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
  published
    property BaseFilter: TBaseFilter read FBaseFilter write FBaseFilter;
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;
  end;

//**********************************************************************
//  TASFWriter
//**********************************************************************

  { This component is designed to create a ASF file or to stream over a
network.}
  TASFWriter = class(TComponent, IFilter)
  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
    {@exclude}
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
  public
    WriterAdvanced2      : IWMWriterAdvanced2;
    WriterNetworkSink    : IWMWriterNetworkSink;
    AudioInput           : IPin;
    { The Video Input Pin. }
```

```
    VideoInput           : IPin;
    AudioStreamConfig    : IAMStreamConfig;
    VideoStreamConfig    : IAMStreamConfig;
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    {@exclude}
    function QueryInterface(const IID: TGUID; out Obj): HResult; override;
stdcall;
  published
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;
    property Profile: TWMPofiles8 read GetProfile write SetProfile;
    property FileName: String read GetFileName write SetFileName;
    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;

//***********************************************************************
//  TDSTrackBar
//***********************************************************************
  {@exclude}
  TTimerEvent = procedure(sender: TObject; CurrentPos, StopPos: Cardinal) of
object ;

  { This control implement a seek bar for a media-player application.
    The seek bar is implemented as a TTrackbar control. }
  TDSTrackBar = class(TTrackBar, IEvent)
  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
    { constructor method. }
    constructor Create(AOwner: TComponent); override;
    { destructor method. }
    destructor Destroy; override;
  published
    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;

//**************************************************************************
//  TColorControl
//**************************************************************************


  TColorControl = class(TPersistent)
  private
    FBrightness : Integer;
    FContrast   : Integer;
    FHue        : Integer;
    FSaturation : Integer;
    FSharpness  : Integer;
    FGamma      : Integer;
    FUtilColor  : Boolean;
    FDefault    : TDDColorControl;
  protected
    { Protected declarations }
    { @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;
    { @exclude }
    function GetContrast : Integer;
    { @exclude }
    function GetHue : Integer;
    { @exclude }
    function GetSaturation : Integer;
    { @exclude }
    function GetSharpness : Integer;
    { @exclude }
    function GetGamma : Integer;
    { @exclude }
    function GetUtilColor : Boolean;
    { @exclude }
    Procedure ReadDefault;
    { @exclude }
    procedure UpdateColorControls;
    { @exclude }
    procedure GetColorControls;
  public
    { Public declarations }
    { @exclude }
    constructor Create(AOwner: TDSVideoWindowEx2);  virtual;

 procedure RestoreDefault;
```

```
  Published
    property Brightness : Integer read GetBrightness write SetBrightness;

    { The
    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;

//**********************************************************************
//  TDSVideoWindowEx2Caps
//**********************************************************************


  TDSVideoWindowEx2Caps = class(TPersistent)
  protected
    { Protected declarations }
    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 declarations }
    { @exclude }
    constructor Create(AOwner: TDSVideoWindowEx2);  virtual;
  published
    { if CanOverlayGraphics return true, you draw on DSVideoWindowEx's canvas
and the
      graphic will bee ontop of the Video.}
    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;

// **********************************************************************
//  TOverlayCallback
//**********************************************************************

  { @exclude }
  TOverlayCallback = class(TInterfacedObject, IDDrawExclModeVideoCallBack)
    AOwner : TObject;
    constructor Create(Owner : TObject); virtual;
    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;
```

```
//***************************************************************************
//  TDSVideoWindowEx2
//***************************************************************************

  { @exclude }
  TRatioModes  = (rmStretched, rmLetterBox, rmCrop);

  { @exclude }
  TOverlayVisibleEvent = procedure (Sender: TObject; Visible : Boolean) of
object;

  { @exclude }
  TCursorVisibleEvent = procedure (Sender: TObject; Visible : Boolean) of
object;

  TDSVideoWindowEx2 = class(TCustomControl, IFilter, IEvent)
  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;
    {@exclude}
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
    {@exclude}
    procedure WndProc(var Message: TMessage); override;
    {@exclude}
    procedure Paint; 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 MyIdleHandler(Sender: TObject; var Done: Boolean);
    {@exclude}
    procedure RefreshVideoWindow;
  public
    { constructor method. }
    constructor Create(AOwner: TComponent); override;
    { destructor method. }
    destructor Destroy; override;

    {@exclude}
    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;
    { @inherited }
    property Canvas;

    property ColorKey : TColor read FColorKey;

    { @link(TDSVideoWindowEx2Caps) }
    property Capabilities : TDSVideoWindowEx2Caps read FCaps;

property OverlayVisible : Boolean read FOverlayVisible;
published


    property AspectRatio : TRatioModes read FAspectMode write SetAspectMode;

    property AutoHideCursor : Integer read FIdleCursor write FIdleCursor;

    property DigitalZoom : Integer read FZoom write SetZoom;

    { The @link(TFilterGraph) component }
```

```
    property FilterGraph: TFilterGraph read FFilterGraph write SetFilterGraph;

    property FullScreenTopMost: boolean read FTopMost write SetTopMost default
false;

    property OnColorKeyChanged: TNotifyEvent read FOnColorKey write FOnColorKey;

    { @link(TColorControl) }
    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;                {@exclude}
    property Visible;              {@exclude}
    property ShowHint;             {@exclude}
    property Anchors;              {@exclude}
    property PopupMenu;            {@exclude}
    property Align;                {@exclude}
    property TabStop default True; {@exclude}
    property OnEnter;              {@exclude}
    property OnExit;               {@exclude}
    property OnKeyDown;            {@exclude}
    property OnKeyPress;           {@exclude}
    property OnKeyUp;              {@exclude}
    property OnCanResize;          {@exclude}
    property OnClick;              {@exclude}
    property OnConstrainedResize;  {@exclude}
    property OnDblClick;           {@exclude}
    property OnMouseDown;          {@exclude}
    property OnMouseMove;          {@exclude}
    property OnMouseUp;            {@exclude}
    property OnMouseWheel;         {@exclude}
    property OnMouseWheelDown;     {@exclude}
    property OnMouseWheelUp;       {@exclude}
    property OnResize;
  end;

////////////////////////////////////////////////////////////////////////////////
//
// TVMRBitmap Class
//
////////////////////////////////////////////////////////////////////////////////
type


  TVMRBitmapOption = (
    vmrbDisable,
    vmrbSrcColorKey,
    vmrbSrcRect
  );
  TVMRBitmapOptions = set of TVMRBitmapOption;

  TVMRBitmap = class
  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

    constructor Create(VideoWindow: TVideoWindow);
    destructor Destroy; override;

    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;
    property ColorKey: COLORREF read GetColorKey write SetColorKey;
    property Options: TVMRBitmapOptions read FOptions write SetOptions;
  end;
implementation
uses ComObj;
End.
```