Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Метод вирівнювання відтворення зображення під час передавання відеотрафіку з використанням емуляції IP-камери»

Виконав: студент 2-го курсу, групи 1КІ-21м
спеціальності 123 — Комп'ютерна інженерія
_____ Вишневський А. В.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. ОТ
_____ Азарова А. О.
(прізвище та ініціали)
«20» 12 2022 р.

Опонент:
_____ Карпінець В.В.
(прізвище та ініціали)
«21» 12 2022 р.

Допущено до захисту
Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О.Д.
(прізвище та ініціали)
«22» 12 2022 р

Вінниця 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень — магістр
Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
Азаров О. Д.
«15» 09 2022 р

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Вишневському Андрію Вікторовичу
(прізвище, ім'я, по-батькові)

1 Тема роботи: «Метод вирівнювання відтворення зображення під час передачі відеотрафіку з використанням емуляції IP-камери»

Керівник роботи: к.т.н., професор кафедри ОТ Азарова Анжеліка Олексіївна затверджені наказом Вінницького національного технічного університету від 15.09.2022 року № 205-А.

2 Строк подання студентом роботи 10.12.2022.

3 Вихідні дані до роботи: технічний опис програмного застосунку, мова програмування C++, середовище розробки CLion.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Аналіз сучасних методів та засобів обробки та передачі даних у системі відеоспостереження, Розробка структури та алгоритмів роботи системи відеоспостереження, Програмна реалізація системи, Тестування системи та її верифікація, Економічна частина.

5 Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): Лістинг RTSP-компоненту, лістинг ONVIF-компоненту, блок-схема алгоритму вирівнювання відтворення зображення, діаграма класів RTSP-модулю, діаграма класів ONVIF-модулю.

6 Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3,4	**Азарова А.О., к.т.н., професор каф. ОТ**		
5	**Небава М.І., к.е.н., професор каф. ЕПВМ**	Небава	Небава

7 Дата видачі завдання_____

8 Календарний план

№ з/п	Назва етапів виконання бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	04.10.2022	Виконано
2	Аналіз та дослідження методів побудови систем відеоспостереження, її переваг та недоліків.	29.10.2022	Виконано
3	Розробка структури та алгоритмів роботи системи відеоспостереження.	04.11.2022	Виконано
4	Програмна реалізація програмного додатку вирівнювання відтворення зображення та засобу комунікації з відеокамерою.	21.11.2022	Виконано
5	Підготовка матеріалів та опис розробки системи відеоспостереження.	30.11.2022	Виконано
6	Оформлення пояснювальної записки	03.12.2022	Виконано
7	Перевірка якості виконання магістерської роботи	10.12.2022	Виконано

Студент _____ Вишневський А.В.
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Азарова А.О.
(підпис) (прізвище та ініціали)

Консультант з економічної частини _____ Небава М. І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.4

Вишневський А. В. Метод вирівнювання відтворення зображення під час передачі відеотрафіку з використанням емуляції IP-камери. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2022. 124 с.

Укр. мовою. Бібліогр.: 26 назв; рис. 18; табл.: 6.

В магістерській кваліфікаційній роботі проаналізовано та досліджено існуючі механізми роботи систем відеоспостереження, можливі проблеми при передачі та відтворенні відеопотоку. Розглянуто основні технології передачі відеоданих та протоколи комунікації з IP-камерою. Розроблено алгоритм вирівнювання відтворення зображення та метод інтеграції програми з протоколом ONVIF.

Розроблено програмний додаток для ретрансляції відеопотоку, коригування та передачі комунікативних повідомлень IP-камери для інтеграції методу вирівнювання відтворення зображення та збереження сумісності з VMS.

Ключові слова: відео, спостереження, VMS, ONVIF.

ABSTRACT

UDC 004.4

Vishnevskyi A.V. The method of equalizing image playback during video traffic transmission using IP camera emulation. Master's thesis on specialty 123 — Computer engineering, educational program — computer engineering. Vinnytsia: VNTU, 2022.

Ukraine language Bibliography: 26 titles; Fig. 18; tab.: 6.

The master's thesis analyzed and researched the existing mechanisms of video surveillance systems, possible problems in the transmission and playback of the video stream. The main video data transfer technologies and communication protocols with IP cameras are considered. An image reproduction alignment algorithm and a program integration method with the ONVIF protocol have been developed.

A software application was developed to relay the video stream, adjust and communicate IP camera messages to integrate the image playback equalization method and maintain VMS compatibility.

Keywords: video, surveillance, VMS, ONVIF.

ЗМІСТ

ВСТУП

В умовах ведення війни проти російських агресорів актуальність використання систем відеоспостереження зростає з кожним днем, що спричинює збільшення обсягу трафіку, що має бути контрольованим. На сучасному ринку є безліч різноманітних пристроїв відеофіксації. У великих системах відеоспостереження особливої популярності набувають IP-відеокамери. IP-відеонагляд базується на цифровій IP-платформі та належить до систем безпеки, що можуть складатися з низки пристроїв. При цьому IP-пристрої підтримують підключення на відстані за допомогою різних гаджетів, зокрема, планшета, смартфона, ноутбука, ПК тощо [1].

У багатьох сферах наукових досліджень та розробок, пов’язаних з обробкою та передаванням даних було зроблено значний вклад в розвиток систем відеоспостереження завдяки численним працям наукових робітників, а саме Шульцрінне Х., Філдінг Р., Тім Бернерс-Лі, Оссо Р. як закордонні науковці, та Захарченко С.М., Крупельницький Л.В., Мордвинцев М.В., Хлєстков О.В. як вітчизняні науковці[2]—[10].

Сучасні системи відеоспостереження можуть містити сотні або тисячі відеокамер, тому зростає ризик втратити зв’язок із будь-якою з них, що спричиняє додаткові часові витрати на з’ясування причини такої події.

Отже, для покращення контролю над ситуацією з кожним днем вдосконалюють існуючі технології збирання, запису, передавання та оброблення даних, проте не всі реалізації дозволяють на високому рівні підтримувати плавність відтворення. Причинами можуть бути тонкощі реалізації протоколів передавання даних чи високорівневість прикладних програмних інтерфейсів та фреймворків, що не дають доступу до його прямої реалізації. Також існує проблема стандартизації, коли не кожна відеокамера здатна комунікувати з певною клієнтською програмою.

Нестійке мобільне мережеве з’єднання та великий обсяг використання Інтернет-трафіку також вказують на потребу у вирівнюванні відтворення

зображення для подальшого вдосконалення методів оброблення даних, автоматичного аналізу та управління передаванням відеоданих, що є вельми **актуальною задачею**.

Метою дослідження магістерської роботи є вдосконалення процесу передавання відео-трафіку із використанням методу вирівнювання відтворення зображення та розширення функціональних можливостей системи відео-спостереження.

Для досягнення поставленої мети було поставлено і вирішено такі **задачі**:

— проаналізовано засоби розроблення ПЗ та протоколів передавання використовуваних даних;

— розроблено алгоритм роботи програми-вирівнювача;

— проаналізовано функції та структурну організацію ПЗ;

— проведено тестування розробленого програмного забезпечення щодо наявності помилок під час роботи.

Об’єкт дослідження — процес вирівнювання відтворення відео-трафіку.

Предмет дослідження — методи оброблення, конвертації та передавання відеоданих засобами мережевого програмування.

У магістерській роботі використовувалися такі **методи дослідження**: методи стиснення даних — для оптимізації процесу передавання трафіку, контейнерування даних із допоміжною інформацією, виявлення перевантаження мережі; метод комп’ютерного моделювання засобами об’єктно-орієнтованого програмування — із метою розроблення відповідного ПЗ мовою C++.

Практичне значення отриманих результатів полягає в розробленні відповідного ПЗ, що уможливлює вирівнювання зображення та підтримку сумісності системи з протоколом ONVIF.

Наукова новизна отриманих результатів магістерської роботи полягає у тому, що:

— вдосконалено метод передавання відеоданих на основі додаткової попередньої буферизації, що, на відміну від існуючих підходів, дозволяє підвищити ефективність зберігання плавності зображення;

— вдосконалено засіб комунікації з клієнтським програмним забезпеченням засобами інтеграції з протоколом ONVIF, що, на відміну від існуючих підходів, дозволяє розширити сумісність ПЗ зі сторонніми програмами.

Публікація. За результатами дослідження, проведеного в магістерській роботі, було опубліковано тези доповіді[11].

Апробація результатів магістерської роботи відбулася на LI науково-технічній конференції підрозділів Вінницького національного технічного університету (м. Вінниця, 2022 р.).

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ЗАСОБІВ ОБРОБКИ ТА ПЕРЕДАЧІ ДАНИХ У СИСТЕМАХ ВІДЕОСПОСТЕРЕЖЕННЯ

1.1 Обгрунтування ролі та важливості якісних та зручних систем відеоспостереження

Під терміном відео розуміють широкий спектр технологій відтворення, обробки, передачі, запису й зберігання візуального матеріалу на моніторах. У побутовому значенні відео означає відеоматеріал, телесигнал або кінофільм. Іншими словами відео — це набір зображень, які змінюються з певною частотою, яка має назву FPS (Кількість кадрів за секунду)

Відео характеризується наступними параметрами:

— кількість кадрів на секунду;

— розгортка;

— роздільна здатність;

— співвідношення сторін екрану;

— кількість кольорів і кольорова розрядність;

— бітова швидкість або ширина відеопотоку;

— оцінка якості відео.

Метою створення систем стеження за потоковим відео є в основному захист рухомих і нерухомих об'єктів, хоча це не єдина можливість використання таких систем.

Сьогодні відеомовлення має величезний спектр застосувань у різних сферах діяльності, від охорони парковок до трансляції відеоданих із супутників і Марса. Такі системи зробили людство гігантським кроком вперед у різних галузях науки і досі допомагають щодня вивчати багато біологічних, хімічних і фізичних процесів.

Зручність систем робить їх доступними практично кожному жителю планети, оскільки для цього не потрібне дороге обладнання.

Тому з призначення відеопотоку можна зробити висновки про його призначення. Основною метою систем відеотранслювання є:

— якість та легкість у використанні;

— швидкість роботи системи;

— автономність роботи, мінімізація втрат даних.

Для швидкої обробки даних потрібно використовувати сучасні способи кодування цифрової інформації. Найбільш ефективним є спосіб «стискання» даних.

Для швидкої обробки даних потрібні сучасні методи кодування цифрової інформації. Найефективнішим способом є «стиснення» даних. Найпоширенішими методами майже у всіх сучасних техніках кодування відеоданих є компресори (кодеки) і контейнери h263 / h264 / h265, які можуть використовуватися для передачі не тільки відеопотоків, але й аудіопотоків: AMV / AVI / RIFF.

Для передачі даних у мережі використовуються протоколи TCP та/або UDP. Для подальшої синхронізації відеопотоків цих протоколів був створений протокол RTSP (Real Time Streaming Protocol), а пізніше RTMP.

Для передачі відео та аудіо даних по протоколу RTSP створено досить багато різних серверних і клієнтських частин. Наприклад, такі клієнтські плеєри, як VLC, Microsoft Milestone VMS і FFplay, а також серверні частини, найвідоміші з яких GStreamer, LibLive555 і FFServer. Незважаючи на те, що розробка деяких модулів велася кілька років, навіть десятиліть, кожен з них все ще має свої недоліки.

У такій системі дуже важлива кросплатформна підтримка, адже клієнту буде зручно використовувати, наприклад, домашній ПК як сервер, а смартфон як клієнт. Сьогодні будь-хто може створити малопотужну, але все ще ефективну систему відеоспостереження, використовуючи лише веб-камеру, комп’ютер і безкоштовне програмне забезпечення.

Хоча майже всі вже мають ресурси для такої системи, такого рішення ще недостатньо для використання на великих підприємствах. Тому були створені IP-камери (див рисунок 1.1).

Рисунок 1.1 — IP-камера

IP-камера — це цифрова камера для відеоспостереження, яка характеризується використанням протоколу IP для передачі відеопотоків у цифровому форматі в мережах Ethernet і Token Ring. Кожна IP-камера є мережевим пристроєм і має власну IP-адресу.

Крім того, ці камери мають власну операційну систему, засновану на ядрі Linux, часто з досить високими технічними характеристиками. Камера налаштовується за допомогою панелі керування в комплекті з операційною системою або через USB-з'єднання за допомогою програмного забезпечення виробника.

Оскільки кожна камера має та використовує власне обладнання, вона може дуже швидко працювати незалежно від інших камер.

У разі аварійного відключення електроенергії деякі камери можуть деякий час працювати автономно. Сьогодні такі камери поширені по всьому світу, де їх використовують у банках і магазинах, офісах і «розумних» будинках. У більшості випадків цього достатньо для користувача.

IP-камери також здебільшого мають вбудований режим день/ніч, який позбавляє користувачів від багатьох проблем, таких як: шум зображення та порушення передачі колірної гами при слабкому освітленні.

Для вирішення цієї проблеми виробники обладнання для відеоспостереження в охоронних цілях пропонують установку монохромних і кольорових камер. Ці камери знімають барвисті та високоякісні зображення від світанку до заходу.

1.2 Аналіз сучасних засобів створення систем відеотранслювання

Для створення найменшої системи відеоспостереження потрібно мати USB веб-камеру або IP-камеру, комп’ютер з операційною системою Linux або Windows, сервер для передачі відеопотоку та програмний засіб відтворення відеопотоку.

Аналіз VLC-програвача. VLC media player (див. рисунок 1.2) — безкоштовний кросплатформний медіапрогравач, розроблений проектом MediaLAN. Можливості плеєра є досить обширними, його можна використовувати не лише у якості медіапрогравача, а ще й як сервера для трансляції відео- та аудиопотоків.[12] Для функціонування програми не потрібно додатково інсталювати нічого, всі функції уже реалізовані у самій програмі. Програвач може відтворювати відеопоток із DVD диску, і потокове незашифроване відео та інтернет-радіо. Також програма має можливість записувати потокові аудіо- та відеодані на комп’ютер. VLC уміє програвати навіть пошкодженні файли — наприклад, з пошкодженими індексами.

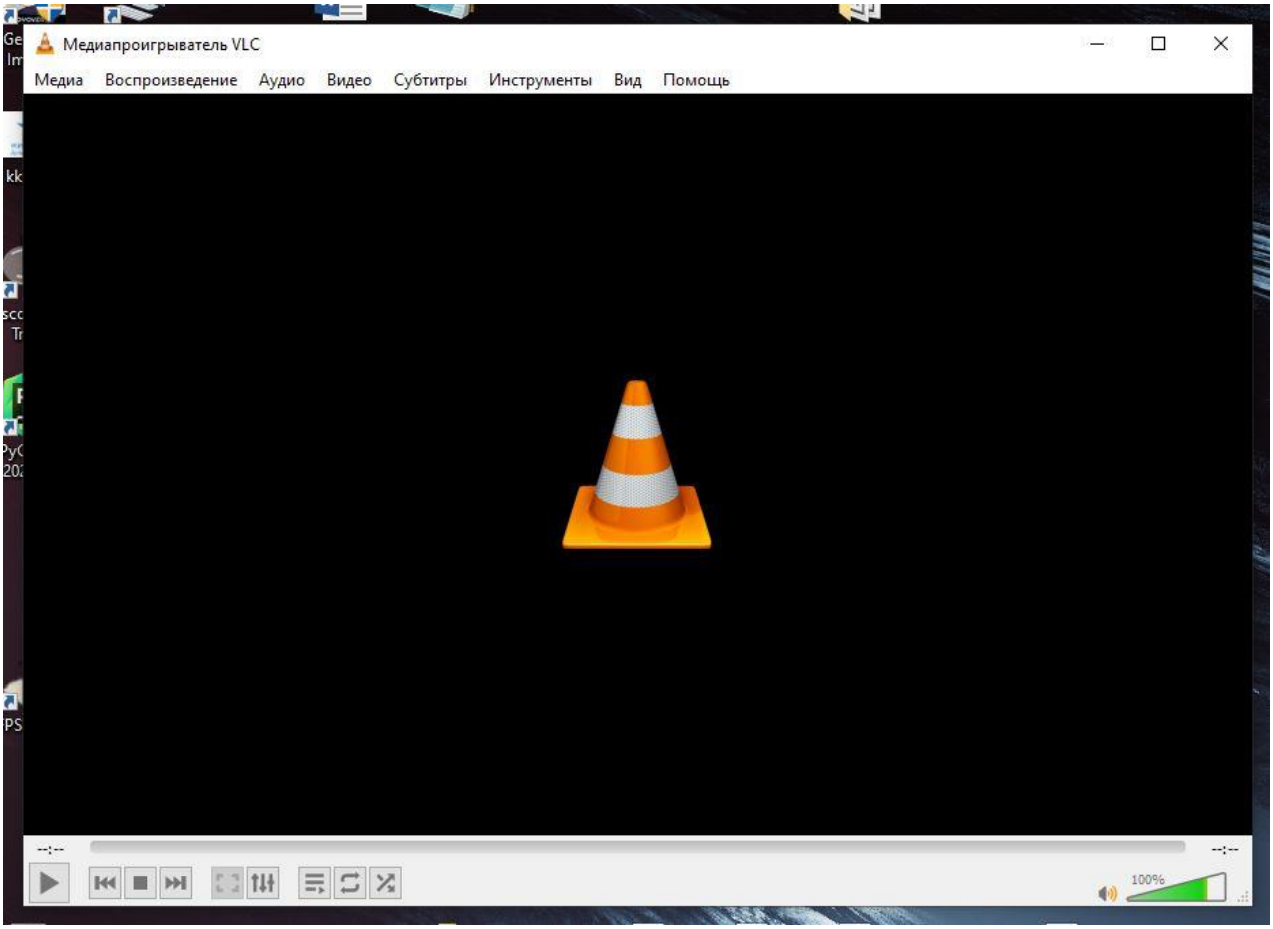

Рисунок 1.2 — VLC-програвач

Спочатку програвач був розроблений студентами паризького університету, але зараз над проектом VideoLAN працює The VLC Team і спільнота розробників, мешкаючих по всьому світі.

Для програвання уже існуючого відеотранслювання досить відкрити меню Media, обрати пункт Play і вказати протокол і лінк до трансляції. Все інше програма налаштує сама. Для більш детального налаштування потрібно відкрити налаштування програми, обрати пункт Video / Input, відкрити розширені налаштування та змінювати необхідні нам налаштування для програвання відеопотоку. [13]

VLC програвач можна використовувати не лише як клієнтську або серверну частину, досить часто його використовують як конвертер, за допомогою якого можливо змінити структуру файлу, конвертувати його у

другий тип кодування, змінити його розширення або видалити деякі потоки даних.

Переваги VLC-програвача:

— надійне програмне забезпечення;

— підтримка великої кількості кодеків та форматів;

— можливість відтворювати відео як з файлу, так і по посиланню.

Недоліки VLC-програвача:

— відсутність механізму перепідключення;

— не надає додаткових можливостей окрім програвання відео.

Аналіз системи керування відео VMS. Система керування відео VMS(Video Management System), також відома як програмне забезпечення для керування відео та сервер керування відео, є компонентом системи відеоспостереження, яка загалом:

— збирає відео з камер та інших джерел;

— записує/зберігає це відео на пристрої зберігання;

— надає інтерфейс як для перегляду відео в реальному часі, так і для доступу до записаного відео.

VMS може бути програмним компонентом мережевого відеореєстратора (NVR) і цифрового відеорекордера (DVR), хоча загалом VMS має тенденцію бути складнішим і надає більше опцій і можливостей, ніж комплектний пристрій NVR.

Через удосконалення технологій необхідно розрізняти VMS та вбудовані функції сучасних мережевих камер безпеки. Багато сучасних мережевих камер пропонують внутрішні можливості для запису та перегляду відео безпосередньо через веб-браузер і без використання VMS. Однак вбудований веб-інтерфейс камери, як правило, є ексклюзивним для самої камери та зазвичай не надає можливості спільного доступу для інших мережевих камер.

За бажанням VMS також може надавати додаткові функції та можливості, такі як розпізнавання людей, підтримку аудіо, аналітику і т.д.

Одним з популярних VMS є Hikcentral Proffessional, розроблений компанією, яка виготовляє відеокамери, Hikvision.

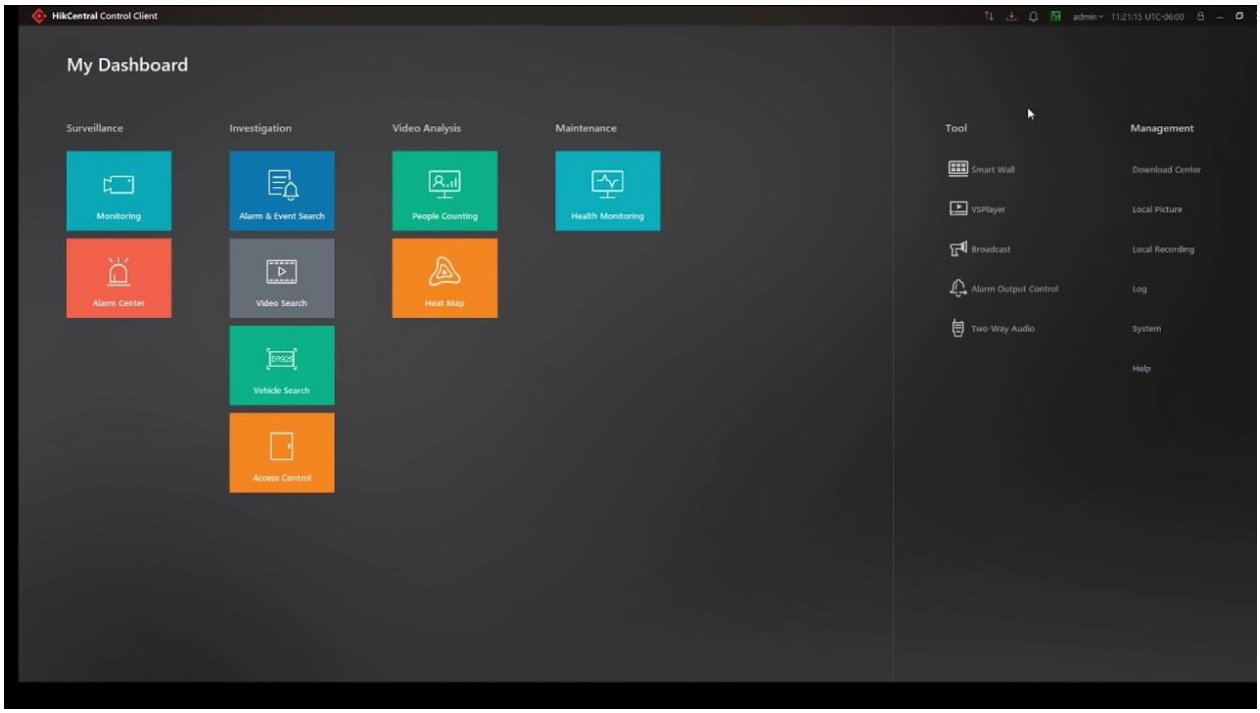

Рисунок 1.3 — Hikcentral Control Client

Для більш зручного використання Hikcentral Proffessional поділений на два компоненти: Hikcentral Control Client та Hikcentral Web Client[14]. Hikcentral Web Client призначений для обліку відеокамер, тобто їх додавання, авторизації, видалення та окремих налаштувань. До Hikcentral Web Client можливо додати камеру двома способами: або через пропрієтарний протокол ISUP, або через стандартизований протокол ONVIF. Додати камеру через RTSP посилання даний VMS можливості не надає. Hikcentral Control Client призначений для перегляду відеопотоку та керування вже доданими камерами.

Переваги використання VMS:

— спеціалізоване ПЗ для роботи з відеокамерами;
— надають додаткові можливості окрім відеотранслювання;
— мають механізм перепідключення до камери;
— надають можливість додати камеру різними способами.

Недоліки використання VMS:

— може виникати більше проблем при підключенні камери;

— можуть бути доволі об’ємними та мати високі системні вимоги.

Аналіз мультимедійного фреймворка GStreamer. GStreamer — мультимедійний фреймворк, написаний на мові програмування C з використанням системи типів GObject. GStreamer є «ядром» мультимедійних застосунків, таких як відеоредактори, потокові сервери, медіаплеєри і конвертери аудіо/відео файлів, VoIP-рішення[15]. У початковий дизайн закладена крос-платформовість; GStreamer працює на Unix-подібних системах, а також на Windows, OS/400 і Symbian OS. GStreamer надає прив'язки для інших мов програмування, таких як Python, C++, Perl, GNU Guile і Ruby. GStreamer є вільним програмним забезпеченням, з ліцензією GNU LGPL[16].

Серед базових можливостей GStreamer:

— локальне відтворення контенту, обробка потокового мовлення і програвання DVD;

— інтеграція з тулкіта для формування GUI-інтерфейсу (GTK+, Clutter);

— автоматичне визначення контейнерів і кодеків;

— функції вилучення метаданих;

— підтримка субтитрів;

— засоби для наочної візуалізації звукового потоку;

— підтримка перемикання на льоту між різними потоками зі звуком і субтитрами;

— переміщення всередині потоку за абсолютною позицією;

— режими уповільнення і прискорення відтворення, перемотування в зворотному порядку і покадровий перегляд;

— автоматичне усунення черезрядковості (дєінтерлейсинг), масштабування і установка колірного балансу;

— підтримка прокидання стисненого звуку;

— підтримка функцій рендерингу текстур бібліотеки Clutter.

Переваги фреймворку GStreamer:

— кросплатформеність та наявність реалізацій на багатьох мовах програмування;

— широкий функціонал.

Недоліки фреймворку GStreamer:

— погана та розмита документація;

— застарілість, відсутність оновлень.

Аналіз фреймворка FFmpeg. FFmpeg — це комплекс вільних комп'ютерних утиліт та програмних бібліотек для маніпуляцій з цифровими відео- та аудіо-матеріалами — запис, конвертація та пакування у різні формати контейнерів. Проект славиться наявністю різних аудіо та відео кодеків[17].

Цей проект складається із декількох компонентів:

Інструменти командного рядку:

— ffmpeg як програма командного рядка для конвертування одного формату відео у інший;

— ffserver як мультимедійний сервер трансляції HTTP і RTSP, що дозволяє здійснювати живі чи записані трансляції;

— ffplay як простий медіа програвач, який використовує в собі SDL і бібліотеки FFmpeg;

— ffprobe як інструмент командного рядка для зображення медіа інформації[12].

Бібліотеки:

— libswresample як бібліотека, що містить функції редискретизації аудіо;

— libavresample як бібліотека, що містить функції редискретизації аудіо з проекту Libav project, подібна до libswresample із бібліотеки ffmpeg;

— libavcodec як бібліотека, що містить усі власні аудіо/відео кодери і декодери FFmpeg;

— libavformat як бібліотека, що містить мультиплексори і демультиплексори для форматів контейнерів аудіо/відео;

— libavutil як допоміжна бібліотека, що містить функції спільні для різних частин FFmpeg;

— libpostproc як бібліотека, що містить старіші функції пост-обробки відео на базі h263;

— libswscale як бібліотека, що містить функції для масштабування зображення відео і конвертації колірного простору/формату пікселів;

— libavfilter є заміною для vhook, що дозволяє перевіряти чи модифікувати відео/аудіо між процесами декодування і кодування. Фільтри були портовані із багатьох проектів, включаючи MPlayer і avisynth[18].

Переваги фреймвоку FFMpeg:

— кросплатформеність;

— зручність у використанні;

— підтримка необхідних кодеків та протоколів;

Недоліки фреймворку FFMpeg:

— недостатньо детально описана документація;

— відсутність зворотньої сумісності з попередніми версіями фреймворку.

Аналіз недоліків та переваг протоколу RTSP та його похідних для створення системи відоспостереження. RTSP — мережевий протокол розроблений IETF в 1998 році і описаний в RFC 2326, є прикладним протоколом, призначеним для використання в системах, що працюють з мультимедіа даними, і що дозволяє клієнтові віддалено управляти потоком даних з сервера, надаючи можливість виконання команд, таких як «Старт», «Стоп», а також доступу за часом до файлів, розташованих на сервері. RTSP не виконує стиску, а також не визначає метод інкапсуляції мультимедійних даних і транспортні протоколи[19]. Передача потокових даних сама по собі не є частиною протоколу RTSP. Більшість серверів RTSP використовують для цього стандартний транспортний протокол реального часу, що здійснює передачу аудіо- і відеоданих. RTSP 2.0 знаходиться в стадії розробки як заміна RTSP 1.0.

RTSP 2.0 заснований на RTSP 1.0, але не має зворотної сумісності з ним в своїй основній версії.

Більшість систем відеоспостереження для своєї реалізації використовують RTSP(Real Time Streaming Protocol) протокол, але він не передає і не стискає дані самостійно [20]. RTSP протокол у свою чергу використовує RTP(Real-time Transport Protocol) у поєднанні з RTCP(Real-Time Transport Control Protocol) протоколами для передачі відеопотоку, проте деякі виробники реалізують власні транспортні протоколи.

RTSP з використанням RTP і RTCP дозволяє здійснення адаптації швидкості передавання. При всій своїй подібності до HTTP, RTSP визначає корисні керуючі послідовності в управлінні відтворенням мультимедіа. Використовується ідентифікатор при необхідності відстежувати одночасні сесії. Як HTTP, RTSP використовує TCP для підтримки з'єднання між кінцевими точками.

Список команд (методів):

— OPTIONS — запит підтримуваних методів;

— DESCRIBE — запит опису контенту, наприклад, у форматі SDP;

— PLAY — запит початку мовлення контента;

— PAUSE — запит тимчасової зупинки мовлення;

— RECORD — запит на записування контента сервером;

— REDIRECT — перенаправлення на інший контент;

— SETUP — запит установки транспортного механізму для медіа-контента;

— ANNOUNCE — оновлення даних опису контента;

— GET_PARAMETER — запит вказаних параметрів в сервера;

— SET_PARAMETER — установка параметрів сервера;

— TEARDOWN — зупинка потоку і звільнення ресурсів.

Протокол RTSP та його похідні є еталоном відеотранслювання, який широко використовується в подібних системах. Переваги даного стеку протоколів:

— стандартизований протокол, який має дуже детальну документацію;

— велика кількість реалізацій в різноманітних фреймворках;

— широке використання як в стандартних відеопрогравачах, так і в багатьох VMS;

З недоліків можна зазначити, що даний стек протоколів має недостатньо коректний механізм вирівнювання кадрів. При «застряганні» кадрів в мобільній мережі можуть виникати затримки, через що різниця між актуальними та наявними кадрами може зростати при недостатньо якісному з’єднанні. Також можна зазначити, що не всі VMS підтримують підключення безпосередньо через протокол RTSP, хоч і після підключення трансляція відбувається саме через даний стек протоколів.

Огляд протоколів ONVIF та PSIA. Для реалізації механізмів комунікації з IP-камерами буде розглянуто технології двох організації, які розробляють стандартизовані протоколи з однойменними назвами — ONVIF та PSIA.

ONVIF (Open Network Video Interface Forum) — галузева міжнародна організація, яка займається розробкою стандартизованих протоколів для взаємодії різного обладнання та програмних засобів, що входять до складу систем безпеки (IP-камер, IP-кодерів, відеореєстраторів, контролерів доступу тощо) .

Міжнародний форум ONVIF заснований компаніями Axis Communications, Bosch Security Systems та Sony у листопаді 2008 року з метою розробки та розповсюдження відкритого стандарту для систем мережевого відеоспостереження.

Станом на липень 2022 року кількість учасників форуму ONVIF перевищила 600 компаній. Першим постачальником систем управління відео із підтримкою стандартів ONVIF стала компанія Genetec[21].

Розробники ONVIF обрали найбільш готові технології та адаптували їх для IP-відеоспостереження. Зокрема специфікації ONVIF побудовані на сучасних веб-сервісах, що описуються мовою WSDL, протоколах RTP/RTSP, SOAP (XML), стандартах відеостиснення H.264, MPEG-4, MJPEG[22].

У межах специфікацій описи сукупності певних функцій за прикладним призначенням об'єднуються у профілі. Зокрема, станом на липень 2015 року розроблено Profile S (для відеоджерел), Profile C (для систем контролю та управління доступом), Profile G (для пристроїв запису відео) та Profile Q (вимоги щодо сумісності пристроїв "з коробки").

Наприклад, специфікації профілю S можуть визначати наступні аспекти взаємодії IP-камери із системами керування або відеозапису (DVR):

— конфігурування мережного інтерфейсу;

— виявлення пристроїв протоколу WS-Discovery;

— керування профілями роботи камери;

— налаштування потокової передачі медіа-даних;

— обробка подій;

— управління приводом PTZ (англ. Pan/Tilt/Zoom - панорамування/нахил/масштабування);

— захист (управління доступом, шифрування).

PSIA (Physical Security Interoperability Alliance) — це глобальний консорціум із понад 65 виробників засобів фізичної безпеки та системних інтеграторів, які зосереджені на сприянні сумісності пристроїв і систем безпеки з підтримкою IP у екосистемі фізичної безпеки, а також у системах автоматизації підприємств і будівель[23].

PSIA просуває та розробляє відкриті специфікації, пов’язані з мережевими технологіями фізичної безпеки, у всіх галузевих сегментах, включаючи відео, зберігання, аналітику, вторгнення та контроль доступу. Його робота аналогічна роботі груп і консорціумів, які розробили стандартизовані методи, які дозволяють різним типам обладнання безперебійно підключатися та обмінюватися даними, наприклад USB і Bluetooth.

PSIA і ONVIF були створені в 2008 році з різницею в кілька місяців для створення інтерфейсів на основі стандартів для апаратних і програмних платформ фізичної безпеки. Це дві групи, які переслідують одну і ту саму фундаментальну мету — забезпечення сумісності систем безпеки на основі IP. Групу ONVIF очолюють Axis Communications, Sony Corporation і Bosch Security Systems, наразі демонструючи 14 сумісних мережевих відеопродуктів від дев'яти компаній. Його членство зросло до 103 компаній, з 12 повноправними членами, 13 членами-учасниками та 78 користувачами. ONVIF випустив свій перший проект специфікацій у листопаді 2008 року[24].

Виробник камер спостереження Axis є дуже потужною силою на ринку IP-камер. Мало того, що третина усіх камер є камерами Axis, багато виробників та інтеграторів відчувають себе в безпеці, користуючись продуктами Axis.

Якщо обидві специфікації продовжать розвиватися, спонсорство ONVIF від Axis може стати потужним рушієм підтримки ONVIF. Хоча у PSIA є виробники з великими продажами аналогових камер (наприклад, Pelco), навряд чи це матиме таку ж силу серед лідерів IP-відеосистем.

У Європі та Азії ONVIF сприймають дуже прихильно (безсумнівно, через підтримку камер Axis, Bosch і Sony). Дійсно, серед багатьох глобальних зв'язків PSIA не знайомий і розглядається як «американець». Хоча це, безумовно, технічно неправильно, таке сприйняття, безумовно, є сильним фактором у створенні імпульсу для його специфікації.

PSIA виступає за розробку специфікацій не тільки для відео — контролю доступу, зберігання тощо, в той час коли ONVIF більш зосереджений на відео — камерах і аналітиці.

1.3 Варіативний вибір засобів для створення системи відеоспостереження під час реалізації задач дипломної роботи

Задля найоптимальнішого вибору засобів розробки системи відеоспостереження буде обрано необхідні фреймворки та протоколи, що найбільше підійдуть для вирішення поставленої задачі.

Для сумісності з відеопрогравачами та VMS буде обрано протокол RTSP та його похідні. Це обгрунтовано тим, що VMS здатні працювати через цей протокол, а засоби комунікації з відеокамерою все одно пов’язані з протоколом RTSP, та потребують його реалізації. Задля усунення недоліків будуть реалізовані та інтегровані додаткові механізми синхронізації відео, що посприяє вирішенню поставлених задач.

Застосунок може бути реалізований на базі фреймворку GStreamer або FFmpeg, вибір буде відбуватись за наступними критеріями:

— наявність достатнього функціоналу для вирішення поставлених задач;

— кросплатформеність;

— зручність у використанні;

— детальність документації;

— актуальність.

GStreamer має достатній функціонал для вирішення поставлених задач, також є кросплатформеним. Щодо зручності у використанні виникають певні сумніви, так як фреймворк є відносно застарілим та не має достатньо детально описану документацію.

До того ж, фреймворк не отримує нових оновлень та підтримується в основному лише спільнотою. Отож, GStreamer підходить для розробки, але його використання значно ускладнить розробку ПЗ та поставить під сумнів надійність отриманих результатів. Тому використання застарілого ПЗ не є пріоритетною задачею.

FFmpeg цілком підходить для вирішення поставлених задач, є кросплатформеним. Документація, у порівнянні з GStreamer, є більш детально описаною. Модульність FFmpeg здатна упростити розробку додатку, використовуючи та інтегруючи лише необхідний функціонал. Також фреймворк є більш відлагодженим та стабільним у роботі, має велику спільноту, широко використовується та не втрачає актуальність.

Протокол комунікації з відеокамерою буде обиратись за наступними критеріями:

— стандартизація;

— поширеність серед VMS;

— простота реалізації;

— наявлість реалізації протоколу в IP-камерах.

PSIA є стандартизованим протоколом, підходить для інтеграції з додатком, цінується через простоту його реалізації та невеликий розмір повідомлень. Але даний протокол є недостатньо поширеним.

Більшість VMS не надають можливості підключення камери через протокол PSIA, а найпоширеніші виробники відеокамер не інтегрують даний протокол в свої камери, що свідчить про те, що використання даного протоколу не вирішить належним чином проблему збереження сумісності із тороннім ПЗ.

ONVIF також є стандартизованим протоколом, для інтеграції з додатком надає достатній функціонал. Має дещо більш складну реалізацію, але це нівелюється тотальною поширеністю серед VMS та наявністю протоколу в проовідних виробників IP-камер. Отже, протокол ONVIF є єдиним варіантом, що може задовольнити потреби у вирішенні задач роботи.

Після аналізу перечислених засобів можна сформувати набір фреймворків та протоколів, які найкращим чином задаовольнять потреби під час розробки програмного забезпечення.

Найсучаснішим вибором фреймворку для створення даного ПЗ на даний момент часу є FFmpeg, на якому буде базуватись RTSP сервер та клієнт для передачі відео. GStreamer є морально застарілим та не має достатньо детальної документації, що значно ускладнить розробку програмного застосунку.

Серед протоколів комунікації з IP-камерами ONVIF є найпоширенішим, тому модуль комунікації буде реалізовано саме на цьому протоколі.

Протокол PSIA в деяких аспектах має ширші функіональні можливості та простіший в реалізації, але в нашому регіоні він не набув широкого використання та не має підтримки більшості відеокамер.

1.4 Висновки до розділу 1

У першому розділі магістерської кваліфікаційної роботи було обгрунтовано роль та важливість якісним систем відеоспостереження. Було наведено основну інформацію про характеристику відео, було наведено інформацію про IP-камеру, їх характеристики та особливості.

Було проведено огляд сучасних засобів розробки систем відеоспостереження, а саме:

— VLC-програвач;
— Hikcentral Control Client;
— мультимедійний фреймворк GStreamer;
— мультимедійний фреймворк FFMpeg;
— протокол відеотранслювання RTSP та його похідні;
— протокол комунікації ONVIF;
— протокол комунікації PSIA.

Також було наведено переваги та недоліки кожного з компонентів.

Для вирішення поставлених задач було проведено порівняльний аналіз оглянутих засобів розробки систем відеоспостереження та на основі цього аналізу було обрано фреймворк FFMpeg як актуальний засіб розробки у сфері відеопроцесингу, RTSP та пов'язані протоколи як оптимальний засіб передачі відеоданих, ONVIF як один з найбільш поширених стандартизованих протоколів комунікації з IP-камерами.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ

2.1 Розробка архітектури системи відеоспостереження

Система відеоспостереження поділятиметься на декілька складових:

— камера, яка безпосередньо знімає та кодує відеопотік;

— програма-посередник, яка отримує потокове відео з камери, накопичує кадри, забезпечує відеотранслювання та його вирівнювання, також підтримує комунікацію камери з клієнтом;

— клієнтська частина(VMS або відеопрогравач), яка отримує потокове відео для подальшого його відтворення.

Схему взаємодії складових системи наведено на рисунку 2.1

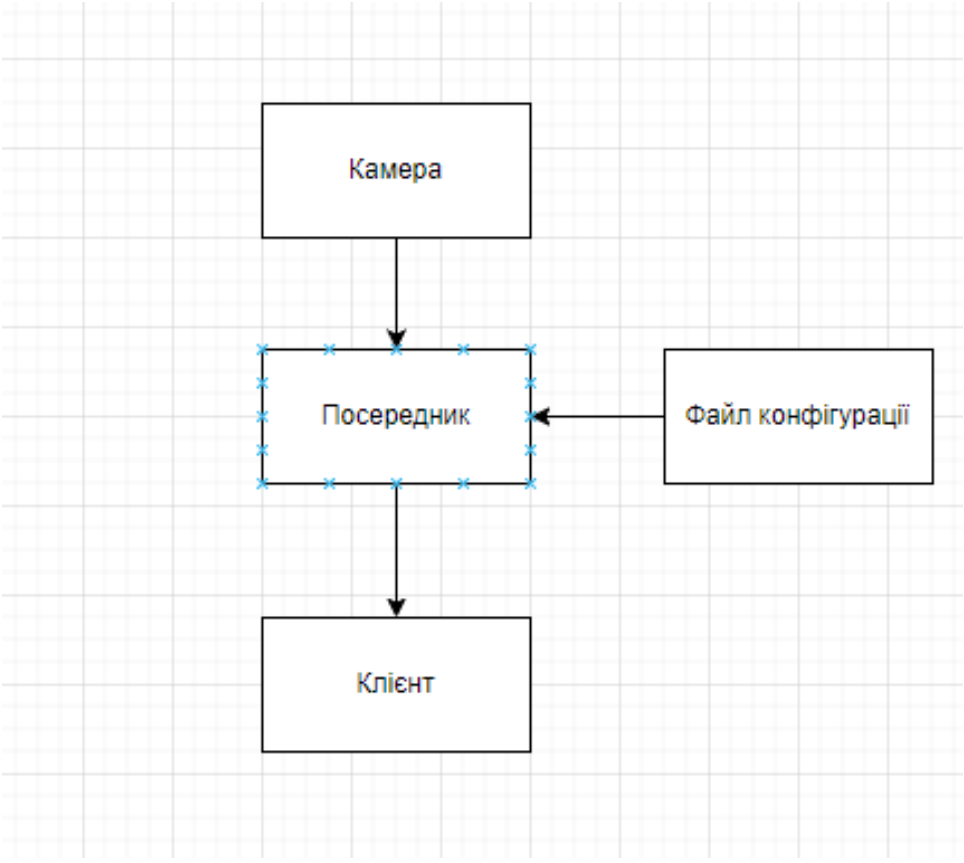

Рисунок 2.1 — Схема взаємодії компонентів

Програма-посередник буде складатись з RTSP-сервера, RTSP-клієнта та модуля ONVIF. RTSP-клієнт буде підключатись безпосередньо до камери та отримуватиме кадри для подальшого їх накопичення, RTSP-сервер буде приймати підключення від VMS та надсилати кадри. Модуль ONVIF виступатиме в ролі проксі-сервера для коректної взаємодії з камерою.

За допомогою конфігураційного файлу користувач може налаштовувати посилання на відеопотік, який потрібно транслювати, порт вихідного потоку та режим роботи модуля ONVIF за необхідності.

2.2 Розробка структури модулю RTSP

Для розробки RTSP-сервера та RTSP-клієнта будемо використовувати бібліотеку FFMpeg.

Налаштування бібліотеки FFMpeg проходить у декілька етапів:

— завантаження найновішої версії з офіційного сайту ffmpeg.org;

— встановлення необхідних залежностей;

— підготовка проекту до компіляції та безпосередня компіляція.

Використання бібліотеки FFMpeg дозволить зчитувати дані з камери та отримувати потрібні метадані про розмір кадрів, кількість кадрів за секунду, презентаційні мітки та інше.

Основною концепцією даних засобів є отримання кадрів з камери, буферизація визначеного об’єму даних та відправка кадрів на клієнт. Накопичення кадрів відбуватиметься постійно, як при первісній буферизації, так і при їх відправці. У випадку нестабільності мережі чи короткочасних перебоїв, тобто коли кадри з камери будуть отримуватись повільніше ніж потрібно, клієнт продовжуватиме брати кадри з резерву в необхідних обсягах, а резерв поповниться коли мережа стабілізується. Таким чином для клієнта картинка залишиться плавною, не зважаючи на перебої.

На рисунку 2.2 зображено алгоритм вирівнювання RTSP-компоненту.

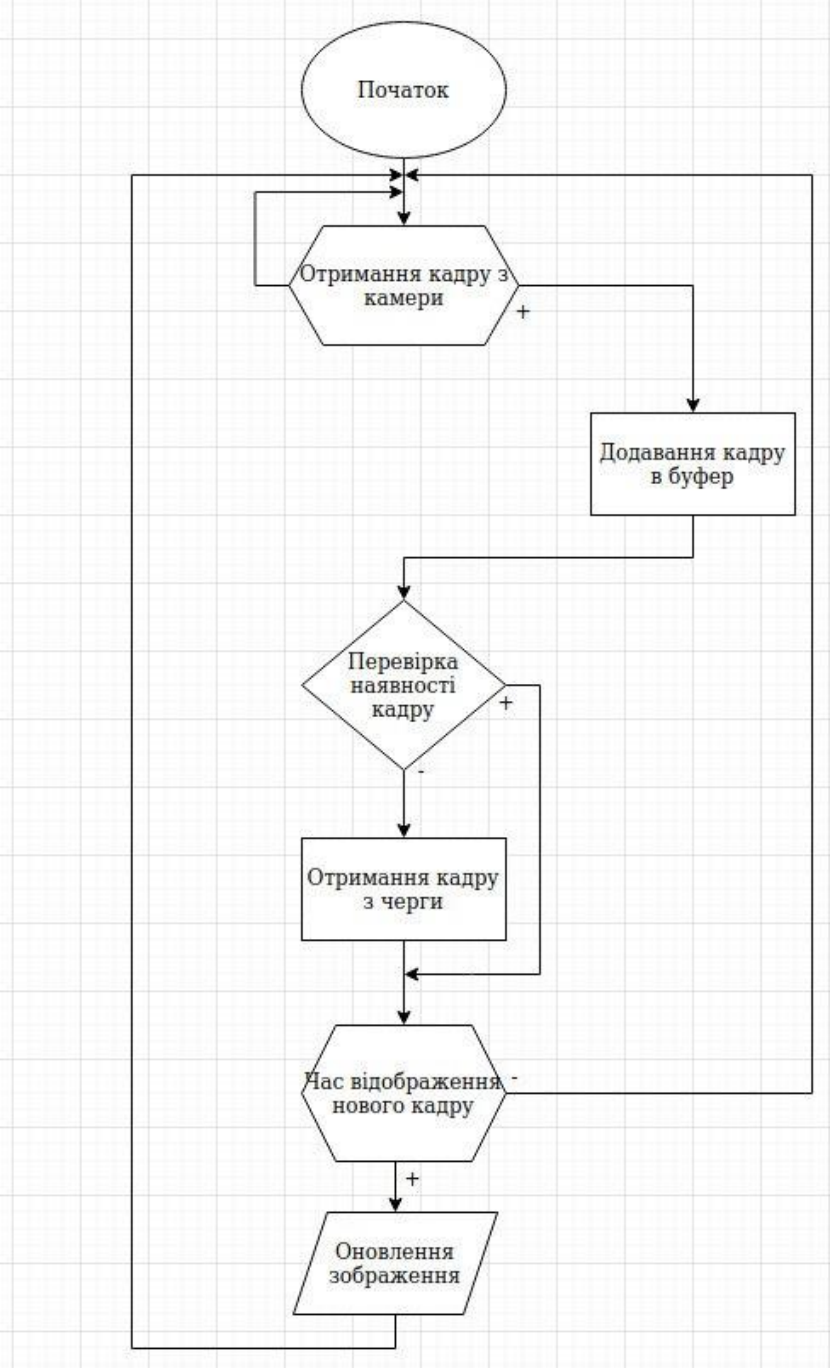

Рисунок 2.2 — Алгоритм вирівнювання RTSP-компоненту

Вирівнювання відтворення зображення може відбуватись завдяки презентаційних міток, значення тривалості відображення кадру та міток декодування. Кадри, що передаються по протоколу RTP, можуть містити тільки тривалість відображення кадру, презентаційні мітки можна сформувати самостійно, починаючи від нульового значення спочатку та додаючи тривалість

відображення при кожному новому кадрі. Таким чином ми матимемо всі потрібні дані для синхронізації відео.

Але лише буферизації може бути недостатньо, при критичних перебоях у зв’язку даний алгоритм може безкінечно застрягати на певному етапі, тому важливо враховувати випадки повного розриву з’єднання та необхідності перепідключення для стабілізації роботи системи.

На рисунку 2.3 зображено схему механізму перепідключення RTSP-компоненту.

Рисунок 2.3 — Схема механізму перепідключення RTSP-компоненту

Є чотири випадки коли RTSP-компонент буде перепідключатись до камери:

— кадр не був розпізнаний більше трьох разів;

— різниця між презентаційними мітками досягла значення більше припустимого;

— VMS ініціює розрив з’єднання;

— буфер вичерпує кадри до нуля.

Таким чином вирішується проблема необхідності перепідключення вручну, будучи спокійним за стабільність роботи застосунку.

Але залишається ще одна проблема, так як відеопакети передаються по мобільній мережі, деяка кількість кадрів може «застрягати» в мережі, через що для застосунку може виникнути випадок, що він отримує більше кадрів, ніж очікує. Щоб уникнути зайвої затримки між актуальними кадрами та тими, що програються, слід реалізувати логіку прискореного програвання, що зображено на рисунку 2.4.

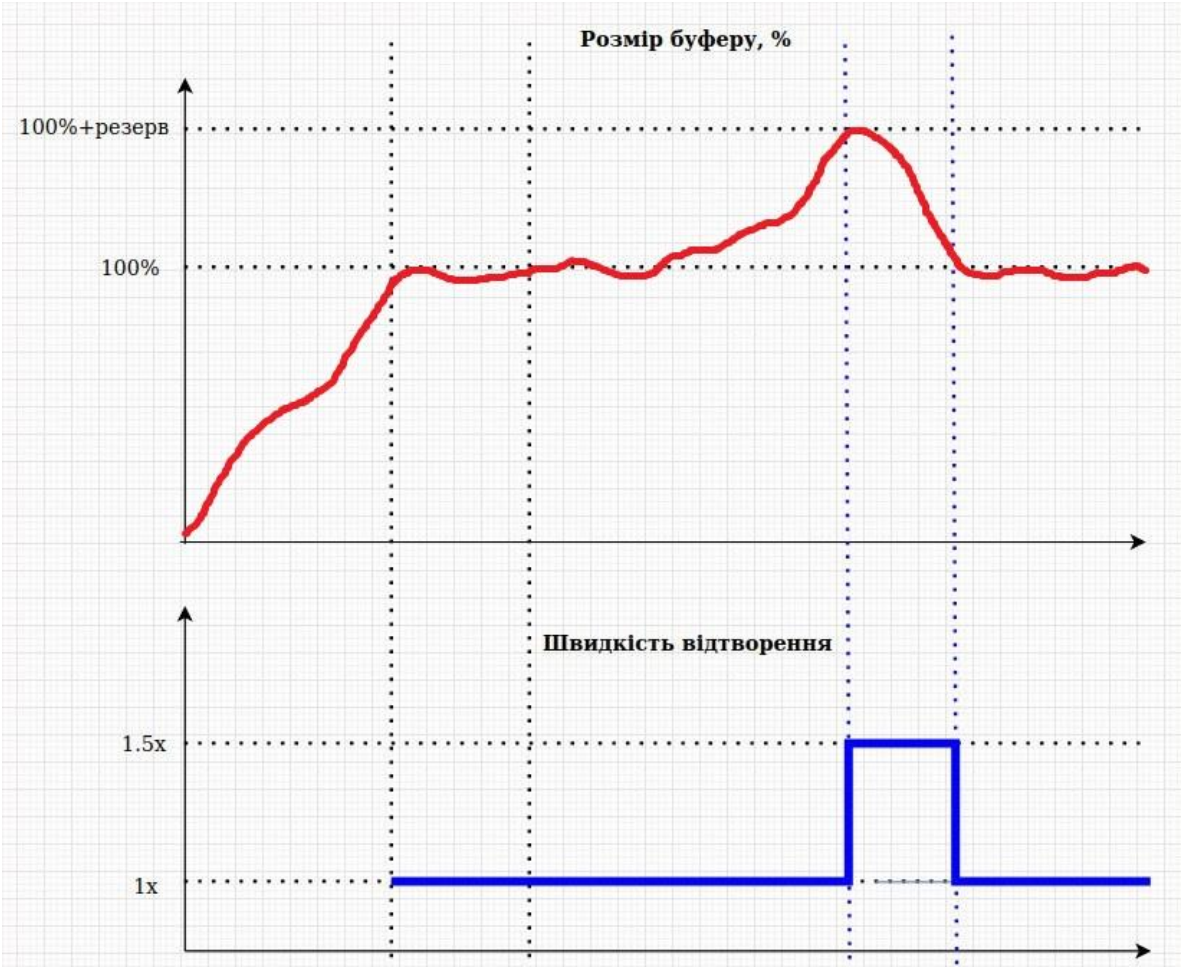

Рисунок 2.4 — Графік роботи механізму прискорення

Для цього окрім основного буферу потрібен ще один невеликий резервний буфер для зберігання «зайвих» кадрів. У випадку, коли цей буфер заповниться, спрацює механізм пришвидшення відтворення відеопотоку, коригуючи презентаційні мітки на менші значення до тих пір, поки резервний буфер не

спорожніє і тоді актуальність кадрів відновиться. Після нормалізації потоку застосунок продовжитиме працювати в штатному режимі.

2.3 Розробка структури модулю ONVIF

Оскільки не всі VMS програми здатні взаємодіяти з камерою безпосередньо по протоколу RTSP, потрібно розробити додатковий модуль, який даватиме можливість взаємодіяти з камерою по протоколу ONVIF, що забезпечить підтримку більшості VMS програм, а також розширить функціональні можливості камери.

ONVIF-компонент для VMS виступатиме в ролі емулятора IP-камери з підтримкою ONVIF протоколу. В даній ситуації немає безпосереднього прямого підключення камери до VMS, ONVIF-сервер камери стає недоступним, а реалізація власного серверу позбавить можливості користуватись додатковими функціями камери накшталт PTZ та отримувати актуальні метадані камери. Також в наповненні ONVIF повідомлень є метадані, що містять інформацію про IP-адреси камери, та посилання на відеопотік. Ця інформація є ключовою для роботи протоколу. Тому більш оптимальним рішенням буде реалізація емулятору, що здатний пересилати повідомлення ONVIF та водночас коригувати їх вміст для коректної роботи.

На рисунку 2.5 зображено схему взаємодії ONVIF-компоненту.

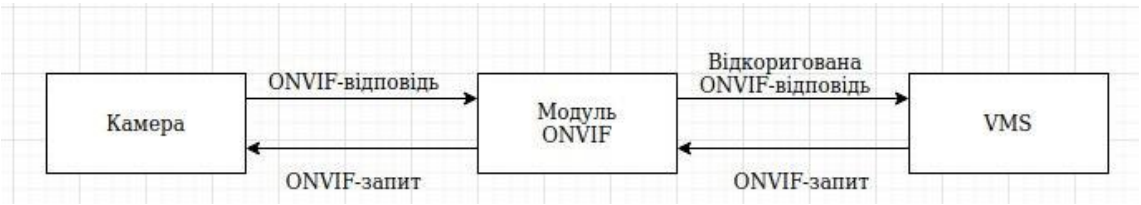

Рисунок 2.5 — Схема взаємодії ONVIF-компоненту

Одним із ONVIF-повідомлень, що потрібно коригувати це «GetCapabilitiesResponse», що є відповіддю на запит «GetCapabilities». На

рисунку 2.5 зображено фрагмент прикладу такого повідомлення отриманого за допомогою програми Wireshark.

Рисунок 2.6 — Фрагмент повідомлення «GetCapabilitiesResponse»

Дане повідомлення містить у своєму тілі посилання на ONVIF-сервіси, такі як «device_sevice», «media_service» та інші. Вкрай важливо, щоб посилання були валідними та актуальними, так як доступ до цих сервісів і відправка подальших запитів відбувається саме за цими посиланнями.

Також буде слід звернути увагу на повідомлення «GetNetworkInterfacesResponse», яке містить інформацію про IP-адресу камери та всі відповіді «ptz_service», за допомогою яких відбувається контроль над положенням камери.

З’являється проблема в тому, що IP-камера формує відповіді, вказуючи свою власну IP-адресу, а для поворотів камери необхідно гарантувати доставку повідомлень безпосередньо до самої камери. Окрім даних сервісів можуть ще бути васні від відеокамери, це також варто враховувати.

Тому емулятор має виконувати функцію проксі-серверу, пересилаючи всі ONVIF-повідомлення, що дасть змогу емулятору видавати себе за повноцінну IP-камеру. Маючи такий функціонал, програмний застосунок зможе коректно інтегруватись з відеокамерою.

2.4 Висновки до розділу 2

У другому розділі магістерської кваліфікаційної роботи було розроблено архітектуру системи відеоспостереження, на основі якої розробку було поділено на декілька компонентів та було побудовано схему взаємодії між ними.

Було розроблено алгоритм вирівнювання зображень, що відбуватиметься в RTSP-компоненті, спроектовано схему механізму перепідключення RTSP-компоненту, що сприятиме більш стабільній роботі системи. Також було розроблено механізм прискорення відтворення кадрів, що впливатиме на формування презентаційних міток та вирішуватиме проблему «застрягання» кадрів в мережі.

Вирівнювання кадрів стабілізує подачу кадрів до VMS, для якого складеться враження, що відеопотік є локальним. Механізм прискорення відтворення кадрів вирішить проблему нарощення зайвої затримки між актуальними кадрами та наявними. Механізм перепідключення забезпечить покращення безперебійності роботи системи відеоспостереження.

Окрім цього, було спроектовано схему взаємодії ONVIF-компоненту в системі та було вирішено, що ONVIF-компонент виступатиме в ролі емулятора відеокамери, який передаватиме через себе повідомлення, попередньо їх коригуючи.

Також було оглянуто особливості деяких елементів ONVIF-повідомлень та знайдено ті дані, які потребуватимуть корекції. Це дасть змогу ефективно користуватись протоколом, підтримуючи його функціонал.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Варіантний аналіз і обгрунтування вибору програмних засобів

Перед тим як починати реалізацію програмного забезпечення, потрібно оглянути доступні програмні засоби та обрати ті, які найкраще підійдуть для розробки даного програмного забезпечення.

Вибір програмних засобів буде відбуватись для наступних компонентів програмного засобу:

— компонент передачі відео;

— компонент комунікації з IP-камерою.

Для розробки компоненту передачі відео та компоненту комунікації з IP-камерою буде проведено огляд таких мов програмування, як C++, Python, Java та C#.

C++ — компільована мова програмування високого рівня, розроблена Б’ярне Страуструпом. Перша версія мови була випущена у 1979 році. Найактуальніша версія мови C++ 17 вийшла у 2017 році. C++ успадкувала синтаксис і основні аспекти мови C. По суті, C++ є надмножиною мови C, на що також вказує початкова назва — «C with classes» — «Сі з класами». Мова C++ підтримує парадигму об’єктно-орієнтованого програмування, характеризується високою швидкодією за рахунок того, що код програми компілюється у машинний код, а також має велику кількість сторонніх бібліотек, що спрощують процес розробки.

До недоліків мови відноситься відсутність автоматичного керування пам’яттю («збір сміття» — «garbage collection»), що є потенційно небезпечним, адже допускає існування помилок, що призводять до втрат пам’яті (явище, коли частина пам’яті, виділеної для виконання програми, не вивільняється після завершення її використання). Натомість, ручне керування пам’яттю надає розробнику більше контролю над роботою програми[25].

C# — мова програмування високого рівня, розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде в корпорації Microsoft. Перша версія мови була випущена у 2002 році. Найактуальніша версія мови C# 7.0 вийшла 7 березня 2017 року. Мова програмування C# була розроблена під впливом C++ та Java, має C-подібний синтаксис. Підтримує парадигму об’єктно-обієнтованого програмування.

Має дещо нижчу швидкодію, ніж мова C++, проте вищу, ніж Java. Характеризується наявністю великої кількості сторонніх бібліотек, автоматичного керування пам’яттю та вбудованою реалізацією деяких шаблонів проектування — наприклад, шаблон Спостерігач (Observer) реалізується мовною конструкцією event.

Java — мова програмування високого рівня, випущена компанією Sun Microsystems у 1995 році. Остання версія Java Standard Edition 10 була випущена 20 березня 2018 року. Він має C-подібний синтаксис. Підтримує парадигму об'єктно-орієнтованого програмування. Він трохи менш продуктивний, ніж мови C++ і C#. Характеризується наявністю великої кількості сторонніх бібліотек. Java використовує автоматичний збирач сміття для керування пам’яттю протягом життя об’єкта. Програміст вирішує, коли створити об’єкт, а віртуальна машина відповідає за звільнення пам’яті після того, як об’єкт більше не потрібен. Коли більше немає посилань на об’єкт, збирач сміття автоматично видаляє його з пам’яті.

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблено Гвідо ван Россумом у 1990 році. Розширені структури даних разом із динамічною семантикою та динамічним зв’язуванням роблять його привабливим для швидкої розробки додатків і як спосіб створення існуючих компонентів. Python підтримує модулі та пакети модулів, які сприяють модульності та повторному використанню коду. Мова програмування Python підтримує кілька парадигм програмування, включаючи: об’єктно-орієнтоване, процедурне, функціональне

та аспектно-орієнтоване. Python має ефективні високорівневі структури даних і простий, але ефективний підхід до об’єктно-орієнтованого програмування. Як інтерпретована, динамічно типізована мова, вона чудово підходить для створення сценаріїв і швидкої розробки додатків, але самі додатки працюватимуть повільніше, ніж ті, що написані на скомпільованих мовах програмування.

У результаті проведеного огляду для розробки було обрано мову C++, на якій будуть створюватись всі складові. Це обгрунтовано потужністю цієї мови програмування, можливістю достатньо глибоко доступатись до сокетів і підтримкою FFMpeg.

3.2 Вибір середовища розробки

Для більш зручної розробки програмних засобів використовуються IDE (інтегровані середовища розробки).

Інтегроване середовище розробки — це комплекс програмних засобів для розробки, що складається з текстового редактора коду, інструментів для компіляції та відлагодження проектів.

Для порівняння середовищ розробки для подальшого вибору буде розглянуто середовища JetBrains CLion, Microsoft Visual Studio та Visual Studio Code.

Visual Studio Code — текстовий редактор призначений для розробки застосунків для хмарних систем та веб-додатків. Visual Studio Code розповсюджується безкоштовно і є кросплатформеним, тобто підтримує всі сучасні операційні системи.

VS Code підтримує різноманітні плагіни, які з легістю можливо інтегрувати в нього, але він більше підходить для написання коду інтерпретованими мовами програмування, тому з VS Code буде не дуже зручно працювати.

Microsoft Visual Studio — інтегроване середовище розробки, що розробляється компанією Microsoft. Підтримує такі мови програмування як C, C++, C#, F#, Visual Basic та інші. Але дане середовище підходить тільки для операційної системи «Windows».

CLion — це IDE для розробки на мовах програмування C та C++, що розробляється компанією JetBrains. Також є кросплатформеним та підтримує всі сучасні операційні системи.

Середовище CLion надає змогу швидко та зручно розробляти систему, підтримує операційну систему Linux та є більш оптимізованим, гнучким та швидкодійним, саме тому було надано вибір даному середовищу для розробки програмного забезпечення на мові C++.

3.3 Схеми класів та програмні компоненти

Діаграма класів — це діаграма, яка надає інформацію про класи програми, їхні методи, поля та зв’язки між ними. Діаграми класів відіграють ключову роль в об'єктно-орієнтованому моделюванні. На діаграмі класи представлені в окремих клітинках, розділених на три частини:

Ім'я класу вказується в першому компоненті. Назви класів виділяються жирним шрифтом і вирівнюються по центру. Назви класів пишуться великими літерами.

Другий компонент визначає поля, які містить клас. Назви полів пишуться малими літерами та вирівнюються за лівим краєм.

Третя частина визначає методи класу. Вони також написані малими літерами та вирівняні за лівим краєм.

Мова UML надає механізми для представлення компонентів класу, тобто методів і полів, з додатковою інформацією, такою як видимість для інших компонентів[26].

Для вказання видимості компонентів класу робиться позначення, яке розташоване перед ім’ям компоненту:

— public (публічний) елемент позначається знаком «+»;

— private(приватний) позначається знаком «-».

Взаємозв’язок — це логічне відношень між сутностями, яке показують на діаграмах класів. В UML представлені наступні види відношень:

— залежність позначає таке відношення між класами, що зміна специфікації класу-постачальника може вплинути на роботу залежного класу, але не навпаки;

— асоціація показує, що об’єкти однієї сутності (класу) пов’язані з об’єктами іншої сутності;

— агрегація різновид асоціації при відношенні між цілим та його складовими;

— композиція як варіант агрегації, коли час існування екземплярів класу контейнера та його складових жорстко пов’язані;

— наслідування показує, що один з двох пов’язаних класів (підтип) є окремою формою другого класу (надтипу), який називається узагальненням першого.

На рисунку 3.1 наведено схему класів програмного модуля RTSP, а саме наслідування класів CVideoDecoder та RTSPServerSink від класу IRunnable. Завдяки IRunnable програмний код стає більш оптимізованим та сприяє униканню зайвих повторень коду. CVideoDecoder відповідає за ініціалізацію, кодування, декодування кадрів, а також їх накопичення. RTSPServerSink відповідає за формування та відправку вихідних відеопакетів, також оброблює протокольні запити.

IRunnable є інтерфейсом який дозволяє перегружати методи prepare() та work().

CVideoDecoder відповідає за декодування відео. Головними методами є OpenFile() та PushNewPacket().

RTSPServerSink відповідає з передачу відео. Головним методом є serverThread().

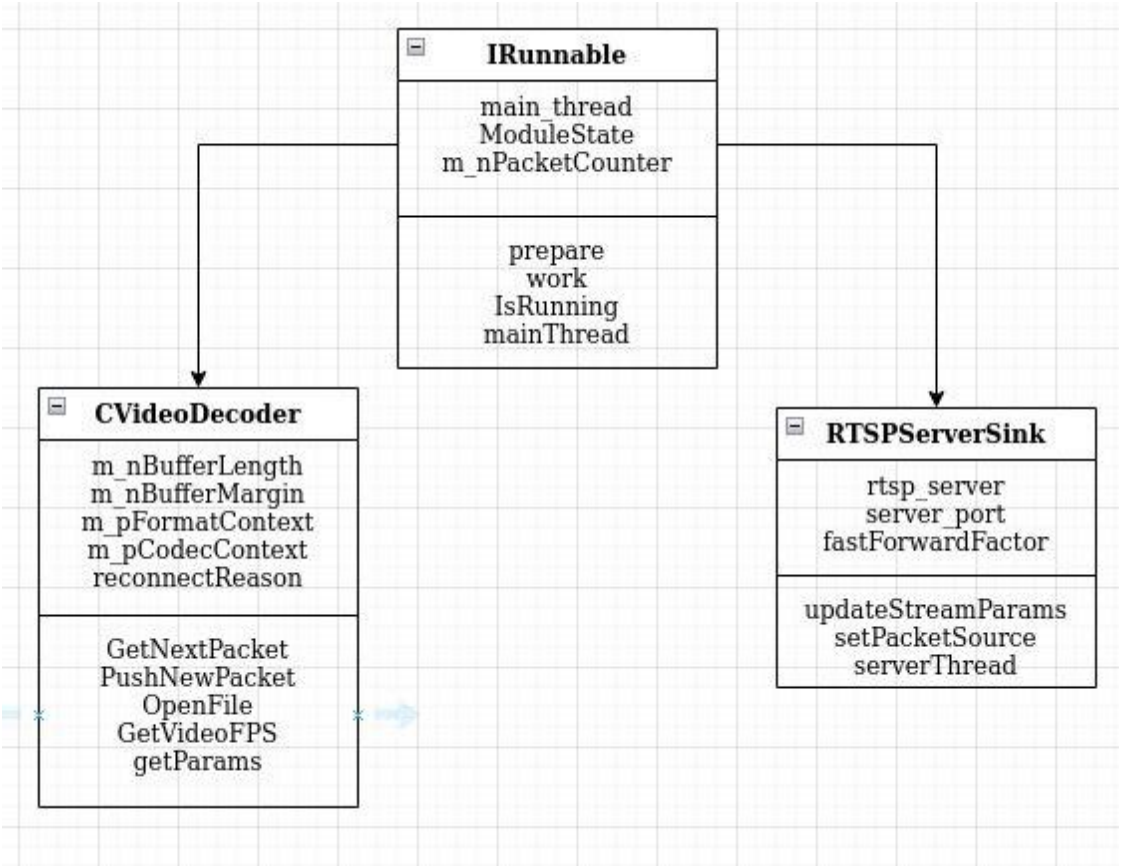

Рисунок 3.1 — Схема класів RTSP модулю

На рисунку 3.2 наведено схему класів програмного модуля ONVIF, а саме належність класу acceptor класові bridge та залежність класу bridge від класу http_helper, яка була створена аналогічно серверній частині. Клас CClient має доволі обширну кількість полів, це обумовлено тим, що значна частина з них — вказівники, зберігання яких в окремому вигляді спрощує структуру програми.

Клас bridge виступає в ролі моста, який підключається до камери, зберігає в собі наступні ключові поля:

— downstream_socket_;

— upstream_socket;

— upstream_buf_;

— upstream_rewrited_buf_;

— httpbuf_.

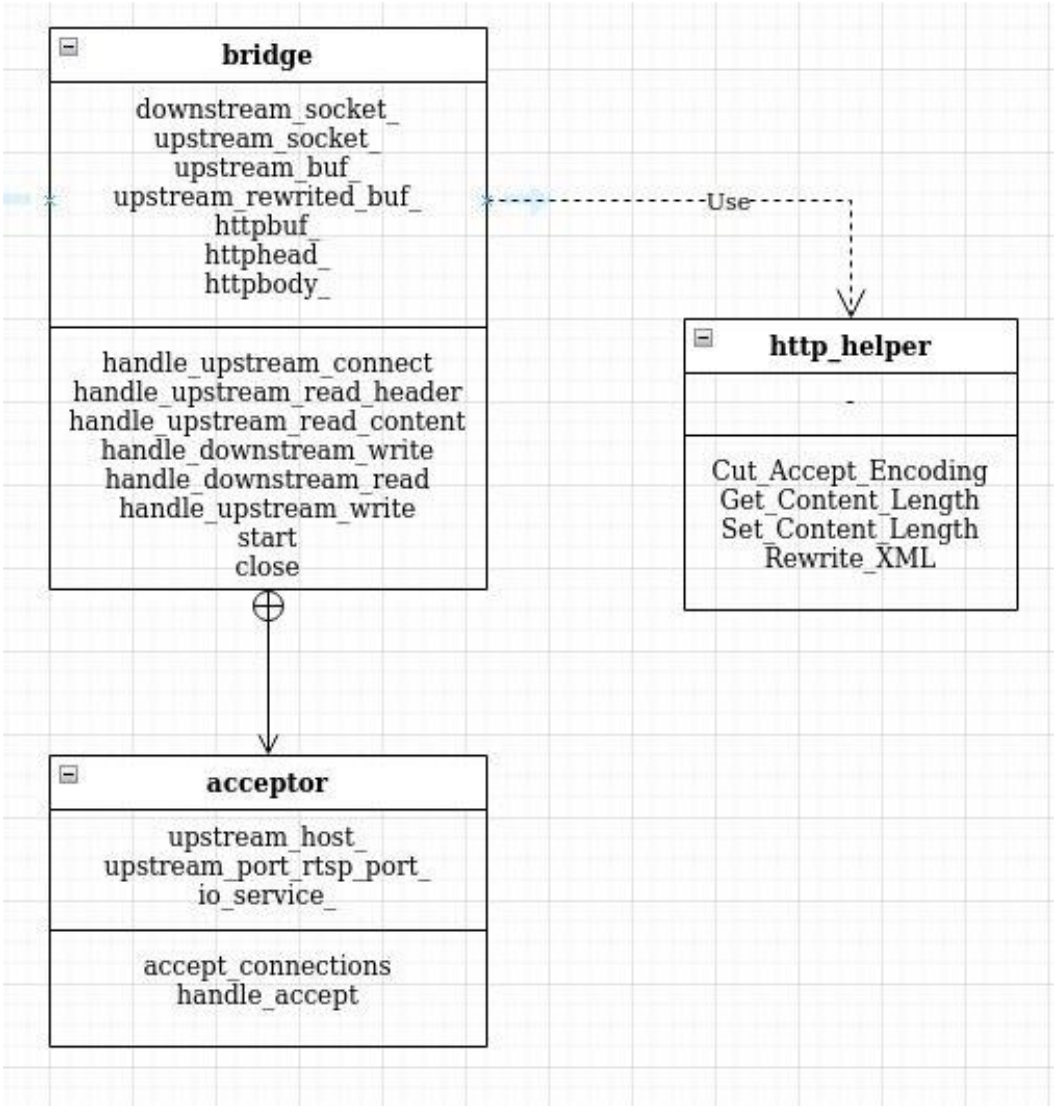

Рисунок 3.2 — Схема класів модулю ONVIF

3.4 Розробка програмних модулів системи

Розробку програмного застосунку буде поділено на декілька етапів. Спочатку потрібно реалізувати основу системи відеоспостереження — RTSP-компонент, що вже забезпечить мінімальний функціонал системи відеоспостереження та інтегрує механізм вирівнювання в систему, що зробить розробку компонентів незалежною один від одного. Після цього буде реалізовано компонент ONVIF, що дозволить комунікувати з VMS, які працюють лише через ONVIF протокол.

Створення RTSP-компоненту відбуватиметься у наступному порядку:

— підключення до камери;

— отримання даних з камери;

— накопичення кадрів в буфер;

— прийняття підключення VMS;

— формування та відправка кадрів користувачеві.

Перший етап реалізований у модулі CVideoDecoder. Функція OpenFile() відкриває для зчитування файл або посилання, на якому йде транслювання відеопотоку.

```
void CVideoDecoder::OpenFile() {
//---
   for (nTryNumber = 0 ; nTryNumber < nMaxTryNumber; nTryNumber++) {
      if (m_pFormatContext)
         CloseFile();
//---
      int ret = avformat_open_input(&m_pFormatContext, sourceURL.c_str(), nullptr, &d);
      if (ret == 0) {
         if ((m_pCodecContext = avcodec_alloc_context3(m_pCodec)) == nullptr)
            continue;
         if  (avcodec_parameters_to_context(m_pCodecContext,  m_pCodecParameters) < 0)
            continue;
         if (avcodec_open2(m_pCodecContext, m_pCodec, nullptr) < 0)
            continue;
//---
      }
   }
}
```

У цьому фрагменті коду описаний процес підключення до відеопотоку за посиланням (наприклад «rtsp://192.168.0.1:554/»).

```
void CVideoDecoder::work() {
//---
   while (1)
   {
//---
      int res = av_read_frame(m_pFormatContext, pPacket);
//---
      if (pPacket->stream_index == m_nVideoStreamIndex)
      {
         pPacket->pts += m_nBasePTSValue;
         pPacket->duration = m_nFrameDuration;
         nLastValidPTS = pPacket->pts;
         fCurrentPTS = GetPTSSeconds(pPacket->pts);
         vPacket.push_back(pPacket);
//---
         for (int j = 0; j < vPacket.size(); j ++)
            PushNewPacket(vPacket[j]);
//---
      }
   }
}
```

В даному циклі програма постійно намагається зчитувати кадри з відеопотоку та акумулювати їх в буфері. Після того як буфер заповниться, RTSPServerSink відкриває порт та починає транслювання після приєднання клієнта.

```
void RTSPServerSink::work() {
//---
```

```
    packet = Decoder->GetNextPacket(bExtraFrames);
    if (bExtraFrames != bFastForwardMode) {
        fStartTime = fCurrentTime;
        nStartPTS = packet->pts;
        bFastForwardMode = bExtraFrames;
    }
    auto fRealTimeElapsed = fCurrentTime - fStartTime;
    auto fPacketTime = Decoder->GetPTSSeconds(packet->pts - nStartPTS);
    if (bFastForwardMode) {
        fPacketTime /= fastForwardFactor;
    }
//---
}
//---
AVPacket* CVideoDecoder::GetNextPacket(bool &bExtraFramesAvailable) {
//---
    if (clear_margin) {
        if (m_qPacket.GetSize() <= defaultQueueSize)
            clear_margin = false;
    } else {
        if (m_qPacket.GetSize() > defaultQueueSize + defaultMarginSize)
            clear_margin = true;
    }
    bExtraFramesAvailable = clear_margin;
//---
}
```

В даному фрагменті коду описаний процес взяття пакету з буферу, визначення презентаційної мітки та запуск пришвидшеного програвання при

умові заповненого маржинального буферу, яка визначається всередині функції GetNextPacket().

```
 void RTSPServerSink::work()
{
//---
   while (!pkts.empty()) {
      auto pkt = &pkts.back();
      {
         xop::AVFrame videoFrame = {0};
         videoFrame.size = pkt->second;
         if (packet->flags != 0 && videoFrame.size > 4)
            videoFrame.type = xop::FrameType::VIDEO_FRAME_I;
         else
            videoFrame.type = xop::FrameType::VIDEO_FRAME_P;
         videoFrame.timestamp = packet->pts;
         videoFrame.buffer.reset(new uint8_t[videoFrame.size]);
         memcpy(videoFrame.buffer.get(), pkt->first.get(), videoFrame.size);
         videoFrame.lastFrame = (pkts.size() == 1) ? 1 : 0;
         rtsp_server->PushFrame(session_id, xop::channel_0, videoFrame);
      }
      pkts.pop_back();
   }
//---
}
```

В даному фрагменті коду описаний процес визначення типу та метаданих кадру, формування відеопакету і відправки його на клієнт.

Після того, як розробили RTSP-компонент, необхідно розробити ONVIF-компонент, який інтегрується в загальну систему. Етапи роботи RTSP-компонентуне зміняться, лише процес підключення VMS буде відбуватись не

безпосередньо через RTSP-посилання, а через ONVIF-проксі систему. Етапи розробки ONVIF-компоненту будуть наступними:

— підключення до ONVIF-серверу камери;

— отримання повідомлення;

— коригування повідомлення та подальша його відправка.

```
void bridge::start(const std::string &upstream_host, unsigned short upstream_port)
{
   upstream_socket_.async_connect(
         ip::tcp::endpoint(
               boost::asio::ip::address::from_string(upstream_host),
               upstream_port),
         boost::bind(&bridge::handle_upstream_connect,
                  shared_from_this(),
                  boost::asio::placeholders::error));
}
```

В даному фрагменті коду ONVIF-компонент починає свою роботу. Після того як VMS почне відправляти запит на підключення, викликається callback-функція, яка підключається безпосередньо до камери для можливості передачі ONVIF-запитів.

```
void bridge::handle_upstream_read_content(const boost::system::error_code &error, const size_t &bytes_transferred)
      if (!httpbody_.str().empty())
      {
         std::string addr = downstream_socket_.local_endpoint().address().to_string();
         std::string port = std::to_string(downstream_socket_.local_endpoint().port());
         http_helper::Rewrite_XML(buf, addr, port, diplom_rtsp_port);
         http_helper::Set_Content_Length(httphead_, buf.length());
      }
      httpbuf_ << httphead_.str();
```

```
httpbuf_ << buf;
upstream_rewrited_buf_.sputn(httpbuf_.str().c_str(), httpbuf_.str().size());
async_write(downstream_socket_,
        upstream_rewrited_buf_,
        boost::bind(&bridge::handle_downstream_write,
                shared_from_this(),
                boost::asio::placeholders::error,
                boost::asio::placeholders::bytes_transferred));
```

В даному фрагменті коду описується процес обробки ONVIF-запиту, який потрапляє всередину функції Rewrite_XML() для коригування за необхідності. Після коригування запиту також коригується значення розміру HTTP тіла Content_Length, після чого повністю сформоване повідомлення передається на іншу сторону.

```
void http_helper::Rewrite_XML(std::string &xml, const std::string &ip, const std::string &port, const std::string &rtsp_port){
//---
 std::string addr = ip + ":" + port;
    std::string rtsp_url = "rtsp://" + ip + ":" + rtsp_port + "/stream";
    if (body.get("trt:GetStreamUriResponse", "NONE") != "NONE")
    {
       body.erase("trt:GetStreamUriResponse.trt:MediaUri.tt:Uri");
       body.put("trt:GetStreamUriResponse.trt:MediaUri.tt:Uri", rtsp_url);
    }
    else if(body.get("tr2:GetStreamUriResponse", "NONE") != "NONE")
//---
}
```

В даному фрагменті методу Rewrite_XML() описується перевірка повідомлення, визначається його тип. У випадку коли повідомлення є відповіддю на запит отримання посилання на відеопотік, видаляється оригінальне посилання

камери та вставляється нове сформоване посилання, на якому транслює RTSP-компонент, враховуючи IP-адресу, де він знаходиться.

Після того як VMS отримає RTSP-посилання, кадри з буферу почнуть відправлятись і процес підключення буде завершеним.

3.5 Висновки до розділу 3

У третьому розділі магістерської кваліфікаційної роботи було проведено огляд потенціальних мов програмування для реалізації розробки та інтегрованих середовищ розробки відповідно.

Для розробки було обрано мову програмування C++ як одну з найбільш швидкодійних да достатньо низькорівневих для вирішення поставлених задач. Також дана мова програмування ідеально підходить для розробки на базі фреймворку FFMpeg та для реалізації проксі-сервера для ONVIF-компоненту. Серед інтегрованих середовищ розробки було обрано середовище CLion як найзручніше у використанні.

Також для кращого розуміння архітектури системи було побудовано схеми класів для кожного з компонентів по стандарту UML. Було проведено огляд цього стандарту.

Після цього розробку було поділено на етапи та було виконано програмну реалізацію компонентів згідно цих етапів. Додатково по кожному з них було наведено фрагменти коду, які включають ключові моменти роботи компонентів.

Було детально описано принцип роботи реалізованих етапів розробки та перехід від компонента до компонента. До того ж, було залишено можливість незалежної роботи RTSP-компоненту задля підтримки роботи з простими або універсальними VMS та відеопрогравачами.

4 ТЕСТУВАННЯ СИСТЕМИ ТА ЇЇ ВЕРИФІКАЦІЯ

4.1 Огляд існуючих видів тестування

Для того щоб провести ефективне тестування системи, потрібно розглянути існуючі види, методи та способи тестування програмних засобів. На основі оглянутих видів тестування буде обрано ті, що підходять найбільше та зможуть в достатній мірі розкрити результати роботи системи.

В першу чергу тестування поділяють на автоматичне та ручне(мануальне) тестування.

Автоматичне тестування являє собою розробка та використання спеціальних скриптів або додаткових програмних застосунків, які автоматично запускають додаток, вводять певні вхідні дані, правильні або навмисно неправильні, перевіряють окремі функції або властивості додатку.

В свою чергу автоматичне тестування поділяється на тестування на рівні коду та GUI-тестування. Тестування на рівні коду являє собою перевірку роботи кожного програмного модулю окремо, а саме окремі класи, функції, методи, інтерфейси та інші компоненти.

GUI-тестування являє собою імітацію дій користувача спеціальними засобами тестування. Даний вид тестування користується більшим попитом через те, що він більш приближений до реальних дій користувача та не потребує прямого втручання в програмний код.

Одним з недоліків автоматичного тестування є в постійній потребі переписування скриптів при будь-якій зміні в коді програмного застосунку, що значно додає ресурсоємності в розробці та самому тестуванні в цілому.

Мануальне тестування є менш ресурсозатратним способом, так як не потребує написання додаткового коду або скриптів. Суть даного підходу заключається в «імітації» потенціального користувача та копіюванні послідовності його дій та пошуку випадків, коли користувач може «зламати» програму, ввівши некоректні дані або дійшовши до вразливості програми, яку

вона не оброблює. Процес мануального тестування займає більше часу, але не потребує попередніх підготовчих заходів.

Також тестування можуть поділяти на наступні види:

— функціональне тестування;

— нефункціональне тестування;

— тестування змін;

— структурне тестування.

Функціональне тестування являє собою перевірку здатності виконувати задачі, які були закладено в програмі та відповідності їх вимогам.

За допомогою нефункціонального тестування перевіряють непрямі властивості програмного застосунку, які не впливають на виконання певних задач, але впливають на зручність та якість програми, такі як швидкодія, оптимізація або ступінь захищеності програми.

Тестування змін являє собою перевірку змін в програмі після оновлень, додавання нового функціоналу чи усунень багів. Використовується при оновленні версій програми.

Структурне тестування передбачає перевірку внутрішньої структури елементів застосунку.

Отже, для найвдалішого тестування буди використано мануальне тестування як краща імітація потенційного користувача. Також будуть використовуватись елементи функціонального тестування для перевірки роботи системи.

4.2 Тестування відеотранслювання та підключення емулятора до VMS

Тестування програмного модуля проводиться методом «чорної скриньки». Він заснований на використанні тестових шаблонів як тестових випадків. Це означає, що буде створено кілька тестів для перевірки правильності роботи основних функцій системи. Тестові випадки, розроблені для перевірки коректності системи, описані в таблиці 4.1.

Таблиця 4.1 – Тест-кейси

Код тест-кейса	Опис тест-кейса		
	Хід тестування		Очікуваний результат
	Дата тестування	Результат	Примітка
001	Перевірка трансляції камери AXIS по RTSP		
	1. Заповнити конфігураційний файл. 2. Запустити додаток. 3. Підключитись до трансляції по VLC.		Поява вікна з трансляцією.
	11.11.2022	Пройдено	-
002	Перевірка камери AXIS на отримання трансляції через ONVIF		
	1. Заповнити конфігураційний файл. 2. Запустити додаток. 3. Підключитись до додатку через ODM		Поява трансляції в ODM
	11.11.2022	Пройдено	-
003	Перевірка трансляції камери Hikvision по RTSP		
	1. Заповнити конфігураційний файл. 2. Запустити додаток. 3. Підключитись до трансляції по VLC		Поява вікна з трансляцією.
	12.11.2022	Пройдено	-
004	Перевірка камери Hikvision на отримання трансляції через ONVIF		
	1. Заповнити конфігураційний файл. 2. Запустити додаток. 3. Підключитись до додатку через ODM		Поява трансляції в ODM.
	12.11.2022	Пройдено	-
005	Перевірка роботи синхронізації відео		
	1. Заповнити конфігураційний файл. 2. Запустити додаток. 3. Підключитись до трансляції		Плавне відтворення відео.
	12.11.2022	Пройдено	-

Тестування додатку по методу тестування «чорної скриньки» пройшло успішно, не було виявлено дефектів. На рисунках 4.1-4.3 зображено результати роботи системи.

Рисунок 4.1 — Підключення через RTSP-компонент

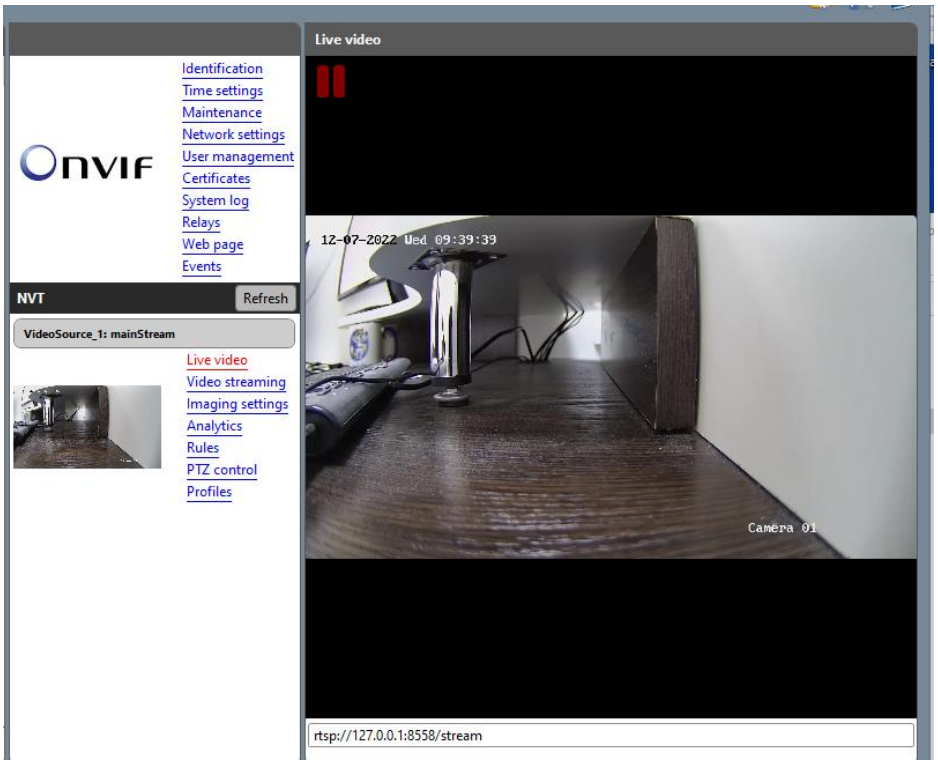

Рисунок 4.2 — Підключення через ONVIF-компонент

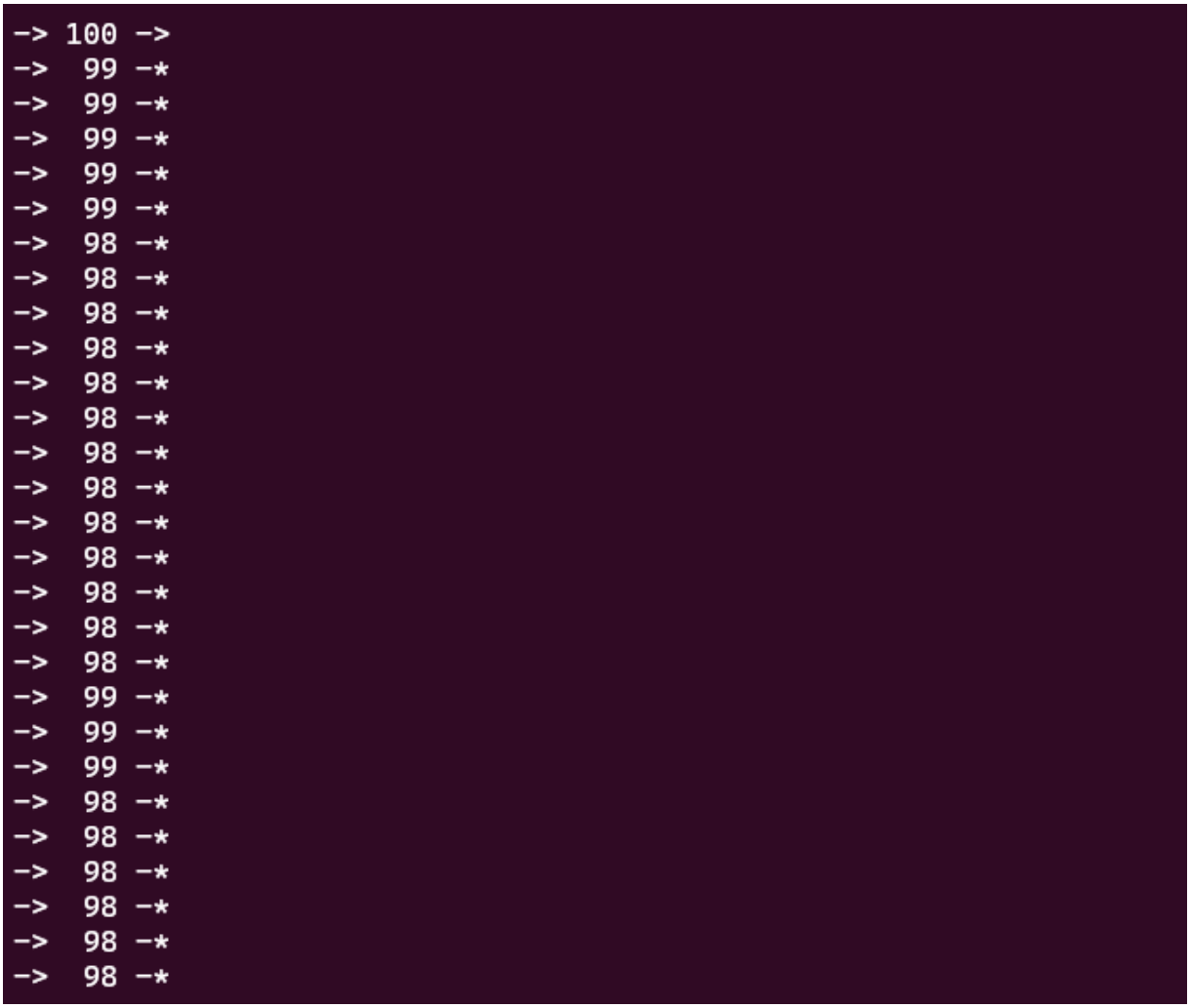

Рисунок 4.3 — Моніторинг наповнення буферу

В ході тестування було продемонстровано можливості програмного додатку, а саме можливість трансляції через відеопрогравачі накшталт VLC та через VMS з підтримкою протоколу ONVIF.

4.3 Верифікація системи

Верифікація — це процес оцінювання системи або її компонентів для визначення того, чи відповідають результати поточної фази розробки умовам, визначеним на початку цієї фази, тобто цілям, термінам виконання, завданням, визначеним на початку поточної фази розвитку проекту, були досягнуті.

Під час перевірки деякі аспекти, створені під час розробки та обслуговування програмного забезпечення, перевіряються на відповідність

іншим аспектам, створеним раніше або використаним як вихідні дані, а також на відповідність цих аспектів і процесу їх розробки правилам і стандартам.

Загалом, основним завданням верифікації є контроль якості програмного забезпечення.

Відповідно до технічного завдання для реалізації завдання розроблена програмна складова, яка дозволяє:

— буферизувати відеопотік;

— проводити постійний процес синхронізації відео;

— підтримувати комунікацію з камерою;

— автоматично перепідключатись до потоку при виникненні неполадок.

На рисунку 4.4 зображено графік наповненості буферу при його розмірі в 10 секунд.

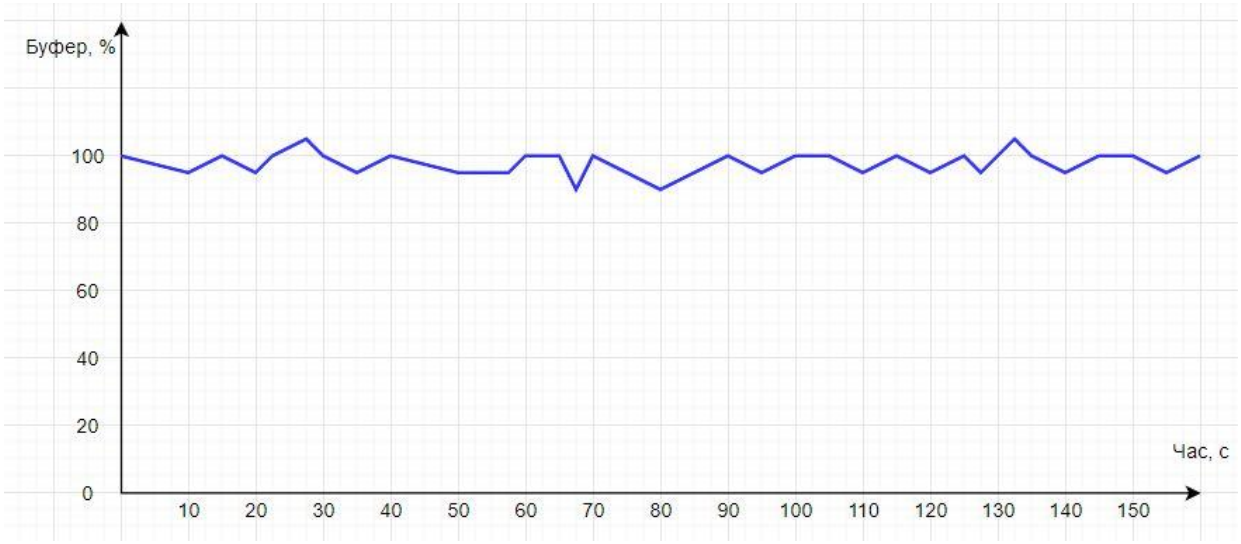

Рисунок 4.4 — Графік наповненості буферу, перший тест

Виходячи з графіку, можна зазначити, що розмір буферу постійно коливається, постійно з’являється нестача кадрів в певні моменти, але згодом мережа стабілізується та буфер вирівнюється. Коливання є відносно незначним, але розмір буферу є також досить великим. На рисунку 4.5 зображено графік наповненості буферу при його розмірі в 5 секунд.

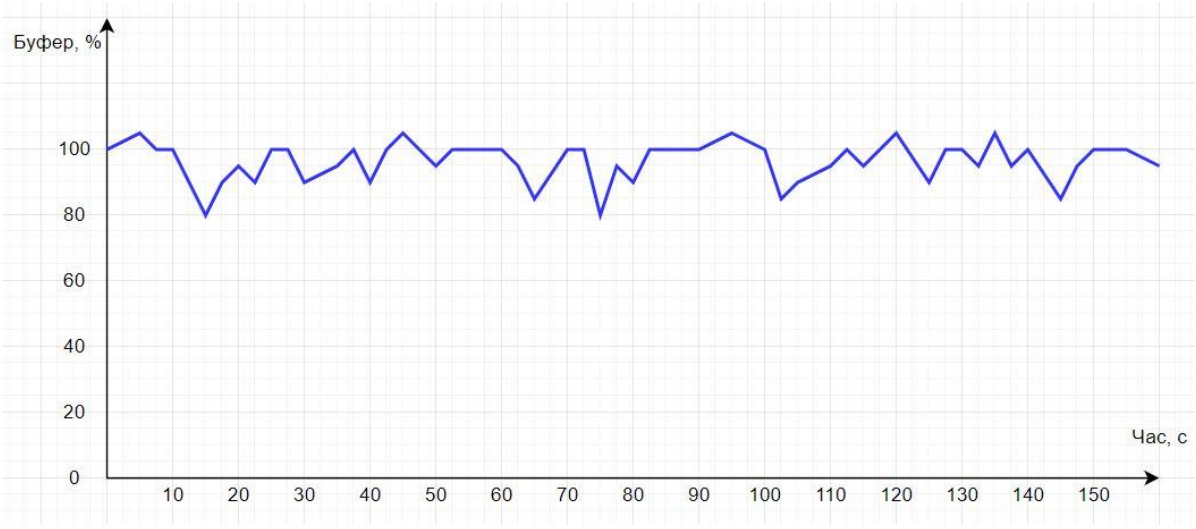

Рисунок 4.5 — Графік наповненості буферу, другий тест

Згідно з другим тестом слід зазначити, що коливання буферу значно зросли при його зменшенні. Так як стабільність мобільної мережі важко відслідкувати, тому оптимальний розмір буферу може змінюватись залежно від ситуації.

4.4 Керівництво оператора

Для початку роботи необхідно, щоб відеокамера та ПК оператора знаходились в одній мережі або мали видимі один для одного IP-адреси. Переконавшись в цьому, потрібно відкрити файл «config.json», який знаходиться в папці з програмою, в полі «stream» вказати RTSP-лінк потокового транслювання камери, в полі «port» вказати вихідний порт транслювання додатку, в полі «proxyPort» вказати вхідний ONVIF порт відеокамери.

Якщо все було зроблено вірно, можна запустити додаток, після чого побачивши стан буфера на рисунку 4.6.

Додаток відкриється, ініціалізується, виведе параметри, з якими запустилась програма та почне накопичувати кадри в буфер.

Рисунок 4.6 — Моніторинг наповнення буферу

Для підключення можна використовувати будь-який RTSP-клієнт або VMS з підтримкою протоколу ONVIF або RTSP.

Для підключення по протоколу RTSP необхідно у відповідному полі ввести посилання у форматі «rtsp://127.0.0.1:XXXX/stream» в залежності від значення вихідного порту, яке було вказано в конфігураційному файлі.

Для підключення по протоколу ONVIF необхідно заповнити необхідні поля в звлежності від VMS. Це можуть бути окремо винесені поля з IP-адресою та ONVIF-портом, або попередньо сформоване посилання, куди потрібно вставити відповідні значення. На рисунку 4.7 зображено приклад такого поля:

Рисунок 4.7 — Посилання на ONVIF-сервіс

ONVIF-порт вираховується за принципом «вихідний RTSP-порт + 10000». Після введення цих даних можна користуватись системою. Окрім перегляду відео, також може бути доступним PTZ та керування додатковими параметрами відеокамери.

4.5 Висновки до розділу 4

У четвертому розділі магістерської кваліфікаційної роботи було оглянуто основні методики та види тестування, а саме:

— функціональне тестування;

— нефункціональне тестування;

— тестування змін;

— структурне тестування;

Також було оглянуто автоматичне та мануальне тестування. Було обрано мануальне тестування, тому що такий метод краще імітує потенційного користувача, не потребує втручання в програмний код та є менш ресурсоємним, тобто не потребує додаткової розробки, написання спеціальних скриптів.

Тестування було проведено по тест-кейсам, які були попередньо написані та сформовані. В ході тестування не було виявлено багів, дефектів тощо.

Окрім цього, було проілюстровано результати роботи програми, показано зовнішній вигляд програми, результат підлючення камери по протоколам RTSP та ONVIF.

Було проведено верифікацію програмного засобу, побудовано графіки наповненості буферу. Було зроблено висновок, що оптимальний розмір буферу може коливатись в залежності від степеню стабільності мережі, тому програма дає можливість коригувати розмір буферу в конфігураційному файлі.

Було оформлено керівництво оператора як коротку інструкцію користувача задля розуміння як правильно користуватись програмним застосунком.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного засобу вирівнювання відтворення зображення, який стане частиною системи відеоспостереження, дозволить оптимізувати передачу відеотрафіку та розширити сумісність зі стороннім програмним забезпеченням.

Особливістю є те, що використовується новий підхід до оптимізації відеотранслювання, а також розробка засобу інтеграції зі стороннім програмним забезпеченням.

Аналогом може бути Happytime Media Server — 14400,00 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах

Продовження табл. 5.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною .	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	3	3	4
Цінова політика	4	3	3
Технічні та споживчі властивості виробу	3	4	4
Експлуатаційні витрати	4	3	3
Ринок збуту	3	4	4
Конкурентоспроможність	4	3	3
Фахівці з технічної і комерційної реалізації	3	4	3
Фінансування	4	4	3
Матеріально-технічна база	4	3	3
Термін реалізації ідеї	3	4	3
Супровідна документація	3	3	4
Сума	41	41	41
Середньоарифметична сума балів	(41+41+41) / 3 = 41		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що програмний продукт відрізняється від існуючих тим, що дана технологія є програмним засобом вирівнювання відтворення зображення, який стане частиною системи відеоспостереження, дозволить оптимізувати передачу відеотрафіку та розширити сумісність зі стороннім програмним забезпеченням, який використовує новий підхід до оптимізації відеотранслювання, а також розробка засобу інтеграції зі стороннім програмним забезпеченням.

5.2 Прогнозування витрат на виконання науково-дослідної(дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \,, \tag{5.1}$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ — число робочих днів в місяці, 23 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	28000	1217,39	40	48695,652
Програміст	25000	1086,96	40	43478,261
Всього				92173,91

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 13 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 13\ \% / 100\ \%, \quad (5.2)$$

де $З_д$ — додаткова заробітня плата;

$З_о$ — основна заробітня плата.

$$З_д = (92173{,}91 \cdot 13\ \% / 100\ \%) = 11982{,}61 \ \text{(грн.)}$$

Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\%, \quad (5.3)$$

де $Н_з$ — нарахування заробітньої платні;

$З_о$ — основна заробітня плата.

$З_д$ — додаткова заробітня плата;

$$Н_з = (92173,91 + 11982,61) \cdot 22\ \% / 100\ \% = 22914,43\ (грн.)$$

Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12}, \qquad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ — термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 25820 грн., термін його корисного використання згідно податкового законодавства — 2 роки, а термін його фактичного використання — 1,74 міс.

$$A_{обл} = \frac{26000}{2} \times \frac{1,73}{12} = 1871,212\ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.5.

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не

амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 3570 грн.

Таблиця 5.5 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія (ARTLINE Gaming X33 v10, GSM модем Sierra Wireless RV50, Xiaomi Mi WiFi Router 4A R4AC)	25820	2	1,74	1871,014
Офісне обладнення (меблі)	20000	4	1,74	724,638
Приміщення	950000	20	1,74	6884,058
Всього				9479,71

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,4 кВт;

Ф — фактична кількість годин роботи обладнання, годин.

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$B_е = 0,9 \cdot 0,4 \cdot 8 \cdot 40 \cdot 6,2 = 714,24 \text{ (грн.)}$$

Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$I_в = 92173,91 * 50\% / 100\% = 46086,96 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 92173,91 * 150 \% / 100 \% = 138261 \text{ (грн.)}$$

Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$B_{заг} = 92173,91+11982,61+22914,43+9479,71+3570+4300+714,24+$$
$$+46086,96+138261 = 329482,73 \text{ грн.}$$

Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн), де} \qquad (5.8)$$

η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 329482,73 / 0,5 = 658965 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів,

дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями;

— розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу.

При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.9)$$

де $\pm\Delta Ц_o$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

Цб — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

ρ — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 6000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *500* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 5000 шт., протягом другого року — на 4000 шт., протягом третього року на 3000 шт.

До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (0*500 + (6000 + 500)*5000)* 0,8333* 0,25) * (1 - 0,18) = 5124999,795 грн.

$\Delta П_2$ = (0*500 + (6000 + 500)*(5000+4000)* 0,8333* 0,25) * (1 - 0,18) = 9993749,600 грн.

$\Delta П_3$ = (0*500 + (6000 + 500)*(5000+4000+3000)* 0,8333* 0,25) * (1 - 0,18) = 13324999,467 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 28443748,86 грн.

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (5124999,795/(1+0,1)^1)+(9993749,600/(1+0,1)^2)+(13324999,467/(1+0,1)^3) = 4659090,72 + 8259297,19 + 10011269,32 = 22929657,23 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 658965 = 1317930{,}93 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (5.12)$$

де $E_{абс}$ — абсолютний економічний ефект.

$$E_{абс} = 22929657{,}23 - 1317930{,}93 = 21611726{,}30 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми

дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

де $T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 21611726{,}30/1317930{,}93} - 1 = 1{,}591$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{\min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (5.15)$$

де $T_{ок}$ — термін окупності

$E_в$ — ефективність витрарт.

$$T_{ок} = 1 / 1{,}591 = 0{,}63 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,63 роки, то фінансування даної наукової розробки є доцільним.

5.5 Висновки до розділу 5

Економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 658965 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,63 роки.

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було оглянуто та проаналізовано основні засоби, методи та способи створення систем відеоспостереження та передавання даних в цілому. Було оглянуто основні та найбільш поширені методи відеотранслювання, способи та протоколи комунікації з відеокамерами. Під час огляду було проаналізовано та перераховано переваги та недоліки кожного засобу.

На основі оглянутих засобів для вирішення поставлених задач було обрано фреймворк FFMpeg як актуальний засіб розробки у сфері відеопроцесингу, RTSP та пов'язані протоколи як оптимальний засіб передачі відеоданих, ONVIF як один з найбільш поширених стандартизованих протоколів комунікації з IP-камерами.

Було спроектовано архітектуру програмного рішення поставленої задачі, описано логіку буферизації відеопотоку. Окрім буферизації також було спроектовано механізм вирівнювання кадрів, що значно покращить сихронізацію відео, логіку перепідключення до камери, що оброблюватиме випадки виникнення критичних перебоїв в мобільній мережі.

Було проаналізовано особливості структури ONVIF-повідомлень, враховано необхідні елементи, що потребують коригування при їх пересиланні. Обране рішення дозволить підтримувати зв’язок з відеокамерою та користуватись в повній мірі її функціоналом.

Для програмної реалізації було оглянуто мови програмування та інтегровані середовища розробки, на основі огляду було обрано мову програмування C++ та інтегроване середовище розробки CLion.

Для більшого розуміння архітектури системи побудовано схеми класів програмних компонентів додатку, було відокремлено компонент відеотранслювання та компонент комунікації через протокол ONVIF.

Було розроблено програмний застосунок, що інтегрує в систему відеоспостереження метод вирівнювання відтворення зображення. Основна ідея

даного застосунку полягає у вдосконаленні стабілізаційних можливостей системи відеоспостереження, не втративши при цьому додатковий функціонал відеокамер.

Також в застосунок було інтегровано компонент комунікації з відеокамерою, пристосований для проксування та редагування повідомлень протоколу ONVIF, для збереження сумісності із стороннім програмним забезпеченням.

В ході розробки було описано етапи розробки та порядок виконання роботи програми.

Також для тестування програмного засобу було оглянуто види та методи тестування та обрано ті, що зможуть ефективно перевірити коректну роботу системи. Були написані тест-кейси для тестування, також було проілюстровано результати роботи програмного додатку. Було проведено верифікацію системи та побудовано графіки наповненості буферу.

Виходячи з побудованих графіків було зроблено висновок, що оптимальний розмір буферу може коливатись в залежності від степеню стабільності мережі, тому програма дає можливість коригувати розмір буферу в конфігураційному файлі.

Було оформлено керівництво оператора як коротку інструкцію користувача задля розуміння як правильно користуватись програмним застосунком.

В економічній часині магістерської кваліфікаційної роботи було проведено технологічний аудит, проаналізовано витрати на розробку, визначено ступінь окупності даної розробки та доведено її конкурентоспроможність.

Інтеграція даного застосунку в існуючі системи відеоспостереження, що працюють через мобільну мережу, покращить стабілізаційні заходи системи, зменшить частоту ручного втручання операторів при раптових погіршеннях якості роботи мобільної мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Системи відеоспостереження : веб-сайт. URL: https://cvt.com.ua/solution/systemy-videosposterezhennya

2. Shulczrinne H. Peer-to-Peer Computing for Mobile Networks: Information Discovery and Dissemination, 2009, Vol. 4, no. 21. P. 224

3. Shulczrinne H. Mobility Protocols and Handover Optimization: Design, Evaluation and Application (IEEE Press), 2015, Vol. 28, no. 6. P. 449

4. Fielding R. Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML, 2004, P. 878.

5. Berners T. Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web, P. 246.

6. Rafael O. Handbook of Emerging Communications Technologies: The Next Decade, 1999, P. 42.

7. Захарченко С. М., Шевчук К.І. Інжиніриг трафіку в комп'ютеринх мережах. Електронне видання XLVI науково-технічної конференції ВНТУ. 2017. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/view/2519/2180

8. Крупельницький Л.В., Жилін М.В., Самко В.В. Комп'ютерна система моніторингу цифрового телевізійного мовлення. Електронне видання XLVIII науково-технічної конференції ВНТУ. 2019. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2019/paper/view/6876/5638

9. Мордвинцев М. В., Хлєстков О. В., Ницюк С. П. Технічні проблеми, пов'язані зі стрімким розвитком систем відеоспостереження, й способи їх вирішення. Економічна та інформаційна безпека : актуальні питання та інновації : матеріали Міжнар. наук.-практ. конф. (м. Дніпро, 4 лист. 2021 р.). Дніпро : ДДУВС, 2021. С. 165-167.

10. Мордвинцев М.В. Стан систем безпеки з використанням технічних засобів відеозапису та відеоспостереження: зарубіжний досвід, перспективи

впровадження в діяльність Національної поліції України / В.А. Коршенко, В.В. Чумак, М.В. Мордвинцев, Д.В. Пашнев // Право і безпека. 2020. № 2(77) С. 86-92.

11. Вишневський А.В. Метод вирівнювання відтворення зображення під час передачі відеотрафіку з використанням емуляції IP-камери. Електронне видання LI науково-технічної конференції ВНТУ. 2022. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15791/13329

12. VLC media player : веб-сайт. URL: https://uk.wikipedia.org/wiki/VLC_media_player

13. Using VLC to test camera stream : веб-сайт. URL: https://support.ipconfigure.com/hc/en-us/articles/115005588503-Using-VLC-to-test-camera-stream

14. Нове програмне забезпечення Hikvision Hikcentral : веб-сайт. URL: https://hikvision.org.ua/uk/nove-programne-zabezpechennya-hikvision-hikcentral/

15. Gstreamer : веб-сайт. URL: https://www.wikiwand.com/uk/GStreamer

16. GStreamer: open source multimedia framework : веб-сайт. URL: https://gstreamer.freedesktop.org/

17. FFmpeg : веб-сайт. URL: https://www.wikizero.com/uk/FFmpeg

18. What is FFmpeg? - api.video : веб-сайт. URL: https://api.video/what-is/ffmpeg

19. Saving RTSP Camera Streams with FFmpeg : веб-сайт. URL: https://medium.com/@tom.humph/saving-rtsp-camera-streams-with-ffmpeg-baab7e80d767

20. Що таке RTSP-посилання та як сформувати його для пристрою Hikvision? : веб-сайт. URL: https://hikvision.org.ua/uk/knowledge_base2/shcho-take-rtsp-posylannya-ta-yak-sformuvaty-yogo-dlya-prystroyu-hikvision/

21. Що таке протокол «ONVIF» в системі відеоспостереження? : веб-сайт. URL: https://tvtdigital.com.ua/shcho-take-protokol-onvif-v-systemi-videosposterezhennia/

22. IP-відеокамери спостереження і ONVIF сумісність : веб-сайт. URL: https://worldvision.com.ua/articles/ip-videokameri-nablyudeniya-i-onvif-sovmestimost

23. ONVIF and PSIA: Guide to Standards in Video Surveillance : веб-сайт. URL: https://www.ifsecglobal.com/video-surveillance/guide-to-standards-in-video-surveillance-onvif-v-psia/

24. Will ONVIF or PSIA win the IP Camera Standards Battle? : веб-сайт. URL: https://ipvm.com/reports/onvif-psia-ip-camera-standards

25. Stroustrup B. The C++ Programming Language, 4th edition / B. Stroustrup Addison-Wesley, 2013 — 1368 ст.

26. What is Unified Modeling Language : веб-сайт. URL: https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. О. Д. Азаров
"__" _____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи на тему:
«Метод вирівнювання відтворення зображення під час передавання відеотрафіку з використанням емуляції IP-камери»
08-23.МКР.002.00.000 ПЗ

Науковий керівник к.т.н., проф. каф. ОТ
_____ Азарова А. О.
Студент групи 1КІ-21м
_____ Вишневський А. В.

ВНТУ 2022

1 Підстава для використання МКР

1.1 Актуальність розробки полягає у вдосконаленні методу синхронізації відтворення зображення шляхом інтеграції засобу додаткової буферизації відеокадрів при збереженні функціоналу комунікації з відеокамерою.

1.2 Наказ про затвердження теми кваліфікаційної роботи.

2 Мета МКР і призначення розробки

2.1 мета роботи — вдосконалення процесу передавання відео-трафіку із використанням методу вирівнювання відтворення зображення та розширення функціональних можливостей системи відеоспостереження;

2.2 призначення розробки — програмна реалізація покращеного методу вирівнювання відтворення зображення, що включає в себе засіб підтримки комунікації з відеокамерою.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих принципів та технологій побудови систем відеоспостереження;

3.2 Розробка структури програмного засобу для вирівнювання відтворення зображення;

3.3 Інтегроване середовище розробки JetBrains CLion;

3.4 Засоби інтеграції з протоколом ONVIF для комунікації з IP-відеокамерою.

4 Технічні вимоги до виконання МКР

4.1 Наявність додатку для буферизації та синхронізації відео;

4.2 Наявність механізму комунікації з IP-камерою.

5 Етапи МКР та очікувані результати

5.1 Робота виконується за п’ять етапів, таблиця А.1.

Таблиця А.1 — Етапи виконання МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасних методів побудови систем відеоспостереження	04.10.2022	15.10.2022	Аналітичний огляд літературних джерел
2	Побудова архітектури системи відеоспостереження, механізмів синхронізації та комунікації	16.10.2022	04.11.2022	2 розділ
3	Практичне застосування та оцінка ефективності розробленого додатку	05.11.2022	30.11.2022	3 і 4 розділи
4	Підготовка економічної частини	30.11.2022	03.12.2022	5 розділ
5	Оформлення пояснювальної записки, графічного матеріалу і/або презентації	04.12.2022	18.12.2022	пояснювальна записка, графічний матеріал і/або презентація

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив опонента, протоколи проходження перевірки на плагіат, анотації до МКР українською та іноземною мовами, нормоконтроль про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні екзаменаційної комісії, затверджено] наказом ректора.

8 Вимоги до оформлення МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

Технічне завдання отримав _____ Вишневський А. В.

ДОДАТОК Б

Лістинг RTSP-компоненту

```
#include "CVideoDecoder.h"
#include "CTimer.h"
#include <fstream>
#define END_OF_BUFFER_SIZE -1
#define CONNECT_TIMEOUT_ADDITION 1          // Constant added to BufferSize
extern "C"
{
#include <libavformat/avformat.h>
#include <libavutil/opt.h>
}
using namespace std;
TSafeVector <CVideoDecoder> CVideoDecoder::vVideoDecoder;
CLogger* CVideoDecoder::pUnrecognized;
bool CVideoDecoder::bInitialized = false;
#ifdef av_err2str
#undef av_err2str
av_always_inline std::string av_err2string(int errnum) {
        char str[AV_ERROR_MAX_STRING_SIZE];
        return av_make_error_string(str, AV_ERROR_MAX_STRING_SIZE, errnum);
    }
#define av_err2str(err) av_err2string(err).c_str()
```

```
#endif  // av_err2str

void CVideoDecoder::SetUnrecognizedLogger(CLogger *pLogger)

{

   pUnrecognized = pLogger;

}

void CVideoDecoder::FFmpegCustomLogHandler(void *avcl, int level, const char
*fmt, va_list vl)

{

   if (level <= AV_LOG_WARNING) {

      char sMessage[128];

      int prefix = 1;

      bool bFound = false;


      av_log_format_line(avcl, level, fmt, vl, sMessage, 128, &prefix);


      for (int i = 0; i < vVideoDecoder.GetSize(); i++) {

         CVideoDecoder

            *pDecoder = vVideoDecoder.GetItem(i);

         if (pDecoder->SearchForObject(avcl)) {

            bFound = true;

            pDecoder->pLogger->Print(CLogger::Level::Debug, sMessage);

            break;

         }

      }
```

```
        if (!bFound)
        {
            pUnrecognized->Print(CLogger::Level::Debug, sMessage);
        }
    }
}
AVPacket* CVideoDecoder::GetNextPacket(bool &bExtraFramesAvailable)
{
    if (reconnectOnExhausted) {
        if (m_qPacket.GetSize() == 0 && !buffering_enabled && !bHaveToReconnect)
{
            PrepareForReconnect(RR_BUFFER_EXHAUSTED);
        }
        if (m_qPacket.GetSize() == 0 || buffering_enabled) {
            return nullptr;
        }
    } else {
        while (m_qPacket.GetSize() == 0) {
            if (GetCurrentState() == STOP || GetCurrentState() == STOPPING)
                return nullptr;
            usleep(10000);
        }
    }
    AVPacket
```

```
        *pPacket = m_qPacket.PopFromQueue();

    pLogger->Print(CLogger::Level::Verbose, "Packet pop from buffer, PTS=%d ",
pPacket->pts);



    if(pPacket->size == END_OF_BUFFER_SIZE)

    {

        pLogger->Print(CLogger::Level::Verbose, "Packet marked as the last before
buffering, PTS=%d ", pPacket->pts);

        buffering_enabled = true;

        av_packet_free(&pPacket);

        return nullptr;

    }

    if (clear_margin) {

        if (m_qPacket.GetSize() <= defaultQueueSize) {

            pLogger->Print(CLogger::Level::Debug, "Margin buffer is empty",
m_nDroppedCounter);

            clear_margin = false;

        }

    } else {

        if (m_qPacket.GetSize() > defaultQueueSize + defaultMarginSize) {

            pLogger->Print(CLogger::Level::Debug, "Margin buffer limit reached",
m_nDroppedCounter);

            clear_margin = true;

        }

    }

    bExtraFramesAvailable = clear_margin;
```

```
    return pPacket;
}
void CVideoDecoder::PushNewPacket(AVPacket *pPacket)
{
    WriteLogInfo();
    if(m_qPacket.GetSize() >= defaultQueueSize && buffering_enabled)
    {
        buffering_enabled = false;
        clear_margin = false;
        pLogger->Print(CLogger::Level::Info, "Buffer full");
    }
    while(m_qPacket.GetSize() >= m_nMaxQueueSize)
    {
        AVPacket *pDropPacket = m_qPacket.PopFromQueue();
        if (pDropPacket == nullptr)
            break;
        pLogger->Print(CLogger::Level::Verbose, "Dropped packet with PTS=%d
(sz=%d, max=%d)", pDropPacket->pts,m_qPacket.GetSize(), m_nMaxQueueSize);


        m_qPacket.LockQueue();
        if (pDropPacket->size == END_OF_BUFFER_SIZE) {
            pLogger->Print(CLogger::Level::Verbose, "Marking next packet as the last
before buffering");
            AVPacket
                *pFrontPacket = m_qPacket.GetFrontItem();
```

```
            if (pFrontPacket)
                pFrontPacket->size = END_OF_BUFFER_SIZE;
        }
        m_qPacket.UnlockQueue();
        av_packet_free(&pDropPacket);
        m_nDroppedCounter++;
    }
    m_qPacket.PushToQueue(pPacket);
    pLogger->Print(CLogger::Level::Verbose, "Packet pushed to the buffer PTS=%d", pPacket->pts);


}
double CVideoDecoder::GetPTSSeconds(int64_t pts)
{
    if (pts == AV_NOPTS_VALUE)
        return 0;
    double fTime = double(pts) / double(m_nTimeBase);
    return fTime;
}
CVideoDecoder::CVideoDecoder(int id, std::string url, bool tcp, CLogger *logger, CLogger *consoleLogger) : sourceURL(url), enabled_tcp(tcp), pLogger(logger), pConsoleLogger(consoleLogger)
{
    assert(pLogger != nullptr);
    assert(pConsoleLogger != nullptr);
```

```
nID = id;
m_pFormatContext = nullptr;
m_pCodecContext = nullptr;
m_pCodec = nullptr;
m_pCodecParameters =  nullptr;
m_pSwsContext = nullptr;
m_pFormatContext = nullptr;
m_nTimeout = 0;
timer = 0;
need_to_interrupt = false;
bHaveToReconnect = false;
m_nBasePTSValue = 0;
m_nActualFPS = 0;
m_fGapAccumulator = 0;
m_fLastLogTime = 0;
m_nDroppedCounter = 0;
m_nPacketCounter = 0;
m_fGapTolerance = 1.0;
m_fGapTolerance = 2.0;
bClientPresent = false;
clear_margin = false;
clearOnVMSDisconnect = false;
reconnectOnVMSDisconnect = true;
reconnectOnExhausted = true;
```

```
    codecID = AV_CODEC_ID_NONE;
    if (!bInitialized) {
        bInitialized = true;
        av_log_set_callback(FFmpegCustomLogHandler);
    }
    vVideoDecoder.AddItem(this);
}
void CVideoDecoder::CloseFile()
{
    if (m_pFormatContext) {
        avformat_close_input(&m_pFormatContext);
        m_pFormatContext = nullptr;
    }
    if (m_pCodecContext) {
        avcodec_free_context(&m_pCodecContext);
        m_pCodecContext = nullptr;
    }
    if (m_pSwsContext) {
        sws_freeContext(m_pSwsContext);
        m_pSwsContext = nullptr;
    }
    if (avbsfContext1) {
        av_bsf_free(&avbsfContext1);
        avbsfContext1 = nullptr;
```

```
    }
    if (avbsfContext2) {
        av_bsf_free(&avbsfContext2);
        avbsfContext2 = nullptr;
    }
    if (avbsfContext3) {
        av_bsf_free(&avbsfContext3);
        avbsfContext3 = nullptr;
    }
    codecID = AV_CODEC_ID_NONE;
    m_pCodecParameters = nullptr;
    m_nActualFPS = 0;
    m_fGapAccumulator = 0;
    m_fLastLogTime = 0;
    m_nDroppedCounter = 0;
}
CVideoDecoder::~CVideoDecoder()
{
    ClearAllPackets();
    CloseFile();
}
bool CVideoDecoder::GetCodecExtraData(uint8_t **ppData, int *pSize)
{
    if (!m_pCodecParameters) {
```

```
        return false;
    }
    *ppData = m_pCodecParameters->extradata;
    *pSize = m_pCodecParameters->extradata_size;


    return true;
}
void CVideoDecoder::GetVideoCodecParameters(AVCodecParameters
*pCodecParameters)
{
    m_pCodecParameters = pCodecParameters;
    m_nWidth = pCodecParameters->width;
    m_nHeight = pCodecParameters->height;
    m_nPixelFormat = (AVPixelFormat)pCodecParameters->format;
}
bool CVideoDecoder::CheckAllStreams()
{
    m_nVideoStreamIndex = -1;
    timer = CTimer::getCurrentTime();
    AVDictionary *d{};
    if (enabled_tcp)
        av_dict_set(&d, "rtsp_transport", "tcp", 0);
    av_dict_set(&d, "stimeout", "30000000", 0);
    need_to_interrupt = true;
```

```
    int ret = avformat_find_stream_info(m_pFormatContext,  &d);
    need_to_interrupt = false;
    av_dict_free(&d);
    if (ret < 0)
        return false;
    pLogger->Print(CLogger::Level::Debug, "Stream info retrieved");

    for (int i = 0; i < m_pFormatContext->nb_streams; i++)
    {
        AVStream
            *pLocalStream = m_pFormatContext->streams[i];
        AVCodecParameters
            *pLocalCodecParameters =  pLocalStream->codecpar;
        AVCodec
            *pLocalCodecIn = avcodec_find_decoder(pLocalCodecParameters->codec_id);

        if (pLocalCodecParameters->codec_type == AVMEDIA_TYPE_VIDEO)
        {
            m_nVideoStreamIndex = i;
            m_pCodec = pLocalCodecIn;
            m_nTimeBase = pLocalStream->time_base.den;
            GetVideoCodecParameters(pLocalCodecParameters);
            codecID = m_pCodec->id;
```

```
        std::string bsf_name;
        switch (codecID)
        {
        case AV_CODEC_ID_H264:
            bsf_name = "h264_metadata";
            break;
        case AV_CODEC_ID_HEVC:
            bsf_name = "hevc_metadata";;
            break;
        default:
            return false;
        }
        const AVBitStreamFilter *bsf1 = av_bsf_get_by_name("remove_extra");
        const AVBitStreamFilter *bsf2 = av_bsf_get_by_name("dump_extra");
        const AVBitStreamFilter *bsf3 = av_bsf_get_by_name(bsf_name.c_str());
        av_bsf_alloc(bsf1, &avbsfContext1);
        av_bsf_alloc(bsf2, &avbsfContext2);
        av_bsf_alloc(bsf3, &avbsfContext3);
        av_opt_set(avbsfContext2->priv_data, "freq", "keyframe", 0);
        av_opt_set(avbsfContext3->priv_data, "aud", "insert", 0);
    }
}
if (!get_extradata_from_stream())
    return false;
```

```
    avcodec_parameters_copy(avbsfContext1->par_in,
(AVCodecParameters*)getParams());

    avcodec_parameters_copy(avbsfContext2->par_in,
(AVCodecParameters*)getParams());

    avcodec_parameters_copy(avbsfContext3->par_in,
(AVCodecParameters*)getParams());

    auto &stream = m_pFormatContext->streams[m_nVideoStreamIndex];

    if (stream->avg_frame_rate.den != 0)

        m_nActualFPS = stream->avg_frame_rate.num / stream->avg_frame_rate.den;

    else if (stream->r_frame_rate.den != 0)

        m_nActualFPS = stream->r_frame_rate.num / stream->r_frame_rate.den;

    if (m_nActualFPS < 1 || m_nActualFPS > 60) {

        pLogger->Print(CLogger::Level::Error, "Incorrect FPS value - %d",
m_nActualFPS);

        return false;

    }

    int bufSize = m_nBufferLength * m_nActualFPS;

    if (bufSize < m_nBufferLengthInFrames)

        bufSize = m_nBufferLengthInFrames;

    SetMaxQueueSize(bufSize, m_nBufferMargin * m_nActualFPS);

    m_fPTSDifference = 1.0 / double(m_nActualFPS);

    m_nFrameDuration = m_nTimeBase / m_nActualFPS;

    char tmp[256];

    sprintf(tmp, "Opened stream #%d - codec: %s, FPS: %d, buffer size : %d sec (%d
frames), margin size: %d sec (%d frames)",

            nID, avcodec_get_name(codecID), m_nActualFPS, m_nBufferLength,
defaultQueueSize, m_nBufferMargin, defaultMarginSize);
```

```
    printf("%s\r\n", tmp);
    pConsoleLogger->Print(CLogger::Level::Debug, "%s", tmp);
    if ((m_nVideoStreamIndex == -1))
        return false;
    av_bsf_init(avbsfContext1);
    av_bsf_init(avbsfContext2);
    av_bsf_init(avbsfContext3);
    return true;
}
int CVideoDecoder::GetActualFPS()
{
    return m_nActualFPS;
}
bool CVideoDecoder::GetVideoDimensions(int *pWidth, int *pHeight)
{
    if (!m_pFormatContext)
        return false;
    *pWidth = m_nWidth;
    *pHeight = m_nHeight;
    return true;
}
void CVideoDecoder::OpenFile() {
    int
        nMaxTryNumber = RETRY_NUMBER,
```

```
    nTryNumber;
  bool
    bRetry = false;
  for (nTryNumber = 0 ; nTryNumber < nMaxTryNumber; nTryNumber++) {
    // Close previous context if any
    if (m_pFormatContext) {
      pLogger->Print(CLogger::Level::Debug, "Closing previous format
context...");
      CloseFile();
    }
    m_pFormatContext = avformat_alloc_context();
    m_pFormatContext->interrupt_callback.callback =
CVideoDecoder::interrupt_callback;
    m_pFormatContext->interrupt_callback.opaque = this;


    // Pause before reconnection
    if (bRetry)
      sleep(5);
    else
      bRetry = true;
    AVDictionary *d{};
    if (enabled_tcp)
      av_dict_set(&d, "rtsp_transport", "tcp", 0);
    av_dict_set(&d, "stimeout", "30000000", 0);
    pLogger->Print(CLogger::Level::Debug, "Trying to open stream...");
```

```
        timer = CTimer::getCurrentTime();
        need_to_interrupt = true;
        int ret = avformat_open_input(&m_pFormatContext, sourceURL.c_str(), nullptr,
&d);
        need_to_interrupt = false;
        av_dict_free(&d);
        if (ret  == 0)
        {
            pLogger->Print(CLogger::Level::Debug, "Checking the stream...");
            if (!CheckAllStreams())
            {
                pLogger->Print(CLogger::Level::Info, "Check stream failed");
                continue;
            }
            pLogger->Print(CLogger::Level::Debug, "Allocating codec context...");
            if ((m_pCodecContext = avcodec_alloc_context3(m_pCodec)) == nullptr)
            {
                pLogger->Print(CLogger::Level::Info, "Failed to allocate codec");
                continue;
            }
            pLogger->Print(CLogger::Level::Debug, "Copying codec params to
context...");
            if (avcodec_parameters_to_context(m_pCodecContext,
m_pCodecParameters) < 0)
            {
```

```
            pLogger->Print(CLogger::Level::Info, "Failed to set stream parameters to
context");
            continue;
        }
        pLogger->Print(CLogger::Level::Debug, "Opening codec...");
        if (avcodec_open2(m_pCodecContext, m_pCodec, nullptr) < 0)
        {
            pLogger->Print(CLogger::Level::Info, "Failed to open codec");
            continue;
        }
        pLogger->Print(CLogger::Level::Debug, "Initializing color conversion...");
        if (!InitColorConversion())
        {
            pLogger->Print(CLogger::Level::Info, "Failed to init color conversion");
            continue;
        }
        pLogger->Print(CLogger::Level::Info, "Connected to stream!");
        return; // Success
    }
    else
    {
        char sError[128];
        av_strerror(ret, sError, 128);
        pLogger->Print(CLogger::Level::Debug, sError);
```

```
            pLogger->Print(CLogger::Level::Debug, "Stream is not available, waiting for
retry...");
        }
        if (GetCurrentState() == STOP || GetCurrentState() == STOPPING) {
            pLogger->Print(CLogger::Level::Info, "Stopped by user");
            return;
        }
    }
    pLogger->Print(CLogger::Level::Info, "Retry number is exceeded. Stream
stopped.");
    SetCurrentState(FAIL);
}
bool CVideoDecoder::InitColorConversion()
{
    if (m_pSwsContext)
        sws_freeContext(m_pSwsContext);
    m_pSwsContext = sws_getContext(m_nWidth, m_nHeight, m_nPixelFormat,
                                   m_nWidth, m_nHeight, AV_PIX_FMT_RGB24,
                                   SWS_BILINEAR, nullptr, nullptr, nullptr);
    if (m_pSwsContext)
        return true;
    else
        return false;
}
```

```
void CVideoDecoder::ApplyFilterToFrames(std::vector <AVPacket*> &vPacket,
AVBSFContext *pBSFilter)

{

   AVPacket

         *pPacket;

   for (int j = 0; j < vPacket.size(); j++) {

      pPacket = vPacket[j];

      av_bsf_send_packet(pBSFilter, pPacket);

      av_packet_free(&pPacket);

   }

   vPacket.clear();

   pPacket = av_packet_alloc();

   while (true) {

      if(av_bsf_receive_packet(pBSFilter, pPacket) == 0) {

         vPacket.push_back(pPacket);

         pPacket = av_packet_alloc();

      } else break;

   }

   av_packet_free(&pPacket);

   return;

}

void CVideoDecoder::work()

{

   if (!m_pFormatContext)
```

```
    return;
int error_counter = 0;
double
        fPreviousPTS = 0,
        fCurrentPTS;
int64_t
        nLastValidPTS = 0;
vector <AVPacket *>
        vPacket;
while (1)
{
    if(!IsRunning())
        return;
    AVPacket *pPacket;
    if ((pPacket = av_packet_alloc()) == nullptr)
        break;
    if (bHaveToReconnect)
    {
        switch (reconnectReason)
        {
            case RR_READ_ERROR:
                break;
            case RR_PTS_GAP:
                break;
```

```
            case RR_BUFFER_EXHAUSTED:
                ClearAllPackets();
                buffering_enabled = true;
                break;
            case RR_CLIENT_DISCONNECTED:
                if (clearOnVMSDisconnect)
                    ClearAllPackets();
                buffering_enabled = true;
                break;
        }
        pLogger->Print(CLogger::Level::Error, std::string("Reconnecting to EA due to ") + reasonDescription);
        pPacket->size = END_OF_BUFFER_SIZE; //End of Buffer packet
        PushNewPacket(pPacket);
        prepare();
        m_nBasePTSValue = nLastValidPTS;
        pLogger->Print(CLogger::Level::Verbose, "New base PTS=%d", m_nBasePTSValue);
        fPreviousPTS = 0;
        bHaveToReconnect = false;
        continue;
    }
    timer = CTimer::getCurrentTime();
    need_to_interrupt = true;
    int res = av_read_frame(m_pFormatContext, pPacket);
```

```
        need_to_interrupt = false;
        if (res < 0)
        {
            ++error_counter;
            pLogger->Print(CLogger::Level::Error, "Packet read error (%s).",
av_err2str(res));
            if(error_counter >= 1)
                PrepareForReconnect(RR_READ_ERROR);
            else
                sleep(1);
            continue;
        }
        if (pPacket->stream_index == m_nVideoStreamIndex)
        {
            if (pPacket->pts == AV_NOPTS_VALUE)
            {
                pLogger->Print(CLogger::Level::Verbose, "Packet with NOPTS ignored");
                av_packet_free(&pPacket);
                continue;
            }
            m_nPacketCounter++;
            pLogger->Print(CLogger::Level::Verbose, "Get new packet, PTS=%d,
switched to %d", pPacket->pts, pPacket->pts + m_nBasePTSValue);
            pPacket->pts += m_nBasePTSValue;
            pPacket->duration = m_nFrameDuration;
```

```
            nLastValidPTS = pPacket->pts;
            fCurrentPTS = GetPTSSeconds(pPacket->pts);
            if (fPreviousPTS == 0)
            {
                fPreviousPTS = fCurrentPTS;
            }
            else
            {
                if (fCurrentPTS - fPreviousPTS > m_fPTSDifference * m_fGapTolerance)
{
                    pLogger->Print(CLogger::Level::Error, "Detected high PTS gap: %f
seconds", fCurrentPTS - fPreviousPTS);
                    m_fGapAccumulator += fCurrentPTS - fPreviousPTS;
                }
                if (m_fGapAccumulator > m_fGapThreshold)
                {
                    av_packet_free(&pPacket);
                    PrepareForReconnect(RR_PTS_GAP);
                    continue;
                }

                fPreviousPTS = fCurrentPTS;
            }
            vPacket.push_back(pPacket);
            ApplyFilterToFrames(vPacket, avbsfContext1);
```

```
            ApplyFilterToFrames(vPacket, avbsfContext2);
            ApplyFilterToFrames(vPacket, avbsfContext3);
            for (int j = 0; j < vPacket.size(); j ++)
                PushNewPacket(vPacket[j]);
            vPacket.clear();
            error_counter = 0;
        }
        else
            av_packet_free(&pPacket);
    }
}
void CVideoDecoder::prepare()
{
    OpenFile();
}
int CVideoDecoder::GetMaxQueueSize()
{
    return defaultQueueSize;
}
bool CVideoDecoder::get_extradata_from_stream()
{
    uint8_t *extradata = m_pFormatContext->streams[m_nVideoStreamIndex]-
>codecpar->extradata;
    int size = m_pFormatContext->streams[m_nVideoStreamIndex]->codecpar-
>extradata_size;
```

```
    if (size < 4) {
        pLogger->Print(CLogger::Level::Debug, "Cant find stream extradata");
        return false;
    }
    auto params = Utils::NAL_splitter(extradata, size);
    if(params.size() < 2) {
        pLogger->Print(CLogger::Level::Debug, "Incorrect stream extradata");
        return false;
    }
    auto sps = &params[0];
    auto pps = &params[1];
    std::string sps_base64 = base64_encode(sps->first.get(), sps->second);
    std::string pps_base64 = base64_encode(pps->first.get(), pps->second);
    stream_extra_data = sps_base64 + ',' + pps_base64;
 //   printf("Converted extradata to SDP packet: %s\n", stream_extra_data.c_str());
    return true;
}
std::string CVideoDecoder::get_extradata()
{
    return stream_extra_data;
}
void *CVideoDecoder::getParams() {
    assert(m_nVideoStreamIndex != -1);
    return m_pFormatContext->streams[m_nVideoStreamIndex]->codecpar;
```

```
}
void *CVideoDecoder::getTimeBase() {
    assert(m_nVideoStreamIndex != -1);
    return &m_pFormatContext->streams[m_nVideoStreamIndex]->time_base;
}
int CVideoDecoder::GetVideoFPS() {
    return m_nActualFPS;
}
void CVideoDecoder::ClearAllPackets()
{
    while (m_qPacket.GetSize()) {
        AVPacket
            *pPacket = m_qPacket.PopFromQueue();
        av_packet_free(&pPacket);
    }

    pLogger->Print(CLogger::Level::Debug, "Packet queue was cleared");
}
void CVideoDecoder::PrepareForReconnect(TReconnectReason r)
{
    bHaveToReconnect = true;
    reconnectReason = r;

    switch (r) {
```

```
        case RR_READ_ERROR : reasonDescription = "read error"; break;

        case RR_PTS_GAP : reasonDescription = "high PTS gap"; break;

        case RR_CLIENT_DISCONNECTED : reasonDescription = "client
disconnection"; break;

        case RR_BUFFER_EXHAUSTED : reasonDescription = "exhausted buffer";
break;

        default : reasonDescription = "unknown reason";

    }

    pLogger->Print(CLogger::Level::Error, std::string("Prepare for reconnect due to ")
+ reasonDescription);

}

void CVideoDecoder::RemoveClients()

{

    bClientPresent = false;

    SetMaxQueueSize(defaultQueueSize, defaultMarginSize);

//    cerr << "Client out, Buffer size set to " << m_nMaxQueueSize << endl;

    pLogger->Print(CLogger::Level::Info, "VMS Client disconnected");

    pLogger->Print(CLogger::Level::Debug, "Max Queue size was decreased to %d",
m_nMaxQueueSize);

    if (!buffering_enabled) {

        if (reconnectOnVMSDisconnect)

            PrepareForReconnect(RR_CLIENT_DISCONNECTED);

    }

}

void CVideoDecoder::AddClients()

{
```

```
    bClientPresent = true;
    SetMaxQueueSize(defaultQueueSize, defaultMarginSize);
    pLogger->Print(CLogger::Level::Info, "VMS Client connected");
    pLogger->Print(CLogger::Level::Debug, "Max Queue size was increased to %d",
m_nMaxQueueSize);
}
void CVideoDecoder::SetBufferLength(int length, int lengthInFrames, int margin)
{
    m_nBufferLength = length;
    m_nBufferLengthInFrames = lengthInFrames;
    m_nBufferMargin = margin;
    m_nTimeout = length;
}
void CVideoDecoder::SetMaxQueueSize(int nMaxBufferSize, int nMarginSize) {
    defaultQueueSize = nMaxBufferSize;
    defaultMarginSize = nMarginSize;
    if (defaultMarginSize > nMaxBufferSize)
        defaultMarginSize = nMaxBufferSize;
    if (bClientPresent)
    {
        m_nMaxQueueSize = nMaxBufferSize * 2;

    } else
        m_nMaxQueueSize = nMaxBufferSize;
```

```
}
int CVideoDecoder::interrupt_callback(void *opaque)
{
    CVideoDecoder
        *pDecoder = (CVideoDecoder*) opaque;
    assert(pDecoder!= nullptr);
    if(!pDecoder->need_to_interrupt)
        return 0;
    auto st = pDecoder->GetCurrentState();
    if (st == STOP || st == STOPPING ||st == FAIL)
        return 1;
    int
        nRealTimeout = pDecoder->m_nTimeout +
CONNECT_TIMEOUT_ADDITION;
    if (nRealTimeout < 10)
        nRealTimeout = 10;
    if(CTimer::getCurrentTime() - pDecoder->timer >= nRealTimeout) {
        pDecoder->pLogger->Print(CLogger::Level::Error, "Timeout exceeded");
        return 1;
    }
    return 0;
}
```

ДОДАТОК В

Лістинг ONVIF-компоненту

```
#include "http_helper.h"
std::string http_helper::Cut_Accept_Encoding(const std::string &buf)
{
    std::string res;
    auto pos = buf.find("Accept-Encoding");
    if (pos != std::string::npos)
    {
        std::string part1 = buf.substr(0, pos);
        std::string part2 = buf.substr(pos);
        part2 = part2.substr(part2.find('\n'));                    //TODO: write regex
        res = part1 + part2.substr(part2.find_first_not_of('\n'));
        return res;
    }
    else
        return "";
}
int http_helper::Get_Content_Length(const std::string &head)
{
    std::smatch m;
    std::regex regex1("Content-Length:[^0-9]*([0-9]+).*");
    std::regex_search(head, m, regex1);
    std::string output = std::regex_replace(
```

```
            m.str(),
            std::regex("[^0-9]*([0-9]+).*"),
            std::string("$1")
    );
    if (output.empty())
        return 0;
    return std::stoi(output);
}
void http_helper::Set_Content_Length(std::stringstream &header, size_t newval)
{
    std::string strhead = header.str();
    std::string output = std::to_string(Get_Content_Length(strhead));
    int pos = strhead.find(output);
    strhead.replace(pos, output.length(), std::to_string(newval));
    header.str(std::string());
    header.clear();
    header << strhead;
}
void http_helper::Rewrite_XML(std::string &xml, const std::string &ip, const
std::string &port, const std::string &rtsp_port)
{
    std::stringstream stream(xml);
    std::stringstream ostream;
    try
```

```
    {
        boost::property_tree::ptree content;
        boost::property_tree::read_xml(stream, content,
boost::property_tree::xml_parser::trim_whitespace);
        boost::property_tree::ptree body;
        std::string path;
        if (content.get("env:Envelope.env:Body", "NONE") != "NONE")
        {
            path = "env:Envelope.env:Body";
            body = content.get_child(path);
        }
        else if(content.get("SOAP-ENV:Envelope.SOAP-ENV:Body", "NONE") !=
"NONE")
        {
            path = "SOAP-ENV:Envelope.SOAP-ENV:Body";
            body = content.get_child(path);
        }
        else
            return;
        std::regex regex (R"((\d+.){3}\d+(?::\d{0,5})?\b)");//Match IP and
port(optional)
        //std::regex(R"((\d+.){3}\d+)")
        std::string addr = ip + ":" + port;
        std::string rtsp_url = "rtsp://" + ip + ":" + rtsp_port + "/stream";
        if (body.get("trt:GetStreamUriResponse", "NONE") != "NONE")
```

```
        {
//          std::string newurl = std::regex_replace(prebuffer_rtsp_url, regex, ip);
            body.erase("trt:GetStreamUriResponse.trt:MediaUri.tt:Uri");
            body.put("trt:GetStreamUriResponse.trt:MediaUri.tt:Uri", rtsp_url);
        }
        else if(body.get("tr2:GetStreamUriResponse", "NONE") != "NONE")
        {
//          std::string newurl = std::regex_replace(prebuffer_rtsp_url, regex, ip);
            body.erase("tr2:GetStreamUriResponse.tr2:Uri");
            body.put("tr2:GetStreamUriResponse.tr2:Uri", rtsp_url);
        }
        else if(body.get("tds:GetServicesResponse", "NONE") != "NONE")
        {

            boost::property_tree::ptree cap = body.get_child("tds:GetServicesResponse");
            for (auto &service : cap)
            {

                for (auto &service_item : service.second)
                {
                    if (service_item.first.find("XAddr") != std::string::npos)
                    {
                        auto old = service_item.second.get_value("NONE");
                        service_item.second.put_value(std::regex_replace(old, regex, addr));
```

```
            }
        }
        if (service.first.find("Extension") != std::string::npos)
        {
            for (auto &extension : service.second)
            {
                for (auto &ext_item : extension.second)
                {
                    if (ext_item.first.find("XAddr") != std::string::npos)
                    {
                        auto old = ext_item.second.get_value("NONE");
                        ext_item.second.put_value(std::regex_replace(old, regex, addr));
                    }
                }
            }
        }
        body.put_child("tds:GetServicesResponse", cap);
    }
    else if(body.get("tds:GetCapabilitiesResponse", "NONE") != "NONE")
    {
        boost::property_tree::ptree cap =
body.get_child("tds:GetCapabilitiesResponse.tds:Capabilities");
        for (auto &service : cap)
```

```
{
    for (auto &service_item : service.second)
    {
        if (service_item.first.find("XAddr") != std::string::npos)
        {
            auto old = service_item.second.get_value("NONE");
            service_item.second.put_value(std::regex_replace(old, regex, addr));
        }
    }
    if (service.first.find("Extension") != std::string::npos)
    {
        for (auto &extension : service.second)
        {
            for (auto &ext_item : extension.second)
            {
                if (ext_item.first.find("XAddr") != std::string::npos)
                {
                    auto old = ext_item.second.get_value("NONE");
                    ext_item.second.put_value(std::regex_replace(old, regex, addr));
                }
            }
        }
    }
}
```

```
            body.put_child("tds:GetCapabilitiesResponse.tds:Capabilities", cap);
        }
        else if(body.get("tds:GetNetworkInterfacesResponse", "NONE") != "NONE")
        {

body.erase("tds:GetNetworkInterfacesResponse.tds:NetworkInterfaces.tt:IPv4.tt:Conf
ig.tt:Manual.tt:Address");

body.put("tds:GetNetworkInterfacesResponse.tds:NetworkInterfaces.tt:IPv4.tt:Config
.tt:Manual.tt:Address", ip);
        }
        else if(body.get("trt:GetSnapshotUriResponse", "NONE") != "NONE")
        {
            auto old = body.get("trt:GetSnapshotUriResponse.trt:MediaUri.tt:Uri",
"NONE");
            std::string newurl = std::regex_replace(old, regex, addr);
            body.erase("trt:GetSnapshotUriResponse.trt:MediaUri.tt:Uri");
            body.put("trt:GetSnapshotUriResponse.trt:MediaUri.tt:Uri", newurl);
        }
        else if(body.get("wsnt:SubscribeResponse", "NONE") != "NONE")
        {
            std::string field;
            if ((field =
body.get("wsnt:SubscribeResponse.wsnt:SubscriptionReference.wsa:Address",
"NONE")) != "NONE")
            {

body.erase("wsnt:SubscribeResponse.wsnt:SubscriptionReference.wsa:Address");
```

```
body.put("wsnt:SubscribeResponse.wsnt:SubscriptionReference.wsa:Address",
std::regex_replace(field, regex, addr));

        }

        else if ((field =
body.get("wsnt:SubscribeResponse.wsnt:SubscriptionReference.wsa5:Address",
"NONE")) != "NONE")

        {


body.erase("wsnt:SubscribeResponse.wsnt:SubscriptionReference.wsa5:Address");


body.put("wsnt:SubscribeResponse.wsnt:SubscriptionReference.wsa5:Address",
std::regex_replace(field, regex, addr));

        }

    }

    else if(body.get("tds:GetDeviceInformationResponse", "NONE") != "NONE")

    {

        body.erase("tds:GetDeviceInformationResponse.tds:Manufacturer");

        body.put("tds:GetDeviceInformationResponse.tds:Manufacturer", "VNTU");



        body.erase("tds:GetDeviceInformationResponse.tds:Model");

        body.put("tds:GetDeviceInformationResponse.tds:Model", "Master");



        body.erase("tds:GetDeviceInformationResponse.tds:FirmwareVersion");

        body.put("tds:GetDeviceInformationResponse.tds:FirmwareVersion",
"0.1.1.0");

    }

    else if(body.get("tds:GetScopesResponse", "NONE") != "NONE")
```

```
        {
            body.erase("tds::GetScopesResponse");
            boost::property_tree::ptree scope;
            scope.put("tds:Scopes.tt:ScopeDef", "Fixed");
            scope.put("tds:Scopes.tt:ScopeItem",
"onvif://www.onvif.org/Profile/Streaming");
            body.put_child("tds:GetScopesResponse", scope);
        }
        content.put_child(path, body);
        boost::property_tree::write_xml(ostream, content);
        xml = ostream.str() + "\r\n";
    }
    catch (boost::property_tree::xml_parser_error& err)
    {
//        std::cout << "NO XML:" << err.message() << "\n";
    }
}
```

ДОДАТОК Г

Алгоритм вирівнювання відтворення зображення

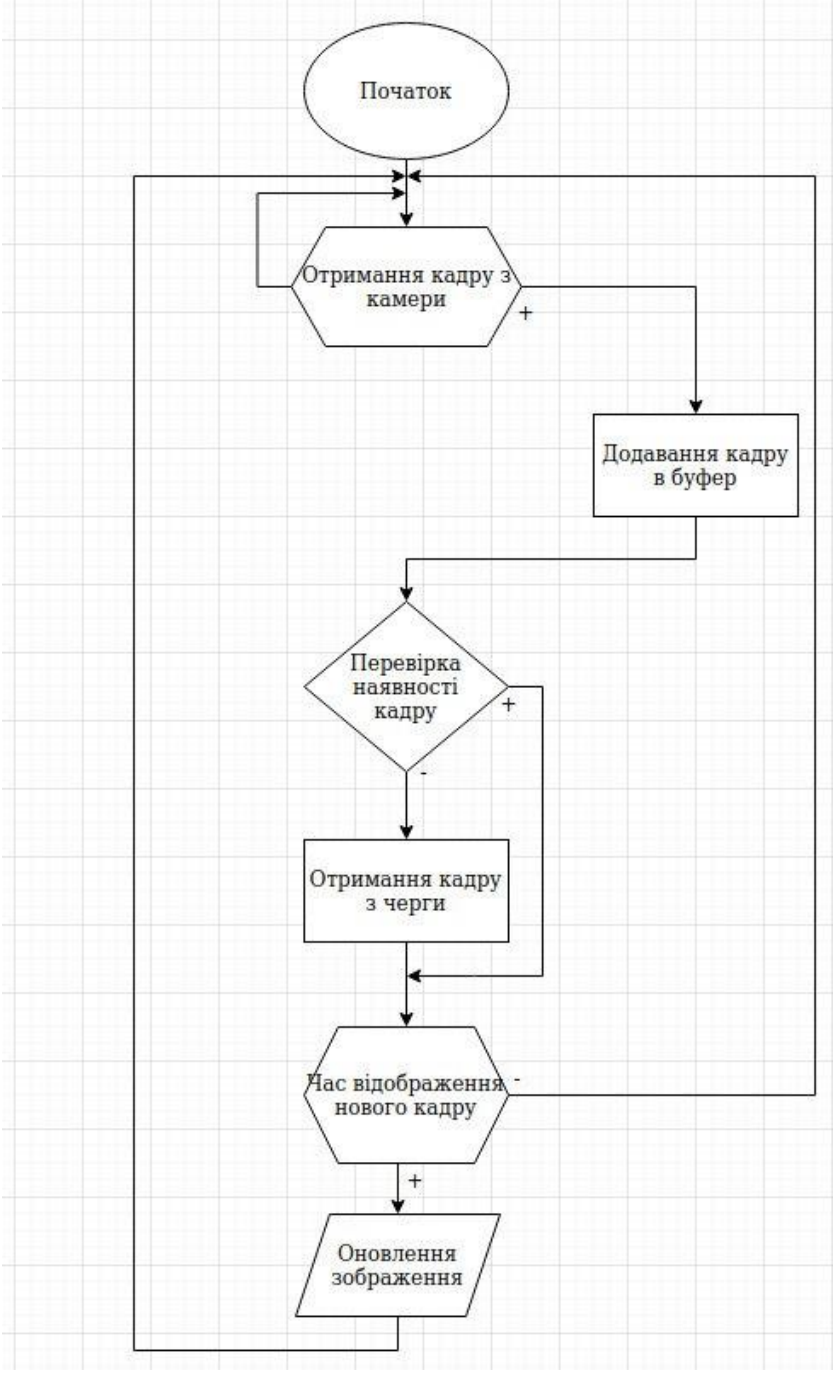

Рисунок Г.1 — Блок-схема алгоритму вирівнювання RTSP-компоненту

ДОДАТОК Д

Діаграма класів RTSP-модулю

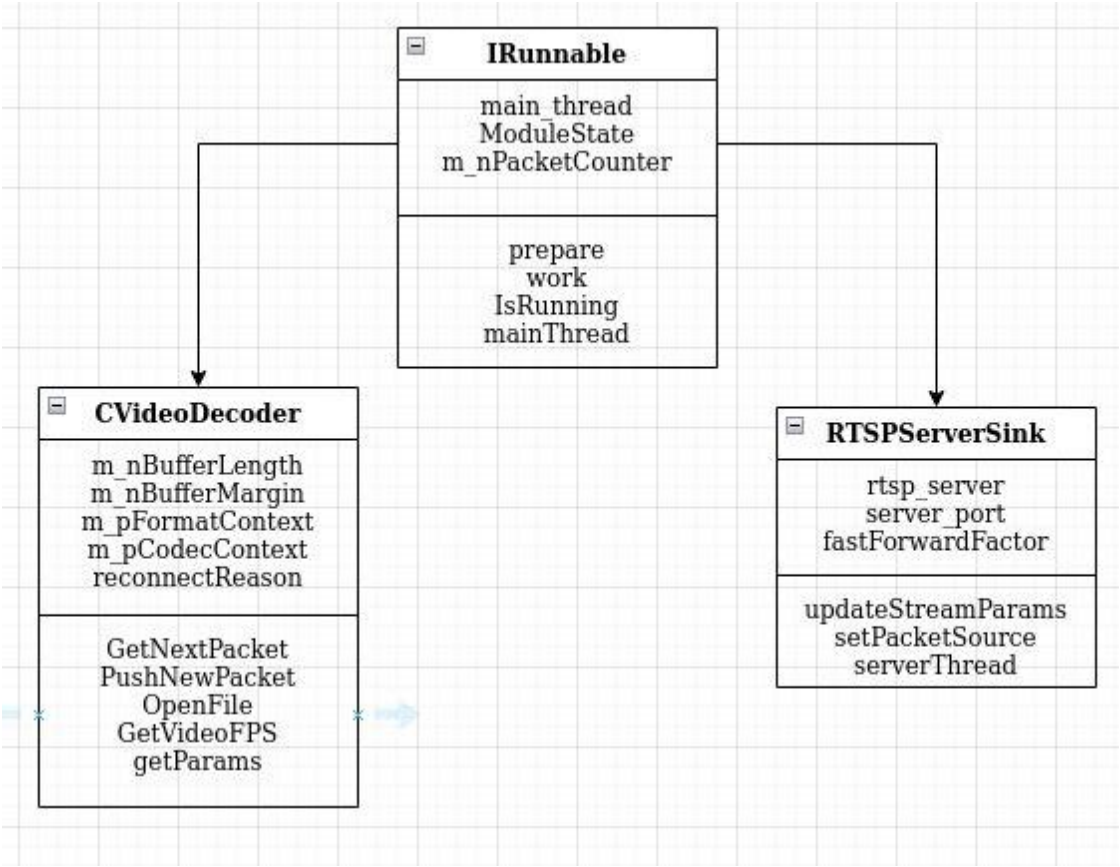

Рисунок Д.1 — Діаграма класів RTSP-модулю

ДОДАТОК Е

Діаграма класів ONVIF-модулю

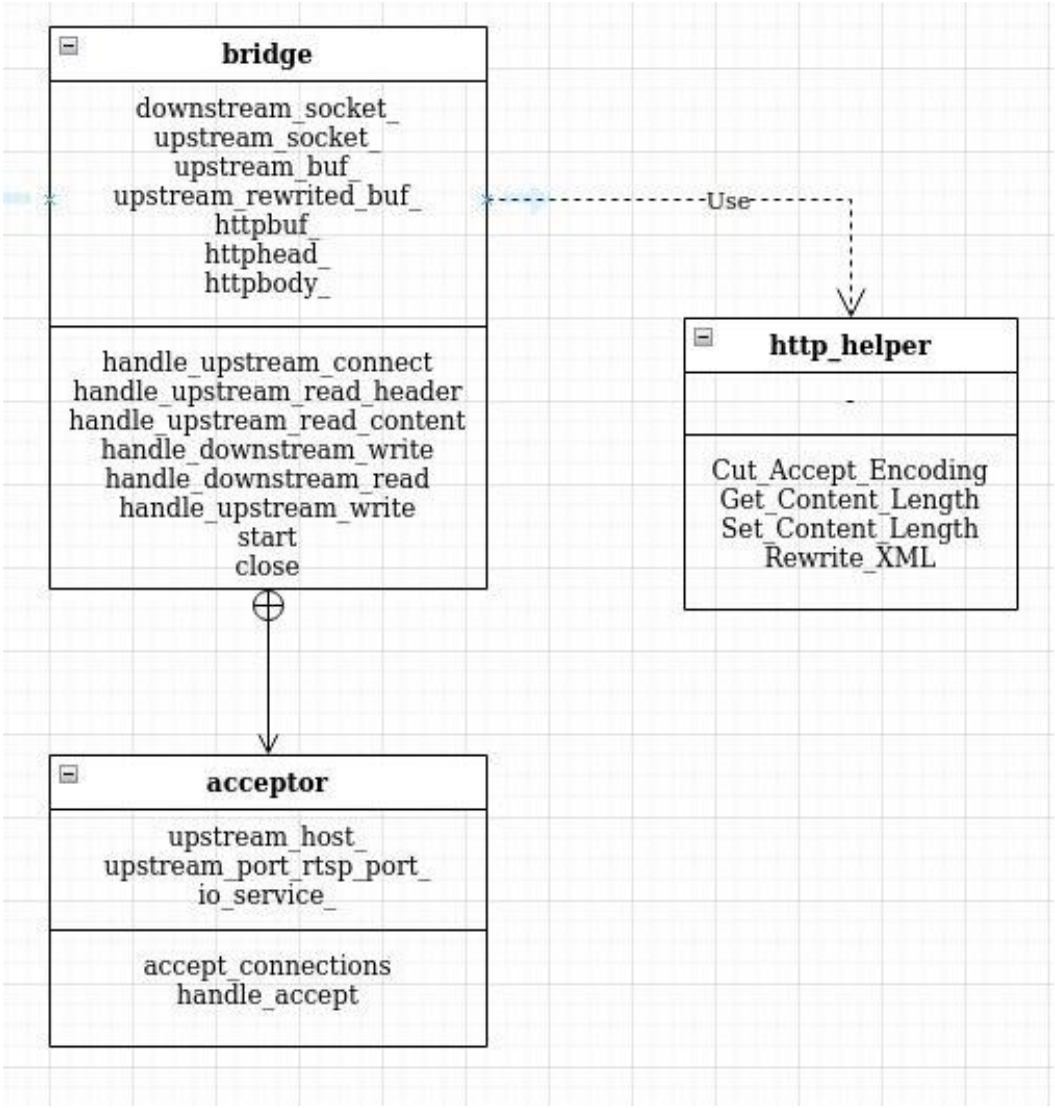

Рисунок Е.1 — Діаграма класів ONVIF-модулю

ДОДАТОК Ж

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: етод вирівнювання відтворення зображення під час передачі відеотрафіку з використанням емуляції IP-камери

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 81.6% Схожість 18.4%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Вишневський А.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Азарова А.О.
(підпис) (прізвище, ініціали)