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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи відеоконференцій за допомогою технології BYOC”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Грабовський М.Д.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

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

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

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

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

Грабовському Максиму Дмитровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи відеоконференцій за допомогою технології BYOC*

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

затверджені наказом вищого навчального закладу № 48-02 від 17.01.2025 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи відеоконференцій за допомогою технології BYOC*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Грабовський М.Д.
(прізвище та ініціали)

АНОТАЦІЯ

Грабовський М.Д. Програмне забезпечення системи відеоконференцій за допомогою технології BYOC. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи відеоконференцій за допомогою технології BYOC.

Метою розробки є програмне забезпечення системи відеоконференцій за допомогою технології BYOC.

Результат роботи – програмна реалізація системи відеоконференцій за допомогою технології BYOC.

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 Visual C#.

Ключові слова: комп'ютерна інженерія, відеоконференція, BYOC

ABSTRACT

Grabovsky M.D. Software for a videoconferencing system using BYOC technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a videoconferencing system using BYOC technology.

The purpose of the development is software for a videoconferencing system using BYOC technology.

The result of the work is a software implementation of a videoconferencing system using BYOC 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 are provided.

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

The program was developed in the Visual C# environment.

Keywords: computer engineering, videoconferencing, BYOC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДКП	–	дискретне косинусне перетворення
ПЗ	–	програмне забезпечення
ЛОМ	–	локальна обчислювальна мережа
CCTV	–	система замкнутого телебачення
IP	–	інтернет протокол
USB	–	універсальна серійна шина

ВСТУП

Актуальність теми. Відеоконференцзв’язок – це технологія, що дозволяє людям бачити й чути один одного, обмінюватися даними й спільно їх обробляти в інтерактивному режимі. Перший крок до створення подібних систем зробила в 1964 р. компанія AT&T, що розробила аудіовізуальну систему електронної взаємодії. А вже наприкінці 1970 р. з'явилися перші системи відеоконференцзв’язку (ВКЗ), які сьогодні найбільше повно відтворять атмосферу реального спілкування.

Як в умовах обмеженого бюджету охопити відеозв'язком широкі маси корпоративних користувачів? Один із привабливих варіантів – використовувати як відеотермінали звичайні ПК.

Сценарій BYOC (Bring Your Own Codec) – «відправляючись у переговірну, захопи свій ВКЗ-кодек (кодек відео-конференц-зв'язку)»: користувач приходить у кімнату для конференцій зі своїм ноутбуком, підключає до нього необхідні периферійні пристрої (камеру, спікерфон, дисплей) і проводить відеоконференцію.

Загалом кажучи, ті, хто має відношення до систем ВКЗ, дивляться на них по-різному. З погляду оператора будь-якої транспортної мережі (LAN, ISDN, Frame Relay, ATM, SDH, xDSL і т.д.), ВКЗ – це не що інше, як регулярний потік даних, критичний до кількості помилок мережі й втраті пакетів. Для оператора зв'язку – це послуга, пов'язана з наданням абонентові широкополосного доступу. Для користувача – послуги зв'язку, які він одержує у своїй корпоративній або загальнодоступній мережі передачі даних, і засіб ведення бізнесу; тому він вимагає надійності, зручності і якості зв'язку на рівні звичної телефонії.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи відеоконференцій за допомогою технології BYOC.

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

– Огляд існуючих систем відеоконференцій за допомогою технології BYOC.

– Дослідження системи відеоконференцій за допомогою технології BYOC.

– Програмна реалізація системи відеоконференцій за допомогою технології BYOC.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі відеоконференцій за допомогою технології BYOC.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеоконференцій за допомогою технології BYOC, 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

У цей час системи ВКЗ широко застосовуються в таких областях, як керування й бізнес, дистанційне навчання й телемедицина. Саме ці області ілюструють традиційне використання ВКЗ і в Україні, і в усьому світі. Крім цього, у світовій практиці розвивається використання ВКЗ у таких областях, як підбор персоналу, телебанкинг (різні системи автентифікації, що впроваджуються через більш часті випадки шахрайства із кредитними картами), реклама й маркетинг (для проведення важливих презентацій і брифінгів для цільових груп клієнтів), моніторинг небезпечних виробництв. Як показала практика, відеоконференції виявляються незамінними для фірм із розгалуженою мережею філій: для координації керування або ефективного рішення поточний бізнес-завдань, що вимагають особистої участі співробітників, зокрема, немає ніякої необхідності щораз відправляти їх у дорогі відрядження.

За кордоном ВКЗ використовують не тільки організації, але й окремі фахівці при виконанні надомних робіт. Очікується, що з підвищенням рівня економічного розвитку ВКЗ буде широко застосовуватися в цих і суміжних областях і в нашій країні. Це обумовлено в першу чергу особливостями геополітичної будови України, розміщенням різних об'єктів, що передають інформацію один одному, на більших відстанях друг від друга. У цей час в Україні системи ВКЗ, наприклад, успішно застосовуються при організації корпоративних тренінгів, у судовій практиці, у військовій справі. Відеоконференцзв’язок знаходить гідне застосування скрізь, де необхідні оперативність в аналізі ситуації й ухваленні рішення; консультація фахівця або спільна робота в режимі віддаленого доступу. Звичайно, відеоконференції ніколи не замінять особистого спілкування, але вони дозволяють домогтися принципово нового рівня спілкування людей, часом розділених багатьма тисячами кілометрів.

1.2 Область застосування

Звичайно виділяють кілька типових завдань, розв'язуваних ВКЗ, причому кожна із цих сфер застосування має свої характерні риси, які враховуються при проектуванні:

- керування – державний або виробниче (бізнес);
- навчання;
- спільна робота;
- надання послуг оператором зв'язку;
- комбіновані варіанти.

У цей час ринок систем відеоконференцзв’язку перебуває на підйомі. По оцінках аналітиків, до 2029 р. у світі його обсяг досягне 10 млрд дол. У цей час щорічний приріст світового ринку встаткування ВКЗ перебуває на рівні 20-25%. По оцінках Wainhouse research (http://www.wainhouse.com), для регіону EMEA характерні ще більш високі темпи росту. Відзначимо, що серед країн регіону EMEA Україна являє собою ринок що найбільше динамічно розвивається як по обсязі продажів, так і по темпах росту (щорічно збільшується на 50%).

По оцінках експертів, високі темпи росту українського ринку ВКЗ збережуться й надалі, оскільки ріст реального сектора економіки, пожвавлення економічного життя регіонів, зміцнення системи державної влади й зростаюча увага держави до соціальних проблем і підвищення якості життя населення вимагає створення систем керування різного призначення й масштабу. Найважливішою технологічною основою таких систем уже сьогодні стали відеоконференцзв’язок і відеотелефонія, що дозволяють керівникові зі свого робочого місця вирішувати будь-які управлінські завдання, зберігаючи звичний стиль спілкування й різко скорочуючи час на поїздки й відрядження.

Прогрес устаткування, зниження його вартості й можливість використовувати як наземну, так і супутникову телекомунікаційну інфраструктури привели до того, що системи ВКЗ впроваджуються в

комерційних і бюджетних організаціях по всій країні. Як будь-які сучасні технології зв'язку, ВКЗ стрімко розвивається, удосконалюються алгоритми обробки сигналу, зростає обчислювальна потужність пристроїв, з'являються нові стандарти й рішення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеоконференцій за допомогою технології BYOC, 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

Класифікація відеоконференцій

Залежно від рівня встаткування, використовуваного для систем відеоконференцзв’язку, розрізняють персональні, групові й студійні відеоконференції. Звичайно персональні відеоконференції – це якийсь початковий рівень технологій ВКЗ. Для їхнього створення потрібні відносно недорогі програмні або програмно-апаратні засоби, застосовувані на робочому місці. Для з'єднання цілком підійде й аналогова телефонна мережа (при відсутності високих вимог до відеозображення). Такий тип відеоконференцзв’язку можна використовувати для неформального спілкування між двома особами, обміну інтерактивною інформацією, пересилання файлів при невеликих витратах часу й фінансів. У спільній роботі з додатками застосовується "дошка оголошень" – спеціалізований додаток, що дає можливість редагувати текстовий або графічний документ всім учасникам сеансу зв'язку.

Групові відеоконференції використовуються для ефективного спілкування великих і середніх груп користувачів при спільній роботі над проектом, для проведення дискусій і виступів, на яких учасник не може бути присутнім особисто. Завдяки високій якості сигналу можливі обмін і перегляд документів, групова робота з додатками. Для організації групових відеоконференцій потрібні старші моделі відеотерміналів, сервер, що забезпечує взаємодія груп користувачів, спеціалізовані програмні продукти для робочих станцій і сервера, ISDN-з'єднання або локальна мережа.

Для створення студійних відеоконференцій необхідне висококласне спеціалізоване встаткування (студійні камери, звукове й контрольне встаткування, монітори) і максимальна пропускна здатність каналів зв'язку (доступ до каналів супутникової й оптоволоконого зв'язку). Такі відеоконференції використовуються для рішення завдань, що вимагають максимуму можливостей з погляду організації обробки інформації більшим числом людей. Типовий приклад таких відеоконференцій – телемости.

По топології звичайно розрізняють два основних типи відеоконференції: " точка-точка" і багатоточечні. Конференції " точка-точка" найбільш прості. Вони мають на увазі з'єднання тільки двох робочих станцій прямо, у той час як багатоточечні відеоконференції здатні охоплювати одночасно кілька десятків користувачів або груп користувачів, але вимагають додаткових витрат на установку й підтримку спеціалізованого пристрою – сервера керування багатоточечними сеансами.

Всі термінали, що беруть участь у конференції, установлюють з'єднання із сервером, що управляє ресурсами відеоконференції, погоджує можливості обробки звуку й відео для терміналів, визначає аудіо- і відеопотоки, які необхідно направляти по багатьом адресам. Приміром, якщо відеоконференцзв’язок із декількома філіями служить тільки для передачі розпоряджень і прийому звітів у реальному режимі часу, то для цієї мети цілком підійде конференція типу "точка-точка" (досить забезпечити кожну філію й головний офіс спеціалізованим терміналом), у той час як для організації наради за участю представників всіх філій потрібно багатоточечна відеоконференція.

Багатоточечні сеанси зв'язку можуть проводитися у двох основних режимах – "активація за голосом" і "безперервна присутність". У першому режимі всі учасники сеансу бачать спікера, а мовець бачить попереднього оратора. У другому на екран кожному учасникові надходить зображення від декількох інших учасників. При цьому екран розділяється на кілька полів (від двох до 16). Якщо полів менше, ніж учасників, то одне з них може працювати в

режимі "активація за голосом". І в тому і в іншому режимі можливий "належному голові контроль" – вибір активного термінала, підключення й відключення терміналів адміністратором відеоконференції. При необхідності включається автоматичний режим адміністрування з можливістю в будь-який момент втрутитися в цей процес.

Безпека

В умовах високої конкуренції в багатьох областях бізнесу передача даних по відкритих мережах неприпустимий – адже навіть мінімальний витік відомостей може привести до краху компанії. Тому при організації відеоконференції на підприємстві немаловажну роль грають питання захисту інформації, особливо при реалізації зв'язку з віддаленими філіями по глобальних мережах. У цей час на українському ринку є програмно-апаратні комплекси криптозахисту, розроблені вітчизняними фахівцями й досить гарні результати, що показали у випробуваннях. В основі такого комплексу звичайно лежить криптографічний шлюз, що гарантує схоронність конфіденційних відомостей шляхом створення захищених тунелів зв'язку. Ці пристрої встановлюються на входах у локальну мережу компанії й забезпечують шифрування/дешифрування інформації за допомогою спеціалізованих програмних або апаратних засобів. Криптографічні комплекси, що використовують українські й закордонні стандарти захисту інформації, гарантують:

- конфіденційність переданих і оброблюваних даних;
- цілісність даних;
- автентифікацію джерела даних;
- приховання топології мережі, яка захищається, і її окремих сегментів;
- захист від аналізу трафіку.

Технічні проблеми

Одна із проблем полягає у швидкості обробки аудіо- і відеопотоку, тобто в ефективному кодуванні переданих і декодуванні одержуваних даних у реальному масштабі часу. Справа в тому, що у відеоконференціях використовуються

спеціальні й досить ефективні алгоритми стиску потоку в десятки (а часом і в сотні) раз. Можна сказати, що передаються не самі аудіо- і відеосигнали, а тільки їхні найважливіші параметри, які дозволяють відновлювати сигнал на прийомному кінці із прийнятною якістю. Якщо приймаюча сторона не встигає обробляти потік, то з'являються пропущені кадри, збої в мовному каналі й т.п.

Алгоритми обробки сигналу досить вимогливі до обчислювальних ресурсів. Хоча й існують їх чисто програмні реалізації, однак вони вимагають значних ресурсів від базової платформи ПК. У результаті навіть на найсучасніших настільних комп'ютерах сильно вповільнюється робота інших додатків, та й прийнятна якість відеозв'язку одержати не вдається. Загальноприйнята світова практика складається у використанні апаратних рішень – спеціалізованих систем відеоконференцій (кодеків), які реалізуються і як плати, що вставляються у вільні слоти ПК, і як функціонально закінчені рішення. Кодеки стискають сигнал і кодують його для каналу зв'язку (відповідно розтискають його й декодують на приймаючій стороні). Однієї з основних тенденцій у виробництві терміналів для відеоконференцзв’язку сьогодні стало зниження вимог до смуги пропущення за рахунок застосування нових алгоритмів відео- і аудіокомпресії (H.264, MPEG і т.д.).

Інша важлива проблема при передачі аудіо- і відеоінформації полягає в тому, що канал зв'язку, по якому передається ця інформація, повинен бути досить швидкісним, іншими словами, мати високу пропускну здатність. Звичайні телефонні канали цілком підходять для передачі аудіосигналу, але якісну передачу відеопотоку вони не забезпечують. Історично проведення відеоконференції мало на увазі зв'язок між терміналами ВКЗ по лініях ISDN (цифрова мережа з інтеграцією послуг). Використання каналів ISDN, а також інших мереж і ліній з гарантованою якістю зв'язку – V.35, E1/T1 і т.д. регламентується серією рекомендацій H.320, розроблених комітетом зі стандартизації телекомунікацій Міжнародного союзу електрозв'язку (ITU-т).

Однак час не стоїть на місці, і в останні роки усе більше широке поширення одержують відеоконференції, що використовують IP-мережі, – як локальні, так і територіально розподілені й глобальні. Хоча встаткування й ПЗ для відеоконференцій по IP-мережах (стандарт H.323) вийшли на ринок порівняно недавно, за прогнозами аналітиків, вони незабаром займуть чільне положення. Так, уже до 2014 р. число розгортань систем ВКЗ на базі мереж IP у світі перевищило аналогічний показник у мережах ISDN і продовжує збільшуватися. В Україні ця тенденція найбільш виражена, тому що тут технологія ISDN просто не встигла зайняти більш-менш міцні позиції. Довівши ж свою ефективність у корпоративних мережах технологія IP-ВКЗ демонструє свої переваги в глобальному масштабі.

Можливість інтеграції ВКЗ із іншими послугами (IP-телефонією, електронною поштою, Web-технологіями й т. інш.), уже розгорнутими в компаніях, а також доступність і поширеність IP-мереж зіграли вирішальну роль у виборі споживачів. Бажаючих будувати мережі ВКЗ на базі технології IP не зупиняє навіть необхідність прикласти певні зусилля (фінансові і технічні), щоб перебороти недоліки даного варіанта – надмірність, нестабільність якості зв'язку, складності безпечного й зручного стикування приватних і суспільних IP-мереж. Компанії – розроблювачі систем ВКЗ підготували широкий спектр рішень подібних проблем на базі економічних приватними й сумісних з іншими виробниками відкритих протоколів.

Звичайно для проведення відеоконференцій використовуються лінії зі смугою пропущення від 128 Кбіт/с до 512 Кбіт/с для ISDN-відеоконференцій і до 1-1,5 Мбіт/с для IP-мереж. Але треба мати на увазі, що для прийнятної якості відео потрібні швидкості порядку 200 Кбіт/с, а високоякісне зображення в гарних системах досягається при швидкостях близько 300 Кбіт/с і вище. Існує думка, що IP-системи вимагають більше широкої смуги пропущення. Це дійсно так – через особливості передачі інформації в мережах з комутацією пакетів (додавання заголовків, службові пакети протоколів RTCP і т.д.) необхідна смуга пропущення

збільшується на 20-30%. Однак практика показує, що якість відеоконференції при використанні трьох BRI-каналів (384 Кбіт/с) або IP-каналу із шириною близько 500 Кбіт/с приблизно однаково.

Стандарти відеоконференцій

В 1990 р. був схвалений перший міжнародний стандарт в області відеоконференцзв'язку – специфікація H.320 для підтримки відеоконференцій по ISDN. Потім ITU схвалив ще цілу серію рекомендацій, що ставляться до відеоконференцзв'язку. Ця серія рекомендацій, часто називана H.32x, крім H.320, містить у собі стандарти H.321-H.324, які призначені для різних типів мереж: H.321 – ATM B-ISDN, H.323 – ЛОМ і Інтернет, H.324 – телефонна мережа загального користування. Прагнення використовувати сформовану структуру IP-мереж привело до появи в 1996 р. стандарту H.323 (Visual Telephone Systems and Terminal Equipment for Local Area Networks which Provide a Non-Guaranteed Quality of Service – "Відеотелефони й термінальне встаткування для локальних мереж з негарантованою якістю обслуговування").

Рекомендації ITU для мультимедійних додатків в обчислювальних мережах, що не забезпечують гарантовану якість обслуговування (мережі пакетної комутації IP і IPX на базі Ethernet, Fast Ethernet і Token Ring), передбачають:

- керування смугою пропущення;
- можливість взаємодії мереж;
- платформну незалежність;
- підтримку багатоточечних конференцій;
- підтримку багатоадресної передачі;
- стандарти для кодеків;
- підтримку групової адресації.

Передача аудіо- і відеоінформації досить інтенсивно навантажує канали зв'язку, і, якщо не стежити за ростом цього навантаження, працездатність критично важливих мережних сервісів може бути порушена. Тому рекомендації

Н.323 передбачають керування смугою пропущення. Допускаються обмеження як числа одночасних з'єднань, так і сумарної смуги пропущення для всіх додатків Н.323. Ці обмеження допомагають зберегти необхідні ресурси для роботи інших мережних додатків. Кожний термінал Н.323 може управляти своєю смугою пропущення в конкретній сесії конференції.

Рекомендації Н.323 пропонують засіб з'єднання учасників відеоконференції в різнорідних мережах. Стандарт не прив'язаний ні до яких конкретних технологічних рішень, пов'язаним з устаткуванням або ПЗ. Взаємодіючі між собою додатка можуть працювати на основі різних платформ, з різними операційними системами. Рекомендації Н.323 дозволяють організувати конференцію із трьома або більше учасниками. Багатоточечні конференції можуть проводитися як з використанням пристрою керування багатоточечною конференцією, так і без нього.

Н.323 підтримує багатоадресну передачу в багатоточечной конференції, якщо мережа підтримує протокол керування груповою адресацією. При багатоадресной передачі один пакет інформації відправляється всім необхідним адресатам без зайвого дублювання. Багатоадресна передача використовує смугу пропущення набагато більш ефективно, оскільки всім, хто включений у список розсилання, відправляється рівно один потік. Установлюються також регламенти для кодування й декодування аудіо- і відеопотоків, для того щоб забезпечити сумісність устаткування різних виробників. Варто відзначити, що в Н.323 існують вимоги, виконання яких обов'язково, і опціональні можливості, при використанні яких також необхідно строго додержуватися стандарту. Крім цього, виробник може включати в мультимедійні продукти й додатки додаткові можливості, якщо вони не суперечать обов'язковим і опціональним вимогам стандарту. Рекомендації Н.323 підтримують з'ясування загальних можливостей користувальницького встаткування й установлюють найкращі із загальних для учасників конференції протоколи кодування, виклику й керування.

У стандарт включений опис термінів "термінал" (terminal), " мультимедіа-шлюз" (gateway), "пристрій керування багатоточечними конференціями" (Multipoint Control Unit, MCU) і "контролер зони", або "воротар" (gatekeeper). Під терміналом розуміють прикінцеве мультимедійне (голос, відео, дані) пристрій, призначений для участі в конференції. Термінали повинні підтримувати протоколи H.245 – для узгодження параметрів з'єднання, Q.931 – для встановлення з'єднання й узгодження параметрів цього з'єднання, канал RAS (Registration/Admission/Status) для взаємодії з контролером зони, протокол RTP/RTCP для роботи з потокуми аудіо- і відеопакетів, протокол G.711 для стиску аудіопотоку. Відповідно до рекомендацій, для термінала H.323 опціональна підтримка відеокодеків, протоколу T.120 і можливостей MCU.

Мультимедіа-Шлюз – це пристрій, призначений для перетворення мультимедійної й керуючої інформації при сполученні різнорідних мереж. Згідно H.323, мультимедіа-шлюз – це опціональний елемент, що може виконувати багато різних функцій. Типове його завдання – перетворення форматів протоколів передачі (наприклад, H.225.0 і H.221). Звичайно мультимедіа-шлюзи служать для підтримки взаємодії між різнорідними мережами.

Пристрій керування багатоточечними конференціями призначено для організації конференцій між трьома й більше учасниками. У цьому пристрої повинен бути присутнім контролер Multipoint Controller (MC) і, можливо, процесори Multipoint Processors (MP). Контролер MC підтримує протокол H.245 і призначений для узгодження параметрів обробки аудіо- і відеопотоків між терміналами. Процесори займаються комутуванням, микшуванням і обробкою цих потоків. Конфігурація багатоточечної конференції може бути централізованою, децентралізованою, гібридною й змішаною. Контролер зони – це рекомендуємий, але не обов'язковий пристрій, що забезпечує мережне керування й виконує роль віртуальної телефонної станції. У число основних функцій контролера зони входять:

– керування викликами і їхньою адресацією;

– забезпечення основними типами обслуговування;

– керування тим, як додатки H.323 використовують смугу пропущення, що повинна забезпечити якість обслуговування;

– керування загальним використанням мережних ресурсів;

– системне адміністрування й забезпечення безпеки.

Незважаючи на те що H.323 визначає контролер зони як необов'язковий компонент, без нього неможливо скористатися різноманітним спектром послуг, передбачених творцями стандарту H.323 для додатків IP-телефонії й мультимедійних телеконференцій.

Сучасні рішення ВКЗ

Компанія Tandberg (http://www.tandberg.net), постачальник устаткування для відеоконференцій, займається розробкою, виробництвом, поставкою й обслуговуванням відеотелекомунікаційних систем більш ніж в 50 країнах миру. Основні відділення Tandberg розташовані в Херндоне (США), у Монреалі (Канада) і в Осло (Норвегія). Компанія має значну частку на європейському, азіатському й тихоокеанському ринках, поширюючи свою продукцію через широку мережу дистриб'юторів і інтеграторів.

Так, на невеликі групові системи доводиться близько 25% оберту цього напрямку бізнесу компанії, на персональні системи – 15%, мобільні системи мають частку 10%, відеосервери – 35%, а на ПЗ й підтримку доводиться 15%.

Устаткування Tandberg

Останнім часом надзвичайно актуальним стало завдання створення терміналів ВКЗ, оптимізованих для мультимедійних застосувань. Рішення цього завдання компанія Tandberg втілила в лінійці спеціалізованих продуктів Tandberg MXP Profile. У неї входять дві клієнтські моделі, розраховані на використання у великій і середній переговірній кімнатах – MXP Profile 6000 і MXP Profile 3000 відповідно, а також інтегроване настільне рішення для індивідуальних користувачів – MXP Profile Set-top Package. Всі продукти лінійки MXP Profile оснащені широкоекранною плазменою панеллю (або РК-дисплеєм), що

забезпечує високу якість візуалізації, чіткість зображення, гарну передачу кольору й підтримуючою найбільш ефективний відеодозвіл. Крім того, убудовані алгоритми обробки звуку, цифрові підсилювачі й акустичні системи забезпечують якісне звучання в широкому діапазоні частот. Всі пристрої сімейства Tandberg MXP Profile прямо підключаються до ПК (через інтерфейс DVI), а також до побутових аудіо- і відеопристроям. Все це розширює можливості використання відеоконференцзв'язку при проведенні нарад за участю керівників вищого рівня, під час презентацій, у дистанційному навчанні.

Продукти сімейства Tandberg MXP Profile підтримують протоколи H.235, H.264, H.323 і який отримує поширення, що все більше широке, протокол SIP. Це, з одного боку, забезпечує зворотну сумісність із практично будь-яким устаткуванням ВКЗ, а з іншого боку – дозволяє як основу системи ВКЗ використовувати сучасні серверні комунікаційні інфраструктури, готові до обробки мультимедійного трафіку, включаючи Microsoft Live Communication Server, IBM Lotus Instant Messaging and Web Conferencing (Sametime) і т.д.

Системи ВКЗ на основі Tandberg Expressway призначені для організації спілкування співробітників територіально розподілених підприємств або сеансів відеоконференцзв'язку між різними організаціями. Система ВКЗ на базі Tandberg Expressway має дві особливості: по-перше, вона дозволяє спростити систему нумерації усередині організації, привласнюючи користувачам замість цифрової IP-адреси зрозумілі кожному ідентифікатори, аналогічні адресам електронної пошти (ім'я@організація.com). Крім того, у цю систему убудована відсутня в інших виробників технологія наскрізної передачі відеоінформації через корпоративні й персональні міжмережні екрани без їхнього переналаштування, що була потрібна дотепер. Все це спрощує проведення відеоконференцій за участю співробітників різних організацій, а також розгортання систем відеоконференцзв'язку на великих підприємствах, що вже створили інфраструктуру Інтернет/інтранет і систему інформаційної безпеки. Високий рівень безпеки інформаційного обміну при цьому зберігається. Ще одна відмінна

риса системи – убудована технологія автоматичного самоналаштування, що реагує на зміни пропускної здатності з'єднання й забезпечує високий рівень якості за рахунок передачі тільки рухливих елементів зображення.

Система відеоконференцзв’язку Tandberg Expressway містить у собі абонентські пристрої, з якими працює користувач, шлюзові контролери (Tandberg Gatekeeper) і контролери з'єднань (Tandberg Border Controller, або TBC), що відповідають, зокрема, за передачу трафіку через міжмережний екран. Як абонентські пристрої можуть використовуватися будь-які відеотермінали, що підтримують стандартний протокол H.323. Контролер з'єднань може працювати з будь-якими міжмережними екранами й установлюється поза захищеною зоною (наприклад, у демілітаризованій зоні або на площадці провайдеру доступу в Інтернет).

Телемедицина

Як вже відзначали, широке застосування технології ВКЗ знайшли в медицині й в освіті, сформувавши найбільше що динамічно розвиваються галузі телемедицини й телеосвіти. При цьому український телемедичний сегмент, що поєднує десятки медичних установ України, уже сьогодні дозволяє реально підвищити якість медичного обслуговування. Найбільш затребувані телемедичні рішення у віддалених і важкодоступних регіонах, а також у медицині катастроф.

Впровадження телемедичних систем сьогодні стає усе більше звичайною справою при інформатизації об'єктів охорони здоров'я. У першу чергу подібні системи забезпечують можливості дистанційної діагностики, консультування, але цим їхнє застосування не обмежується. Мережа відеоконференцзв’язку може використовуватися для рішення наступних завдань:

– регулярних телемедичних консультацій пацієнтів з віддаленими медичними центрами;

– віддаленого навчання й читання лекцій провідними медичними спеціалістами країни й миру; дистанційного навчання співробітників територіально рознесених підрозділів, проведення семінарів і конференцій;

– проведення регулярних нарад керівного состава й структурних підрозділів; учасники нарад спілкуються з використанням засобів аудіо- і відеозв'язку, немає необхідності збирати їх в одному місці;

– заходів, пов'язаних з оперативним керуванням і прийняттям рішень;

– презентаційних і маркетингових заходів.

Нові сфери застосування мобільної телемедицини виникають постійно. Наприклад, послуги телемедицини й систем ВКЗ стають актуальними для власників великих приватних катерів і яхт. Адже застосування телемедичних терміналів на порядок збільшує безпеку подорожей, дозволяє в будь-якій точці світу одержувати кваліфіковані медичні консультації, підтримувати ділові контакти й зв'язок з рідними в режимі відеоконференцзв’язку.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою Visual C#. Ця мова обрана виходячи з наступних міркувань. Visual C# – строго типізована об’єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна розробляти звичайні клієнтські додатки Windows, веб-служби XML, розподілені компоненти, додатки типу “ сервер-клієнт”, додатки баз даних і багато яких інших. В Visual C# є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою Visual C# версії 5.0 і .NET Framework версії 4.5.

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих мов, як правило, зможуть домогтися ефективної роботи з мовою Visual C# за

дуже короткий час. Синтаксис Visual C# робить простіше те, що було складно в C++, і забезпечує потужні можливості, такі як типи значень Nullable, перерахування, делегати, лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. Visual C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В Visual C# 5.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об'єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод `Main` – точку входу додатка – інкапсулюється у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

На додаток до основних описаних об'єктно-орієнтованих принципів, мова Visual C# спрощує розробку компонентів програмного забезпечення завдяки декільком інноваційним конструкціям мови, у число яких входять наступні:

– Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.

– Властивості, що виступають у ролі методів доступу для закритих змінних-членів.

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

– LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32, у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови Visual C# у порівнянні з C і C++ простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі Visual C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою Visual C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками.

Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску у файлі, що виконується, називаному складанням, з розширенням EXE або DLL у більшості випадків. Складання містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки.

При виконанні програми на Visual C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки дотримані, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, що

відносяться до автоматичного збору сміття, обробки виключень і керуванню ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим кодом" у протиставлення "некерованому коду", що компілюється в машинний код, призначений для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом Visual C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

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 відеоконференцій за допомогою технології BYOC.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Відеоконференція – це сеанс зв'язку між двома користувачами або групою користувачів, незалежно від їхнього місця розташування, при цьому, учасники бачать і чують один одного відповідно до правил, обумовленим видом відеоконференції.

Відеоконференції проводяться за умови використання спеціальних засобів, які можуть бути реалізовані як на основі апаратних рішень і систем, так і у вигляді програмного забезпечення для ПК, мобільних пристроїв або браузерів.

Для забезпечення учасників звуком і картинкою використовується різне периферійне встаткування: камери, екрани, мікрофони, спікерфони, гарнітури, конгрес-системи й проектори. Як середовище передачі даних може використовуватися як мережа підприємства, побудована по різних принципах, так і глобальна мережа інтернет.

Сучасні відео- і аудіо- кодеки, спеціалізовані мережні протоколи, різні алгоритми обробки сигналів дозволяють домогтися якісного зв'язку практично на будь-яких каналах зв'язку.

Найчастіше під час сеансу відеоконференції необхідна демонстрація різних медіаданих, для цього системи відеоконференцій дозволяють захоплювати й передавати віддаленим учасникам презентації, зображення робочого стола або окремих його вікон, а так само різні по форматах документи. Досягається це за рахунок використання спеціального програмного забезпечення, додаткових камер (наприклад, документальних камер), захвата сигналу з відеовиходів ноутбуків, ПК і інших систем, включаючи медичні комплекси.

Підсумуємо. Відеоконференція – це високотехнологичний сучасний інструмент спілкування, призначений для підвищення ефективності ведення

бізнесу, оптимізації бізнес-процесів, прискорення прийняття рішень і економії засобів на відрядженнях.

Види відеоконференцій

Існує два основних типи відеоконференцій – персональна й групова. Персональна відеоконференція має на увазі сеанс відеозв'язку, у якому бере участь усього два абоненти. Під груповими ж відеоконференціями маються на увазі всі інші види відеоконференцій. Різні устояні правила відображення учасників відеоконференції для кожної йз сторін називаються видами відеоконференцій. Пропонуємо розібратися в цьому питанні докладніше!

Відеоконференції на-1

Тут все просто: беруть участь два абоненти, обоє бачать і чують один одного одночасно. Відразу обмовимося, що під час будь-якого сеансу відеоконференції можуть використовуватися різні інструменти для спільної роботи, такі, як обмін текстовими повідомленнями, файлами, презентаціями й іншими медіаданими.

Симетричні відеоконференції

Вони ж відеоконференції з постійною присутністю, від англ. Continuous Presence. Так називають сеанс відеоконференції, у якому беруть участь більше 2 чоловік і всіх учасників бачать і чують один одного одночасно. Природно, відеоконференція має на увазі повнодуплексне спілкування. Інакше кажучи, це аналог круглого стола, де у всіх рівні права. Групова відеоконференція підходить для зустрічей, де потрібно максимальна участть кожного учасника.

Відеоконференції з активацією за голосом

Назва такого режиму пішло від англійського позначення Voice Activated Switching (VAS). Ця відеоконференція припускає наступний формат спілкування: всі учасники сеансу чують і бачать на своїх екранах тільки виступаючого доповідача, у той час, як він сам бачить себе, або попереднього оратора. Можливі невеликі варіації даного механізму, але суть залишається наступної: сервер ВКЗ відслідковує голосову активність абонентів і перемикає трансльоване всім

учасникам, зображення, на спікера. У даного режиму є істотні недоліки, наприклад, помилкові спрацьовування на шум, кашель або дзвінок мобільного телефону.

Селекторні відеоконференції

Режим, у якому учасники діляться на два види: доповідачі й слухачі, де кожний зі слухачів може стати доповідачем (з дозволу організатора конференції). Провідної такої конференції сам призначає доповідачів і може видалити їх з відео-трибуни в будь-який момент.

Цей режим може так само називатися рольовою відеоконференцією. Селекторна відеоконференція використовується найчастіше при проведенні веб-конференцій (вебінарів).

Відеоконференції для дистанційного утворення

Спеціальний режим, у якому всі учасники (учні) бачать і чують тільки одного користувача, що віщає (викладача), а він бачить і чує всіх учнів. Учні не відволікаються один на одного, а викладач їх контролює.

Відеотрансляція

Вид відеоконференції, у якому доповідач віщає на широку аудиторію слухачів, при цьому, він не бачить і не чує їх. Інші учасники бачать і чують тільки доповідача. Зворотний зв'язок можливий тільки через текстовий чат. Найчастіше, для згладжування зміни мережних умов, у ході трансляції вноситься значна затримка до декількох секунд між що віщає й слухачами.

Устаткування для відеоконференцій

Залежно від місця й способу підключення до сеансу відеоконференції, може знадобитися різне периферійне встаткування.

Відеоконференції в переговірній кімнаті або конгрес-залі

Щоб якісно обладнати переговірну кімнату, необхідно дотримати безлічі нюансів. Природно, чим їх більше, тим вище вартість підготовки. У першу чергу, необхідно правильно розрахувати й установити систему звукопідсилення, на цю тему на одній з Відео+Конференцій була гарна доповідь. Якщо зал невеликий, то

буде досить установити один або трохи спікерфонів (це спеціальні пристрої, що сполучають у собі один або кілька мікрофонів і динаміків, і призначені для усунення луна й шумів).

Далі буде потрібно PTZ відеокамера, від звичайної її відрізняє можливість повертатися, нахилятися нагору й долілиць, а також наближати й віддаляти. Така камера може як у ручному, так і автоматичному режимах (для цього буде потрібно спец. устаткування) перемикатися між особами доповідачів і залом. Як система відображення рекомендується використання двох РК екранів великої діагоналі: один для відео учасників, другий для презентацій і іншого контенту.

Ну й не останнє місце займає інтер'єр приміщення: гарна освітленість, контрастний, але не яскраве тло на стінах, шумопоглинальні панелі та інше. Як видно, вартість устаткування переговірної кімнати залежно від обраного рішення відеоконференцій, периферійного встаткування й обробки, може відрізнятися на порядок.

Відеоконференції на робочому місці

Існує вже безліч готових наборів і комплексів, що включають у себе все необхідне, але зайве місце, що займає, на столі. Тому найчастіше, а також з метою економії, як термінал відеоконференції використовують звичайний робітник ПК, благо різниці в якості, при правильному виборі периферії, між ним і спеціалізованими апаратними системами, немає.

Для підготовки ПК до сеансу відеоконференції буде потрібно гарна веб-камера, на жаль, більшість убудованих у моноблоки й ноутбуки камер не придатні для відеоконференцій. Гарнітура (бажано USB-Гарнітура) або портативний спікерфон, що підключається до ПК через USB інтерфейс.

Мобільні відеоконференції

Одна з переваг відеоконференцій – це їхня мобільність. Їх можна використовувати, навіть перебуваючи в поїздці або на ходу. Пристрій, що може виступати як термінал відеоконференцій – смартфон, планшетний комп'ютер або навіть годинники. На них досить установити спеціальний додаток. Про всі

іншому вужі подбали виробники цих пристроїв: фронтальна камера, потужний центральний процесор, апаратна підтримка відеокодеків (яка в тому числі потрібна й для перегляду фільмів або YouTube), ну а гарний динамік і мікрофон – це саме собою що розуміє. Такий спосіб проведення відеоконференцій дозволить вам бути завжди на зв'язку зі своїми колегами, партнерами по бізнесі, друзями або родичами поза залежністю від обставин.

З іншого боку, існує ряд складностей, пов'язаних з мобільними відеоконференціями, деякі галузі ще треба буде розв'язати, щоб зробити їх по-справжньому зручними й популярними, як на ПК.

Що впливає на якість відеоконференцій?

На відміну від звичних нам електронних комунікацій, таких, як електронна пошта або обмін повідомленнями, відеоконференції відносять до так званих комунікацій у реальному часі (від англ. Real Time Communications), які накладають більше серйозні вимоги, як на термінали відеоконференцій, так і на канали зв'язку, їх єднальні.

Всі ми звикли судити про якість з'єднання за його швидкістю, що в контексті відеоконференції буде не зовсім вірно. Заявлена швидкість може швидко змінюватися в часі, може знижуватися під навантаженням, може кардинально відрізнятися від напрямку передачі. У той час, як все це критично важливо для відеоконференцій, де рівномірність і передбачуваність потоку даних найбільш важливі. Системі відеоконференцій не складно підлаштувати відеопотік під широкий діапазон значень від 64 кб/с до, скажемо, 4 Мб/с, залежно від виду конференції і якості сигналу учасників. Набагато складніше в реальному часі адаптувати ширину каналу під умови, що змінюються, кожного з учасників сеансу зв'язку.

У реальних умовах на перше місце при оцінці якості відеоконференцій виходить тип архітектури, використовуваної для організації відеоконференцій, і здатність цієї архітектури працювати в умовах, що постійно змінюються:

– Потужність ЦП прикінцевих терміналів. Паралельно сеансу зв'язку користувач може почати виконувати ресурсномісткі завдання.

– Можливості захвата відео на камері термінала. Камера може мати відмінний дозвіл, але давати “зашумлену” картинку низької якості при недоліку освітлення.

– Можливості відображення відеоконференції на екрані термінала. Наприклад, користувач вийшов з повноекранного режиму й тепер йому не треба пересилати відео у високій якості.

– Ширина каналу між сервером відеоконференцій і між учасниками. Це найбільш часта ситуація. Варіацій у неї може бути багато: хтось в організації почав накачувати з мережі великий обсяг даних і різко скоротив ресурси мережі на відеоконференцію. Або ж ви, спілкуючись по відео зі смартфона, потрапили в багатолюдне місце, і найближча базова станція вашого оператора зв'язку вже не може гарантувати вам колишню швидкість і якість з'єднання.

Найпростішим рішенням даної проблеми є тверде резервування, як апаратних, так і мережних ресурсів системи відеоконференцій. Але при цьому, таке рішення найдорожче. На щастя, наука й технології не стоять на місці, і сучасні системи відеоконференцій можуть гарантувати відмінна якість зв'язку в будь-яких умовах за рахунок застосування сучасних програмних архітектур. Давайте зупинимося на цьому питанні докладніше.

Типи архітектур систем відеоконференцій

Проведення будь-якої групової відеоконференції, мабуть, вимагає наявність механізму й способу організації передачі даних між її учасниками. У виді того, що передача між учасниками прямо за принципом повнозв’язного графа (кожний до кожного) мало застосовна на практиці, те розглянемо варіанти з використанням деякого медіума, назвемо його "сервер ВКЗ", тобто системи, що працює по топології “зірка” (від центра до кожного).

У традиційних апаратних ВКЗ системах такий сервер називається MCU, у програмних устояні назви немає. Завдання сервера – комутація й обробка потоків

під час групової відеоконференції. Сервер ВКЗ є ядром ВКЗ інфраструктури, ресурсами якої користуються ВКЗ термінали.

Традиційно було прийнято ділити по типі архітектури всі рішення на програмні й апаратні, але в 2014 такий розподіл з ряду причин стало не актуальним. У першу чергу тому, що є як апаратні рішення, що використовують архітектуру, властивим програмним рішенням (на основі перемикання й SVC), так і програмні, що копіюють принцип роботи MCU. А, по-друге, всі провідні виробники намагаються свою ВКЗ інфраструктуру перекласти на віртуалізованні середовища й поставляти як програмне забезпечення.

Архітектура відеоконференцій на основі мікшування (MCU)

Під час відеоконференції сервер ВКЗ приймає відеопотоки від кожного з учасників, декодує їх, зменшує, склеює в нову картинку необхідної якості й дозволу для кожного з учасників (не забувайте про реальні умови, описані вище), заново кодує й відправляє. Все це вимагає величезної обчислювальної потужності, вносить затримки на обробку на сервері, а також може погіршувати якість у результаті перестиску даних.

Масштабованість такої архітектури вкрай низька, навіть із урахуванням можливості її віртуалізації, тому ціна на подібну інфраструктуру вкрай висока, і в сучасних реаліях такі витрати просто не виправдані.

Архітектура відеоконференцій на основі мультиплексування (Switching)

Це класичний підхід до побудови програмних систем ВКЗ, по такому принципі, наприклад, працює Skype. На відміну від MCU, сервер ВКЗ у цьому випадку не перекодовує, створює копії вхідних потоків і пересилає їх іншим учасникам “як є”. Виходить, що кожний з терміналів одержує відразу кілька відеопотоків у повній якості, які він просто не може відобразити одночасно. Терміналу доводиться зменшувати дозвіл кожного вхідного відео від кожного з учасників на своїй стороні, або просити зменшувати його перед відправленням, що погіршує якість відео для всіх інших учасників.

Плюс у цієї схеми один: інфраструктура не вимоглива до ресурсів і навіть рядовий ПК може витримати сотні таких конференцій одночасно. Але от мінусів значно більше: терміналу (звичайно це простий ПК) доводиться декодувати не один, а відразу кілька потоків, а серверу ВКЗ потрібно в кілька разів більша вихідна ширина каналу, щоб умістити в себе всі створені їм копії потоків.

Додамо до цьому реальні умови, і одержимо систему, яка проблемно працює з кількістю учасників більше, ніж 3, і різко погіршує якість відео для всіх, при приєднанні до неї мобільного абонента, не здатного "переварити" вихідна якість картинки, що відправляється іншими абонентами.

Архітектура відеоконференцій на основі масштабованого відео кодування (SVC)

Дана архітектура сполучає в собі всі переваги мікшуючого підходу й при цьому позбавлена недоліків систем на основі мультиплексування. Вона дешева, миттєво масштабується й працює на будь-яких платформах. Це стало можливим завдяки розвитку технологій обробки сигналів і стиску даних.

Суть полягає в тому, що термінал стискає свій відеопотік шарами: кожний додатковий шар підвищує дозвіл відео, його якість і кіл-у кадрів у секунду. Якщо канал між терміналом і сервером ВКЗ гарний, то термінал відправляє максимально можливе кіл-у шарів. Варто помітити, що шар – це не окремий відеопотік меншої якості, а повноцінна різниця між ним і попереднім шаром. Тим самим, SVC потік усього на 15-20% відрізняється по ширині каналу від не SVC-Потоку, і значно менше необхідної суми смуги пропущення незалежних потоків.

Сервер ВКЗ, одержавши SVC-Потік із шарами, просто відтинає зайві без перекодування, тільки лише за рахунок викидання з нього пакетів з даними за певними правилами. Тим самим, дозволяючи на лету створювати індивідуальні набори відеопотоків ("розкладки" вікон) для кожного з учасників групової відеоконференції залежно від його реальних умов зв'язку.

Використання сучасних протоколів і кодеків

Для організації відеоконференцзв'язку між різним програмним забезпеченням і встаткуванням сторонніх виробників використовуються стандартні протоколи передачі даних:

– H.239 – комунікаційний протокол підтримки двох медіапотоків від різних джерел. Підходить для відеоконференцій, у яких зображення виводиться на два різних екрани (приміром, у відеопереговірній, коли на одному екрані – зображення доповідача, на другому – супровідна презентація).

– H.323 – протокол передачі даних по мережах з негарантованою пропускною здатністю. Застосовується й у персональних, і в багаточечних відеоконференціях.

– SIP – мережний протокол установки з'єднання між клієнтськими додатками різних виробників, що прийшов на зміну стандарту H.323. Використовується у відеоконференцзв'язку й IP-телефонії.

Стиск і відтворення звуку й відео під час сеансу конференцзв'язку здійснюється за допомогою використання аудіо й відеокодеків:

– H.264 – стандарт стиску відео, що забезпечує високий рівень стиску відеопотоку зі збереженням первісної якості.

– H.264 Scalable Video Coding (SVC) – кодек з компенасацією відсутніх даних, що передає відео з використанням декількох шарів. Стійкий до помилок у мережі, наприклад таким, як втрата пакетів.

– H.265 – стандарт стиску відео, у якому застосовуються більше ефективні алгоритми кодування, чим в H.264. Серед особливостей даного відеокодеку можна виділити підвищену стійкість до втрати пакетів при передачі медіаданних і мінімальну затримку сигналу під час відеоконференцій. Цей стандарт підтримує формати UltraHD: 4K и 8K.

– Opus – аудіокодек для стиску звуку, що відрізняється високою продуктивністю й масштабованістю.

– G.722.1 Annex C – стандарт стиску широкополосного аудіо сигналу.

– VP8 – відеокодек з підвищеною стійкістю до втрати кадрів і високою швидкістю декодування відеопотоків.

– VP9 – відкритий стандарт стиску відео, споконвічна мета якого складалася в поліпшенні характеристик кодеків VP8 і H.265. У першому випадку (у порівнянні з VP8) основним завданням розроблювачів стало зменшення бітрейта на 50% зі збереженням споконвічної якості відео, у другому (у порівнянні з H.265) – значне поліпшення ефективності стиску відеопотоків.

3.2 Розробка структурної схеми

Сценарій BYOC (Bring Your Own Codec) – «відправляючись у переговірну, захопи свій ВКЗ-кодек (кодек відео-конференц-зв'язку)»: користувач приходить у кімнату для конференцій зі своїм ноутбуком, підключає до нього необхідні периферійні пристрої (камеру, спікерфон, дисплей) і проводить відеоконференцію

Як в умовах обмеженого бюджету охопити відеозв'язком широкі маси корпоративних користувачів? Один із привабливих варіантів – використовувати як відеотермінали звичайні ПК.

Час Flash – технології, що донедавна використовувалася для реалізації більшості вебінарів і відеосервісів в Інтернеті, а також деяких систем відео-конференц-зв'язку, – підходить до кінцю. Після 2 січня 2017 року у світі залишилося лише з десяток сайтів, для користувачів яких збережеться підтримка Flash у найпоширенішому браузері Chrome. При відвідуванні користувачами інших сайтів Chrome, швидше за все, повідомить про «застарілий плагін AdobeFlash». В інших браузерах підтримка залишиться, але з обліком того, що ця технологія не розвивається, її дні полічені.

Розробка Flash була почата компанією FutureWave, що створила пакет анімації FutureSplash Animator. В 1996 році її придбала компанія Macromedia, що перейменувала FutureSplash Animator в Flash. Після того, як в 2005 році

Macromedia була поглинена Adobe, Macromedia Flash стала офіційно називатися Adobe Flash. Flash «пожвавила» Web за рахунок векторної анімації, а потім і відео (до появи Flash єдине, чим Web-сайти відрізнялися від газет по зовнішньому вигляді, були анімовані зображення gif). Споконвічно Flash була реалізована як плагін до браузерів, потім підтримку цієї технології виробники почали вбудовувати в браузери. У роки розквіту цієї технології (середина 2000-х років) створювалися навіть сайти цілком на базі Flash. Потенційно Flash могла б повністю замінити HTML – можливо, так би й трапилося, якби технологію придбала Google, а не Adobe.

Хронологія заходу Flash приблизно така. У квітні 2010 року Том Крча заявив, що HTML5 зможе замінити Flash через 7 років. А через місяць Стив Джобс заборонив її використання на iPhone (до травня 2010 року технологія підтримувалася майже всіма платформами). Через півтора року, восени 2011-го, Adobe заявила про припинення підтримки мобільних платформ, а незабаром після цього практично заморозила розвиток технології. Як припускає Ярослав Городецький, видимо, після декількох невдач компанія змирилася з поразкою в боротьбі з титанами, що бажали смерті Flash, Google і Apple.

Зараз галузь поступово відмовляється від Flash. Більшість гравців переорієнтуються на технологію WebRTC. Apple активно розвиває розроблену ще в 2010 році (період заборони Flash на iPhone) технологію HTTP Live Streaming (HLS). Цей комунікаційний протокол для потокової передачі медіа на основі HTTP є частиною програмного забезпечення QuickTime, Safari, OS X і iOS.

Якщо багато систем для вебінарів уже переписані або активно переробляються під WebRTC, то для інших додатків можливі проблеми при відмові від підтримки Flash. Як відзначає Ярослав Городецький, Flash – сама популярна платформа браузерних ігор, що використовують багато розроблювачів, у тому числі українська AlternativaPlatform. Більшість движків відеореклами, у тому числі й український AdFox, дотепер використовують Flash. Після блокування Flash, імовірно, багато хто з них стануть працювати некоректно, що приведе до падіння ринку інтернет-реклами.

WebRTC

Для користувачів ВКЗ відмова від Flash повинен пройти безболісно. Згідно даним дослідження, проведеного OSP Data в 2016 році, компаній, у яких для ВКЗ застосовується технологія Flash, не так багато – близько 5%. У два рази більше компаній уже застосовують молоду технологію WebRTC, за допомогою якої учасники сеансу ВКЗ можуть підключатися до конференції із браузера без установки додаткового додатка. WebRTC – новий стандарт комунікацій, що має всі передумови для того, щоб стати домінуючої на ринку.

Нагадаємо, що самі по собі браузери не можуть відкрити медіасесію (для обміну голосом і відео), тому для реалізації цієї функціональності споконвічно була потрібна установка спеціальних додатків або програмних модулів (плагінів). Далеко не завжди це було прийнятно для користувачів. Більше того, виникали складності з установкою, необхідністю обновляти ПЗ й т.д. Складності додалися, коли кілька років назад виробники браузерів почали реалізовувати політики, націлені на заборону використання плагінів. Першими це зробили Chrome і Firefox, на частку яких, за різними оцінками, доводиться близько 50 і 15% всіх установлених браузерів у світі відповідно. Наприклад, як було сказано в заяві Chrome, опублікованому у вересні 2013 року, «архітектура Web-додатків 90-х років, заснована на плагінах, стає занадто громіздкої й уразливої через численні помилки, тому Chrome планує припинити підтримку модулів, що завантажуються, (плагінів) у найближчі роки». У результаті сьогодні більшість браузерів підтримують тільки плагіни з дозволеного списку.

Не дивно, що в останні роки різко прискорилася розробка браузерних технологій, що дозволяють відмовитися від плагінів. У частині реалізації функцій відеозв'язку WebRTC сьогодні стає основною альтернативою завантажуються модулям, що. Будучи частиною стандарту HTML5, WebRTC (Real-Time Communications) дозволяє реалізувати можливості телефонії, відеозв'язку й обміну файлами усередині браузера без завантаження плагіна (або зовнішнього

програмного клієнта). При цьому обмін інформацією може здійснюватися в режимі реального часу через простий програмний інтерфейс JavaScript API.

Поява технології WebRTC означає, що звичні додатки для телефонної й відеозв'язку (такі, наприклад, як Jabber і Skype) легко можуть бути замінені убудованими в Web-сторінки клієнтами, а точніше – компактним кодом, що виконується браузером. При цьому відпадає необхідність в установці додаткових додатків – браузер сам проробить всю необхідну роботу. Як тільки канал передачі даних (data channel) буде повністю підтримуватися браузером, на його базі можна буде реалізувати керування віддаленим робочим столом, передачу файлів, онлайн-гри й текстовий чат у режимі реального часу, а також будь-який інший функціонал, що вимагає обміну даними.

Технологія WebRTC істотно спрощує розробку нових додатків. Більше того, сполучення WebRTC з іншими Web-технологіями відкриває шлях до нових можливостей. Наприклад, WebGL і HTML5, застосовувані разом з WebRTC, дозволять реалізувати раніше недоступну функціональність, наприклад відеоефекти в додатках Web-трансляції або відеозв'язку. Причому зробити це можна буде з мінімальними витратами.

Безумовно, WebRTC має й свої недоліки/обмеження. Cisco указує на те, що в рамках WebRTC поки не реалізована організація сеансу, а також немає функції перевірки доступності (presence). Відкриття сеансу й перевірка присутності повинні бути виконані власним кодом додатка, що використовує виклики WebRTC. Крім того, WebRTC підтримує тільки спілкування за схемою «точка – точка»; підтримка схем «точка – багатоточка» і «багатоточка – багатоточка» відсутня.

Для інтеграції терміналів WebRTC (звичайних комп'ютерів з підтримуючу дану технологію браузерами) із класичними системами ВКЗ у більшості випадків потрібні шлюзи.

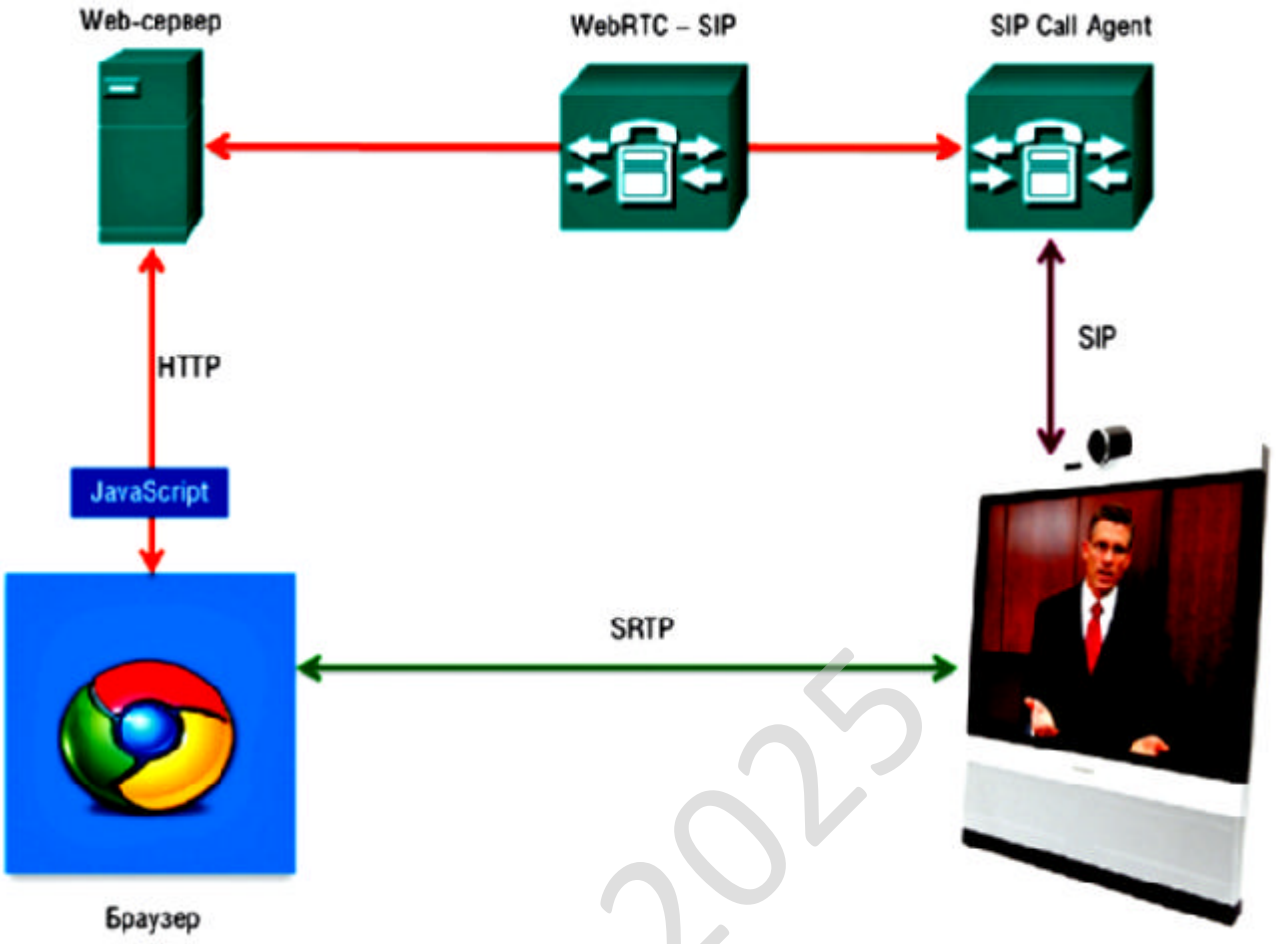

Рисунок 3.1 – Структурна схема системи

В усьому світі вже встановлено більше мільярда копій браузерів, що підтримують технологію WebRTC. По суті, кожний комп'ютер, на якому є такий браузер, – це потенційний термінал ВКЗ, що істотно розширює число можливих учасників сеансів відео-конференц-зв'язку.

Що ж стосується виробників систем ВКЗ, те, як уважає фахівець Cisco, у нових умовах конкурентну перевагу одержать ті з них, хто забезпечить у своїх продуктах убудовану підтримку відеозв'язку «із браузера». Ну а в найбільшому виграші виявляться, звичайно, користувачі, які одержують можливість практично без додаткових витрат прилучитися до сучасних комунікаційних сервісів.

Головне – звук

Важливу роль у будь-якій відеоконференції грає звук. Із трьох компонентів типового сеансу ВКЗ – відео, звук і контент (презентація) – експерти практично одностайно головним визнали саме звук. Це легко з'ясовно: голосом можна донести головну інформацію й навіть переказати презентацію. До речі, у рішеннях ряду виробників у випадку погіршення якості зв'язку саме передача голосу здійснюється з максимальним пріоритетом. Інакше кажучи, зображення може пропасти, а голосовий зв'язок залишитися.

Звук – головне джерело інновацій у переговірній кімнаті». По його даним, на аудіосистему доводиться 10-20% вартості всієї системи ВКЗ, і заощаджувати на цьому компоненті не коштує. Ідеальним рішенням для голосового зв'язку він називає гарнітуру: мікрофон максимально наближений до рота, динамік – у юшці, мінімум перешкод, найкраща якість. Однак це виняткове персональне рішення. У більшості випадків сеанси ВКЗ проводяться в переговірних кімнатах, де за столом в екрана збираються трохи співробітників. У цьому випадку потрібні групові засоби голосового зв'язку. Часто це спікерфони – пристрої, у яких об'єднані мікрофони й динаміки.

При видаленні мікрофона від спікера потужність сигналу який доходить до його (мікрофона) стрімко падає. Тому важливо зробити мікрофон максимально чутливим. Але, крім корисного сигналу (голос спікера), мікрофон захоплює й всі шуми, які необхідно забрати з переданого далі сигналу. Експерт виділяє чотири типи шумів:

– рівномірний (наприклад, від працюючого кондиціонера або проектора);

– луна (коли в мікрофон надходить сигнал від розташованого поруч динаміка й інша сторона чує себе ж із затримкою);

– кімнатний шум від реверберації (відбиття сигналу від стін і інших перешкод);

– шум від різних сторонніх джерел.

Для того щоб забрати сторонні шуми, у мікрофоні застосовуються досить складні технології фільтрації й постпроцесінгу. Наприклад, для вирізання рівномірного шуму сигнал розділяється на спектральні складові, після чого віддаляються ті з них, які переносять цей шум. Ефективним способом боротьби з реверберацією служить технологія формування спрямованого аудіолуча (слухового конуса). У цьому випадку мікрофон «націлюється» на мовець і «сприймає» тільки його голос, а не різні відбиття.

У класичній переговірній кімнаті звуки ззовні блокують стіни. Тим часом всі частіше сеанси ВКЗ проводять у відкритому офісі, коли учасники конференції не відділені ніякими стінами. У цьому випадку застосовуються різні методи для звукової ізоляції.

Звичайні ПК можуть прийти на зміну спеціалізованим кодекам групового відео-конференц-зв'язку в переговірних кімнатах. За аналогією із широко відомою концепцією BYOD, пропонується термін BYOC (Bring Your Own Codec) – «відправляючись у переговірну, захопи свій ВКЗ-кодек». Сценарій простій: прийшовши в кімнату для конференцій зі своїм ноутбуком, користувач підключає до нього необхідні периферійні пристрої (камеру, спікерфон, дисплей), після чого сеанс ВКЗ можна проводити з використанням програмного клієнта через хмарний сервіс.

Реалізацію такого сценарію спрощує той факт, що зараз практично всю периферію можна підключити по стандартному інтерфейсі USB. Більше того, широке поширення технології USB Video device Class (UVC) ставить під погрозу майбутнє інтерфейсу HDMI для підключення PTZ-камер. При приєднанні камери до ПК через такий інтерфейс з'являється можливість управляти функціями PTZ і контролювати параметри стиску потоку, при цьому не потрібні ні карта захвата, ні спеціальні драйвери.

Ще одним фактором, що потенційно спрощує реалізацію концепції BYOC, стала поява на ринку рішень «усе в одному», що поєднують в одному пристрої високоякісну відеокамеру й спікерфон, а також укомплектованих пультом

дистанційного керування функціями камери й пристрої голосного зв'язку. Такі продукти, наприклад, пропонує компанія Logitech.

Здається, що спеціалізовані кодеки так просто не поступляться своє місце ПК із програмними клієнтами. Спеціалізоване встаткування завжди забезпечує кращі характеристики для «свого» додатка, чим пристрою широкого профілю, обтяжені багатьма зайвими (для даного додатка) функціями. Крім того, робота на одному універсальному пристрої декількох додатків чревата зниженням надійності, що неприйнятно в ситуаціях, коли відеозв'язок має критично важливе значення для оперативного керування або інших бізнес-процесів. Нарешті, спеціалізовані пристрої, оптимізовані для відеокомунікацій, у принципі, можна зробити навіть дешевше високопродуктивних ПК. У кожному разі протистояння «кодеки проти ПК» підстьобне конкуренцію, а виходить, споживач в остаточному підсумку повинен виграти.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.3.

Зі схеми видно, що розроблена система складається з наступних частин:

1. Припустима розв’язна здатність захвату відео.

– для локальних відеокамер 384x288, 480x360, 560x420, 640x480, 720x540, 768x576 (напівкадр і повний кадр).

– для мережних камер 176x144, 240x180, 320x240, 352x240, 352x288, 384x288, 480x360, 560x420, 640x480, 704x420, 704x576, 720x540, 720x576, 768x576, 1280x720, 1280x960, 1280x1024, 1600x1200.

2. Програмний детектор руху.

Програмний детектор руху працює на основі порівняння кількості змінених пікселів у сусідніх відеокадрах із заданим у налаштуваннях граничним значенням для кожної відеокамери.

Додатково включає наступні технології:

– налаштування рівня шуму;

– виділення областей аналізу накладенням маски;

– групування в тимчасові сеанси руху.

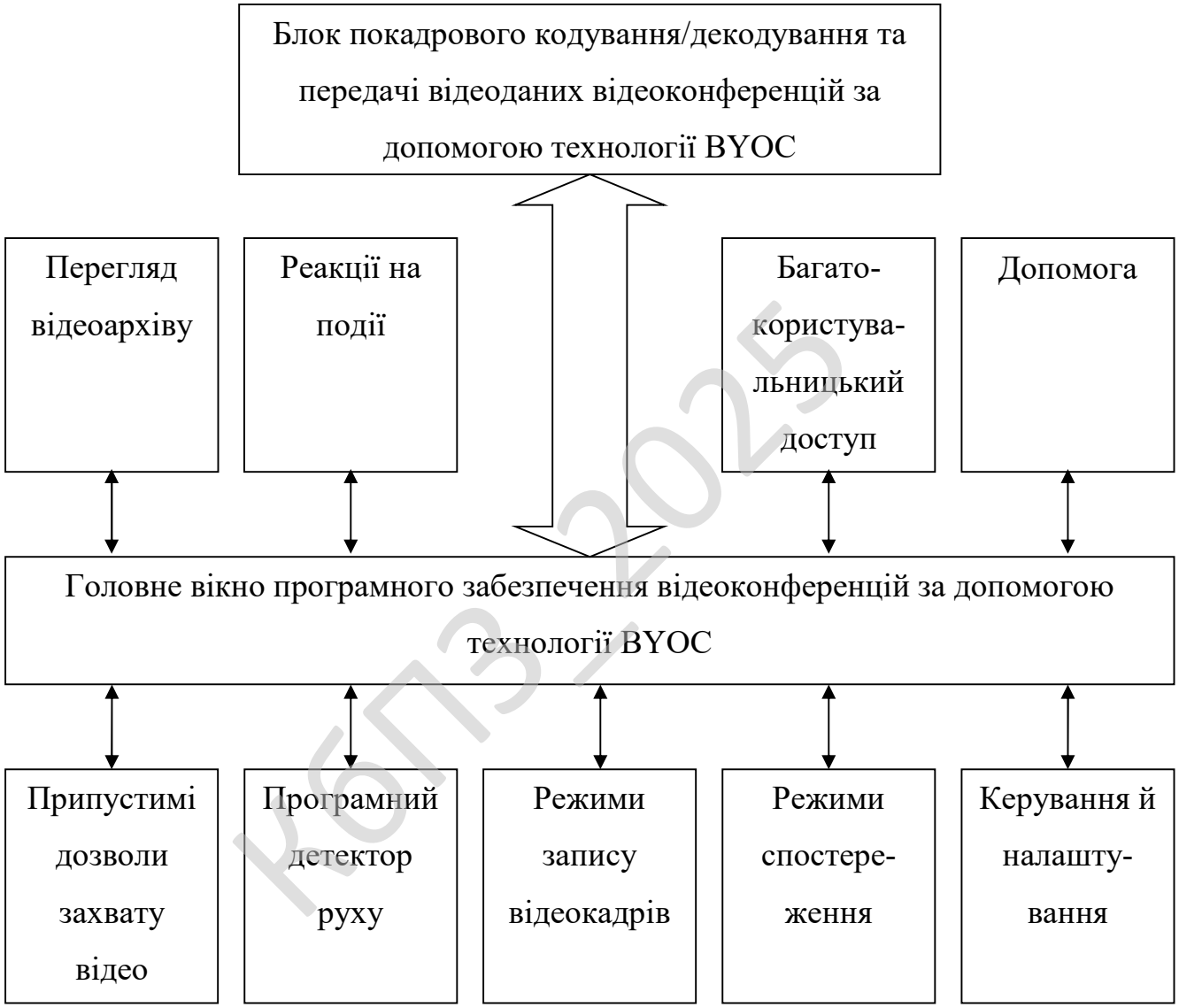

Рисунок 3.3 – Функціональна схема системи

3. Режими запису відеокадрів.

– без запису на жорсткий диск,

– запис відеокадрів по спрацьовуванню детектора;

– запис по детектору + запис певної кількості відеокадрів до/після спрацьовування детектора (режими передзапису й післязапису).

Відеокадри можуть зберігатися у файлах наступного формату:

– файли у форматі покадрового кодування та передачі відеоданих на основі ієрархічного кодування;

– окремі файли JPEG;

– фільми AVI, стиск кодеком MJPEG.

Відеоархів – циклічний (кільцевий), з видаленням частини старих даних при досягненні вільного місця жорсткого диска певної величини.

Забезпечено можливість накладення (малювання) тексту й часу на збережений відеокадр.

4. Режими спостереження.

Перегляд у режимі реального часу можливий:

– за допомогою локальної програми, що виводить зображення форматах квадратор, мультиекран, поліекран і ін.;

– за допомогою веб-браузера з дозволених віддалених комп'ютерів.

5. Керування й налаштування.

Реалізована для зареєстрованих користувачів по паролю.

6. Перегляд відеоархіву.

Реалізований для зареєстрованих користувачів по паролю.

7. Реакції на події.

– Програвання звукового файлу на початок і закінчення сеансу руху.

– Повідомлення через e-mail по початку сеансу руху із вкладеним JPEG файлом першого кадру руху (прим: по мережі на SMTP сервер).

– Виконання користувальницьких скриптів і завдань.

8. Багатокористувальницький доступ.

У системі передбачені 4 групи користувачів:

– інсталятори (установники),

– адміністратори,

– оператори архіву,

– оператори спостереження.

Кількість користувачів і віддалених робочих місць програмно не обмежено.

9. Допомога.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.4. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

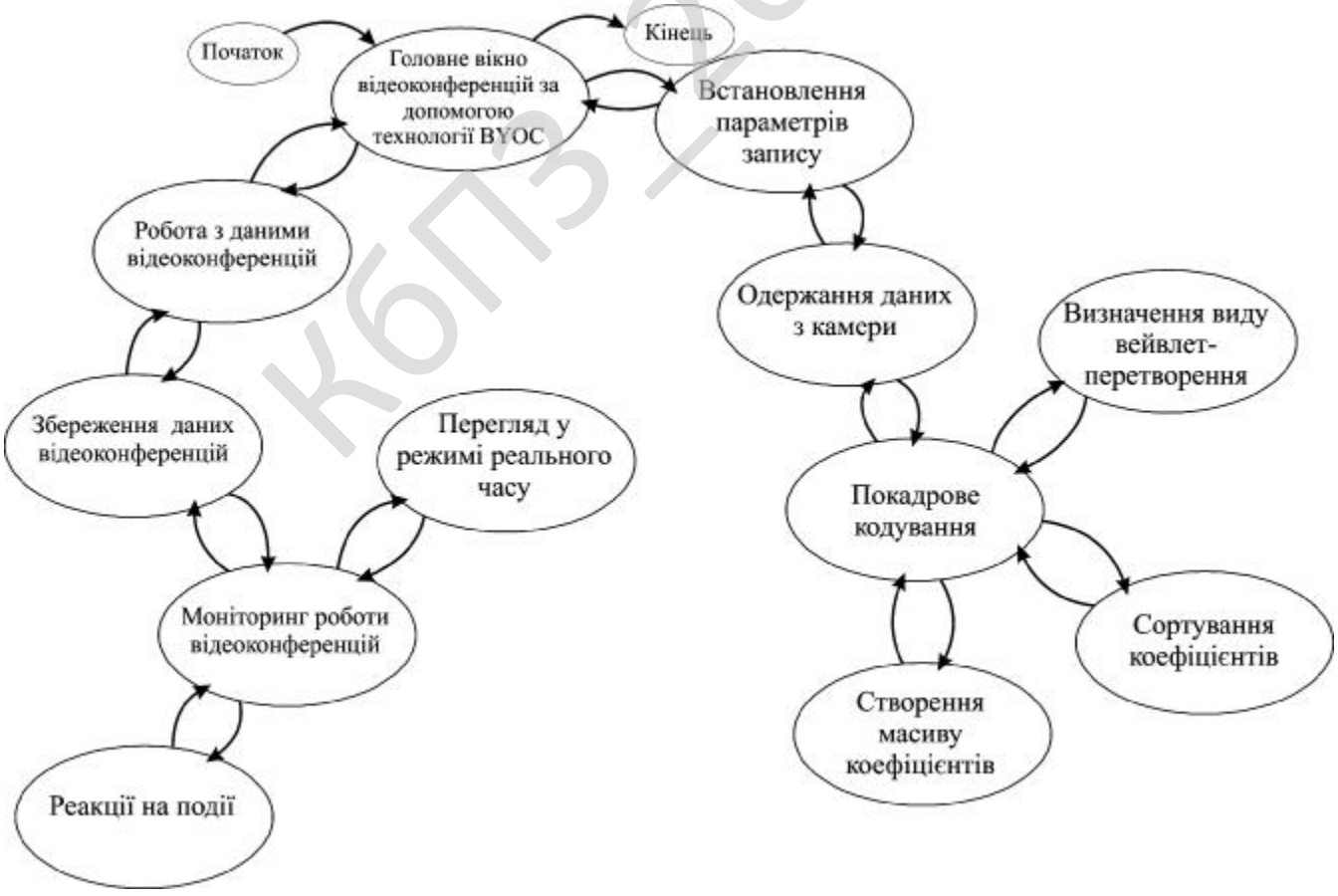

Рисунок 3.4 – Діаграма взаємодії процесів

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

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

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

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

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

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

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

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації;

– Можливість самореєстрації нових користувачів;

– Багатомовний інтерфейс (у тому числі українська мова);

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

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

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

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

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

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

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

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

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

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

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

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

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

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

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

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

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником;

2. Призначений – призначений відповідальний за виправлення дефекту;

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

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

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

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

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

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

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

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

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

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми).

Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю відеоконференцій за допомогою технології BYOC.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

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

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

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

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії.

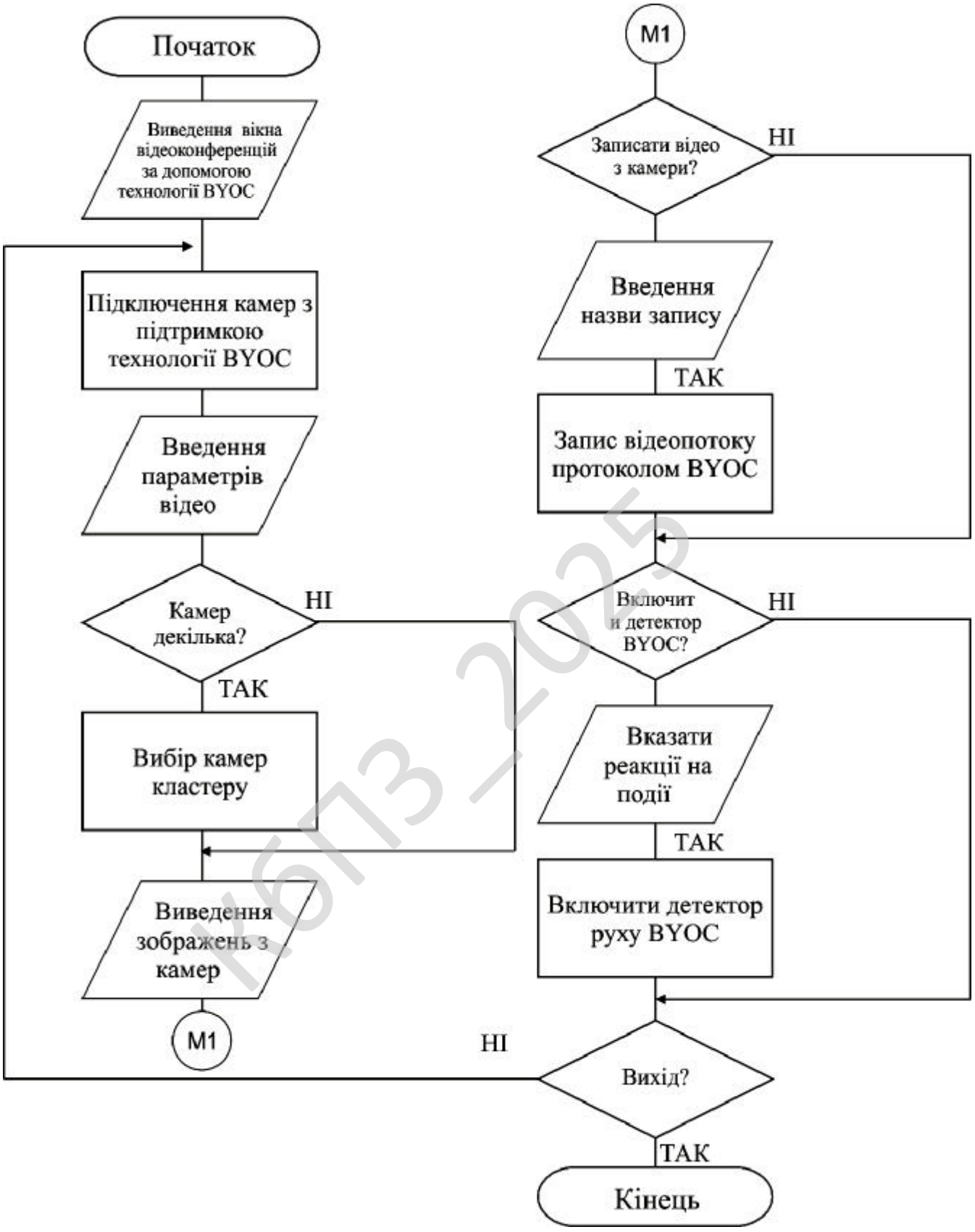

Рисунок 4.1 – Блок-схема основної програми

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

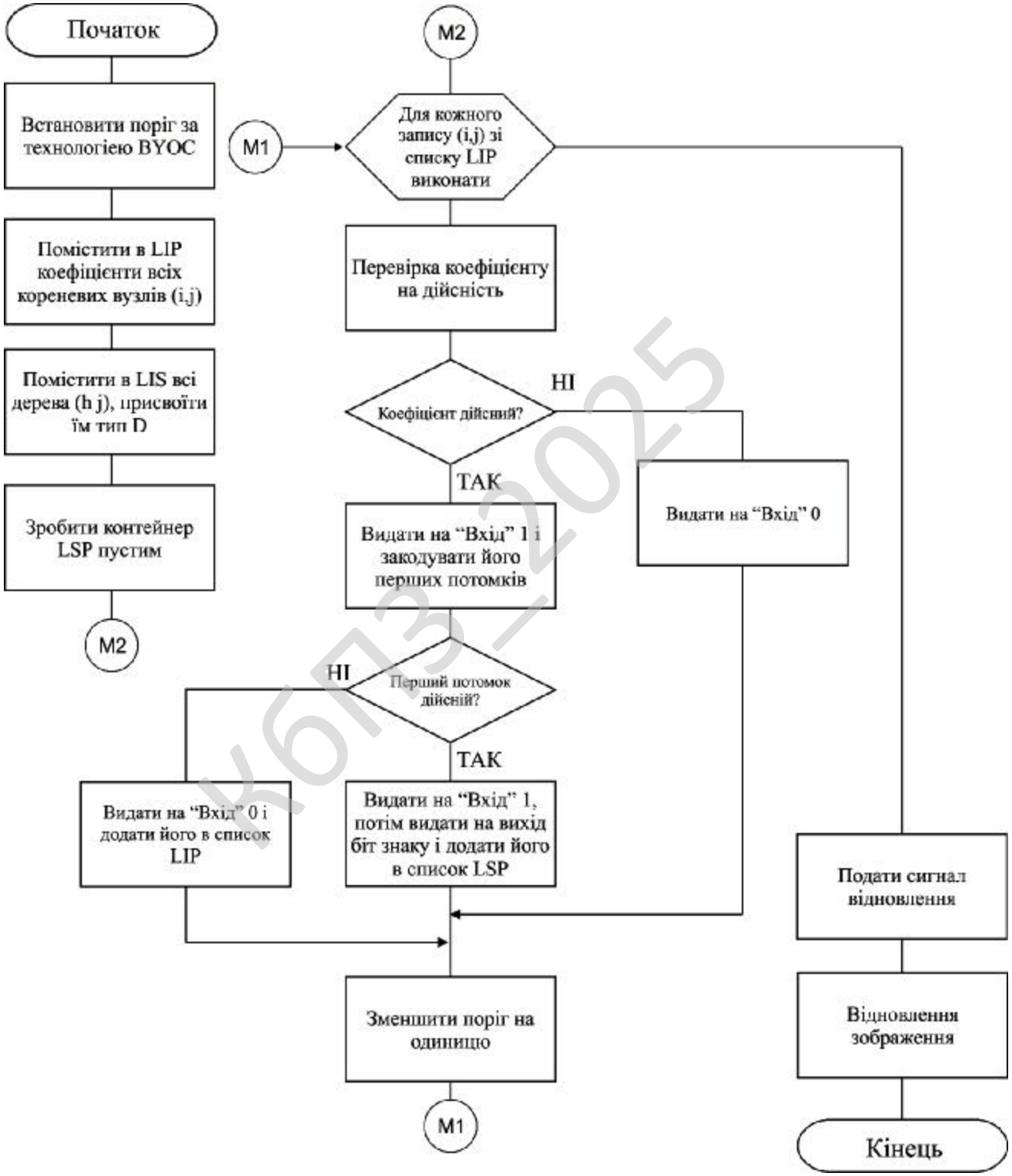

Рисунок 4.2 – Блок-схема роботи підпрограми

Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності.

Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

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

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

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

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

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

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

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

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

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

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

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

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

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

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

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі

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

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

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

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

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з

класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SEED – у криптографії симетричний блоковий криптоалгоритм на основі Мережі Фейстеля, розроблений Корейським агентством інформаційної безпеки (Korean Information Security Agency, KISA) в 1998 році. В алгоритмі використовується 128-бітний блок і ключ довжиною 128 біт.

Алгоритм одержав широке поширення й використовується фінансовими й банківськими структурами, виробничими підприємствами й бюджетними установами Південної Кореї, оскільки 40-бітний SSL не забезпечує на даний момент мінімально необхідного рівня безпеки. Агентством по захисту інформації специфіковане використання шифру SEED у протоколах TLS і S/MIME.

У той же час, алгоритм SEED не реалізований у більшості сучасних браузерів і інтернет-додатків, що утрудняє його використання в даній сфері поза межами Південної Кореї.

SEED являє собою мережу Фейстеля з 16 раундами, 128-бітовими блоками й 128-бітовим ключем.

Алгоритм використовує дві 8×8 таблиці підстановки, які, як такі з Safer, виведені з дискретного зведення в ступінь (у цьому випадку, x^{247} і x^{251} – плюс деякі «несумісні операції»).

Це є деякою подібністю с MISTY1 у рекурсивності його структури: 128-бітовий повний шифр – мережа Фейстеля з F-функцією, що впливає на 64-бітові половини, у той час як сама F-функція – Мережа Фейстеля, складена з G-функції, що впливає на 32-розрядні половини. Однак рекурсія не простягнеться далі, тому що G-функція – не Мережа Фейстеля.

В G-функції 32-розрядне слово розглядають як чотири 8-бітових байта, кожний з яких проходить через одну або іншу таблицю підстановки, потім

поєднується в поміркованo комплексному наборі булевих функцій таким чином, що кожний біт виводу залежить від 3 з 4 вхідних байтів.

SEED має складний ключовий розклад, генеруючи тридцять два 32-розрядних додаткових символу, використовуючи G-функції на серіях обертань вихідного неопрацьованого ключа, комбінованого зі спеціальними раундовими константами (як в TEA) від «Золотого співвідношення» (англ. Golden ratio).

Згідно з дослідженнями KISA, алгоритм SEED «надійно протистоїть відомим атакам».

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ відеоконференцій за допомогою технології BYOC яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Камери; Детектор руху; Параметри; Довідка.

– Розділу виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ відеоконференцій за допомогою технології BYOC.

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

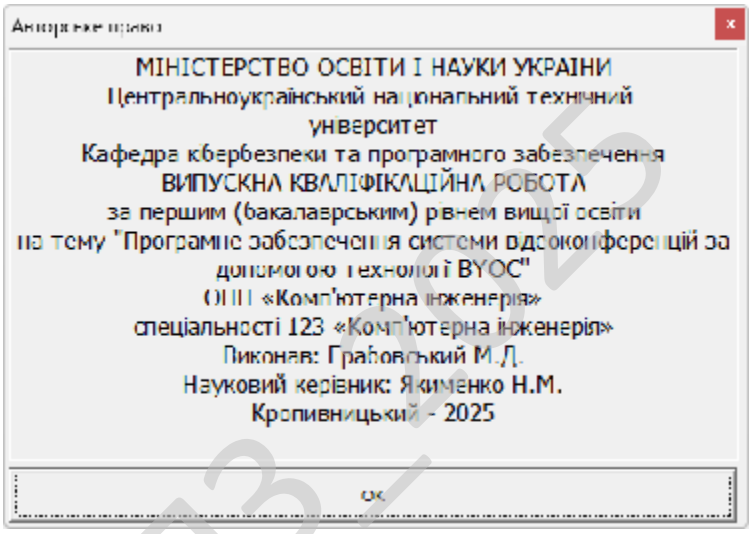

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є

унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

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

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються

в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям.

Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником.

Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи відеоконференцій за допомогою технології BYOC.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем відеоконференцій за допомогою технології BYOC.

– Досліджена система відеоконференцій за допомогою технології BYOC.

– На основі отриманих результатів досліджень створена програмна реалізація системи відеоконференцій за допомогою технології BYOC.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання відеоконференцій за допомогою технології BYOC.

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 Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи відеоконференцій за допомогою технології BYOC. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної

техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SEED.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

7. Peter Shirley, Steve Marschner. Fundamentals of Computer Graphics. 2009

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

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

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

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

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

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

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

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

16. 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.

17. 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.

18. 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.

19. 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,

20. 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.

21. 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

22. 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.

23. 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.

24. 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.

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

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

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

43. 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.

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

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи відеоконференцій за допомогою технології BYOC.

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. № 48-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи відеоконференцій за допомогою технології BYOC.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи відеоконференцій за допомогою технології BYOC;

– 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 Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 74 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи відеоконференцій за допомогою технології BYOC

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 42

Літера: РП

Кропивницький – 2025 року

Файл MultimodeVideoBYOCSource.cs – робота з відеокамерами

```
namespace multisource
{
	using System;
	using System.Collections;
	using VideoBYOCsource;
	using SPIHT;
	using mSPIHT;

	/// <summary>
	/// MultimodeVideoBYOCSource - абстрактний клас для відеоресурсів, з
підтримкою
	/// мульти робочого режиму
	/// </summary>
	public abstract class MultimodeVideoBYOCSource : IVideoBYOCSource
	{
		protected IVideoBYOCSource	VideoBYOCSource;
		protected StreamType	streamType;
		private ArrayList	delegates = new ArrayList();

		// Нова подія у фреймі
		public event CameraEventHandler NewFrame
		{
			add
			{
				VideoBYOCSource.NewFrame += value;
				delegates.Add((object) value);
			}
			remove
			{
				VideoBYOCSource.NewFrame -= value;
				delegates.Remove((object) value);
			}
		}

		// Конструктор
		public MultimodeVideoBYOCSource()
		{
		}

		// StreamType властивості
		public virtual StreamType StreamType
		{
			get { return streamType; }
			set
			{
				// покращує потоковий тип, якщо відео джерело не
рухається
				if ((streamType != value) && (!VideoBYOCSource.Running))
				{
					streamType = value;

					// зберігає дані
					object	userData = VideoBYOCSource.UserData;
					string	login = VideoBYOCSource.Login;
					string	password = VideoBYOCSource.Password;

					// створює нове базове відео джерело
					switch (streamType)
					{
						case StreamType.SPIHT:
							VideoBYOCSource = new SPIHTSource();
							break;
						case StreamType.MSPIHT:
							VideoBYOCSource = new MSPIHTSource();
```

```
                                    break;
                                }

                                // бере дані та повертає у нове відеоджерело
                                VideoBYOCSource.Login         = login;
                                VideoBYOCSource.Password      = password;
                                VideoBYOCSource.UserData      = userData;

                                // додає делегування до NewFrame події
                                foreach (object handler in delegates)
                                      VideoBYOCSource.NewFrame +=
(CameraEventHandler) handler;

                                UpdateVideoBYOCSource();
                          }
                    }
              }

              // Властивості відеоджерела
              public abstract string VideoBYOCSource
              {
                    get;
                    set;
              }

              // Властивості підключення
              public string Login
              {
                    get { return VideoBYOCSource.Login; }
                    set { VideoBYOCSource.Login = value; }
              }

              // Властивості паролювання
              public string Password
              {
                    get { return VideoBYOCSource.Password; }
                    set { VideoBYOCSource.Password = value; }
              }

              // FramesReceived властивості
              public int FramesReceived
              {
                    get { return VideoBYOCSource.FramesReceived; }
              }

              // BytesReceived властивості
              public int BytesReceived
              {
                    get { return VideoBYOCSource.BytesReceived; }
              }

              // UserData властивості
              public object UserData
              {
                    get { return VideoBYOCSource.UserData; }
                    set { VideoBYOCSource.UserData = value; }
              }

              // Беремо стан відеоджерела
              public bool Running
              {
                    get { return VideoBYOCSource.Running; }
              }

              // Починаємо отримувати відеофрейми
              public void Start()
              {
                    VideoBYOCSource.Start();
              }
```

```
            // Закінчуємо отримувати відеофрейми
            public void SignalToStop()
            {
                  VideoBYOCSource.SignalToStop();
            }

            // Чекаємо закінчення
            public void WaitForStop()
            {
                  VideoBYOCSource.WaitForStop();
            }

            // Закінчення роботи
            public void Stop()
            {
                  VideoBYOCSource.Stop();
            }

            // Обновлюємо відеоджерело
            protected abstract void UpdateVideoBYOCSource();
      }
}
```

Файл CameraInfo.cs – отримання інформації про камеру

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace CameraViewer
{
	/// <summary>
	/// інформація з CameraInfo.
	/// </summary>
	public class CameraInfo : System.Windows.Forms.Form
	{
		private Camera camera;

		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.Label widthLabel;
		private System.Windows.Forms.Label label2;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.Label nameLabel;
		private System.Windows.Forms.Label label5;
		private System.Windows.Forms.Label descriptionLabel;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.Label providerLabel;
		private System.Windows.Forms.PictureBox pictureBox1;
		private System.Windows.Forms.Label heightLabel;
		private System.Windows.Forms.Button closeButton;
		private System.Windows.Forms.PictureBox pictureBox2;
		/// <summary>
		/// Необхідні змінні розробника.
		/// </summary>
		private System.ComponentModel.Container components = null;

		// Camera властивості
		public Camera Camera
		{
			get { return camera; }
			set { camera = value; }
		}

		// Конструктор
		public CameraInfo()
		{
			//
			// Необхідно для підтримки Windows Form Designer
			//
			InitializeComponent();

			//
			// ПРИМІТКА Додаємо любий конструктор коду після виклику
InitializeComponent
			//
		}

		/// <summary>
		/// Очищуємо використані ресурси.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
```

```
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Необхідний метод розробника – не модифікується
        /// зміст цього методу з редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.widthLabel = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.nameLabel = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.descriptionLabel = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.providerLabel = new System.Windows.Forms.Label();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.heightLabel = new System.Windows.Forms.Label();
            this.closeButton = new System.Windows.Forms.Button();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(8, 136);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(40, 14);
            this.label1.TabIndex = 0;
            this.label1.Text = "Width:";
            //
            // widthLabel
            //
            this.widthLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.widthLabel.Location = new System.Drawing.Point(60, 135);
            this.widthLabel.Name = "widthLabel";
            this.widthLabel.Size = new System.Drawing.Size(60, 16);
            this.widthLabel.TabIndex = 1;
            this.widthLabel.TextAlign =
System.Drawing.ContentAlignment.TopCenter;
            //
            // label2
            //
            this.label2.Location = new System.Drawing.Point(140, 136);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(40, 14);
            this.label2.TabIndex = 2;
            this.label2.Text = "Height:";
            //
            // label3
            //
            this.label3.Location = new System.Drawing.Point(8, 9);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(40, 14);
            this.label3.TabIndex = 3;
            this.label3.Text = "Name:";
            //
            // nameLabel
            //
            this.nameLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.nameLabel.Location = new System.Drawing.Point(60, 8);
            this.nameLabel.Name = "nameLabel";
            this.nameLabel.Size = new System.Drawing.Size(200, 16);
            this.nameLabel.TabIndex = 4;
```

```
                    //
                    // label5
                    //
                    this.label5.Location = new System.Drawing.Point(8, 30);
                    this.label5.Name = "label5";
                    this.label5.Size = new System.Drawing.Size(65, 14);
                    this.label5.TabIndex = 5;
                    this.label5.Text = "Description:";
                    //
                    // descriptionLabel
                    //
                    this.descriptionLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
                    this.descriptionLabel.Location = new System.Drawing.Point(8,
47);
                    this.descriptionLabel.Name = "descriptionLabel";
                    this.descriptionLabel.Size = new System.Drawing.Size(252, 40);
                    this.descriptionLabel.TabIndex = 6;
                    //
                    // label4
                    //
                    this.label4.Location = new System.Drawing.Point(8, 96);
                    this.label4.Name = "label4";
                    this.label4.Size = new System.Drawing.Size(50, 14);
                    this.label4.TabIndex = 7;
                    this.label4.Text = "Provider:";
                    //
                    // providerLabel
                    //
                    this.providerLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
                    this.providerLabel.Location = new System.Drawing.Point(60,
95);
                    this.providerLabel.Name = "providerLabel";
                    this.providerLabel.Size = new System.Drawing.Size(200, 16);
                    this.providerLabel.TabIndex = 8;
                    //
                    // pictureBox1
                    //
                    this.pictureBox1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
                    this.pictureBox1.Location = new System.Drawing.Point(8, 120);
                    this.pictureBox1.Name = "pictureBox1";
                    this.pictureBox1.Size = new System.Drawing.Size(252, 2);
                    this.pictureBox1.TabIndex = 9;
                    this.pictureBox1.TabStop = false;
                    //
                    // heightLabel
                    //
                    this.heightLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
                    this.heightLabel.Location = new System.Drawing.Point(200,
135);
                    this.heightLabel.Name = "heightLabel";
                    this.heightLabel.Size = new System.Drawing.Size(60, 16);
                    this.heightLabel.TabIndex = 10;
                    this.heightLabel.TextAlign =
System.Drawing.ContentAlignment.TopCenter;
                    //
                    // closeButton
                    //
                    this.closeButton.DialogResult =
System.Windows.Forms.DialogResult.OK;
                    this.closeButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
                    this.closeButton.Location = new System.Drawing.Point(98, 175);
                    this.closeButton.Name = "closeButton";
                    this.closeButton.TabIndex = 11;
                    this.closeButton.Text = "Close";
```

```
                //
                // pictureBox2
                //
                this.pictureBox2.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
                this.pictureBox2.Location = new System.Drawing.Point(9, 163);
                this.pictureBox2.Name = "pictureBox2";
                this.pictureBox2.Size = new System.Drawing.Size(252, 2);
                this.pictureBox2.TabIndex = 12;
                this.pictureBox2.TabStop = false;
                //
                // CameraInfo
                //
                this.AcceptButton = this.closeButton;
                this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
                this.CancelButton = this.closeButton;
                this.ClientSize = new System.Drawing.Size(270, 206);
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                                this.pictureBox2,

                                this.closeButton,

                                this.heightLabel,

                                this.pictureBox1,

                                this.providerLabel,

                                this.label4,

                                this.descriptionLabel,

                                this.label5,

                                this.nameLabel,

                                this.label3,

                                this.label2,

                                this.widthLabel,

                                this.label1});
                this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedToolWindow;
                this.MaximizeBox = false;
                this.MinimizeBox = false;
                this.Name = "CameraInfo";
                this.Opacity = 0.85;
                this.ShowInTaskbar = false;
                this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterParent;
                this.Text = "Camera info";
                this.Load += new System.EventHandler(this.CameraInfo_Load);
                this.ResumeLayout(false);

            }
            #endregion

            // On load
            private void CameraInfo_Load(object sender, System.EventArgs e)
            {
                  if (camera != null)
                  {
                        nameLabel.Text = camera.Name;
                        descriptionLabel.Text = camera.Description;
                        providerLabel.Text = camera.Provider.Name;
```

```
                        if (camera.Width != -1)
                        {
                              widthLabel.Text = camera.Width.ToString();
                              heightLabel.Text = camera.Height.ToString();
                        }
                        else
                        {
                              widthLabel.Text = string.Empty;
                              heightLabel.Text = string.Empty;
                        }
                  }
            }
      }
}
```

Файл Multiplexer.cs – мультиплексування (для роботи з декількома камерами)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;

namespace CameraViewer
{
	/// <summary>
	/// Загальний опис для мультиплексора.
	/// </summary>
	public class Multiplexer : System.Windows.Forms.Panel
	{
		private const int	MaxRows = 5;
		private const int	MaxCols = 5;
		private CameraWindow[,] camWindows;

		private bool	fitToWindow = false;
		private bool	singleCameraMode = true;
		private bool	camerasVisible = false;

		private int	rows = 1;
		private int	cols = 1;
		private int	cellWidth = 320;
		private int	cellHeight = 240;

		private CameraWindow	lastClicked;

		private CameraViewer.CameraWindow cameraWindow1;
		private CameraViewer.CameraWindow cameraWindow2;
		private CameraViewer.CameraWindow cameraWindow3;
		private CameraViewer.CameraWindow cameraWindow4;
		private CameraViewer.CameraWindow cameraWindow5;
		private CameraViewer.CameraWindow cameraWindow6;
		private CameraViewer.CameraWindow cameraWindow7;
		private CameraViewer.CameraWindow cameraWindow8;
		private CameraViewer.CameraWindow cameraWindow9;
		private CameraViewer.CameraWindow cameraWindow10;
		private CameraViewer.CameraWindow cameraWindow11;
		private CameraViewer.CameraWindow cameraWindow12;
		private CameraViewer.CameraWindow cameraWindow13;
		private CameraViewer.CameraWindow cameraWindow14;
		private CameraViewer.CameraWindow cameraWindow15;
		private CameraViewer.CameraWindow cameraWindow16;
		private CameraViewer.CameraWindow cameraWindow17;
		private CameraViewer.CameraWindow cameraWindow18;
		private CameraViewer.CameraWindow cameraWindow19;
		private CameraViewer.CameraWindow cameraWindow20;
		private CameraViewer.CameraWindow cameraWindow21;
		private CameraViewer.CameraWindow cameraWindow22;
		private CameraViewer.CameraWindow cameraWindow23;
		private CameraViewer.CameraWindow cameraWindow24;
		private CameraViewer.CameraWindow cameraWindow25;
		/// <summary>
		/// Опис змінних розробника.
		/// </summary>
		private System.ComponentModel.Container components = null;

		// FitToWindow властивості
		[DefaultValue(false)]
		public bool FitToWindow
		{
			get { return fitToWindow; }
			set
```

```
                {
                        fitToWindow = value;

                        if ((camWindows[0, 0].AutoSize = (!fitToWindow &&
singleCameraMode)) == true)
                        {
                                camWindows[0, 0].UpdatePosition();
                        }
                        else
                        {
                                UpdateSize();
                        }
                }
        }
        // SingleCameraMode властивості
        [DefaultValue(true)]
        public bool SingleCameraMode
        {
                get { return singleCameraMode; }
                set
                {
                        singleCameraMode = value;
                        if (!fitToWindow)
                                camWindows[0, 0].AutoSize = value;
                }
        }
        // CamerasVisible властивості
        [DefaultValue(false)]
        public bool CamerasVisible
        {
                get { return camerasVisible; }
                set
                {
                        camerasVisible = value;

                        // Показувати/приховувати усі камери
                        for (int i = 0; i < rows; i++)
                        {
                                for (int j = 0; j < cols; j++)
                                {
                                        camWindows[i, j].Visible = value;
                                }
                        }
                }
        }
        // Rows властивості
        [DefaultValue(1)]
        public int Rows
        {
                get { return rows; }
                set
                {
                        rows = Math.Max(1, Math.Min(MaxRows, value));
                        UpdateVisiblity();
                        UpdateSize();
                }
        }
        // Cols властивості
        [DefaultValue(1)]
        public int Cols
        {
                get { return cols; }
                set
                {
                        cols = Math.Max(1, Math.Min(MaxCols, value));
                        UpdateVisiblity();
                        UpdateSize();
                }
        }
```

```
		// CellWidth
		[DefaultValue(320)]
		public int CellWidth
		{
			get { return cellWidth; }
			set
			{
				cellWidth = Math.Max(50, Math.Min(800, value));
				UpdateSize();
			}
		}
		// CellHeight
		[DefaultValue(240)]
		public int CellHeight
		{
			get { return cellHeight; }
			set
			{
				cellHeight = Math.Max(50, Math.Min(800, value));
				UpdateSize();
			}
		}
		// Контекстне меню у вікні камери
		[DefaultValue(null)]
		public ContextMenu CamerasContextMenu
		{
			get { return camWindows[0, 0].ContextMenu; }
			set
			{
				for (int i = 0; i < MaxRows; i++)
				{
					for (int j = 0; j < MaxCols; j++)
					{
						camWindows[i, j].ContextMenu = value;
					}
				}
			}
		}
		// Камера при останньому нажатті
		[Browsable(false)]
		public Camera ContextCamera
		{
			get { return (lastClicked == null) ? null :
lastClicked.Camera; }
		}

		// Конструктор
		public Multiplexer()
		{
			// Цей виклик використовується у Windows.Forms Form Designer.
			InitializeComponent();

			// ПРИМІТКА Додається ініціалізація після виклику InitForm
			camWindows = new CameraWindow[MaxRows, MaxCols];

			// row 1
			camWindows[0, 0] = cameraWindow1;
			camWindows[0, 1] = cameraWindow2;
			camWindows[0, 2] = cameraWindow3;
			camWindows[0, 3] = cameraWindow4;
			camWindows[0, 4] = cameraWindow5;
			// row 2
			camWindows[1, 0] = cameraWindow6;
			camWindows[1, 1] = cameraWindow7;
			camWindows[1, 2] = cameraWindow8;
			camWindows[1, 3] = cameraWindow9;
			camWindows[1, 4] = cameraWindow10;
			// row 3
			camWindows[2, 0] = cameraWindow11;
```

```
            camWindows[2, 1] = cameraWindow12;
            camWindows[2, 2] = cameraWindow13;
            camWindows[2, 3] = cameraWindow14;
            camWindows[2, 4] = cameraWindow15;
            // row 4
            camWindows[3, 0] = cameraWindow16;
            camWindows[3, 1] = cameraWindow17;
            camWindows[3, 2] = cameraWindow18;
            camWindows[3, 3] = cameraWindow19;
            camWindows[3, 4] = cameraWindow20;
            // row 5
            camWindows[4, 0] = cameraWindow21;
            camWindows[4, 1] = cameraWindow22;
            camWindows[4, 2] = cameraWindow23;
            camWindows[4, 3] = cameraWindow24;
            camWindows[4, 4] = cameraWindow25;
      }

      /// <summary>
      /// Очищуємо усі ресурси використовувані користувачем.
      /// </summary>
      protected override void Dispose( bool disposing )
      {
            if( disposing )
            {
                  if(components != null)
                  {
                        components.Dispose();
                  }
            }
            base.Dispose( disposing );
      }

      #region Component Designer generated code
      /// <summary>
      /// Необхідний метод для підтримки розробника - не модифікується
      /// контент цього методу з редактором коду.
      /// </summary>
      private void InitializeComponent()
      {
            this.cameraWindow1 = new CameraViewer.CameraWindow();
            this.cameraWindow2 = new CameraViewer.CameraWindow();
            this.cameraWindow3 = new CameraViewer.CameraWindow();
            this.cameraWindow4 = new CameraViewer.CameraWindow();
            this.cameraWindow5 = new CameraViewer.CameraWindow();
            this.cameraWindow6 = new CameraViewer.CameraWindow();
            this.cameraWindow7 = new CameraViewer.CameraWindow();
            this.cameraWindow8 = new CameraViewer.CameraWindow();
            this.cameraWindow9 = new CameraViewer.CameraWindow();
            this.cameraWindow10 = new CameraViewer.CameraWindow();
            this.cameraWindow11 = new CameraViewer.CameraWindow();
            this.cameraWindow12 = new CameraViewer.CameraWindow();
            this.cameraWindow13 = new CameraViewer.CameraWindow();
            this.cameraWindow14 = new CameraViewer.CameraWindow();
            this.cameraWindow15 = new CameraViewer.CameraWindow();
            this.cameraWindow16 = new CameraViewer.CameraWindow();
            this.cameraWindow17 = new CameraViewer.CameraWindow();
            this.cameraWindow18 = new CameraViewer.CameraWindow();
            this.cameraWindow19 = new CameraViewer.CameraWindow();
            this.cameraWindow20 = new CameraViewer.CameraWindow();
            this.cameraWindow21 = new CameraViewer.CameraWindow();
            this.cameraWindow22 = new CameraViewer.CameraWindow();
            this.cameraWindow23 = new CameraViewer.CameraWindow();
            this.cameraWindow24 = new CameraViewer.CameraWindow();
            this.cameraWindow25 = new CameraViewer.CameraWindow();
            this.SuspendLayout();
            //
            // cameraWindow1
            //
```

```
                this.cameraWindow1.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow1.Camera = null;
                this.cameraWindow1.Location = new System.Drawing.Point(285,
17);
                this.cameraWindow1.Name = "cameraWindow1";
                this.cameraWindow1.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow1.TabIndex = 0;
                this.cameraWindow1.Text = "cameraWindow1";
                this.cameraWindow1.Visible = false;
                this.cameraWindow1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow2
                //
                this.cameraWindow2.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow2.Camera = null;
                this.cameraWindow2.Location = new System.Drawing.Point(151,
17);
                this.cameraWindow2.Name = "cameraWindow2";
                this.cameraWindow2.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow2.TabIndex = 1;
                this.cameraWindow2.Text = "cameraWindow2";
                this.cameraWindow2.Visible = false;
                this.cameraWindow2.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow3
                //
                this.cameraWindow3.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow3.Camera = null;
                this.cameraWindow3.Location = new System.Drawing.Point(419,
17);
                this.cameraWindow3.Name = "cameraWindow3";
                this.cameraWindow3.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow3.TabIndex = 2;
                this.cameraWindow3.Text = "cameraWindow3";
                this.cameraWindow3.Visible = false;
                this.cameraWindow3.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow4
                //
                this.cameraWindow4.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow4.Camera = null;
                this.cameraWindow4.Location = new System.Drawing.Point(553,
17);
                this.cameraWindow4.Name = "cameraWindow4";
                this.cameraWindow4.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow4.TabIndex = 3;
                this.cameraWindow4.Text = "cameraWindow4";
                this.cameraWindow4.Visible = false;
                this.cameraWindow4.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow5
                //
                this.cameraWindow5.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow5.Camera = null;
                this.cameraWindow5.Location = new System.Drawing.Point(17,
54);
                this.cameraWindow5.Name = "cameraWindow5";
                this.cameraWindow5.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow5.TabIndex = 4;
                this.cameraWindow5.Text = "cameraWindow5";
```

```
                this.cameraWindow5.Visible = false;
                this.cameraWindow5.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow6
                //
                this.cameraWindow6.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow6.Camera = null;
                this.cameraWindow6.Location = new System.Drawing.Point(17,
91);
                this.cameraWindow6.Name = "cameraWindow6";
                this.cameraWindow6.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow6.TabIndex = 5;
                this.cameraWindow6.Text = "cameraWindow6";
                this.cameraWindow6.Visible = false;
                this.cameraWindow6.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow7
                //
                this.cameraWindow7.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow7.Camera = null;
                this.cameraWindow7.Location = new System.Drawing.Point(17,
91);
                this.cameraWindow7.Name = "cameraWindow7";
                this.cameraWindow7.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow7.TabIndex = 6;
                this.cameraWindow7.Text = "cameraWindow7";
                this.cameraWindow7.Visible = false;
                this.cameraWindow7.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow8
                //
                this.cameraWindow8.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow8.Camera = null;
                this.cameraWindow8.Location = new System.Drawing.Point(17,
91);
                this.cameraWindow8.Name = "cameraWindow8";
                this.cameraWindow8.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow8.TabIndex = 7;
                this.cameraWindow8.Text = "cameraWindow8";
                this.cameraWindow8.Visible = false;
                this.cameraWindow8.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow9
                //
                this.cameraWindow9.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow9.Camera = null;
                this.cameraWindow9.Location = new System.Drawing.Point(151,
91);
                this.cameraWindow9.Name = "cameraWindow9";
                this.cameraWindow9.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow9.TabIndex = 8;
                this.cameraWindow9.Text = "cameraWindow9";
                this.cameraWindow9.Visible = false;
                this.cameraWindow9.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow10
                //
                this.cameraWindow10.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow10.Camera = null;
```

```
                this.cameraWindow10.Location = new System.Drawing.Point(328,
80);
                this.cameraWindow10.Name = "cameraWindow10";
                this.cameraWindow10.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow10.TabIndex = 9;
                this.cameraWindow10.Text = "cameraWindow10";
                this.cameraWindow10.Visible = false;
                this.cameraWindow10.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow11
                //
                this.cameraWindow11.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow11.Camera = null;
                this.cameraWindow11.Location = new System.Drawing.Point(8,
152);
                this.cameraWindow11.Name = "cameraWindow11";
                this.cameraWindow11.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow11.TabIndex = 10;
                this.cameraWindow11.Text = "cameraWindow11";
                this.cameraWindow11.Visible = false;
                this.cameraWindow11.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow12
                //
                this.cameraWindow12.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow12.Camera = null;
                this.cameraWindow12.Location = new System.Drawing.Point(88,
152);
                this.cameraWindow12.Name = "cameraWindow12";
                this.cameraWindow12.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow12.TabIndex = 11;
                this.cameraWindow12.Text = "cameraWindow12";
                this.cameraWindow12.Visible = false;
                this.cameraWindow12.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow13
                //
                this.cameraWindow13.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow13.Camera = null;
                this.cameraWindow13.Location = new System.Drawing.Point(228,
152);
                this.cameraWindow13.Name = "cameraWindow13";
                this.cameraWindow13.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow13.TabIndex = 12;
                this.cameraWindow13.Text = "cameraWindow13";
                this.cameraWindow13.Visible = false;
                this.cameraWindow13.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow14
                //
                this.cameraWindow14.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow14.Camera = null;
                this.cameraWindow14.Location = new System.Drawing.Point(248,
152);
                this.cameraWindow14.Name = "cameraWindow14";
                this.cameraWindow14.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow14.TabIndex = 13;
                this.cameraWindow14.Text = "cameraWindow14";
                this.cameraWindow14.Visible = false;
                this.cameraWindow14.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
```

```
                //
                // cameraWindow15
                //
                this.cameraWindow15.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow15.Camera = null;
                this.cameraWindow15.Location = new System.Drawing.Point(388,
152);
                this.cameraWindow15.Name = "cameraWindow15";
                this.cameraWindow15.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow15.TabIndex = 14;
                this.cameraWindow15.Text = "cameraWindow15";
                this.cameraWindow15.Visible = false;
                this.cameraWindow15.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow16
                //
                this.cameraWindow16.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow16.Camera = null;
                this.cameraWindow16.Location = new System.Drawing.Point(528,
152);
                this.cameraWindow16.Name = "cameraWindow16";
                this.cameraWindow16.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow16.TabIndex = 15;
                this.cameraWindow16.Text = "cameraWindow16";
                this.cameraWindow16.Visible = false;
                this.cameraWindow16.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow17
                //
                this.cameraWindow17.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow17.Camera = null;
                this.cameraWindow17.Location = new System.Drawing.Point(17,
189);
                this.cameraWindow17.Name = "cameraWindow17";
                this.cameraWindow17.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow17.TabIndex = 16;
                this.cameraWindow17.Text = "cameraWindow17";
                this.cameraWindow17.Visible = false;
                this.cameraWindow17.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow18
                //
                this.cameraWindow18.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow18.Camera = null;
                this.cameraWindow18.Location = new System.Drawing.Point(157,
189);
                this.cameraWindow18.Name = "cameraWindow18";
                this.cameraWindow18.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow18.TabIndex = 17;
                this.cameraWindow18.Text = "cameraWindow18";
                this.cameraWindow18.Visible = false;
                this.cameraWindow18.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow19
                //
                this.cameraWindow19.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow19.Camera = null;
                this.cameraWindow19.Location = new System.Drawing.Point(297,
189);
                this.cameraWindow19.Name = "cameraWindow19";
```

```
                this.cameraWindow19.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow19.TabIndex = 18;
                this.cameraWindow19.Text = "cameraWindow19";
                this.cameraWindow19.Visible = false;
                this.cameraWindow19.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow20
                //
                this.cameraWindow20.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow20.Camera = null;
                this.cameraWindow20.Location = new System.Drawing.Point(17,
261);
                this.cameraWindow20.Name = "cameraWindow20";
                this.cameraWindow20.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow20.TabIndex = 19;
                this.cameraWindow20.Text = "cameraWindow20";
                this.cameraWindow20.Visible = false;
                this.cameraWindow20.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow21
                //
                this.cameraWindow21.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow21.Camera = null;
                this.cameraWindow21.Location = new System.Drawing.Point(17,
298);
                this.cameraWindow21.Name = "cameraWindow21";
                this.cameraWindow21.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow21.TabIndex = 20;
                this.cameraWindow21.Text = "cameraWindow21";
                this.cameraWindow21.Visible = false;
                this.cameraWindow21.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow22
                //
                this.cameraWindow22.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow22.Camera = null;
                this.cameraWindow22.Location = new System.Drawing.Point(17,
335);
                this.cameraWindow22.Name = "cameraWindow22";
                this.cameraWindow22.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow22.TabIndex = 21;
                this.cameraWindow22.Text = "cameraWindow22";
                this.cameraWindow22.Visible = false;
                this.cameraWindow22.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow23
                //
                this.cameraWindow23.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow23.Camera = null;
                this.cameraWindow23.Location = new System.Drawing.Point(17,
335);
                this.cameraWindow23.Name = "cameraWindow23";
                this.cameraWindow23.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow23.TabIndex = 22;
                this.cameraWindow23.Text = "cameraWindow23";
                this.cameraWindow23.Visible = false;
                this.cameraWindow23.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow24
                //
```

```
                this.cameraWindow24.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow24.Camera = null;
                this.cameraWindow24.Location = new System.Drawing.Point(17,
335);
                this.cameraWindow24.Name = "cameraWindow24";
                this.cameraWindow24.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow24.TabIndex = 23;
                this.cameraWindow24.Text = "cameraWindow24";
                this.cameraWindow24.Visible = false;
                this.cameraWindow24.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // cameraWindow25
                //
                this.cameraWindow25.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.cameraWindow25.Camera = null;
                this.cameraWindow25.Location = new System.Drawing.Point(17,
335);
                this.cameraWindow25.Name = "cameraWindow25";
                this.cameraWindow25.Size = new System.Drawing.Size(75, 64);
                this.cameraWindow25.TabIndex = 24;
                this.cameraWindow25.Text = "cameraWindow25";
                this.cameraWindow25.Visible = false;
                this.cameraWindow25.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.cameraWindow_MouseDown);
                //
                // Мультиплексер
                //
                this.Controls.AddRange(new System.Windows.Forms.Control[] {
                                this.cameraWindow25,
                                this.cameraWindow24,
                                this.cameraWindow23,
                                this.cameraWindow22,
                                this.cameraWindow21,
                                this.cameraWindow20,
                                this.cameraWindow19,
                                this.cameraWindow18,
                                this.cameraWindow17,
                                this.cameraWindow16,
                                this.cameraWindow15,
                                this.cameraWindow14,
                                this.cameraWindow13,
                                this.cameraWindow12,
                                this.cameraWindow11,
                                this.cameraWindow10,
                                this.cameraWindow9,
                                this.cameraWindow8,
                                this.cameraWindow7,
```

```
                                this.cameraWindow6,

                                this.cameraWindow5,

                                this.cameraWindow4,

                                this.cameraWindow3,

                                this.cameraWindow2,

                                this.cameraWindow1});
                  this.Size = new System.Drawing.Size(424, 376);
                  this.Resize += new
System.EventHandler(this.Multiplexer_Resize);
                  this.ResumeLayout(false);

            }
            #endregion


            // Закриваємо усі камери
            public void CloseAll()
            {
                  for (int i = 0; i < MaxRows; i++)
                  {
                        for (int j = 0; j < MaxCols; j++)
                        {
                              camWindows[i, j].Camera = null;
                        }
                  }
            }

            // Беремо зображення з камери у спеціальну позицію для
мультиплексера
            public void SetCamera(int row, int col, Camera camera)
            {
                  if ((row >= 0) && (col >= 0) && (row < MaxRows) && (col <
MaxCols))
                  {
                      camWindows[row, col].Camera = camera;
                  }
            }

            // Встановлюємо розмір  мультиплексера
            public void SetSize(int rows, int cols, int cellWidth, int
cellHeight)
            {
                  this.rows = rows;
                  this.cols = cols;
                  this.cellWidth = cellWidth;
                  this.cellHeight = cellHeight;
                  UpdateSize();
            }

            // Оновлюємо зображення камери
            private void UpdateVisiblity()
            {
                  if (camerasVisible)
                  {
                        for (int i = 0; i < MaxRows; i++)
                        {
                              for (int j = 0; j < MaxCols; j++)
                              {
                                    camWindows[i, j].Visible = ((i < rows) && (j
< cols));
                              }
                        }
                  }
```

```
            }

            // Оновлюємо розмір та місце зображення камери
            private void UpdateSize()
            {
                  int width, height;

                  if (!fitToWindow)
                  {
                        // стандартні ширина та висота
                        width = cellWidth;
                        height = cellHeight;
                  }
                  else
                  {
                        // розраховуємо ширину та висоту камери для відображення
у вікні
                        width = (ClientRectangle.Width / cols) - 4;
                        height = (ClientRectangle.Height / rows) - 4;
                  }

                  // установлюємо позицію перегляду
                  int startX = (ClientRectangle.Width - cols * (width + 4)) / 2;
                  int startY = (ClientRectangle.Height - rows * (height + 4)) /
2;

                  this.SuspendLayout();

                  for (int i = 0; i < rows; i++)
                  {
                        for (int j = 0; j < cols; j++)
                        {
                              camWindows[i, j].Location = new Point(startX +
(width + 4) * j + 1, startY + (height + 4) * i + 1);
                              camWindows[i, j].Size = new Size(width + 2, height
+ 2);
                        }
                  }

                  this.ResumeLayout(false);
            }

            // Змінюємо розмір
            private void Multiplexer_Resize(object sender, System.EventArgs e)
            {
                  UpdateSize();
            }

            // Миша виключає працю з камерою
            private void cameraWindow_MouseDown(object sender,
System.Windows.Forms.MouseEventArgs e)
            {
                  lastClicked = (CameraWindow) sender;
            }
      }
}
```

Файл VideoBYOCStream.cs – робота з відео

```
namespace stream
{
	using System;
	using System.Drawing;
	using System.Drawing.Imaging;
	using System.IO;
	using System.Threading;
	using System.Runtime.InteropServices;
	using System.Net;

	using VideoBYOCsource;
	using dshow;
	using dshow.Core;

	/// <summary>
	/// VideoBYOCStream – поток подачи відео
	/// </summary>
	public class VideoBYOCStream : IVideoBYOCSource
	{
		private string    source;
		private object    userData = null;
		private int       framesReceived;

		private Thread    thread = null;
		private ManualResetEvent stopEvent = null;

		// нова подія у фреймі
		public event CameraEventHandler NewFrame;

		// VideoBYOCSource властивості
		public virtual string VideoBYOCSource
		{
			get { return source; }
			set { source = value; }
		}
		// Властивості підключення
		public string Login
		{
			get { return null; }
			set { }
		}
		// Властивості паролювання
		public string Password
		{
			get { return null; }
			set {  }
		}
		// FramesReceived властивості
		public int FramesReceived
		{
			get
			{
				int frames = framesReceived;
				framesReceived = 0;
				return frames;
			}
		}
		// BytesReceived властивості
		public int BytesReceived
		{
			get { return 0; }
		}
		// UserData властивості
		public object UserData
		{
```

```
        get { return userData; }
        set { userData = value; }
    }
    // Отримуємо стан вихідного відео
    public bool Running
    {
        get
        {
            if (thread != null)
            {
                if (thread.Join(0) == false)
                    return true;

                // Якщо стан не заданий, звільнюємо ресурси
                Free();
            }
            return false;
        }
    }


    // Конструктор
    public VideoBYOCStream()
    {
    }

    // Починаємо роботу
    public void Start()
    {
        if (thread == null)
        {
            framesReceived = 0;

            // Створюємо подію
            stopEvent   = new ManualResetEvent(false);

            // Створюємо й стартуємо нову подію
            thread = new Thread(new ThreadStart(WorkerThread));
            thread.Name = source;
            thread.Start();
        }
    }

    // Сигнал події до остановки роботи
    public void SignalToStop()
    {
        // Остановлюємо подію
        if (thread != null)
        {
            // Сигнал остановки
            stopEvent.Set();
        }
    }

    // Чекаємо остановки події
    public void WaitForStop()
    {
        if (thread != null)
        {
            // Чекаємо остановки події
            thread.Join();

            Free();
        }
    }

    // Подія помилки
    public void Stop()
    {
```

```
            if (this.Running)
            {
                  thread.Abort();
                  // WaitForStop();
            }
      }

      // Визволяємо ресурси
      private void Free()
      {
            thread = null;

            // Випуск події
            stopEvent.Close();
            stopEvent = null;
      }

      // Точка входу події
      public void WorkerThread()
      {
            bool  failed = false;

            // Граббер
            Grabber     grabber = new Grabber(this);

            // Об'єкти
            object      graphObj = null;
            object      sourceObj = null;
            object      grabberObj = null;

            // Інтерфейси
            IGraphBuilder     graph = null;
            IBaseFilter       sourceBase = null;
            IBaseFilter       grabberBase = null;
            ISampleGrabber    sg = null;
            IFileSourceFilter fileSource = null;
            IMediaControl     mc = null;
            IMediaEventEx     mediaEvent = null;

            int   code, param1, param2;

            while ((!failed) && (!stopEvent.WaitOne(0, true)))
            {
                  try
                  {
                        // Встановлюємо тип фільтру графіки
                        Type srvType =
Type.GetTypeFromCLSID(Clsid.FilterGraph);
                        if (srvType == null)
                              throw new ApplicationException("Failed
creating filter graph");

                        // Створюємо фільтр графіки
                        graphObj = Activator.CreateInstance(srvType);
                        graph = (IGraphBuilder) graphObj;

                        // Беремо тип фільтру вікна програвання джерела
                        srvType =
Type.GetTypeFromCLSID(Clsid.WindowsMediaSource);
                        if (srvType == null)
                              throw new ApplicationException("Failed
creating WM source");

                        // Створюємо вікно програвання джерела
                        sourceObj = Activator.CreateInstance(srvType);
                        sourceBase = (IBaseFilter) sourceObj;

                        // Беремо тип простого грабера
```

```
                              srvType =
Type.GetTypeFromCLSID(Clsid.SampleGrabber);
                              if (srvType == null)
                                    throw new ApplicationException("Помилка
створення  простого грабера");

                              // Створення  простого грабера
                              grabberObj = Activator.CreateInstance(srvType);
                              sg = (ISampleGrabber) grabberObj;
                              grabberBase = (IBaseFilter) grabberObj;

                              // Додаємо фільтр графічного джерела
                              graph.AddFilter(sourceBase, "source");
                              graph.AddFilter(grabberBase, "grabber");

                              // Визначаємо тип медіа
                              AMMediaType mt = new AMMediaType();
                              mt.majorType = MediaType.VideoBYOC;
                              mt.subType = MediaSubType.RGB24;
                              sg.SetMediaType(mt);

                              // Редактуємо файл
                              fileSource = (IFileSourceFilter) sourceObj;
                              fileSource.Load(this.source, null);

                              // Підключаємо піни
                              if (graph.Connect(DSTools.GetOutPin(sourceBase,
0), DSTools.GetInPin(grabberBase, 0)) < 0)
                                    throw new ApplicationException("Failed
connecting filters");

                              // Беремо тип медіа
                              if (sg.GetConnectedMediaType(mt) == 0)
                              {
                                    VideoBYOCInfoHeader vih =
(VideoBYOCInfoHeader) Marshal.PtrToStructure(mt.formatPtr,
typeof(VideoBYOCInfoHeader));

                                    grabber.Width = vih.BmiHeader.Width;
                                    grabber.Height = vih.BmiHeader.Height;
                                    mt.Dispose();
                              }

                              // рендер
                              graph.Render(DSTools.GetOutPin(grabberBase, 0));

                              //
                              sg.SetBufferSamples(false);
                              sg.SetOneShot(false);
                              sg.SetCallback(grabber, 1);

                              // вікно
                              IVideoBYOCWindow win = (IVideoBYOCWindow)
graphObj;
                              win.put_AutoShow(false);
                              win = null;

                              // Беремо подію інтерфейсу
                              mediaEvent = (IMediaEventEx) graphObj;

                              // Беремо управління медіа
                              mc = (IMediaControl) graphObj;

                              // запускаємо
                              mc.Run();

                              while (!stopEvent.WaitOne(0, true))
                              {
                                    Thread.Sleep(100);
```

```
                                    // беремо подію
                                    if (mediaEvent.GetEvent(out code, out
param1, out param2, 0) == 0)
                                    {
                                          // визначаємо параметри
                                          mediaEvent.FreeEventParams(code,
param1, param2);

                                          //
                                          if (code == (int) EventCode.Complete)
                                          {
                                                break;
                                          }
                                    }
                              }

                              mc.StopWhenReady();
                        }
                              // Визначаємо виключення
                        catch (Exception e)
                        {
                              System.Diagnostics.Debug.WriteLine("----: " +
e.Message);
                              failed = true;
                        }
                              // Фіналізуємо блок
                        finally
                        {
                              // визначаємо усі об'єкти
                              mediaEvent = null;
                              mc = null;
                              fileSource = null;
                              graph = null;
                              sourceBase = null;
                              grabberBase = null;
                              sg = null;

                              if (graphObj != null)
                              {
                                    Marshal.ReleaseComObject(graphObj);
                                    graphObj = null;
                              }
                              if (sourceObj != null)
                              {
                                    Marshal.ReleaseComObject(sourceObj);
                                    sourceObj = null;
                              }
                              if (grabberObj != null)
                              {
                                    Marshal.ReleaseComObject(grabberObj);
                                    grabberObj = null;
                              }
                        }
                  }
            }

            // новий фрейм для обробки
            protected void OnNewFrame(Bitmap image)
            {
                  framesReceived++;
                  if (NewFrame != null)
                        NewFrame(this, new CameraEventArgs(image));
            }

            // Граббер
            private class Grabber : ISampleGrabberCB
            {
                  private VideoBYOCStream parent;
```

```
                private int width, height;

                // Width властивості
                public int Width
                {
                      get { return width; }
                      set { width = value; }
                }
                // Height властивості
                public int Height
                {
                      get { return height; }
                      set { height = value; }
                }

                // Конструктор
                public Grabber(VideoBYOCStream parent)
                {
                      this.parent = parent;
                }

                //
                public int SampleCB(double SampleTime, IntPtr pSample)
                {
                      return 0;
                }

                // Повертаємо метод, який вказує на буфер взірця
                public int BufferCB(double SampleTime, IntPtr pBuffer, int
BufferLen)
                {
                      // створюємо нову картинку
                      System.Drawing.Bitmap img = new Bitmap(width, height,
PixelFormat.Format24bppRgb);

                      // блокуємо дані бітової площини
                      BitmapData bmData = img.LockBits(
                            new Rectangle(0, 0, width, height),
                            ImageLockMode.ReadWrite,
                            PixelFormat.Format24bppRgb);

                      // копіюємо дані зображення
                      int srcStride = bmData.Stride;
                      int dstStride = bmData.Stride;

                      int dst = bmData.Scan0.ToInt32() + dstStride * (height -
1);
                      int src = pBuffer.ToInt32();

                      for (int y = 0; y < height; y++)
                      {
                            Win32.memcpy(dst, src, srcStride);
                            dst -= dstStride;
                            src += srcStride;
                      }

                      // розблокуємо дані бітової площини
                      img.UnlockBits(bmData);

                      // Увідомляємо батьків
                      parent.OnNewFrame(img);

                      // Будуємо картинку
                      img.Dispose();

                      return 0;
                }
          }
     }
```

```
}
```

Файл VideoBYOCStreamSetupPage.cs – робота з відео (інтерфейс)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using VideoBYOCsource;

namespace stream
{
	/// <summary>
	/// Основні дескриптори для VideoBYOCStreamSetupPage.
	/// </summary>
	public class VideoBYOCStreamSetupPage : System.Windows.Forms.UserControl,
IVideoBYOCSourcePage
	{
		private bool completed = false;
		private System.Windows.Forms.TextBox urlBox;
		private System.Windows.Forms.Label label1;
		/// <summary>
		/// Опис змінних розробника.
		/// </summary>
		private System.ComponentModel.Container components = null;

		// стан змінюваної події
		public event EventHandler StateChanged;

		// Конструктор
		public VideoBYOCStreamSetupPage()
		{
			// Цей виклик використовується у Windows.Forms Form Designer.
			InitializeComponent();

		}

		/// <summary>
		/// Очищуємо усі ресурси використовувані користувачем.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Component Designer generated code
		/// <summary>
		/// Необхідний метод для підтримки розробника - не модифікується
		/// контент цього методу з редактором коду.
		/// </summary>
		private void InitializeComponent()
		{
			this.urlBox = new System.Windows.Forms.TextBox();
			this.label1 = new System.Windows.Forms.Label();
			this.SuspendLayout();
			//
			// urlBox
			//
			this.urlBox.Anchor = ((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
```

```
                        | System.Windows.Forms.AnchorStyles.Right);
                this.urlBox.Location = new System.Drawing.Point(50, 10);
                this.urlBox.Name = "urlBox";
                this.urlBox.Size = new System.Drawing.Size(240, 20);
                this.urlBox.TabIndex = 1;
                this.urlBox.Text = "";
                this.urlBox.TextChanged += new
System.EventHandler(this.urlBox_TextChanged);
                //
                // label1
                //
                this.label1.Location = new System.Drawing.Point(10, 13);
                this.label1.Name = "label1";
                this.label1.Size = new System.Drawing.Size(30, 14);
                this.label1.TabIndex = 0;
                this.label1.Text = "&URL:";
                //
                // VideoBYOCStreamSetupPage
                //
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                                this.urlBox,

                                this.label1});
                this.Name = "VideoBYOCStreamSetupPage";
                this.Size = new System.Drawing.Size(300, 150);
                this.ResumeLayout(false);

        }
        #endregion

        // Completed властивості
        public bool Completed
        {
                get { return completed; }
        }

        // Показуємо сторінку
        public void Display()
        {
                urlBox.Focus();
                urlBox.SelectionStart = urlBox.TextLength;
        }

        // Додаємо сторінку
        public bool Apply()
        {
                return true;
        }

        // Конфігуруємо об’єкт зображення
        public object GetConfiguration()
        {
                StreamConfiguration config = new StreamConfiguration();

                config.source = urlBox.Text;

                return (object) config;
        }

        // Встановлюємо конфігурацію
        public void SetConfiguration(object config)
        {
                StreamConfiguration cfg = (StreamConfiguration) config;

                if (cfg != null)
                {
                        urlBox.Text = cfg.source;
                }
```

```
            }

            // Змінюємо URL
            private void urlBox_TextChanged(object sender, System.EventArgs e)
            {
                  completed = (urlBox.TextLength != 0);

                  if (StateChanged != null)
                        StateChanged(this, new EventArgs());
            }
      }
}
```

Файл SPIHTSource.cs – алгоритм кодування SPIHT

```
namespace SPIHT
{
	using System;
	using System.Drawing;
	using System.IO;
	using System.Threading;
	using System.Net;

	using VideoBYOCsource;

	/// <summary>
	/// SPIHTSource - SPIHT скачувач
	/// </summary>
	public class SPIHTSource : IVideoBYOCSource
	{
		private string    source;
		private string    login = null;
		private string    password = null;
		private object    userData = null;
		private int       framesReceived;
		private int       bytesReceived;
		private bool      useSeparateConnectionGroup = false;
		private bool      preventCaching = false;
		private int       frameInterval = 0;          // інтервал подання
фреймів у мілісекундах

		private const int bufSize = 512 * 1024;   // розмір буферу
		private const int readSize = 1024;        // розмір блоку для читання

		private Thread    thread = null;
		private ManualResetEvent stopEvent = null;

		// нова подія у фреймі
		public event CameraEventHandler NewFrame;

		// SeparateConnectioGroup властивості
		// indicates to open WebRequest in separate connection group
		public bool SeparateConnectionGroup
		{
			get { return useSeparateConnectionGroup; }
			set { useSeparateConnectionGroup = value; }
		}
		// PreventCaching властивості
		// Якщо властивості є правильними, то керуємо параметриу URL. Цей
клієнт повинен бути встановлений на проксі-сервері.
		public bool PreventCaching
		{
			get { return preventCaching; }
			set { preventCaching = value; }
		}
		// FrameInterval властивості - інтервал між фреймами
		Якщо властивості встановлені в 100, тоді джерело формує 10 фреймів
		// в секунду
		public int FrameInterval
		{
			get { return frameInterval; }
			set { frameInterval = value; }
		}
		// VideoBYOCSource властивості
		public virtual string VideoBYOCSource
		{
			get { return source; }
			set { source = value; }
		}
		// Властивості підключення
```

```
public string Login
{
      get { return login; }
      set { login = value; }
}
// Властивості паролювання
public string Password
{
      get { return password; }
      set { password = value; }
}
// FramesReceived властивості
public int FramesReceived
{
      get
      {
            int frames = framesReceived;
            framesReceived = 0;
            return frames;
      }
}
// BytesReceived властивості
public int BytesReceived
{
      get
      {
            int bytes = bytesReceived;
            bytesReceived = 0;
            return bytes;
      }
}
// UserData властивості
public object UserData
{
      get { return userData; }
      set { userData = value; }
}
// Отримуємо стан вихідного відео
public bool Running
{
      get
      {
            if (thread != null)
            {
                  if (thread.Join(0) == false)
                        return true;

                  // Якщо стан не заданий, звільнюємо ресурси
                  Free();
            }
            return false;
      }
}

// Конструктор
public SPIHTSource()
{
}

// Починаємо роботу
public void Start()
{
      if (thread == null)
      {
            framesReceived = 0;
            bytesReceived = 0;

            // Створюємо подію
            stopEvent   = new ManualResetEvent(false);
```

```
                        // Створюємо й стартуємо нову подію
                        thread = new Thread(new ThreadStart(WorkerThread));
                        thread.Name = source;
                        thread.Start();
                  }
            }

            // Сигнал події до остановки роботи
            public void SignalToStop()
            {
                  // Остановлюємо подію
                  if (thread != null)
                  {
                        // Сигнал остановки
                        stopEvent.Set();
                  }
            }

            // Чекаємо остановки події
            public void WaitForStop()
            {
                  if (thread != null)
                  {
                        // Чекаємо остановки події
                        thread.Join();

                        Free();
                  }
            }

            // Подія помилки
            public void Stop()
            {
                  if (this.Running)
                  {
                        thread.Abort();
                        WaitForStop();
                  }
            }

            // Визволяємо ресурси
            private void Free()
            {
                  thread = null;

                  // Випуск події
                  stopEvent.Close();
                  stopEvent = null;
            }

            // Точка входу події
            public void WorkerThread()
            {
                  byte[]                  buffer = new byte[bufSize];   // буфер
читання потоку
                  HttpWebRequest    req = null;
                  WebResponse       resp = null;
                  Stream                  stream = null;
                  Random                  rnd = new Random((int)
DateTime.Now.Ticks);
                  DateTime          start;
                  TimeSpan          span;

                  while (true)
                  {
                        int   read, total = 0;

                        try
```

```
                        {
                              start = DateTime.Now;

                              // створюємо запрос
                              if (!preventCaching)
                              {
                                    req = (HttpWebRequest)
WebRequest.Create(source);
                              }
                              else
                              {
                                    req = (HttpWebRequest)
WebRequest.Create(source + ((source.IndexOf('?') == -1) ? '?' : '&') + "fake=" +
rnd.Next().ToString());
                              }
                              // встановлюємо логін та пароль
                              if ((login != null) && (password != null) &&
(login != ""))
                                    req.Credentials = new
NetworkCredential(login, password);
                              // встановлюємо найменування группи підключення
                              if (useSeparateConnectionGroup)
                                    req.ConnectionGroupName =
GetHashCode().ToString();
                              // отримуємо відповідь
                              resp = req.GetResponse();

                              // отримуємо відповідь потоку
                              stream = resp.GetResponseStream();

                              // цикл
                              while (!stopEvent.WaitOne(0, true))
                              {
                                    // перевіряємо загальне читання
                                    if (total > bufSize - readSize)
                                    {
                                          total = 0;
                                    }

                                    // Читаємо наступний блок у потоці
                                    if ((read = stream.Read(buffer, total,
readSize)) == 0)
                                          break;
                                    total += read;
                                    // Додаємо лічильник зчитаних байт
                                    bytesReceived += read;
                              }
                              if (!stopEvent.WaitOne(0, true))
                              {
                                    // додаємо лічильник фреймів
                                    framesReceived++;
                                    // остановка читання зображення
                                    if (NewFrame != null)
                                    {
                                          Bitmap      bmp = (Bitmap)
Bitmap.FromStream(new MemoryStream(buffer, 0, total));
                                          // Клієнт увідомлення
                                          NewFrame(this, new
CameraEventArgs(bmp));
                                          // Будуємо картинку
                                          bmp.Dispose();
                                          bmp = null;
                                    }
                              }
                              // Чекаємо в циклі ?
                              if (frameInterval > 0)
                              {
                                    // діапазон часу
                                    span = DateTime.Now.Subtract(start);
```

```
                                    // засипання у мілісекундах
                                    int msec = frameInterval - (int)
span.TotalMilliseconds;
                                    while ((msec > 0) && (stopEvent.WaitOne(0,
true) == false))
                                    {
                                          // засипання ...
                                          Thread.Sleep((msec < 100) ? msec :
100);
                                          msec -= 100;
                                    }
                              }
                        }
                        catch (WebException ex)
                        {
                              System.Diagnostics.Debug.WriteLine("=============:
" + ex.Message);
                              // чекаємо у циклі наступну спробу
                              Thread.Sleep(250);
                        }
                        catch (Exception ex)
                        {
                              System.Diagnostics.Debug.WriteLine("=============:
" + ex.Message);
                        }
                        finally
                        {
                              // помилка запиту
                              if (req != null)
                              {
                                    req.Abort();
                                    req = null;
                              }
                              // закриваємо потоку
                              if (stream != null)
                              {
                                    stream.Close();
                                    stream = null;
                              }
                              // close response
                              if (resp != null)
                              {
                                    resp.Close();
                                    resp = null;
                              }
                        }
                        // need to stop ?
                        if (stopEvent.WaitOne(0, true))
                              break;
                  }
            }
      }
}
```

Файл SPIHTSourcePage.cs – алгоритм кодування SPIHT **(інтерфейс)**

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using VideoBYOCsource;

namespace SPIHT
{
	/// <summary>
	/// Основні дескриптори для SPIHTSourcePage.
	/// </summary>
	public class SPIHTSourcePage : System.Windows.Forms.UserControl,
IVideoBYOCSourcePage
	{
		private static int[] frameIntervals = new int[] {0, 100, 142, 200,
333, 1000,
                                    5000, 10000, 15000, 20000, 30000, 60000};
		private bool completed = false;
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.TextBox urlBox;
		private System.Windows.Forms.Label label2;
		private System.Windows.Forms.TextBox loginBox;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.TextBox passwordBox;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.ComboBox rateCombo;
		/// <summary>
		/// Опис змінних розробника.
		/// </summary>
		private System.ComponentModel.Container components = null;

		// стан змінюваної події
		public event EventHandler StateChanged;

		// Конструктор
		public SPIHTSourcePage()
		{
			// Цей виклик використовується у Windows.Forms Form Designer.
			InitializeComponent();

			//
			rateCombo.SelectedIndex = 0;
		}

		/// <summary>
		/// Очищуємо усі ресурси використовувані користувачем.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Component Designer generated code
		/// <summary>
		/// Необхідний метод для підтримки розробника - не модифікується
		/// контент цього методу з редактором коду.
```

```
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.urlBox = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.loginBox = new System.Windows.Forms.TextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.passwordBox = new System.Windows.Forms.TextBox();
            this.label4 = new System.Windows.Forms.Label();
            this.rateCombo = new System.Windows.Forms.ComboBox();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(10, 13);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(41, 14);
            this.label1.TabIndex = 0;
            this.label1.Text = "&URL:";
            //
            // urlBox
            //
            this.urlBox.Anchor = ((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right);
            this.urlBox.Location = new System.Drawing.Point(70, 10);
            this.urlBox.Name = "urlBox";
            this.urlBox.Size = new System.Drawing.Size(220, 20);
            this.urlBox.TabIndex = 1;
            this.urlBox.Text = "";
            this.urlBox.TextChanged += new
System.EventHandler(this.urlBox_TextChanged);
            //
            // label2
            //
            this.label2.Location = new System.Drawing.Point(10, 43);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(35, 14);
            this.label2.TabIndex = 2;
            this.label2.Text = "&Login:";
            //
            // loginBox
            //
            this.loginBox.Anchor = ((System.Windows.Forms.AnchorStyles.Top
| System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right);
            this.loginBox.Location = new System.Drawing.Point(70, 40);
            this.loginBox.Name = "loginBox";
            this.loginBox.Size = new System.Drawing.Size(220, 20);
            this.loginBox.TabIndex = 3;
            this.loginBox.Text = "";
            //
            // label3
            //
            this.label3.Location = new System.Drawing.Point(10, 73);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(60, 14);
            this.label3.TabIndex = 4;
            this.label3.Text = "&Password:";
            //
            // passwordBox
            //
            this.passwordBox.Anchor =
((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right);
            this.passwordBox.Location = new System.Drawing.Point(70, 70);
            this.passwordBox.Name = "passwordBox";
```

```
                this.passwordBox.Size = new System.Drawing.Size(220, 20);
                this.passwordBox.TabIndex = 5;
                this.passwordBox.Text = "";
                //
                // label4
                //
                this.label4.Location = new System.Drawing.Point(10, 103);
                this.label4.Name = "label4";
                this.label4.Size = new System.Drawing.Size(63, 14);
                this.label4.TabIndex = 6;
                this.label4.Text = "&Frame rate:";
                //
                // rateCombo
                //
                this.rateCombo.Anchor =
((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                      | System.Windows.Forms.AnchorStyles.Right);
                this.rateCombo.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
                this.rateCombo.Items.AddRange(new object[] {

        "Uncontrolled",

        "10 фреймів у секунду",

        "7 фреймів у секунду",

        "5 фреймів у секунду",

        "3 фреймів у секунду",

        "1 фреймів у секунду",

        "12 фреймів у хвилину",

        "6 фреймів у хвилину",

        "4 фреймів у хвилину",

        "3 фреймів у хвилину",

        "2 фреймів у хвилину",

        "1 фреймів у хвилину"});
                this.rateCombo.Location = new System.Drawing.Point(70, 100);
                this.rateCombo.Name = "rateCombo";
                this.rateCombo.Size = new System.Drawing.Size(220, 21);
                this.rateCombo.TabIndex = 7;
                //
                // SPIHTSourcePage
                //
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                            this.rateCombo,

                            this.label4,

                            this.passwordBox,

                            this.label3,

                            this.loginBox,

                            this.label2,

                            this.urlBox,

                            this.label1});
```

```
                this.Name = "SPIHTSourcePage";
                this.Size = new System.Drawing.Size(300, 150);
                this.ResumeLayout(false);

            }
            #endregion

            // Completed властивості
            public bool Completed
            {
                get { return completed; }
            }

            // Показуємо сторінку
            public void Display()
            {
                urlBox.Focus();
                urlBox.SelectionStart = urlBox.TextLength;
            }

            // Додаємо сторінку
            public bool Apply()
            {
                return true;
            }

            // Конфігуруємо об’єкт зображення
            public object GetConfiguration()
            {
                SPIHTConfiguration      config = new SPIHTConfiguration();

                config.source     = urlBox.Text;
                config.login      = loginBox.Text;
                config.password   = passwordBox.Text;
                config.frameInterval =
frameIntervals[rateCombo.SelectedIndex];

                return (object) config;
            }

            // Встановлюємо конфігурацію
            public void SetConfiguration(object config)
            {
                SPIHTConfiguration      cfg = (SPIHTConfiguration) config;

                if (cfg != null)
                {
                    urlBox.Text = cfg.source;
                    loginBox.Text = cfg.login;
                    passwordBox.Text = cfg.password;
                    rateCombo.SelectedIndex = Array.IndexOf(frameIntervals,
cfg.frameInterval);
                }
            }

            // Змінюємо URL
            private void urlBox_TextChanged(object sender, System.EventArgs e)
            {
                completed = (urlBox.TextLength != 0);

                if (StateChanged != null)
                    StateChanged(this, new EventArgs());
            }
        }
}
```

Файл About.cs - довідка

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace CameraAbout {
      public class About : System.Windows.Forms.Form {

            #region system stuff
            private System.Windows.Forms.PictureBox pictureBox1;
            private System.Windows.Forms.RichTextBox richTextBox1;
            private System.Windows.Forms.Button button1;
            private System.ComponentModel.Container components = null;

            public About() {
                  InitializeComponent();
            }

            protected override void Dispose( bool disposing ) {
                  if( disposing ) {
                        if(components != null) {
                              components.Dispose();
                        }
                  }
                  base.Dispose( disposing );
            }
            #endregion

            #region Windows Form Designer generated code
            /// <summary>
            /// Необхідний метод для підтримки розробника - не модифікується
            /// контент цього методу з редактором коду.
            /// </summary>
            private void InitializeComponent() {
                  System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(About));
                  this.pictureBox1 = new System.Windows.Forms.PictureBox();
                  this.richTextBox1 = new System.Windows.Forms.RichTextBox();
                  this.button1 = new System.Windows.Forms.Button();
                  this.SuspendLayout();
                  //
                  // pictureBox1
                  //
                  this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
                  this.pictureBox1.Location = new System.Drawing.Point(8, 8);
                  this.pictureBox1.Name = "pictureBox1";
                  this.pictureBox1.Size = new System.Drawing.Size(152, 192);
                  this.pictureBox1.TabIndex = 0;
                  this.pictureBox1.TabStop = false;
                  //
                  // richTextBox1
                  //
                  this.richTextBox1.BackColor = System.Drawing.Color.Black;
                  this.richTextBox1.BorderStyle =
System.Windows.Forms.BorderStyle.None;
                  this.richTextBox1.ForeColor = System.Drawing.Color.White;
                  this.richTextBox1.Location = new System.Drawing.Point(176,
16);
                  this.richTextBox1.Name = "richTextBox1";
                  this.richTextBox1.Size = new System.Drawing.Size(304, 184);
                  this.richTextBox1.TabIndex = 1;
                  this.richTextBox1.Text = @"БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА
```

на тему:

Програмне забезпечення відеоконференцій за допомогою технології BYOC

Керівник: Якименко Н.М.

Розробив: студент Грабовський Максим Дмитрович
гр. КІ-22-мб

м. Кропивницький 2025";

```
                    //
                    // button1
                    //
                    this.button1.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
                    this.button1.ForeColor = System.Drawing.Color.White;
                    this.button1.Location = new System.Drawing.Point(16, 168);
                    this.button1.Name = "button1";
                    this.button1.Size = new System.Drawing.Size(136, 24);
                    this.button1.TabIndex = 2;
                    this.button1.Text = "&Close";
                    this.button1.Click += new
System.EventHandler(this.button1_Click);
                    //
                    // About
                    //
                    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
                    this.BackColor = System.Drawing.Color.Black;
                    this.ClientSize = new System.Drawing.Size(480, 208);
                    this.Controls.Add(this.button1);
                    this.Controls.Add(this.richTextBox1);
                    this.Controls.Add(this.pictureBox1);
                    this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.None;
                    this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
                    this.Name = "About";
                    this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "About";
                    this.ResumeLayout(false);

            }
            #endregion

            #region events

            private void button1_Click(object sender, System.EventArgs e) {
                  this.Close();
            }

            #endregion
      }
}
```