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 широкомовного HD-відео з застосуванням технології MBMS”

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

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 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

Висоцькому Володимиру Геннадійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи широкомовного HD-відео з застосуванням технології MBMS*

2. Керівник роботи *Петренюк Володимир Ілліч, 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 широкомовного HD-відео з застосуванням технології MBMS*

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
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Висоцький В.Г.
(прізвище та ініціали)

АНОТАЦІЯ

Висоцький В.Г. Програмне забезпечення системи широкомовного HD-відео з застосуванням технології MBMS. 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 широкомовного HD-відео з застосуванням технології MBMS.

Метою розробки є програмне забезпечення системи широкомовного HD-відео з застосуванням технології MBMS.

Результат роботи – програмна реалізація системи широкомовного HD-відео з застосуванням технології MBMS.

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 Python.

Ключові слова: комп'ютерна інженерія, MBMS

ABSTRACT

Vysotsky V.G. Software for a broadcast HD video system using MBMS 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 broadcast HD video system using MBMS technology.

The purpose of the development is to develop software for a broadcast HD video system using MBMS technology.

The result of the work is a software implementation of a broadcast HD video system using MBMS 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 OS Windows 10/11.

The program was developed in the Python environment.

Keywords: computer engineering, MBMS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

СВІ	–	системи виведення інформації
ТВЧ	–	Телебачення Високої Чіткості
AHD	–	Analog High Definition
AVC	—	Advanced Video Coding (прогресивне кодування відео)
HD	—	High Definition Video
HD-CVI	—	High Definition Coaxial Video Interface
HD-TVI		High Definition Transport Video Interface
LTE		Long-Term Evolution, стандарт зв'язку четвертого покоління
MBMS		Multimedia Broadcast Multicast Service

ВСТУП

Актуальність теми. MBMS (Multimedia Broadcast Multicast Service) – це послуга, яку пропонують стільникові мережі 3GPP (3rd Generation Partnership Project) для розповсюдження мультимедійного вмісту та конкурує з такими технологіями, як DVB-H (Digital Video Broadcasting Handhelds) і DMB (Digital Multimedia Broadcasting) Мобільні відеокодери з можливостями стільникової передачі часто називають мобільними передавачами, польовими пристроями, відеопередавачами або навіть зв'язаними стільниковими кодерами. Це пристрій, який забезпечує трансляцію відео через об'єднані стільникові мережі (3G, 4G, 5G). Вони оснащені декількома стільниковими модемами, які можна використовувати для об'єднання кількох стільникових мережевих з'єднань для збільшення пропускної здатності та надійності. Використовується в складних або віддалених середовищах, коли підключення до IP-мережі або невідоме, або непередбачуване, або просто непрактично, мобільний передавач відео використовує стільникові модеми для кодування та передачі відео якості трансляції з низькою затримкою.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи широкомовного HD-відео з застосуванням технології MBMS.

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

– Огляд існуючих систем широкомовного HD-відео з застосуванням технології MBMS.

– Дослідження системи широкомовного HD-відео з застосуванням технології MBMS.

– Програмна реалізація системи широкомовного HD-відео з застосуванням технології MBMS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі широкомовного HD-відео з застосуванням технології MBMS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи широкомовного HD-відео з застосуванням технології MBMS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для забезпечення широкомовного HD-відео. Переглянути HD-відео за його високої роздільної якості можна тільки на пристроях відтворення (Монітори, ТБ-панелі, проектори...), що підтримують розв’язувана здатність екрана не менш 1280x720 точок.

Основними носіями HD-контенту є диски збільшеної щільності даних, наприклад Blu-ray. Або переглядати через телеприймач, що приймає й відтворює телепередачі у ТВЧ-сигналі.

Для кодування й декодування відеопотоку потрібні значні обчислювальні потужності процесора. Щоб полегшити навантаження на процесор, іменитіі виробники відеокарт не дуже давно включили підтримку апаратного декодування HD-відео у свої нові моделі. Звичайно міткою Full-HD позначається підтримка телевізором повного дозволу 1920x1080 точок.

Міткою HD-Ready позначається підтримка телевізором дозволу менше 1920x1080 точок, наприклад якщо в телевізора розв’язувана здатність становить 1024x768 точок, то він зможе показати вхідний HD-сигнал, але при цьому перетворить (стисне) картинку до розмірів 1024x768, чим зменшить чіткість вхідного сигналу

1.2 Область застосування

Областю застосування є LTE-мережі. Стандарт LTE – Long-Term Evolution – або стандарт зв'язку четвертого покоління вважається перспективним напрямком розвитку мереж.

При використанні станцій LTE радіус покриття досягає від 5 км (оптимально) до 30 км або навіть 100 км (при необхідності). За спостереженнями аналітиків, впровадження стандарту LTE дозволить операторам знизити собівартість послуг передачі даних в 6 разів у порівнянні з 3G. У результаті вартість мобільного інтернету для кінцевих користувачів знизиться. Впровадження цих сервісів також дозволить операторам одержати джерело для додаткового заробітку. Однак при цьому можливо падіння прибутку від голосового зв'язку, що вільно буде здійснюватися через інтернет телефонію. Інакше кажучи, підвищується швидкість передачі даних і, відповідно, підвищується якість послуг, що у свою чергу, сприяє поширенню сучасних мультимедійних сервісів (соціальні мережі, он-лайн і мережні ігри, різні інтерактивні додатки, відеодзвінки, відеоконференції й ін.).

Українські аналітики відзначають, що коли саме стане можливим надавати доступ до зв'язку 4G такій кількості людей щорічно, поки не зрозуміло. Самі "проблемні" регіони для будівництва мережі 4G – це регіони з низькою купівельною спроможністю й з малою щільністю населення – таких регіонах в операторів можуть істотно затягтися строки окупності мереж через низький попит при високих витратах. Основна проблема на цьому тлі – низька економічна ефективності розвитку зв'язку, тому що будівництво мереж зажадає високих витрат, окупність яких при меншій потенційній чисельності абонентів буде значно вище. І головне – до настільки високих темпів нарощування бази користувачів LTE не готові ні українські стільникові оператори, що тільки недавно одержали відповідні частоти, ні виробники смартфонів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи широкомовного HD-відео з застосуванням технології MBMS, 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

HD-SDI/CVI/TVI/AHD: деякі особливості нових форматів

Базові параметри нового формату відповідають стандартним значенням 720p і 1080p. Даний формат припускає горизонтальний розв'язувана здатність в 800 і 1200 ТВЛ зі співвідношенням сторін 16:9 при прогресивному розкладанні. Спочатку здавалося, що формат буде монополізований патентовласником. Однак у наш час, коли "все вже винайдено", удержати що-небудь одному виробникові не представляється можливим. Технічні рішення створюються сьогодні шляхом компіляції інтегральних рішень, розроблених здебільшого великими світовими виробниками. Крім того, основні технічні рішення розроблені багато років тому. В остаточному підсумку перше HD-телебачення було все-таки аналоговим, а багато проблем того років нині успішно вирішуються завдяки сучасній елементній базі великої інтеграції з високими характеристиками.

Порівняння конкурентних HD-технологій

Самітність Dahua, розроблювача й тримача патентів, тривало недовго. Намітилися вже три конкуруючі компанії, не вважаючи великого творця елементної бази NextChip, що володіють аналогічними розробками. Формально пропонуються три технології з досить схожими характеристиками. Це HD-CVI (High Definition Coaxial Video Interface) від компанії Dahua, а також HD-TVI (High Definition Transport Video Interface), пропонована компаніями Hikvision і Hi Sharp. Компанія Hi Sharp одночасно робить камери типу AHD (Analog High Definition), які практично повністю ідентичні HD-TVI. Як видно, у них Hi Sharp використовує елементну базу компанії NextChip, що і є творцем технології й

самого терміна AHD. І швидше за все саме факт виробництва елементної бази дає підстави заявляти про наступність саме технології AHD для третіх сторін.

Незважаючи на те, що для всіх технологій заявлені обоє дозволу HD (720p і 1080p), реально камери FullHD виробляються поки тільки компаніями Dahua і Hikvision.

Кожний з розроблювачів пропонує порівняльний аналіз характеристик всіх форматів, застосовуваних у відеоспостереженні. У них, природно, новий формат демонструє істотні достоїнства, і вивчення розходжень і особливостей може доставити багато радості фахівцям. Наприклад, системи IP і SDI істотно програють із дозволу аналогового виходу, хоча порівнювати параметри можливих технологічних (допоміжних) виходів не зовсім правильно. Очевидна дешевина кабелю для CVI, аналогічного типовому аналоговому CCTV. Однак, як показує практика, для CVI кабель потрібно практично ідентичний системі SDI, особливо з урахуванням заявленої дальності. При цьому економія виходить тільки на відсутності репітерів при дальності більше 100 м, як в SDI. Головним порівняльним фактором у цей час є дальність передачі сигналу. Але стосовно до аналогового способу передачі говорити про дальність передачі не завжди коректно. Для цифрових сигналів визначальними є можливість синхронізації або число помилок, що обмежують швидкість при пакетній передачі, тому можна досить точно визначити дальність передачі для кожного виду кабелю. У випадку аналогової передачі якість зображення помітно погіршується при збереженні досить упевненої синхронізації, тому визначити граничну довжину кабелю досить складно, не з огляду на зміни передачі кольору й дозволу одержуваного зображення.

Черговий прапор, під яким іде "революція" відеоспостереження, – це передача якісного FullHD-Зображення аналоговим методом по звичайному коаксіальному кабелі на 500 м. Під цим "соусом" багато менеджерів повідомляють, що формат HD-SDI зі своїми 100 м – це щось застаріле й віджиле. Так, можна вже констатувати, що надлишкова якість HD-SDI-Зображення професійного класу практично не затребуваний ринком безпеки через високу

ціну встаткування. Якщо цифрова передача "піксель у піксель" незжатого зображення в телецентрах і між ними є неодмінною умовою підтримки загального стандарту якості, то у відеоспостереженні це виявилося надмірністю, що не покриває інші виникаючі складності.

Горизонтальний розв'язувана здатність можна оцінити в 1000-1050 ТВЛ для HD-SDI і 900-950 ТВЛ для HD-CVI. Камери підключені на коротких кабелях в 1 м. Необхідно враховувати, що результати є сукупними для камер і реєстраторів з кодеком стиску H.264 (невідомим) і переформатуванням з формату у формат при виводі. Можна вважати, що це системний результат.

На жаль, використовувана у випробуваннях відносно сучасна таблиця з наявністю вимірювальних елементів більше 600 ТВЛ мало того що має формат 4:3, так і приблизно на 20% завищує оцінку горизонтального дозволу. Це з'ясувалося при порівнянні геометрії фрагмента 600 ТВЛ із телевізійною таблицею 1972 року з Московського телецентру, де цей розв'язувана здатність максимальне. Тому отримані оцінки необхідно помножити на 1,33, щоб привести розв'язувана здатність до формату 16:9, і додатково помножити на 0,8, щоб скорегувати позитивну погрішність. Отриманий поправочний коефіцієнт 1,064 говорить про те, що в цьому випадку можна орієнтуватися на прямі дані.

Недоліки формату високої чіткості

Чи не так все безхмарно в новому форматі HD-CVI (він же TVI, він же AHD)? Ще на початку "важкого шляху" HD-SDI теж говорилося про звичайний коаксіальний кабель від старої системи відеоспостереження. Можливо, для США або Західної Європи, де цей кабель у найгіршому разі RG-59, а найчастіше RG-6, це справедливо. У Україні ж ці кабелі, як правило, КВК-3, КВК-2 або КВТ-2, але зустрічається й екранованe проведення ШГЕС-2 або 4, тому очікувати "чудес" і навіть простого функціонування не доводиться.

Необхідні розроблювачами кабелі 3С и 5С є близькими загальним типовим моделям RG-59 і RG-6. Можна сказати, що для української дійсності це вже не зовсім "будь-які РК-75", як заявляє виготовлювач. Причому це цілком

гідні варіанти й для системи HD-SDI зі смугою 1,5 Ггц і несучою частотою імпульсної послідовності 750 Мгц.

Спектр аналогового відеосигналу CVI (TVI або AHD) навіть для FullHD не перевищує 26 Мгц, що для коаксіального кабелю, нормованого звичайно до 1000 Мгц, не є чимсь нездійсненним. Разом з тим це перевищує по спектрі типовий телевізійний сигнал більш ніж в 4 рази й пред'являє до кабелів зв'язку трохи підвищені вимоги.

Як не дивно, чіткість зображення при кабелі в 500 м навіть трохи вище, ніж при довжині 1 м. Навіть для дешевого кабелю Rexant розв’язувана здатність не знижується до 800 ТВЛ і при дальності 200 м. Але зі збільшенням втрат на лінії сильно міняється передача кольору. Очевидно, що небажано збільшувати дальність більше 500 м навіть при гарному кабелі, а дешеві кабелі не слід застосовувати довше ніж 75-100 м.

Примітно, що керування камерою також порушується при довжині кабелю більше 500 м. Це технічний феномен, що характеризує повну закритість інформації про принципи передачі даних.

Строго говорячи, обіцянки забезпечувати розв’язувану здатність 1920x1080p і нормування як 1080p розв’язувану здатність камер і реєстраторів HD-CVI, а також аналогічних HD-TVI і AHD є принципово нездійсненними. У системі відбувається перехід на аналоговий сигнал з перетворенням яскравості в амплітуду й передачею кольоровості квадратурною модуляцією що піднесе, за принципом дії практично аналогічної стандарту PAL, з наступним перетворенням по суті композитного сигналу в цифрову форму для запису й виводу у форматі HDMI. Цим порушується принцип передачі "піксель у піксель" і змушує оцінювати розв’язувану здатність системи HD-CVI як в аналогових системах – тільки в телевізійних лініях. Крім того, додаткові перетворення "аналог – цифра – аналог – цифра" при некоректному їхньому виконанні можуть служити причиною артефактів і муару. Варто також насторожено ставитися до заяв розроблювачів про "високу захищеність" аналогової передачі HD-Зображення

амплітудною модуляцією по коаксіальному кабелі. Коаксіальний кабель мало захищений до 5-10 Мгц, а передача інформації зміною амплітуди є самим чутливим до перешкод методом передачі. Як видно, по чутливості до перешкод система HD-CVI повинна бути подібна типовий аналогової CCTV.

Резюмуючи вищесказане, можна зробити вивід про перспективність нового формату (CVI, TVI або AHD) для відеоспостереження, якщо, звичайно, ціна встаткування буде істотно нижче, ніж у систем HD-SDI. Але, на жаль, тільки збільшення дальності передачі зображення, що одночасно з переходом в аналог привело до деякого зниження його якості, не зробить новий формат або формати особливо привабливими.

Короткий огляд систем HD-SDI

Споконвічне призначення інтерфейсу HD-SDI – транспортування незжатого цифрового зображення із синхронним звуком у межах телевізійної студії. Технологія HD-SDI ґрунтується на стандарті SMPTE 292M. Даний стандарт описує цифрові послідовні інтерфейси й передбачає передачу цифрового незжатого відео по коаксіальному кабелі хвильовим опором 75Ом. Використовувана технологія в сучасних віщальних станціях стандарту HDTV, початку своє життя на ринку охоронного відеоспостереження 12 липня 2009р. Чому ж технологія HD-SDI не одержала стрімкого впровадження й не домоглася дотепер успіху? Причина – ціна. Мікросхеми, що формують зображення HD-Якості, донедавна залишалися неприйнятно дорогими.

Сьогодні вже стають доступні HD-SDI камери, які підтримують можливість передачі незжатого відеосигналу і які підключаються прямо до HD-SDI відеореєстраторів. Ці пристрої здатні транслювати зображення з розв’язною здатністю 1920x1080p/1280x720p без затримок на відстані 100-150м (без проміжного посилення), що вигідно їх відрізняє від IP-камер. Проте, говорити про витиснення із займаної ніші цифрових засобів відеоспостереження не доводиться, тому що при всьому бажанні HD-SDI камери поки не можуть дати дозволу більше, ніж в 3 мегапікселя. На українському ринку систем

відеоспостереження продукція цього виду поки представлена не так широко, як скажемо аналогові пристрої. Одним з першовідкривачів устаткування формату HD-SDI у нас у країні став бренд Partizan, що зараз працює над поповненням лінійки камер високого дозволу.

У чому ж основні переваги таких камер? Неможливо залишити без уваги високу світлочутливість матриць, на яких вони працюють. Завдяки додатковій обробці сигналу процесором DSP2 якість зображення в нічний час наближається до рівня професійного відеоустаткування. При включенні ІЧ-підсвічування, такі камери здатні гарантувати відмінну розпізнаваємість дрібних деталей навіть у повній темряві. Цим показником устаткування HD-SDI значно перевершує аналогові й IP-камери.

Високий рівень деталізації – ще одна відмітна риса пристроїв цього класу. Розв'язувана здатність в 3 мегапікселя дозволяє камері забезпечити така якість зображення, при якому можна без праці розглянути дрібні деталі. Функціональний набір HD-SDI устаткування надзвичайно різноманітний і в ньому є все необхідне для побудови надійної системи. Звичні можливості цифрових і аналогових камер тут втілюються не тільки силами матриці, але й процесором, що згадується вже, обробки відеосигналу. Завдяки цьому, розроблювачам удалося розкрити традиційні «фішки» у новому світлі, з небаченою раніше міццю й ефективністю. При надлишковому висвітленні, причиною якого можуть бути сонячні відблиски або світло автомобільних фар, функція маскування (HLC) зафарбовує занадто яскраві ділянки зображення, вирівнює загальну експозицію зображення й без праці дозволяє визначати номерні знаки або інші важливі деталі. Необхідно відзначити новаторський підхід HD-SDI камер до роботи зі світлом. Залежно від освітленості, устаткування цього типу може «витягати» із затемнених ділянок погано помітні фрагменти зображення, віддаючи всій картинці рівномірну насиченість світлом.

HD-SDI – це зовсім новий підхід до втілення мрії про безпеку. Він синтезував у собі все краще аналогових і IP технологій, зберігаючи при цьому

свою самобутність. Ринок відеоспостереження не стоїть на місці. Кожна нова пропозиція одних виробників народжує аргументована технічна відповідь їхніх конкурентів, змушуючи постійно обертатися колесо технічного прогресу. Іноді це є причиною появи зовсім нових продуктів, таких як HD-SDI камери відеоспостереження й HD-SDI відео реєстратори.

Мережні API якості за запитом покращують потокове відео

5G підштовхнув телекомунікаційну галузь до дедалі більшої адаптації своїх бізнес-функцій, від мережевих операцій до управління екосистемою партнерів. Deutsche Telekom Group очолює цю галузеву трансформацію, оскільки ми рухаємося до прискореної глобалізації та цифровізації всередині нашої галузі та за її межами. Майбутні зміни будуть масовими та принесуть значні виклики, перевернувши світ з ніг на голову. Нам потрібно переконатися, що наша галузь залишається на висоті, щоб гарантувати подальший успіх наших партнерів і клієнтів. Однією з наших зусиль є створення стійкої екосистеми 5G на арені інтерфейсу прикладного програмування (API), де ми були одним із перших операторів, які відкрили нашу мережу 5G для наших клієнтів через API та запропонували пряму взаємодію з нею.

Відкриття нашої мережі означає, що ми робимо дані доступними для наших клієнтів через визначені API, таким чином надаючи нову послугу даних і стаючи Telco як послуга. Ми тісно співпрацюємо з нашими клієнтами, щоб спільно тестувати всі API, отримуючи відповідні вимоги безпосередньо від них. З грудня 2022 року ми інтегрували п’ять API у нашу живу мережу: Quality-on-Demand (QoD), перевірку місцезнаходження, статус роумінгу, перевірку номера та заміну SIM-карти. Відтоді ми разом із нашими партнерами розробили кілька варіантів використання Proof-of-of-Concept (PoC), щоб випробувати наші API в режимі реального часу, зокрема сценарії автоматизованого паркування автомобіля працівником автомобіля та польоту безпілотника.

Випадок успішного використання Broadpeak

PoC був проведений з Broadpeak, чиє програмне забезпечення CDN ми розгорнули на Telekom Edge Cloud у Берліні. Використовуючи платформу програмованого підключення нашого партнера Microsoft Azure, Broadpeak скористався перевагами нашого QoD API, щоб покращити якість відео та надати кінцевим клієнтам більш стабільну послугу потокового передавання. Цей тип API оптимізує мережу, забезпечуючи найвищу якість, навіть якщо умови мережі є складними.

PoC було проведено в нашій тестовій лабораторії Hubraum 5G у Кракові, Польща, де ми протестували кілька сценаріїв потокового передавання на пристроях з обмеженою якістю мережі, яка миттєво покращилася після запуску QoD API. QoD API забезпечив стабільне з'єднання з низькою затримкою, оптимізувавши трафік до пристрою. Наші лабораторні тести показали надзвичайні результати щодо якості потокового передавання. Штучно підвищуючи навантаження на клітинку до межі, користувачі, які користуються перевагами QoD Network API, бачили плавні потокові сеанси з високою роздільною здатністю, тоді як усі інші відчували зависання відео та низьку якість зображення. Оптимальна трансляція сподобається користувачам, які дивляться події в прямому ефірі, наприклад футбольні чи тенісні матчі. Це також дуже важливо в таких сценаріях, як дистанційні медичні процедури або безпілотні літальні апарати. Подивіться наше відео нижче.

API Deutsche Telekom забезпечують мережеві інновації

Ми віримо, що мережеві API Deutsche Telekom можуть сприяти розвитку всіх сучасних сфер інновацій. Відкриття нашої мережі, а також комерційний запуск нашого API MagentaBusiness знаменують наш перехід до інноваційної ролі Telco as a Service, оскільки ми можемо пропонувати клієнтам нові дані по-новому. Стає можливим партнерство, і кожна галузь із IT-підтримкою може отримати вигоду, оскільки вони можуть використовувати інформацію, отриману

з мережі, для оптимізації своїх програм і, таким чином, створювати додаткову вартість для своїх кінцевих клієнтів.

Щоб переконатися, що API є легкодоступними та придатними для використання, ми створили унікальні тестові стенди в наших інноваційних центрах hubraum у Берліні, Кракові та разом із T-Mobile US у Сіетлі в рамках нашої програми раннього доступу 5G. У 2021 році підтримано створення Telco Global API Alliance CAMARA, що об'єднало великих операторів у всьому світі, а також Linux Foundation і GSMA. Метою CAMARA є забезпечення безперебійного доступу до мережевих можливостей 4G і 5G шляхом просування стандартизації відкритих і глобальних API.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

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 широкомовного HD-відео з застосуванням технології MBMS.

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

Під час трансляції з телевізійної студії за допомогою кабелів SDI або приватної IP-мережі камери можна безпосередньо підключити до обладнання для трансляції за допомогою входів SDI або Ethernet для SMPTE ST 2110 або NDI. Під час трансляції на відстані без приватного IP-з'єднання також може бути хорошим варіантом використання кодера реального часу, наприклад Makito X4, який підтримує SRT через Інтернет. Однак, якщо доступ до Інтернету обмежений або недоступний, або якщо камера має бути портативною, мобільний передавач відео є ідеальним вибором. Мобільні відеопередавачі пропонують мовникам більшу гнучкість, дозволяючи легко розширити їх пропускну здатність за допомогою стільникового зв'язку для:

– Спортивні трансляції в прямому ефірі – чим більший майданчик, тим важче всюди провести кабельну мережу. Незалежно від того, чи йдеться про катання на лижах, велоспорт або масштабні події, як-от TCS New York City Marathon, мобільні відеокодери дозволяють мовникам забезпечувати дистанційне охоплення кількома камерами для покращених прямих трансляцій і розширеного охоплення подій.

– Збір новин – завжди будьте готові до новин за допомогою портативних, економічно ефективних і надійних відеорішень, якими можна легко користуватися з будь-якої точки світу.

– Живі розваги – для музичних і культурних фестивалів, де недоступні мережеві ресурси.

У цих випадках передача мобільного відео через зв'язаний стільниковий зв'язок і мережу 5G пропонує переваги якості відео трансляції з субсекундною затримкою (усього 80 мс), спрощене виробництво за допомогою кількох камер,

підвищену гнучкість, спрощену логістику та можливість передавати відео з будь-якого місця.

Як мережі 5G змінюють гру для мовників

Зрозуміло, що мовники дуже зацікавлені у розкритті потенціалу 5G для покращення живого виробництва. 74% опитаних мовників уже використовують або планують використовувати технологію 5G для мовлення в найближчі два роки. Пропонуючи значно більшу ємність, меншу затримку та краще покриття, ніж 4G, 5G готова зробити широкосмуговий доступ до Інтернету повсюдним і розширити робочі процеси на основі IP для будь-якого типу живих подій.

У найближчі роки очікується значне зростання впровадження технології 5G для трансляції відео. Використання 5G замість фіксованих мережевих підключень оптимізує налаштування виробництва та зробить можливими нові захоплюючі способи запису відео з мобільних камер, особливо для трансляцій спортивних трансляцій.

Початковий етап розгортання 5G, також відомий як NSA (неавтономна архітектура), додає технологію мобільної передачі 5G до існуючих базових мереж 4G, щоб збільшити пропускну здатність до 50%. Другий етап SA (автономна архітектура) триватиме протягом наступних кількох років, який спирається на нову базову мережу 5G на основі гнучкої хмари та віртуалізованих технологій. Цей етап забезпечить ще більшу пропускну здатність, меншу затримку та низку інших інновацій, які принесуть користь мовникам.

Серія Pro від Haivision мобільних відеопередавачів, які монтуються на камеру, забезпечують низьку затримку в будь-якій мережі, використовуючи кодування відео в реальному часі, стільниковий зв’язок і 5G для передачі живого відео через мобільні мережі.

Призначені для збору новин і створення подій у прямому ефірі (наприклад, спортивні змагання, концерти та церемонії). Польові пристрої Haivision поєднують у собі бездоганне 4K/UHD-зйомку за допомогою кількох камер із критично важливою передачею з будь-якої точки світу.

Передавачі, що не менш важливо, прості у використанні, з вбудованим РК-сенсорним екраном, що означає, що користувачі можуть розпочати пряму трансляцію лише двома дотиками. Серія передавачів Haivision Pro забезпечує відмінну якість живого відео, низьку затримку, оптимальне енергоспоживання, вагу та розмір для портативних, автомобільних або фіксованих додатків. Відео в реальному часі через мобільні мережі підтримується та захищено запатентованою інтелектуальною технологією IP-з'єднання Safe Streams Transport (SST), яка отримала нагороду Emmy®. SST одночасно об'єднує кілька мережевих з'єднань, динамічно адаптує бітрейт відео відповідно до коливань пропускної здатності мережі, захищає потоковий вміст і підтримує повторну передачу втрачених даних.

Коли використовується фіксований IP-відеокодер?

Якщо мобільні відеокодери настільки гнучкі, коли і навіщо слід використовувати спеціальні IP-відеокодери? Відповідь полягає в тому, що якщо ви можете використовувати дротовий зв'язок, будьте дротовим. Коли доступні дротові мережі, вони майже завжди будуть швидшими, дешевшими, коли доступний Інтернет, і надійнішими. Однак, якщо вам потрібен мобільний відеокодер, найкращий вибір стільникового зв'язку та доступ до технології 5G – це ваш найкращий вибір для низької затримки, високоякісного та зручного відео в реальному часі.

3.2 Розробка структурної схеми

Система бездротової передачі відео з декількома надсиланнями і прийомом – це технологія бездротового зв'язку, яка надсилає кілька джерел відеоданих на один приймач. Ця технологія широко використовується у військовій, аерокосмічній галузі, роботах, безпілотних літальних апаратах та інших галузях, з високим рівнем безпеки та надійності в реальному часі. Ця стаття надасть детальний вступ до основного принципу, структури композиції та

сценаріїв застосування бездротової системи передачі відео з декількома надсиланнями та прийомом.

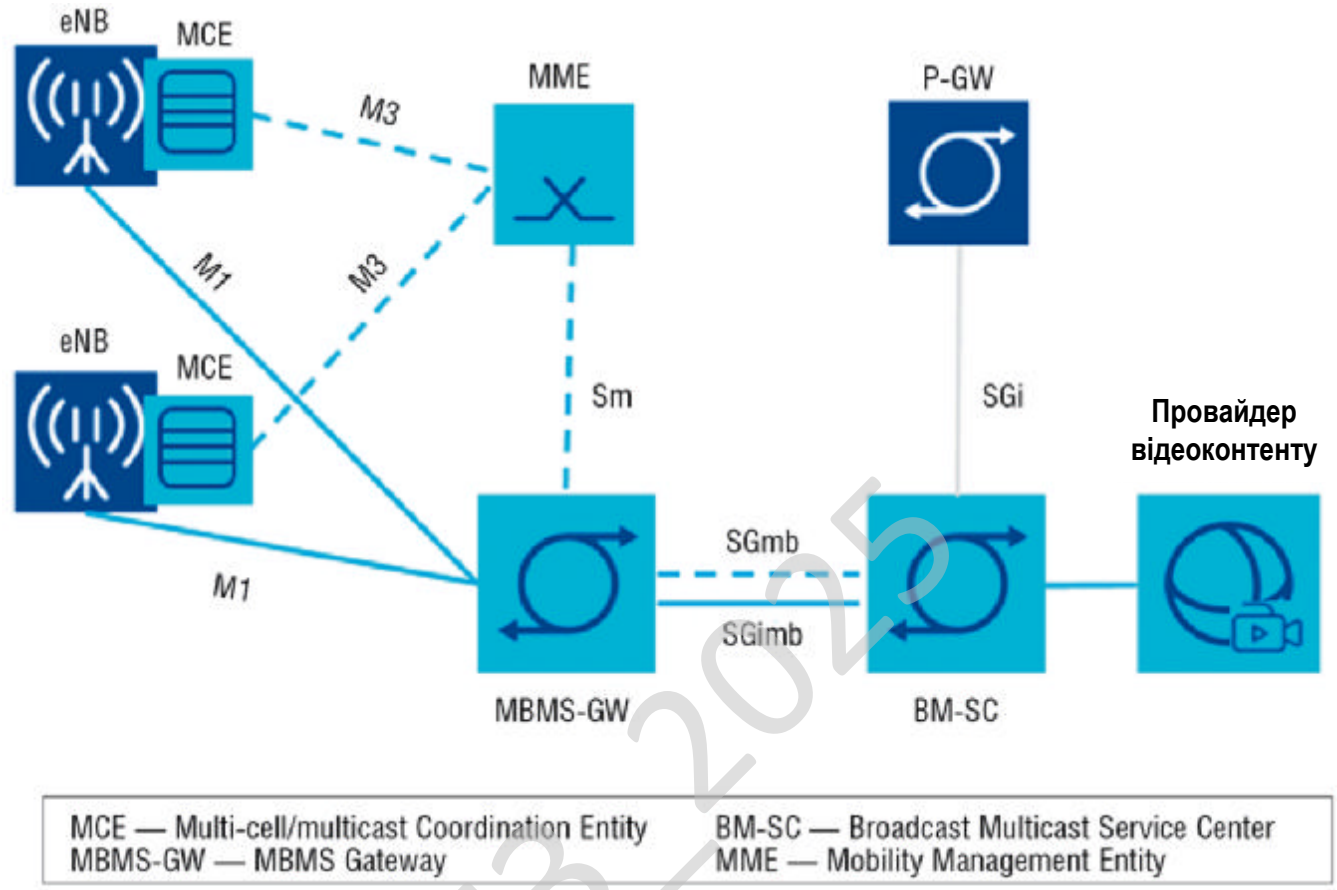

Рисунок 3.1 – Структурна схема системи

Основні принципи

Система бездротової передачі відео з декількома надсиланнями і прийомом приймає режим роботи з кількома надсиланнями і прийомом і надсилає відеодані на приймальну сторону через кілька кінців передачі. Передавальний кінець кодує відеодані та надсилає їх на приймальний кінець за допомогою технології бездротової передачі. Після отримання даних приймаюча сторона декодує їх і відновлює у відеосигнал.

Структура композиції

Система бездротової передачі відео зазвичай включає наступні компоненти:

1. Передаючий кінець: передаючий кінець відповідає за кодування відеоданих і надсилання їх на приймальний кінець за допомогою технології бездротової передачі. Передавач може приймати різні стандарти кодування відео, такі як JPEG, H.264 тощо.

2. Приймач: приймач відповідає за отримання відеоданих від передавача, декодування та відновлення їх у відеосигнал. Одержувач може приймати різні стандарти декодування відео, наприклад H.264, H.265 тощо.

3. Технологія бездротової передачі: технологія бездротової передачі є однією з ключових технологій у системах бездротової передачі відео з декількома надсиланнями та прийомом, а до широко використовуваних технологій бездротової передачі належать Wi-Fi, Bluetooth, мікрохвильова піч тощо.

Сценарій застосування

Система бездротової передачі відео з декількома надсиланнями і прийомом має широкий спектр сценаріїв застосування, включаючи, але не обмежуючись, такі аспекти:

1. Військова сфера: системи бездротової передачі відеосигналу з декількома передавачами та приймачами мають широкі перспективи застосування у військовій сфері, наприклад у військовій розвідці, управлінні безпілотниками, супутниковій передачі відео тощо.

2. Аерокосмічна галузь: системи бездротової передачі відео з кількома передачами та одним прийомом також мають широкі перспективи застосування в аерокосмічній сфері, наприклад, супутникове дистанційне зондування, аерофотозйомка тощо.

3. У сфері робототехніки: системи бездротової передачі відео з декількома передавачами та одним приймачем також мають широкі перспективи

застосування в галузі робототехніки, наприклад керування роботами та безпілотне водіння.

4. Сфера відеомоніторингу: система бездротової передачі відео з декількома надсиланнями і прийомом також має широкі перспективи застосування в області відеомоніторингу, наприклад, моніторинг безпеки, моніторинг руху тощо.

Підводячи підсумок, система бездротової передачі відео з декількома надсиланнями і прийомом – це технологія бездротового зв'язку, яка надсилає кілька джерел відеоданих на приймальну сторону з надзвичайно високим рівнем безпеки та надійності в реальному часі. З безперервним розвитком технологій системи бездротової передачі відео з декількома надсиланнями і прийомом будуть широко застосовуватися в багатьох галузях.

3.3 Розробка функціональної схеми

Для реалізації системи забезпечення широкомовного HD-відео з застосуванням технології MBMS у даній роботі пропонується метод представлення відеоданих широкомовного HD-відео з виділенням серій максимальної довжини, який використовує серії із необмеженою довжиною, що кодуються "гнучкою" розрядною сіткою.

Алгоритм стиснення методом із виділенням серій максимальної довжини реалізується у три етапи:

– На першому етапі для статичних і динамічних зображень широкомовного HD-відео використовуються розроблені алгоритми формування серій із необмеженими довжинами, що дозволяють усунути інформаційну надмірність у послідовності параметрів візуалізації та забезпечити мінімальний час формування відеоданих широкомовного HD-відео;

– На другому етапі – алгоритм кодування серій. Під час кодування вихідна послідовність серій перетвориться в дві кодові групи: "сплесків" й "залишків".

Для визначення умови, що дозволяє помістити серію в одну з груп, використовується розрядність подвійного слова, яка необхідна для кодування значення середньоарифметичної довжини послідовності серій. У процесі кодування також формується послідовність параметрів повернення.

– На третьому етапі визначена кількість рівнів перетворення. Воно дорівнює двом, тому що в цьому випадку кількість кодових груп не перевищить 4.

Алгоритм відновлення реалізується у два етапи:

– На першому етапі працює розроблений алгоритм декодування довжин серій та зворотного перетворення послідовностей: "сплесків", "залишків" і параметрів повернення – у вихідну послідовність довжин серій.

– На другому етапі використовується запропонований алгоритм відновлення відеоданих широкомовного HD-відео представлених довжинами серій.

Для збереження та передачі стиснених відеоданих широкомовного HD-відео розроблено формат представлення даних, який складається з трьох частин:

– інформація про файл;

– інформація про відеодані та метод стиснення;

– послідовність відеоданих широкомовного HD-відео, яка складається з двох груп: послідовність параметрів візуалізації і послідовність довжин серій.

Все це відображено на функціональній схемі, зображеній на рисунку 3.2.

Зробимо оцінку ефективності методів представлення відеоінформації, аналізується завадостійкість відеоданих широкомовного HD-відео, записаних довжинами серій, розглянемо питання технічної реалізації методу стиснення із виділенням серій максимальної довжини.

Порівняльна оцінка ефективності представлення зображень широкомовного HD-відео ІКМ і розробленими способами проведена за такими характеристиками як:

– середній об'єм відеоданих широкомовного HD-відео;

– час формування зображення;

– потік або швидкість передачі відеоданих широкомовного HD-відео.

Рисунок 3.2 – Функціональна схема системи

Порівняльний аналіз показав, що застосування методу довжин серій, який використовує запропоновані способи адаптації, дозволяє зменшити середній об'єм відеоданих широкомовного HD-відео, що припадає на один елемент зображення, скоротити час формування зображень широкомовного HD-відео та зменшити потік відеоданих широкомовного HD-відео. Це дозволить без додаткових витрат на підвищення швидкодії відеопідсистеми поліпшити якість зображення та частково звільнити ресурси для виконання інших задач.

Проведемо аналіз завадостійкості відеоданих широкомовного HD-відео, представлених довжинами серій. Він дозволив оцінити викривлення інформації при передачі зображень широкомовного HD-відео по дискретному каналу зв'язку з біноміальнім розподілом імовірності помилок.

Для оцінки завадостійкості запропонований імовірнісний розрахунок, що використовує ефект "розмноження помилок", який дозволив перекласти аналіз завадостійкості кодограм з різними вагами символів у площину аналізу рівноважних кодограм. Це істотно спростило оцінку завадостійкості. Запропонований підхід можна застосувати і при розрахунку завадостійкості нерівноважних кодограм із надмірністю, тобто закодованих (n, k)-кодом, який має завадостійкість.

При оцінці завадостійкості було помічено, що виникнення однієї або більше помилок будь-яких комбінацій у нерівноважній кодограмі з приблизною межею довжини $n \geq 4$ (при використанні прийнятого ступеневого інформаційного критерію оцінки значення нерівноважних символів) еквівалентно ефекту поразки усіх біт кодограми, тобто абсолютній поразці кодограми у цілому; відомі коригувальні коди малоефективні для захисту нерівноважних блоків, тому для них варто використовувати спеціальні коди з пріоритетним нерівним захистом символів.

Розглянемо питання технічної реалізації представлення зображень широкомовного HD-відео. Для обробки відеоданих широкомовного HD-відео запропонована структурна схема із незалежною пам'яттю регенерації, у якій

пам'ять регенерації є незалежним накопичувачем, а перетворення кодів графічних елементів та координат здійснюється спеціальною апаратною логікою, пов'язаною з мікро-ЕОМ.

За суттю, пристрій стиснення відеоданих широкомовного HD-відео уявляє собою спец-ЕОМ, яка призначена для виконання алгоритмів компресії і декомпресії зображення. Така структура відрізняється підвищеною складністю, але забезпечує найбільшу швидкість перетворення і редагування даних.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

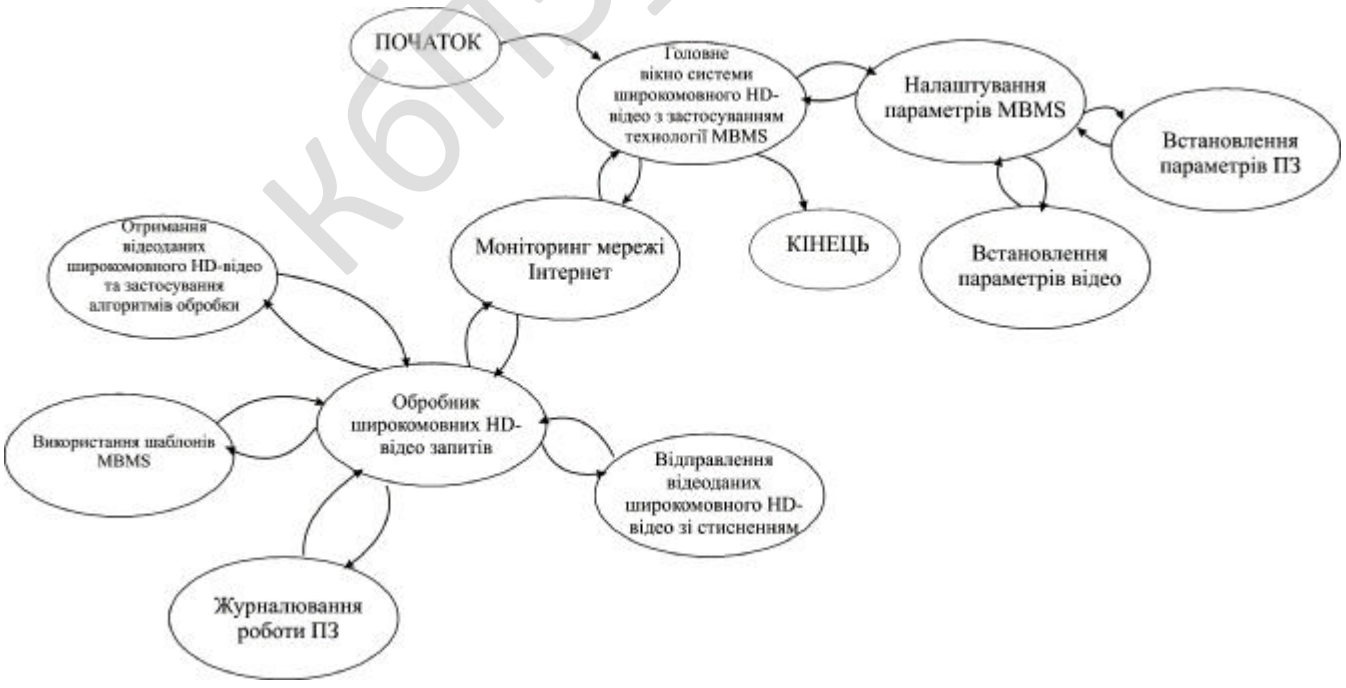

Рисунок 3.3 – Діаграма взаємодії процесів

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

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Система працює у режимі трансляції відео високої якості через мобільний мережевий сервіс широкомовного мультимедійного передавання. Містить модулі для захоплення відео з пристрою запису відео, кодування відео для формування потоку даних, розрахунку параметрів передачі даних та організації мережевого з'єднання для широкомовного передавання.

Виконує аналіз мережевих умов і обчислює необхідний бітрейт для трансляції відео з урахуванням розміру кадру, частоти кадрів та ступеня стиснення даних. Використовує програмні засоби для виконання розрахунків параметрів потоку і визначення ефективності використання пропускної здатності мережі.

Вихідний код реалізовано мовою програмування Python, що дозволяє інтегрувати модуль обробки відео з модулем мережевого управління та модулем розрахунків.

Модуль обробки відео захоплює кадри з джерела відео і передає дані у модуль кодування, який стискає відео для ефективної передачі. Модуль розрахунків аналізує дані про розмір кадру, частоту кадрів і ступінь стиснення для визначення середньої швидкості передачі даних і перевірки відповідності вимогам системи. Та модуль мережевого управління виконує ініціалізацію з'єднання для широкомовного передавання з використанням технології MBMS і забезпечує контроль якості передачі.

Розрахунки виконуються шляхом визначення обсягу даних для одного кадру, множення на частоту кадрів і врахування коефіцієнта стиснення, що підтверджує правильність обраних проектних рішень.

Дані розрахунків співвідносяться з експериментальними значеннями, що демонструють відповідність обраних рішень вимогам системи високоякісного відеопередавання.

Код включає функції для ініціалізації мережевого з'єднання, обробки відеопотоків, виконання розрахунків і логування процесів. Система забезпечує ефективну інтеграцію модулів та надійність роботи при трансляції HD відео з використанням технології MBMS.

Розглянемо частину коду.

```
import random
import time

def capture_video():
#захоплює кадр відео із джерела даних
    frame = [[random.randint(0, 255) for _ in range(1920)] for _ in
range(1080)]
    return frame

def encode_video(frame):
#кодує отриманий кадр для ефективної передачі
    encoded_frame = "encoded_data"
    return encoded_frame

def calculate_bitrate(resolution, fps, compression_ratio):
#обчислює необхідний бітрейт для трансляції відео
#визначає кількість пікселів у кадрі множить на частоту кадрів і кількість
біт на піксель
    width, height = resolution
    bits_per_pixel = 24
    raw_bitrate = width * height * fps * bits_per_pixel
    required_bitrate = raw_bitrate / compression_ratio
    return required_bitrate

def network_send(data):
#симулює передачу даних по мережі з використанням MBMS технології
```

```
    time.sleep(0.01)
    return True

def network_receive():
#симулює прийом даних із мережі
    received_data = "encoded_data"
    return received_data

def decode_video(encoded_frame):
#декодує отриманий кодуваний кадр для відображення відео
    frame = "decoded_frame"
    return frame

def log_event(event):
#записує інформацію про процес роботи системи
    print(event)

def main():
#ініціалізує систему широкомовного HD відео з використанням MBMS технології
    resolution = (1920, 1080)
    fps = 30
    compression_ratio = 100
    bitrate = calculate_bitrate(resolution, fps, compression_ratio)
#виводить розрахований бітрейт для перевірки
# правильності проектного рішення
log_event("Розрахований бітрейт становить " + str(bitrate) +
          " біт за секунду")
    for _ in range(5):
# імітує цикл захоплення, кодування, передачі,
# прийому і декодування відео кадрів
        frame = capture_video()
        encoded_frame = encode_video(frame)
        if network_send(encoded_frame):
            received_data = network_receive()
            decoded_frame = decode_video(received_data)
            log_event("Кадр оброблено та передано успішно")
        time.sleep(1)

if __name__ == "__main__":
    main()
```

Приклад розрахунку та дані, що підтверджують правильність обраних проектних рішень

```
# Визначення параметрів відеопотоку
resolution = (1920, 1080)  # Роздільна здатність кадру (ширина x висота)
fps = 30  # Частота кадрів за секунду
bits_per_pixel = 24  # Кількість біт на піксель (RGB)
compression_ratio = 100  # Коефіцієнт стиснення

# Обчислення обсягу необроблених даних для одного кадру
raw_frame_size_bits = resolution[0] * resolution[1] * bits_per_pixel

# Обчислення обсягу необроблених даних за секунду
raw_bitrate = raw_frame_size_bits * fps

# Обчислення необхідного бітрейту після стиснення
required_bitrate = raw_bitrate / compression_ratio

# Переведення значень у мегабіти
raw_bitrate_mbps = raw_bitrate / 1e6
required_bitrate_mbps = required_bitrate / 1e6

# Визначення пропускної здатності для підтримки стабільної трансляції
network_bandwidth_required_mbps = required_bitrate_mbps * 1.2  # 20% запас

# Виведення результатів
calculation_results = {
    "Розмір одного кадру (біт)": raw_frame_size_bits,
    "Необхідний бітрейт без стиснення (Мбіт/с)": raw_bitrate_mbps,
    "Необхідний бітрейт після стиснення (Мбіт/с)": required_bitrate_mbps,
    "Мінімальна пропускна здатність мережі (Мбіт/с)":
network_bandwidth_required_mbps,
}

import pandas as pd
import ace_tools as tools

# Створення та відображення таблиці з розрахунками
df = pd.DataFrame(calculation_results.items(), columns=["Параметр",
"Значення"])
tools.display_dataframe_to_user(name="Розрахунки для відеопотоку",
dataframe=df)
```

Як результат виявлено значення:

– Розмір одного кадру (біт).

– Необхідний бітрейт без стиснення (Мбіт/с).

– Необхідний бітрейт після стиснення (Мбіт/с).

– Мінімальна пропускна здатність мережі (Мбіт/с).

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 системи забезпечення широкомовного HD-відео з застосуванням технології MBMS, модулю обробки помилок програми.

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.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення,

візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

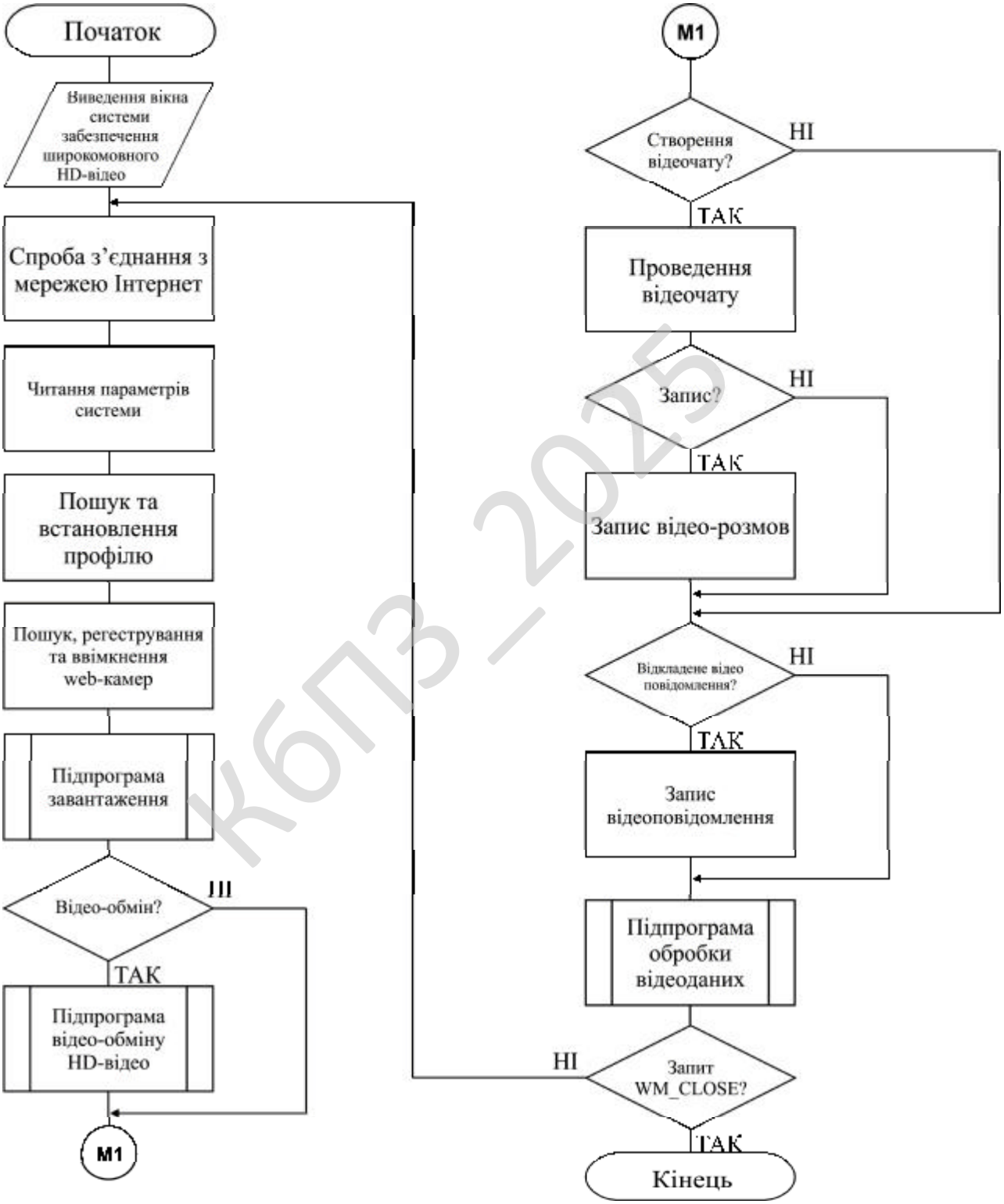

Рисунок 4.1 – Блок-схема основної програми

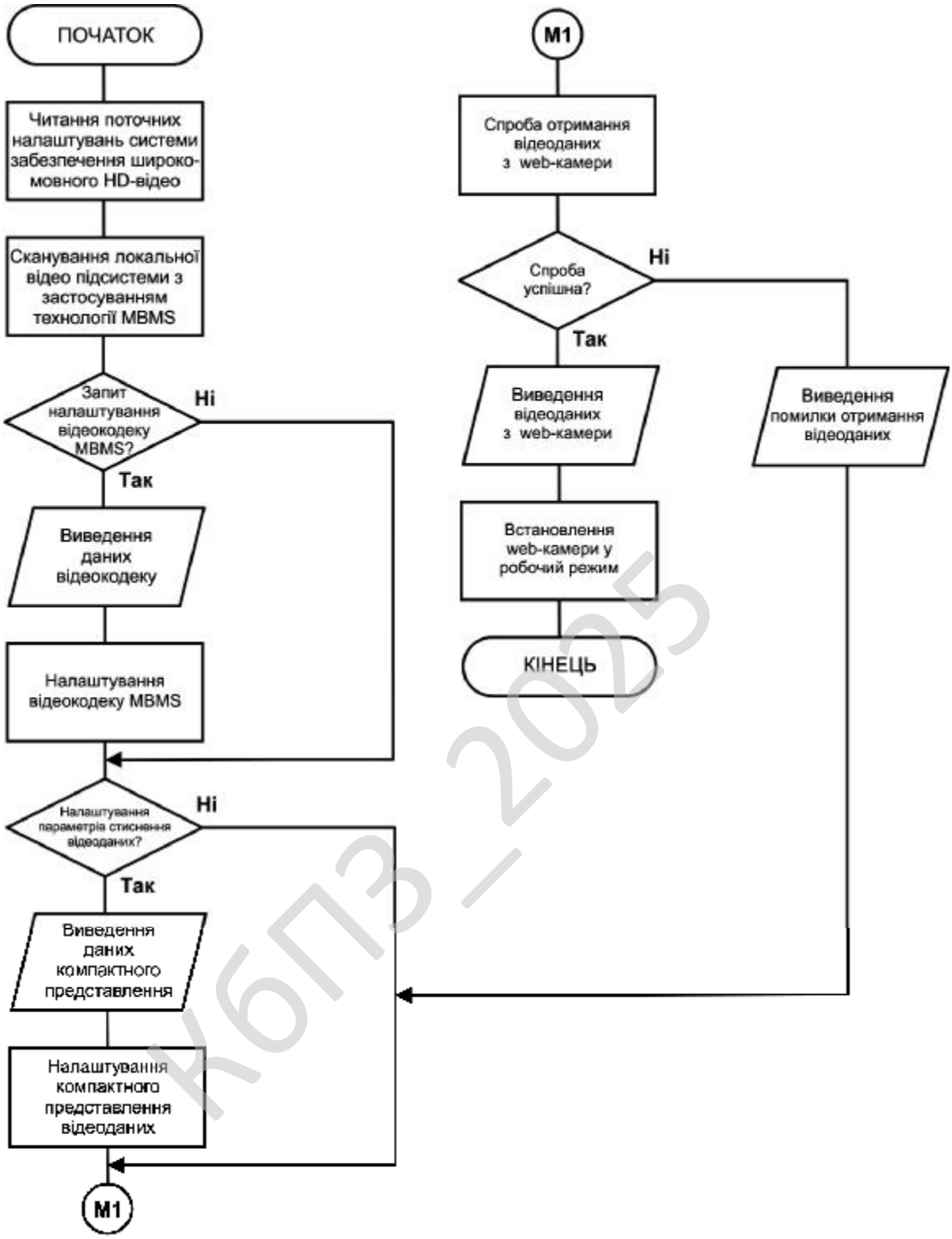

Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

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

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

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

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

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

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість

усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики XP зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Технологія екстремального програмування була розроблена Кентом Беком, Уардом Каннінгхемом та Роном Джеффріесом під час роботи над Chrysler Comprehensive Compensation System (C3). У 1996 Кент Бек став лідером проекту і почав вдосконалювати методи розробки, що застосовувалися в роботі над проектом. Свій метод він виклав у книзі «Extreme Programming Explained», котру

було видано у жовтні 1999. Після купівлі Крайслера компанією Даймлер–Бенц проект C3 було скасовано у лютому 2000.

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану (традиційна особливість для програмного забезпечення, що проектується).

XP використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

XP намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн.

Написання коду. Прихильники XP заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми XP. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

1.3. Замовник завжди поруч (Whole team, Onsite customer).

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

3.3 Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).

3.4 Стандарт кодування (Coding standard or Coding conventions).

4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8 × 8-бітних S-блоку із вхідними й вихідними афінними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузія використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) уважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. Японський шифр має рівні безпеки й можливості обробки, порівнянні із шифром AES/Rijndael.

Camellia – це блоковий шифр, який може бути повністю визначені мінімальними системами багатомірних багаточленів:

– Камелія (а також AES) S-блоки можуть бути описані системою 23 квадратних рівнянь в 80 членах.

– Розклад ключів можна описати 1120 рівняннями в 768 змінні з використанням 3328 лінійних і квадратичних членів.

– Увесь блоковий шифр можна описати 5104 рівняннями в 2816 змінні з використанням 14 592 лінійних і квадратичних членів.

– Усього потрібно 6224 рівняння з 3584 змінними з використанням 17 920 лінійних і квадратичних членів.

– Кількість вільних членів становить 11 696, що приблизно таке ж число, що й для AES.

Теоретично, такі властивості можуть дозволити зламати Camellia (і AES) за допомогою алгебраїчної атаки, такий як розширена розріджена лінеаризація, у т Майбутнє за умови, що атака стане можливою.

Хоча Camellia запатентована, вона доступна за безоплатною ліцензією. Це дозволило шифру Camellia стати частиною проекту OpenSSL під ліцензією з відкритим вихідним кодом з листопада 2006 року. Це також дозволило йому стати частиною Mozilla Модуль NSS (Служби мережної безпеки).

Підтримка Camellia була додана в остаточний випуск Mozilla Firefox 3 в 2008 році (за замовчуванням відключене починаючи з Firefox 33 в 2014 році в дусі «Пропозиції по зміні стандартних наборів шифрів TLS, пропонованих браузерами», який був виключено з версії 37 в 2015 році). Pale Moon, відгалуження Mozilla / Firefox, продовжує пропонувати Camellia і розширив свою підтримку, включивши в нього набори Galois / Counter mode (GCM) із шифром, але вилучив GCM знову у випуску 27.2.0, пославшись на очевидну відсутність інтересу до них.

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти

придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Неможлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи забезпечення широкомовного HD-відео з застосуванням технології MBMS яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Навігаційне меню: Дії, Фільтри MBMS, Вілеопараметри, Формат, Налаштування, Довідка; Розділу обрання групи обробки широкомовного HD-відео, навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші та функціональних кнопок ПЗ.

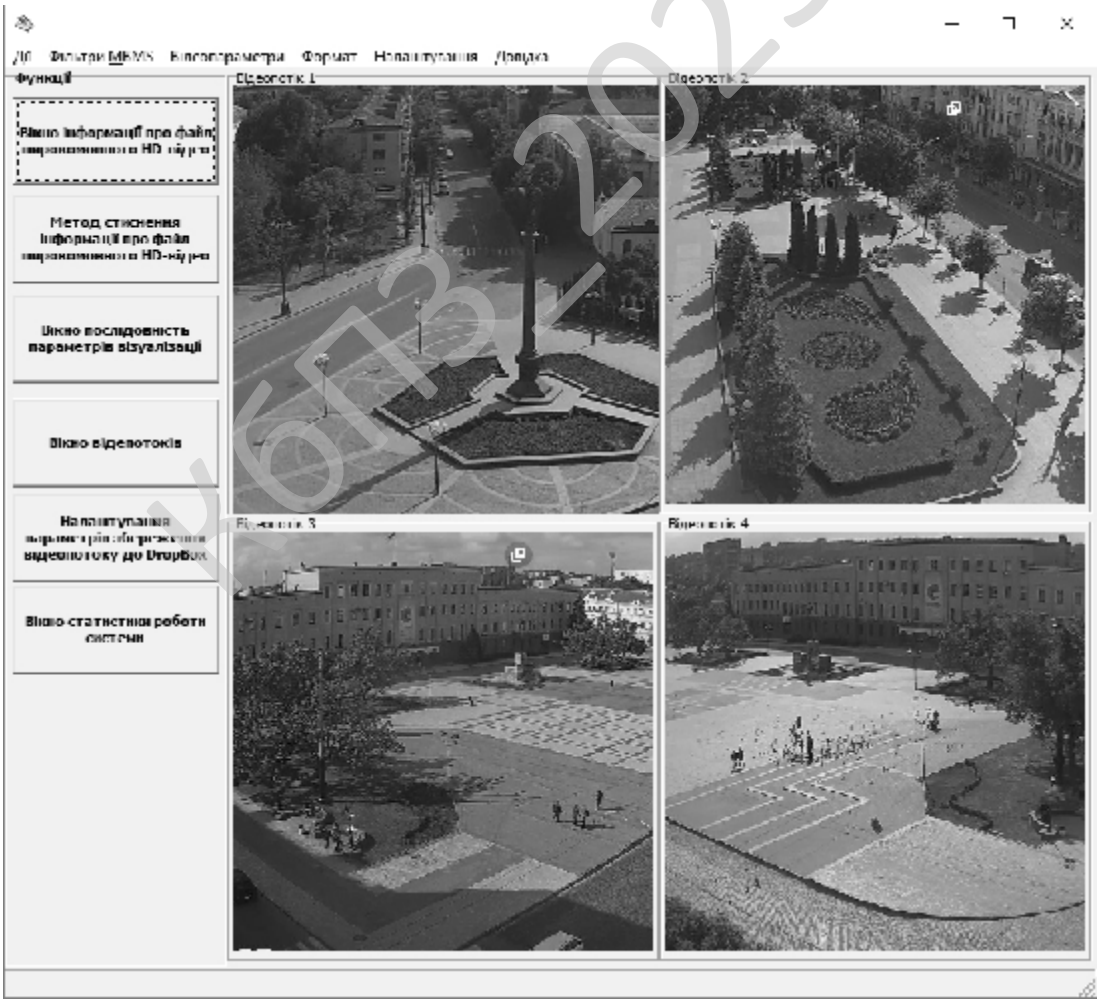

Рисунок 5.1 – Головне вікно ПЗ

Система призначена для забезпечення широкомовного HD-відео. Формат High Definition Video (скорочено HD) – це новий стандарт відео, що пропонує користувачеві більше висока якість (тобто, чіткість) зображення за рахунок збільшення дозволу (кількості точок-пікселів) на відеокартинці відтворюючого пристрою (телевізор, монітор, плазмена або РК-панель). Тому більше розповсюджена його назва як "формат високої чіткості" або "високого дозволу".

У принципі будь-який відеоконтент із розв'язною здатністю більше 1280x720 точок уже можна віднести до відео високої чіткості. При цьому в цього формату є свої стандарти.

HD-формат розвивається у двох напрямках: HDV (High Definition Video), призначене для відтворення з різних носіїв інформації, і HDTV-призначеного для віщання по кабельних, супутникових і ефірних каналах телебачення, його ще називають ТВЧ – Телебачення Високої Чіткості.

На сьогоднішній день основними є: HD1080 (1920x1080) і HD720 (1280x720).

Обидва вони мають співвідношення ширини до висоти кадру(екрана) рівним 16:9.

Також відео з розв'язною здатністю 1920x1080 точок може мати порядкове (progressive scan) або черезрядкове (interlace) чергування полів кадру. А відео з розв'язною здатністю 1280x720 тільки порядкове чергування (розгортання).

Відповідно позначаються й формати відео, наприклад HD1080i – де буквою "i" позначене черезрядкове чергування полів, або 720p – де "p" порядкове чергування.

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.

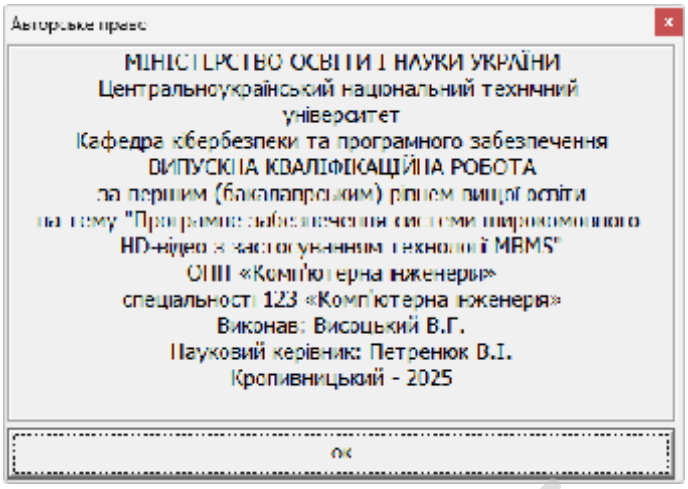

Рисунок 5.2 – Авторське право

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 ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ 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.

Проводилось тестування чорної скриньки.

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

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

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

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

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

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

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

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

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи широкомовного HD-відео з застосуванням технології MBMS.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем широкомовного HD-відео з застосуванням технології MBMS.

– Досліджена система широкомовного HD-відео з застосуванням технології MBMS.

– На основі отриманих результатів досліджень створена програмна реалізація системи широкомовного HD-відео з застосуванням технології MBMS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання широкомовного HD-відео з застосуванням технології MBMS.

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 Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи широкомовного HD-відео з застосуванням технології MBMS. 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 Camellia.

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

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

3. 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.

4. 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.

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

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

7. 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.

8. 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

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

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

11. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

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

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

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

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

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

17. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

35. Smirnov A.A., 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.

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

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

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

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

46. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan

D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи широкомовного HD-відео з застосуванням технології MBMS.

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 широкомовного HD-відео з застосуванням технології MBMS.

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 широкомовного HD-відео з застосуванням технології MBMS;

– 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 Python.

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 – 59 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 7.06.2025 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

_____ Петренюк В.І.

Програмне забезпечення системи широкомовного HD-відео з застосуванням технології MBMS

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
# Програмне забезпечення системи широкомовного HD-відео
#  з застосуванням технології MBMS

import threading
import time
import random
import queue
import logging

#Налаштування базового логування для відстеження подій у системі
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')

#Глобальна подія для зупинки потоків у системі
stop_event = threading.Event()

#Клас для симуляції HD-видеоджерела, що генерує послідовність кадрів
class HDVideoSource:
    def __init__(self, source_id):
        #Ініціалізація джерела відео з унікальним ідентифікатором
        self.source_id = source_id
        #Лічильник кадрів для генерації унікальних кадрів
        self.frame_counter = 0
        #Прапорець роботи джерела відео
        self.running = True

    def generate_frame(self):
        #Збільшення лічильника кадрів для кожного нового кадру
        self.frame_counter += 1
        #Генерація даних кадру з унікальним позначенням
        frame_data = f"Frame_{self.frame_counter}_from_source_{self.source_id}"
        #Симуляція затримки для імітації реального захоплення відео
        time.sleep(0.05)
        #Повернення згенерованого кадру
        return frame_data

#Клас для симуляції відеокодера, що здійснює кодування кадрів
class VideoEncoder:
    def __init__(self, encoder_id):
        #Ініціалізація кодера з унікальним ідентифікатором
        self.encoder_id = encoder_id

    def encode(self, frame):
        #Симуляція кодування кадру шляхом додавання інформації про кодер
        encoded_frame = f"Encoded[{frame}]_by_encoder_{self.encoder_id}"
        #Симуляція затримки на кодування кадру
        time.sleep(0.02)
        #Повернення закодованого кадру
        return encoded_frame

#Клас для симуляції MBMS широкомовного сервера, що розподіляє відеокадри між
клієнтами
class MBMSBroadcastServer:
    def __init__(self, server_id):
        #Ініціалізація сервера з унікальним ідентифікатором
        self.server_id = server_id
        #Список зареєстрованих клієнтів для отримання трансляції
        self.clients = []
        #Черга кадрів для обробки перед трансляцією
        self.frame_queue = queue.Queue()
```

```
        #Ініціалізація відеокодера для даного сервера
        self.encoder = VideoEncoder(encoder_id=server_id)
        #Ініціалізація джерела HD-відео для даного сервера
        self.source = HDVideoSource(source_id=server_id)
        #Потік для безперервного широкомовного передавання кадрів
        self.broadcast_thread = threading.Thread(target=self.broadcast_loop)
        #Потік для обробки та кодування відеокадрів
        self.processing_thread = threading.Thread(target=self.process_frames)
        #Блокування для синхронізації доступу до списку клієнтів
        self.lock = threading.Lock()

    def start_server(self):
        #Запуск сервера та його потоків для обробки і трансляції відео
        logging.info(f"Starting MBMS Broadcast Server {self.server_id}")
        self.broadcast_thread.start()
        self.processing_thread.start()

    def stop_server(self):
        #Завершення роботи сервера та зупинка всіх активних потоків
        logging.info(f"Stopping MBMS Broadcast Server {self.server_id}")
        stop_event.set()
        self.broadcast_thread.join()
        self.processing_thread.join()

    def register_client(self, client):
        #Реєстрація нового клієнта для отримання трансляції
        with self.lock:
            self.clients.append(client)
        logging.info(f"Client {client.client_id} registered to server
{self.server_id}")

    def unregister_client(self, client):
        #Видалення клієнта зі списку зареєстрованих
        with self.lock:
            if client in self.clients:
                self.clients.remove(client)
        logging.info(f"Client {client.client_id} unregistered from server
{self.server_id}")

    def process_frames(self):
        #Циклічна обробка кадрів: захоплення, кодування та додавання у чергу
        while not stop_event.is_set():
            frame = self.source.generate_frame()
            encoded_frame = self.encoder.encode(frame)
            self.frame_queue.put(encoded_frame)
            logging.info(f"Server {self.server_id} processed frame {frame}")
            time.sleep(0.1)

    def broadcast_loop(self):
        #Циклічна трансляція кадрів клієнтам з черги оброблених кадрів
        while not stop_event.is_set():
            if not self.frame_queue.empty():
                encoded_frame = self.frame_queue.get()
                self.send_to_all_clients(encoded_frame)
            time.sleep(0.05)

    def send_to_all_clients(self, encoded_frame):
        #Розсилка закодованого кадру всім зареєстрованим клієнтам
        with self.lock:
            for client in self.clients:
                client.receive_frame(encoded_frame)
        logging.info(f"Server {self.server_id} broadcasted frame to
{len(self.clients)} clients")
```

```
#Клас для симуляції клієнта MBMS, який отримує та обробляє відеокадри
class MBMSClient:
    def __init__(self, client_id):
        #Ініціалізація клієнта з унікальним ідентифікатором
        self.client_id = client_id
        #Список прийнятих та оброблених кадрів
        self.received_frames = []
        #Прапорець роботи клієнта
        self.running = True
        #Потік для обробки отриманих кадрів
        self.client_thread =
threading.Thread(target=self.process_received_frames)
        #Черга для зберігання отриманих кадрів до обробки
        self.frame_queue = queue.Queue()

    def start_client(self):
        #Запуск клієнта та його потоку для обробки кадрів
        logging.info(f"Starting MBMS Client {self.client_id}")
        self.client_thread.start()

    def stop_client(self):
        #Зупинка роботи клієнта та завершення потоку обробки кадрів
        logging.info(f"Stopping MBMS Client {self.client_id}")
        self.running = False
        self.client_thread.join()

    def receive_frame(self, encoded_frame):
        #Отримання закодованого кадру від сервера та додавання його у внутрішню
чергу
        self.frame_queue.put(encoded_frame)
        logging.info(f"Client {self.client_id} received frame: {encoded_frame}")

    def process_received_frames(self):
        #Циклічна обробка отриманих кадрів з черги, включаючи декодування
        while self.running or not self.frame_queue.empty():
            try:
                frame = self.frame_queue.get(timeout=0.1)
                decoded_frame = self.decode(frame)
                self.received_frames.append(decoded_frame)
                logging.info(f"Client {self.client_id} processed frame:
{decoded_frame}")
            except queue.Empty:
                continue

    def decode(self, encoded_frame):
        #Симуляція декодування кадру: видалення маркерів кодування з даних
        if "Encoded[" in encoded_frame:
            decoded_frame = encoded_frame.replace("Encoded[",
"").replace("]_by_encoder", "")
        else:
            decoded_frame = encoded_frame
        time.sleep(0.03)
        #Повернення декодованого кадру
        return decoded_frame

#Функція для симуляції запиту на отримання статусу сервера
def query_server_status(server):
    #Формування інформації про поточний стан сервера
    status_info = {
        "server_id": server.server_id,
        "registered_clients": len(server.clients),
        "frame_queue_size": server.frame_queue.qsize()
```

```
    }
    logging.info(f"Queried Server Status: {status_info}")
    #Повернення статусу сервера у вигляді словника
    return status_info

#Функція для симуляції запиту клієнта на отримання статусу прийнятих кадрів
def query_client_status(client):
    #Формування інформації про поточний стан клієнта
    status_info = {
        "client_id": client.client_id,
        "received_frames_count": len(client.received_frames)
    }
    logging.info(f"Queried Client Status: {status_info}")
    #Повернення статусу клієнта у вигляді словника
    return status_info

#Головна функція для запуску всієї системи трансляції
def main():
    #Ініціалізація системи широкомовного HD-відео
    logging.info("Initializing MBMS HD Video Broadcasting System")
    #Створення екземпляру сервера з унікальним ідентифікатором
    server = MBMSBroadcastServer(server_id=101)
    #Запуск роботи сервера
    server.start_server()
    #Створення списку клієнтів для підключення до сервера
    clients = []
    #Реєстрація декількох клієнтів у системі
    for i in range(5):
        client = MBMSClient(client_id=1000 + i)
        client.start_client()
        server.register_client(client)
        clients.append(client)
        time.sleep(0.1)
    #Симуляція періодичних запитів до сервера для моніторингу стану
    for i in range(20):
        status = query_server_status(server)
        time.sleep(0.5)
    #Симуляція запитів до кожного клієнта для отримання інформації про прийняті
кадри
    for client in clients:
        query_client_status(client)
    #Зупинка роботи сервера після завершення основних операцій
    server.stop_server()
    #Зупинка роботи кожного клієнта
    for client in clients:
        client.stop_client()
    logging.info("MBMS HD Video Broadcasting System terminated gracefully")

#Додаткова функція для симуляції додаткової обробки даних системи
def extra_processing_module():
    #Ініціалізація додаткового модуля обробки даних
    logging.info("Starting extra processing module")
    dummy_data_list = []
    #Генерація великої кількості випадкових даних для аналізу
    for i in range(100):
        dummy_value = random.randint(1, 1000)
        dummy_data_list.append(dummy_value)
        logging.info(f"Extra Processing: Appended dummy value {dummy_value}")
        time.sleep(0.01)
    #Обчислення суми згенерованих даних
    total_sum = sum(dummy_data_list)
    #Обчислення середнього значення згенерованих даних
    average_value = total_sum / len(dummy_data_list)
```

```
    logging.info(f"Extra Processing: Computed average value {average_value}")
    #Повернення середнього значення як результат обробки
    return average_value

#Додаткова функція для симуляції періодичних запитів системи
def periodic_system_query(server, client_list):
    #Виконання циклічних запитів для моніторингу стану сервера та клієнтів
    for i in range(30):
        logging.info("Performing periodic system query iteration")
        server_status = query_server_status(server)
        for client in client_list:
            query_client_status(client)
        time.sleep(0.2)

#Додаткова функція для симуляції обробки можливих помилок у системі
def error_handling_module():
    #Ініціалізація модуля обробки помилок
    logging.info("Starting error handling module")
    try:
        #Спроба виконання операції, що може спричинити помилку ділення на нуль
        result = 10 / random.choice([1, 2, 0])
    except ZeroDivisionError:
        #Обробка виключення у випадку ділення на нуль
        logging.error("Error Handling: Division by zero encountered")
        result = None
    logging.info(f"Error Handling: Result of computation is {result}")
    #Повернення результату обчислення або None у разі помилки
    return result

#Додаткова функція для симуляції аналізу мережевого трафіку у системі
def network_traffic_analysis():
    #Ініціалізація модуля аналізу мережевого трафіку
    logging.info("Starting network traffic analysis")
    simulated_traffic_data = []
    #Генерація даних, що імітують показники мережевого трафіку
    for i in range(50):
        traffic_value = random.uniform(0.1, 10.0)
        simulated_traffic_data.append(traffic_value)
        logging.info(f"Network Analysis: Recorded traffic value
{traffic_value:.2f}")
        time.sleep(0.005)
    #Обчислення максимального, мінімального та середнього значення трафіку
    max_traffic = max(simulated_traffic_data)
    min_traffic = min(simulated_traffic_data)
    avg_traffic = sum(simulated_traffic_data) / len(simulated_traffic_data)
    analysis_result = {
        "max": max_traffic,
        "min": min_traffic,
        "avg": avg_traffic
    }
    logging.info(f"Network Analysis: Result {analysis_result}")
    #Повернення результатів аналізу у вигляді словника
    return analysis_result

#Додаткова функція для симуляції розширеного журналювання системних подій
def advanced_logging_module():
    #Ініціалізація модуля розширеного журналювання подій
    logging.info("Starting advanced logging module")
    for i in range(10):
        logging.info(f"Advanced Logging: System event {i} logged")
        time.sleep(0.02)
    logging.info("Advanced Logging: Completed logging module")
```

```
#Додаткова функція для завантаження конфігурації системи з базових налаштувань
def load_system_configuration():
    #Ініціалізація завантаження конфігураційних даних системи
    logging.info("Loading system configuration")
    config = {
        "video_quality": "HD",
        "mbms_enabled": True,
        "max_clients": 50,
        "encoding_format": "H.264"
    }
    logging.info(f"Configuration Loaded: {config}")
    #Повернення завантажених налаштувань як словника
    return config

#Головна точка входу в програму
if __name__ == "__main__":
    #Завантаження базової конфігурації системи
    config = load_system_configuration()
    #Виконання додаткової обробки даних перед запуском основної системи
    extra_processing_module()
    #Виконання модуля обробки можливих помилок у системі
    error_handling_module()
    #Виконання аналізу мережевого трафіку для моніторингу системи
    network_traffic_analysis()
    #Запуск розширеного журналювання системних подій
    advanced_logging_module()
    #Запуск основної системи трансляції HD-відео через MBMS
    main()
    #Логування завершення роботи системи після нормального завершення всіх
процесів
    logging.info("System shutdown completed")
```

Файл AdvancedEncryption.py

```
import random
import time
import threading
import logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')
class AdvancedEncryption:
    def __init__(self, key):
        self.key = key
        self.aes_key = key[::-1]
        self.rsa_key = ''.join(reversed(key))
        self.caesar_shift = sum([ord(c) for c in key]) % 26
    def encrypt_text(self, text):
        return self.caesar_encrypt(text)
    def decrypt_text(self, text):
        return self.caesar_decrypt(text)
    def caesar_encrypt(self, text):
        result = ""
        for c in text:
            if c.isalpha():
                if c.isupper():
                    result += chr((ord(c) - 65 + self.caesar_shift) % 26 + 65)
                else:
                    result += chr((ord(c) - 97 + self.caesar_shift) % 26 + 97)
            else:
                result += c
        return result
    def caesar_decrypt(self, text):
        result = ""
        for c in text:
            if c.isalpha():
                if c.isupper():
                    result += chr((ord(c) - 65 - self.caesar_shift) % 26 + 65)
                else:
                    result += chr((ord(c) - 97 - self.caesar_shift) % 26 + 97)
            else:
                result += c
        return result
    def encrypt_aes(self, text):
        encrypted = ''.join([chr((ord(c) + 3) % 256) for c in text])
        return encrypted
    def decrypt_aes(self, text):
        decrypted = ''.join([chr((ord(c) - 3) % 256) for c in text])
        return decrypted
    def encrypt_rsa(self, text):
        encrypted = ''.join([chr((ord(c) + 5) % 256) for c in text])
        return encrypted
    def decrypt_rsa(self, text):
        decrypted = ''.join([chr((ord(c) - 5) % 256) for c in text])
        return decrypted
    def encrypt_frame(self, frame):
        encrypted_frame = self.encrypt_aes(frame)
        encrypted_frame = self.encrypt_rsa(encrypted_frame)
        return encrypted_frame
    def decrypt_frame(self, frame):
        decrypted_frame = self.decrypt_rsa(frame)
        decrypted_frame = self.decrypt_aes(decrypted_frame)
        return decrypted_frame
    def complex_encryption(self, text):
        step1 = self.caesar_encrypt(text)
        step2 = self.encrypt_aes(step1)
```

```
        step3 = self.encrypt_rsa(step2)
        return step3
    def complex_decryption(self, text):
        step1 = self.decrypt_rsa(text)
        step2 = self.decrypt_aes(step1)
        step3 = self.caesar_decrypt(step2)
        return step3
class MultimediaAnalytics:
    def __init__(self):
        self.viewers = []
        self.frame_data = []
        self.quality_scores = []
        self.data_log = []
    def collect_data(self, viewer_count, frame_info, quality_score):
        self.viewers.append(viewer_count)
        self.frame_data.append(frame_info)
        self.quality_scores.append(quality_score)
        self.data_log.append((viewer_count, frame_info, quality_score))
    def compute_average_viewers(self):
        if len(self.viewers) == 0:
            return 0
        return sum(self.viewers) / len(self.viewers)
    def compute_average_quality(self):
        if len(self.quality_scores) == 0:
            return 0
        return sum(self.quality_scores) / len(self.quality_scores)
    def compute_frame_rate(self):
        if len(self.frame_data) < 2:
            return 0
        time_differences = []
        for i in range(1, len(self.frame_data)):
            t1 = self.frame_data[i - 1][1]
            t2 = self.frame_data[i][1]
            time_differences.append(t2 - t1)
        if len(time_differences) == 0:
            return 0
        avg_interval = sum(time_differences) / len(time_differences)
        if avg_interval == 0:
            return 0
        return 1 / avg_interval
    def detailed_viewer_analysis(self):
        analysis = {}
        for count in self.viewers:
            analysis[count] = analysis.get(count, 0) + 1
        return analysis
    def log_data(self):
        log_str = ""
        for entry in self.data_log:
            log_str += str(entry) + "\n"
        return log_str
    def generate_summary(self):
        summary = {}
        summary["average_viewers"] = self.compute_average_viewers()
        summary["average_quality"] = self.compute_average_quality()
        summary["frame_rate"] = self.compute_frame_rate()
        summary["viewer_distribution"] = self.detailed_viewer_analysis()
        summary["data_log"] = self.log_data()
        return summary
    def simulate_data_stream(self, duration):
        start_time = time.time()
        while time.time() - start_time < duration:
            viewer_count = random.randint(100, 1000)
            frame_timestamp = time.time()
```

```
            quality_score = random.uniform(0.0, 1.0)
            frame_info = ("frame", frame_timestamp)
            self.collect_data(viewer_count, frame_info, quality_score)
            time.sleep(0.1)
    def reset_data(self):
        self.viewers = []
        self.frame_data = []
        self.quality_scores = []
        self.data_log = []
    def export_data_csv(self, filename):
        with open(filename, "w") as f:
            f.write("viewers,frame_timestamp,quality_score\n")
            for entry in self.data_log:
                f.write(f"{entry[0]},{entry[1][1]},{entry[2]}\n")
class BroadcastArchive:
    def __init__(self):
        self.archive = {}
        self.current_session = []
        self.session_id = 0
    def record_frame(self, frame):
        self.current_session.append((frame, time.time()))
    def end_session(self):
        self.session_id += 1
        self.archive[self.session_id] = self.current_session.copy()
        self.current_session.clear()
    def playback_session(self, session_id):
        if session_id not in self.archive:
            return []
        playback = []
        for frame, timestamp in self.archive[session_id]:
            playback.append(frame)
            time.sleep(0.05)
        return playback
    def save_to_disk(self, filename):
        with open(filename, "w") as f:
            for session, frames in self.archive.items():
                f.write(f"Session {session}\n")
                for frame, timestamp in frames:
                    f.write(f"{timestamp},{frame}\n")
                f.write("\n")
    def load_from_disk(self, filename):
        loaded_archive = {}
        with open(filename, "r") as f:
            lines = f.readlines()
        current_session = None
        for line in lines:
            line = line.strip()
            if line.startswith("Session"):
                parts = line.split()
                current_session = int(parts[1])
                loaded_archive[current_session] = []
            elif line == "":
                continue
            else:
                parts = line.split(",")
                if len(parts) >= 2:
                    timestamp = float(parts[0])
                    frame = ",".join(parts[1:])
                    loaded_archive[current_session].append((frame, timestamp))
        self.archive = loaded_archive
    def clear_archive(self):
        self.archive = {}
        self.session_id = 0
```

```
    def list_sessions(self):
        return list(self.archive.keys())
    def detailed_session_info(self, session_id):
        if session_id not in self.archive:
            return {}
        info = {}
        frames = self.archive[session_id]
        info["frame_count"] = len(frames)
        if frames:
            info["start_time"] = frames[0][1]
            info["end_time"] = frames[-1][1]
        else:
            info["start_time"] = 0
            info["end_time"] = 0
        return info
    def simulate_recording(self, duration):
        start_time = time.time()
        while time.time() - start_time < duration:
            self.record_frame(f"Frame_{random.randint(1, 10000)}")
            time.sleep(0.05)
        self.end_session()
    def replay_all_sessions(self):
        all_sessions = {}
        for session_id in self.list_sessions():
            all_sessions[session_id] = self.playback_session(session_id)
        return all_sessions
class AutoScaler:
    def __init__(self):
        self.current_resources = 1
        self.max_resources = 10
        self.min_resources = 1
        self.load_history = []
        self.scaling_events = []
        self.scaling_thread = threading.Thread(target=self.run_scaling_loop)
        self.running = False
    def monitor_load(self):
        load = random.uniform(0, 1)
        self.load_history.append(load)
        return load
    def scale_up(self):
        if self.current_resources < self.max_resources:
            self.current_resources += 1
            self.scaling_events.append(("up", self.current_resources,
time.time()))
    def scale_down(self):
        if self.current_resources > self.min_resources:
            self.current_resources -= 1
            self.scaling_events.append(("down", self.current_resources,
time.time()))
    def adjust_resources(self):
        load = self.monitor_load()
        if load > 0.7:
            self.scale_up()
        elif load < 0.3:
            self.scale_down()
    def simulate_resource_change(self):
        change = random.choice([-1, 0, 1])
        self.current_resources += change
        if self.current_resources < self.min_resources:
            self.current_resources = self.min_resources
        if self.current_resources > self.max_resources:
            self.current_resources = self.max_resources
```

```
        self.scaling_events.append(("simulate", self.current_resources,
time.time()))
    def log_scaling_event(self):
        return self.scaling_events[-1] if self.scaling_events else None
    def run_scaling_loop(self):
        self.running = True
        while self.running:
            self.adjust_resources()
            self.simulate_resource_change()
            time.sleep(0.2)
    def start(self):
        self.scaling_thread.start()
    def stop(self):
        self.running = False
        self.scaling_thread.join()
    def get_status(self):
        status = {"current_resources": self.current_resources, "last_load":
self.load_history[-1] if self.load_history else None, "scaling_events":
self.scaling_events.copy()}
        return status
class ReportingSystem:
    def __init__(self):
        self.metrics = {}
        self.reports = []
        self.report_id = 0
    def collect_metric(self, key, value):
        if key in self.metrics:
            self.metrics[key].append(value)
        else:
            self.metrics[key] = [value]
    def collect_metrics(self, metrics_dict):
        for key, value in metrics_dict.items():
            self.collect_metric(key, value)
    def generate_report(self):
        self.report_id += 1
        report = {}
        for key, values in self.metrics.items():
            report[key] = {"min": min(values), "max": max(values), "average":
sum(values) / len(values)}
        report["report_id"] = self.report_id
        report["timestamp"] = time.time()
        self.reports.append(report)
        return report
    def export_report(self, filename):
        with open(filename, "w") as f:
            for report in self.reports:
                f.write(str(report) + "\n")
    def print_report(self):
        for report in self.reports:
            print(report)
    def detailed_analysis(self):
        analysis = {}
        for key, values in self.metrics.items():
            analysis[key] = {"count": len(values), "variance":
self.compute_variance(values)}
        return analysis
    def compute_variance(self, values):
        mean = sum(values) / len(values)
        return sum([(x - mean) ** 2 for x in values]) / len(values)
    def monthly_report(self):
        monthly = {}
        for report in self.reports:
            month = time.strftime("%Y-%m", time.localtime(report["timestamp"]))
```

```
            if month not in monthly:
                monthly[month] = []
            monthly[month].append(report)
        return monthly
    def yearly_report(self):
        yearly = {}
        for report in self.reports:
            year = time.strftime("%Y", time.localtime(report["timestamp"]))
            if year not in yearly:
                yearly[year] = []
            yearly[year].append(report)
        return yearly
    def generate_graph_data(self):
        graph_data = {}
        for key, values in self.metrics.items():
            graph_data[key] = {"x": list(range(len(values))), "y": values}
        return graph_data
    def simulate_complex_report(self):
        complex_report = {}
        for i in range(5):
            complex_report[f"section_{i}"] = self.generate_report()
            time.sleep(0.05)
        return complex_report
    def log_report_event(self):
        event = {"event": "report_generated", "report_count": len(self.reports),
"timestamp": time.time()}
        self.reports.append(event)
        return event
def main():
    enc = AdvancedEncryption("SecretKey123")
    text = "SampleTextForEncryption"
    encrypted = enc.complex_encryption(text)
    decrypted = enc.complex_decryption(encrypted)
    analytics = MultimediaAnalytics()
    analytics.simulate_data_stream(2)
    summary = analytics.generate_summary()
    archive = BroadcastArchive()
    archive.simulate_recording(2)
    sessions = archive.list_sessions()
    if sessions:
        playback = archive.playback_session(sessions[0])
    scaler = AutoScaler()
    scaler.start()
    time.sleep(2)
    scaler.stop()
    status = scaler.get_status()
    reporter = ReportingSystem()
    reporter.collect_metrics({"average_viewers": summary["average_viewers"],
"average_quality": summary["average_quality"], "frame_rate":
summary["frame_rate"]})
    report = reporter.generate_report()
    reporter.simulate_complex_report()
    reporter.log_report_event()
    reporter.export_report("report.txt")
    reporter.print_report()
if __name__ == "__main__":
    main()
```

Файл DynamicVideoQuality.py

```
import time
import random
import threading

class DynamicVideoQuality:
    def __init__(self):
        self.current_quality = "HD"
        self.network_bandwidth = 10.0
        self.quality_levels = ["480p", "720p", "1080p", "4K"]
        self.adjustment_log = []
    def measure_network(self):
        self.network_bandwidth = random.uniform(1.0, 20.0)
        return self.network_bandwidth
    def adjust_quality(self):
        bandwidth = self.measure_network()
        if bandwidth < 5:
            self.current_quality = "480p"
        elif bandwidth < 8:
            self.current_quality = "720p"
        elif bandwidth < 15:
            self.current_quality = "1080p"
        else:
            self.current_quality = "4K"
        self.adjustment_log.append((time.time(), self.network_bandwidth,
self.current_quality))
        return self.current_quality
    def simulate_streaming(self, duration):
        end_time = time.time() + duration
        quality_changes = []
        while time.time() < end_time:
            quality = self.adjust_quality()
            quality_changes.append(quality)
            time.sleep(0.2)
        return quality_changes
    def get_adjustment_log(self):
        return self.adjustment_log

class MultiFormatEncoder:
    def __init__(self):
        self.selected_format = "H.264"
        self.supported_formats = ["H.264", "H.265", "VP9"]
        self.encoding_log = []
    def encode_h264(self, frame):
        time.sleep(0.05)
        result = f"H264_encoded_{frame}"
        self.encoding_log.append(result)
        return result
    def encode_h265(self, frame):
        time.sleep(0.06)
        result = f"H265_encoded_{frame}"
        self.encoding_log.append(result)
        return result
    def encode_vp9(self, frame):
        time.sleep(0.07)
        result = f"VP9_encoded_{frame}"
        self.encoding_log.append(result)
        return result
    def choose_encoder(self, frame):
        bandwidth = random.uniform(1.0, 20.0)
        if bandwidth < 6:
            self.selected_format = "H.264"
```

```
            return self.encode_h264(frame)
        elif bandwidth < 12:
            self.selected_format = "H.265"
            return self.encode_h265(frame)
        else:
            self.selected_format = "VP9"
            return self.encode_vp9(frame)
    def batch_encode(self, frames):
        encoded_frames = []
        for frame in frames:
            encoded = self.choose_encoder(frame)
            encoded_frames.append(encoded)
        return encoded_frames
    def get_encoding_log(self):
        return self.encoding_log

class AdaptiveErrorRecovery:
    def __init__(self):
        self.error_log = []
        self.recovery_log = []
    def detect_error(self, frame):
        if "error" in frame:
            return True
        if random.random() < 0.1:
            return True
        return False
    def recover_frame(self, frame):
        recovered = frame.replace("error", "recovered")
        self.recovery_log.append((frame, recovered))
        return recovered
    def process_frame(self, frame):
        if self.detect_error(frame):
            self.error_log.append((time.time(), frame))
            frame = self.recover_frame(frame)
        return frame
    def process_frames(self, frames):
        processed_frames = []
        for frame in frames:
            processed_frames.append(self.process_frame(frame))
        return processed_frames
    def get_error_log(self):
        return self.error_log
    def get_recovery_log(self):
        return self.recovery_log

class InteractiveUserInterface:
    def __init__(self):
        self.running = True
        self.commands = {
            "1": self.show_quality,
            "2": self.start_encoder,
            "3": self.recover_error,
            "4": self.exit_ui
        }
        self.quality_module = DynamicVideoQuality()
        self.encoder_module = MultiFormatEncoder()
        self.error_module = AdaptiveErrorRecovery()
    def show_menu(self):
        print("Menu:")
        print("1. Show Current Video Quality")
        print("2. Encode Sample Frame")
        print("3. Process Frame for Error Recovery")
        print("4. Exit")
```

```
    def show_quality(self):
        quality = self.quality_module.adjust_quality()
        print(f"Current Quality: {quality}")
    def start_encoder(self):
        sample_frame = f"frame_{random.randint(1,1000)}"
        encoded = self.encoder_module.choose_encoder(sample_frame)
        print(f"Encoded Frame: {encoded}")
    def recover_error(self):
        sample_frame = f"frame_error_{random.randint(1,1000)}"
        processed = self.error_module.process_frame(sample_frame)
        print(f"Processed Frame: {processed}")
    def exit_ui(self):
        self.running = False
        print("Exiting UI")
    def start_ui(self):
        while self.running:
            self.show_menu()
            choice = input("Enter command: ")
            if choice in self.commands:
                self.commands[choice]()
            else:
                print("Invalid command")
            time.sleep(0.1)
    def simulate_ui_interaction(self, commands_list):
        for cmd in commands_list:
            if cmd in self.commands:
                self.commands[cmd]()
            time.sleep(0.1)
        self.running = False

class DRMSystem:
    def __init__(self):
        self.licenses = {}
        self.user_keys = {}
        self.drm_log = []
    def generate_license(self, user_id):
        license_key = f"LICENSE_{random.randint(10000,99999)}_{user_id}"
        self.licenses[user_id] = license_key
        self.drm_log.append((user_id, license_key))
        return license_key
    def validate_license(self, user_id, license_key):
        valid = self.licenses.get(user_id) == license_key
        self.drm_log.append((user_id, license_key, valid))
        return valid
    def encrypt_key(self, key):
        encrypted = ''.join([chr((ord(c) + 7) % 256) for c in key])
        self.drm_log.append(("encrypt", key, encrypted))
        return encrypted
    def decrypt_key(self, encrypted_key):
        decrypted = ''.join([chr((ord(c) - 7) % 256) for c in encrypted_key])
        self.drm_log.append(("decrypt", encrypted_key, decrypted))
        return decrypted
    def assign_user_key(self, user_id, key):
        encrypted_key = self.encrypt_key(key)
        self.user_keys[user_id] = encrypted_key
        return encrypted_key
    def validate_user_key(self, user_id, key):
        encrypted = self.user_keys.get(user_id, "")
        decrypted = self.decrypt_key(encrypted)
        valid = decrypted == key
        self.drm_log.append((user_id, key, valid))
        return valid
    def renew_license(self, user_id):
```

```
            new_license = self.generate_license(user_id)
            self.drm_log.append((user_id, new_license, "renewed"))
            return new_license
        def get_drm_log(self):
            return self.drm_log

def main():
    d_quality = DynamicVideoQuality()
    quality_changes = d_quality.simulate_streaming(3)
    print("Quality Changes:", quality_changes)
    m_encoder = MultiFormatEncoder()
    frames = [f"frame_{i}" for i in range(10)]
    encoded_frames = m_encoder.batch_encode(frames)
    print("Encoded Frames:", encoded_frames)
    a_error = AdaptiveErrorRecovery()
    sample_frames = [f"frame_{i}" for i in range(10)]
    sample_frames.append("frame_error_extra")
    processed_frames = a_error.process_frames(sample_frames)
    print("Processed Frames:", processed_frames)
    ui = InteractiveUserInterface()
    def ui_thread():
        ui.simulate_ui_interaction(["1", "2", "3", "4"])
    t = threading.Thread(target=ui_thread)
    t.start()
    t.join()
    drm = DRMSystem()
    license1 = drm.generate_license("user1")
    valid1 = drm.validate_license("user1", license1)
    key_encrypted = drm.assign_user_key("user1", "UserSecretKey")
    valid_key = drm.validate_user_key("user1", "UserSecretKey")
    renewed_license = drm.renew_license("user1")
    print("DRM License:", license1, valid1, key_encrypted, valid_key,
renewed_license)
if __name__ == "__main__":
    main()
```

Файл buffers.py

```
from collections import deque
from io import BytesIO
from threading import Event, Lock


class Chunk(BytesIO):
    """A single chunk, part of the buffer."""

    def __init__(self, buf):
        self.length = len(buf)
        BytesIO.__init__(self, buf)

    @property
    def empty(self):
        return self.tell() == self.length


class Buffer:
    """Simple buffer for use in single-threaded consumer/filler.

    Stores chunks in a deque to avoid inefficient reallocating
    of large buffers.
    """

    def __init__(self):
        self.chunks = deque()
        self.current_chunk = None
        self.closed = False
        self.length = 0
        self.written_once = False

    def _iterate_chunks(self, size):
        bytes_left = size

        while bytes_left:
            try:
                current_chunk = self.current_chunk or
Chunk(self.chunks.popleft())
            except IndexError:
                break

            data = current_chunk.read(bytes_left)
            bytes_left -= len(data)

            if current_chunk.empty:
                self.current_chunk = None
            else:
                self.current_chunk = current_chunk

            yield data

    def write(self, data):
        if not self.closed:
            data = bytes(data)  # Copy so that original buffer may be reused
            self.chunks.append(data)
            self.length += len(data)
            self.written_once = True

    def read(self, size=-1):
        if size < 0 or size > self.length:
            size = self.length
```

```
        if not size:
            return b""

        data = b"".join(self._iterate_chunks(size))
        self.length -= len(data)

        return data

    def close(self):
        self.closed = True


class RingBuffer(Buffer):
    """Circular buffer for use in multi-threaded consumer/filler."""

    def __init__(self, size=8192 * 4):
        Buffer.__init__(self)

        self.buffer_size = size
        self.buffer_lock = Lock()

        self.event_free = Event()
        self.event_free.set()
        self.event_used = Event()

    def _check_events(self):
        if self.length > 0:
            self.event_used.set()
        else:
            self.event_used.clear()

        if self.is_full:
            self.event_free.clear()
        else:
            self.event_free.set()

    def _read(self, size=-1):
        with self.buffer_lock:
            data = Buffer.read(self, size)

            self._check_events()

        return data

    def read(self, size=-1, block=True, timeout=None):
        if block and not self.closed:
            if not self.event_used.wait(timeout) and self.length == 0:
                raise OSError("Read timeout")

        return self._read(size)

    def write(self, data):
        if self.closed:
            return

        data_left = len(data)
        data_total = len(data)

        while data_left > 0:
            self.event_free.wait()

            if self.closed:
```

```
                return

            with self.buffer_lock:
                write_len = min(self.free, data_left)
                written = data_total - data_left

                Buffer.write(self, data[written : written + write_len])
                data_left -= write_len

                self._check_events()

    def resize(self, size):
        with self.buffer_lock:
            self.buffer_size = size

            self._check_events()

    def wait_free(self, timeout=None):
        return self.event_free.wait(timeout)

    def wait_used(self, timeout=None):
        return self.event_used.wait(timeout)

    def close(self):
        Buffer.close(self)

        # Make sure we don't let a .write() and .read() block forever
        self.event_free.set()
        self.event_used.set()

    @property
    def free(self):
        return max(self.buffer_size - self.length, 0)

    @property
    def is_full(self):
        return self.free == 0


__all__ = ["Buffer", "RingBuffer"]
```