Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)

Кафедра обчислювальної техніки
(повна назва кафедри)

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему:
"Програмний засіб для обробки даних та автоматизованого управління в системі відеоспостереження"

Виконав: студент 2 курсу, групи КІ —19м
спеціальності:
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки)

Бабій Вадим Ігорович
(прізвище та ініціали)

Керівник: д.т.н., проф. Азаров О. Д.
(прізвище та ініціали)

м. Вінниця — 2020 рік

ВСТУП

Відео — мабуть один із найпоширеніших термінів по всьому світу. Використання відеозйомки вже стало повсякденною нормою. Це чудовий спосіб запам’ятати момент із життя, або ж відпочити з друзями за переглядом фільму.

Хоч і більшість людей використовує відеозйомку майже щодня для різних розважаючих та індивідуальних цілей, проте це далеко не єдиний спосіб використання відео. Відеозйомка вже давно опередила зір людини, якщо в середньому людина може бачити не більше 40-45 перемінних зображень за секунду, то професійні відеокамери можуть обробити понад тисячу кадрів за секунду. За допомогою цього людство зробило величезний крок у розвитку технологій, було винайдено тисячі нових приладів, виведено сотні нових законів та вивчено десятки тисяч фізичних та біологічних процесів.

Для полегшення життя матері досить мати у кімнаті немовляти камеру та дистанційно переглядати за його діями, для поліпшення транспортного трафіку — досить встановити камеру на світлофорі та регулювати рух дистанційно. Можна уявити та створити будь-що: зафіксувати траєкторію польоту кулі, зафіксувати якесь явище, розповсюдити якісь дані, тощо. Фіксація природних явищ, перегляд тих зон, в яких людина не має можливості перебувати, зліт ракет, відеотрансляція з марсоходу, космічні явища, все це підштовхує людей до швидшого прогресу майже у всіх галузях наук.

Розробка системи відеоспостереження дозволить людям передбачати якісь небажані втручання інших людей, катастрофи, чи пограбування. Досить лише встановити камери відеоспостереження та спостерігати за їх трансляцією не покидаючи зону комфорту. Такий підхід розширює місця для робочого персоналу компаній або ж запроваджує окремі компанії по захисту та охороні об’єкту.

Базуючись на розглянутому, завдання подальшого вдосконалення методу обробки даних, автоматичного аналізу та управління передачі відеоданих є **актуальною задачею**. Перебійне мобільне мережеве з’єднання та великий об’єм

використання інтернет-трафіку вказують на актуальність даної проблеми та вимагає подальшого вирішення.

Метою дослідження магістерської роботи є вдосконалення методу передачі відео-трафіку та мінімізація втрат зображень із використанням автоматичної системи управління.

Задачі дослідження магістерської роботи:

— здійснити аналіз способів побудови систем онлайн відеотрансляцій та їх віддаленого запису;

— запропонувати кращий підхід для мінімізації втрат даних та збереження відеопотоку;

— розробити алгоритм та механізм автоматизованого управління системою відеоспостереження.

Об’єкт дослідження магістерської роботи — процес отримання, аналізу, оптимізації та збереження відео-трафіку шляхом автоматизованого механізму управління.

Предмет дослідження магістерської роботи — процедури опрацювання, конвертування, передавання та збереження відеоданих в рамках запропонованого методу мінімізації.

Методи дослідження магістерської роботи: використання алгоритмів ущільнення даних для оптимізації передавання трафіку, контейнерування даних з допоміжною інформацією, методи виявлення перевантаження мережі, джитер-буферування кадрів. У роботі використано принципи об’єктно-орієнтованого програмування мовами C++ та Python 3.6 для реалізації запропонованого підходу.

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

— удосконалено метод ущільнення та контейнерування відеоданих, що дозволяє економніше використовувати пропускну здатність інтернет мережі;

— удосконалено метод відправки та отримки відеоданих, який дозволяє мінімізувати втрати відеокадрів.

Практичне значення одержаних результатів магістерської роботи:

— розроблено алгоритм розподілу пріоритетів а також максимально допустимої вихідної пропускної здатності для кожної камери системи;

— розроблено програмне забезпечення мовою C++ для управління вихідною пропускною здатністю та оптимізацією трафіку;

— розроблено програму для зчитування розшифрування та зберігання або програвання відеопотоку;

— розроблено сервер мовою Python 3.6 із web-інтерфейсом для дистанційного мануального налаштування режимів роботи камер.

Апробація результатів магістерської роботи: зроблено доповідь на 49 регіональній науково-технічній конференції професорсько-викладацького складу, співробітників та студентів університету з участю працівників науково-дослідних організацій та інженерно-технічних працівників підприємств м. Вінниці та області.

Результати роботи опубліковані у матеріалах 49 регіональної науково-технічної конференції ВНТУ[11].

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ЗАСОБІВ ОПРАЦЮВАННЯ ДАНИХ У СИСТЕМАХ ВІДЕОСПОСТЕРЕЖЕННЯ

Відео — під цим терміном розуміють широкий спектр технологій запису, опрацювання, передачі, зберігання й відтворення візуального матеріалу на моніторах.У побутовому значенні відеоозначає 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.

Кількість (частота) кадрів в секунду це число нерухомих зображень, що змінюють один одного при показі 1 секунди відеозапису і створюють ефект руху об'єктів на екрані. Чим більше частота кадрів, тим більш плавним і природним буде здаватися рух. Мінімальний показник, при якому рух буде сприйматися однорідним, приблизно 16 кадрів в секунду (це значення індивідуально для кожної людини). У звуковому кінематографі частота зйомки і проекції стандартизована з 1932 року і становить 24 кадру в секунду. Системи телебачення PAL і SÉCAM використовують 25 кадрів в секунду (25 fps або 25 Гц), а система NTSC використовує 30 кадрів в секунду (точніше, 29,97 fps через необхідність кратного відповідності частоті піднесе). Комп'ютерне відео хорошої якості, як правило, використовує частоту 30 кадрів в секунду. Верхня гранична частота мерехтіння, що сприймається людським мозком, в середньому становить від 39 до 42 Гц і індивідуальна для кожної людини, а також залежить від умов спостереження. Деякі сучасні професійні відеокамери

можуть знімати з частотою до 120 кадрів в секунду. Спеціальні камери знімають з частотою до 1000 кадрів в секунду, що необхідно, наприклад, для детального вивчення структури вибуху або траєкторії польоту кулі.

Колірна модель — математична модель опису представлення кольорів у вигляді кортежів чисел (зазвичай з трьох, рідше чотирьох значень), званих колірними координатами або компонентами. Всі можливі значення кольорів, що задаються моделлю, визначають колірний простір.

RGB (абревіатура англійських слів red, green, blue — червоний, зелений, синій) або ЧЗС — кольорова модель, яка описує спосіб кодування кольору для відтворення кольору за допомогою трьох кольорів, які прийнято називати основними. Вибір основних кольорів обумовлений особливостями фізіології сприйняття кольору сітківкою ока.

RGB-модель є адитивною, де кольори отримують за рахунок додавання до чорного кольору. При відсутності випромінювання — немає ніякого кольору — чорний, змішання всіх трьох в певній пропорції — дає білий. Якщо колір екрану, освітленого кольоровим прожектором, позначається в RGB як (r1, g1, b1), а колір того ж екрану, освітленого іншим прожектором, — (r2, g2, b2), то при освітленні двома прожекторами колір екрану буде позначатися як (r1 + r2, g1 + g2, b1 + b2).

Зображення в даній колірній моделі складається з трьох каналів. При змішуванні основних випромінювань, наприклад, синього (B) і червоного (R), виходить пурпурний (M, magenta), зеленого (G) і червоного (R) — жовтий (Y, yellow), зеленого (G) і синього (B) — ціановий (C, cyan). При змішуванні всіх трьох основних випромінювань виходить білий колір (W, white).

Проектор і монітор здатний показати обмежену кількість кольорів. Їх кількість залежить від їх якості. Всі кольори кодуються в бітах, відведених на кожен піксель. За допомогою 1 біта можна закодувати лише 2 кольори — зазвичай чорний та білий. 2 біти дозволяють закодувати 4 кольори, тобто 2 в другій степені. 3 біти закодують вісім кольорів, 2 в 3 степені і так далі.

У компютерній техніці є стандарт — на один піксель 32 біта, або ж 4 байта αRGB, байт α використовується лише для кодування прозорості пікселя. При його

обробці колір пікселя буде змінено в залежності від значення α-байта і кольору попереднього пікселя (який стане «видно» через «прозорий» піксель), а потім α-байт буде відкинутий, і буде відображено лише RGB колір.

1.1 Обґрунтування важливості якісних та зручних систем відеоспостереження

Метою створення системи потокового відеотраслювання є насамперед захист рухомих та нерухомих об’єктів, хоча й це не єдина можливість використання таких систем. На даний момент відеотранслювання займає надзвичайно велику область використання у різних сферах діяльності, від охорони автомобільних парковок, до транслювання відеоданих з супутників та планети Марс. Такі системи дозволили людству зробити великий крок у різних сферах науки та досі допомагають щоденно вивчати дуже багато біологічних, хімічних та фізичних процесів. Зручність систем дозволяє використовувати їх майже кожній людині на планеті, тому що для цього не потрібно бути власником дорогого апаратного обладнання.

Таким чином, з мети потокового відеотранслювання можна зробити висновки щодо його завдання.

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 ущільнення даних. Найпоширеніші способи, які використовуються майже у всіх сучасних технологіях кодування відеоданих — це компрессори (кодеки) h263 / h264 / h265 та контейнери, за допомогою яких можливо передавати не лише відео-, а і аудіопотоки: AMV / AVI / RIFF та інші.

Для передачі даних у мережі використовуються TCP та/або UDP протоколи. Але для синхронізації відеопотоку цих протоколів недостатньо, саме тому був створений протокол RTSP (Real-TimeStreamProtocol), а згодом і RTMP.

Для передачі відео- та аудіоданих на основі протоколу RTSP створено досить велику кількість різних серверних та клієнтських частин. Наприклад такі клієнтські програвачі, як VLC, Microsoft Milestone VMS, FFplay та багато інших, і також серверні частини, найвідоміші з яких GStreamer, LibLive555 та FFServer. Хоч і розробка деяких модулів триває вже декілька, а то і декілька десятків років, всеодно у кожного з них є свої недоліки.

У таких системах дуже важлива підтримка кросплатформності, тому що клієнту буде дуже зручно використовувати наприклад домашній персональний комп'ютер як сервер, а смартфон як клієнтську частину. На сьогоднішній день вже кожен бажаючий може зробити малофункціональну, але все ж, систему відеоспостереження використовуючи лише веб-камеру, персональний комп'ютер та безкоштовне програмне забезпечення.

Хоч і ресурси для такої системи вже є майже у кожного, але такого рішення недостатньо для її використання на великих підприємствах. Саме тому були створені IP-камери (див рисунок 1.1). IP-камера — цифрова відеокамера для відеоспостереження, особливістю якої є передача відеопотоку в цифровому форматі у мережі Ethernet і TokenRing, що використовує IP протокол. Кожна IP-камера є мережевим пристроєм.

Також ці камери мають свою операційну систему, яка базується на ядрі Linux, та має зазвичай досить високі технічні характеристики. Налаштування камери здійснюється через панель управління, яка прив'язана до операційної системи, або ж через USB підключення, за допомогою програмного забезпечення виробника. Оскільки кожна камера має і використовує свою власну апаратну частину, це дозволяє їй працювати досить швидко та незалежно від інших.

Деякі камери мають можливість працювати деякий час автономно, у разі аварійного відключення електроенергії. На сьогодні такі камери широко поширені по всьому світу, їх використовують у банках та магазинах, офісах та «розумних» домах. І здебільшого цього вистачає користувачам.Налаштування можуть бути змінені під будь-які потреби користувача.

Кожна IP-камера є мережевим пристроєм і має свою IP-адресу і використовує свою власну апаратну частину. Саме тому така камера є досить актуальною та забезпечує стабільну роботу.

Рисунок 1.1 — IP-камера

IP-камера також здебільшого має вбудований режим День/Ніч, який дозволяє користувачам позбавитись цілої низки проблем, а саме таких як: зашумлення зображення при поганому освітленні та порушення передачі гамми кольорів.

1.2 Сучасні засоби для створення систем відеотранслювання

Для створення найменшої системи відеоспостереження потрібно мати USB веб-камеру або IP-камеру, комп'ютер з операційною системою Linux або Windows, та RTSP сервер, наприклад OBS Studio, VLC, GStreamer або інші.

Найпоширенішим способом створення відеотранслювання є відеопрогравач VLC, за допомогою якого можливо транслювати не тільки дані з камери, а також відеофайли, робочий стіл комп'ютера або смартфону, зображення або інший відеопотік з протоколом RTSP. У сучасному світі дуже стрімко розвиваються різні стрім-платформи, такі як Twich, YouTubeGaming, Instagram, Facebook та ін.. Для створення відеопотоку користувачу потрібно зробити мінімальну кількість дій, все інше зробить за нього програма.

На сьогоднішній день у тренді знаходиться відеотранслювання ігор, яке здійснюється за допомогою безкоштовної програми OBS Studio. Для створення такого транслювання потрібно встановити саму программу та налаштувати її на створення спеціальних «сцен» (див. рисунок 1.2) та створити обліковий запис. Так виконуючи мінімальну кількість дій ми вже можемо перетворити свій персональний комп'ютер або ж смартфон у повноцінну систему відеотранслювання.

Потрібно зауважити, що OBS Studio не єдина програма для створення відеострімінгу, їх ще є сотні, а то і тисячі, але багато із них є платними, тому переважна більшість користувачів користуються саме безкоштовною OBS Studio. Цей продукт дозволяє нам швидко та без спеціальної підготовки налаштувати відеотранслювання, але в ньому є дуже багато різних можливостей.

Програма не лише транслює робочий стіл, вона дозволяє робити окреме захоплення вікон, картинок, відеопотоків з інших трансляцій, передачі звуку, захоплення зображення з таких пристроїв як веб-камери, IP-камери, смартфони, та має можливість працювати з API різних стрімінг-платформ, що дозволяють підключати різні віджети, нотифікації, та багато іншого.

OpenBroadcasterSoftware підтримує широку гаму плагінів для розширення функціональності програми. Вони завантажуються як DLL файли з нативним кодом,

проте доступний wrapper, який додав підтримку плагінів, написаних на .NET Framework. Це забезпечує програмі великий набір гнучких налаштувань під кожного користувача.

Цей програмний додаток найчастіше використовується на стрімінг-сервісах, таких як: Twitch.tv, Youtube, HBOX та багато інших.

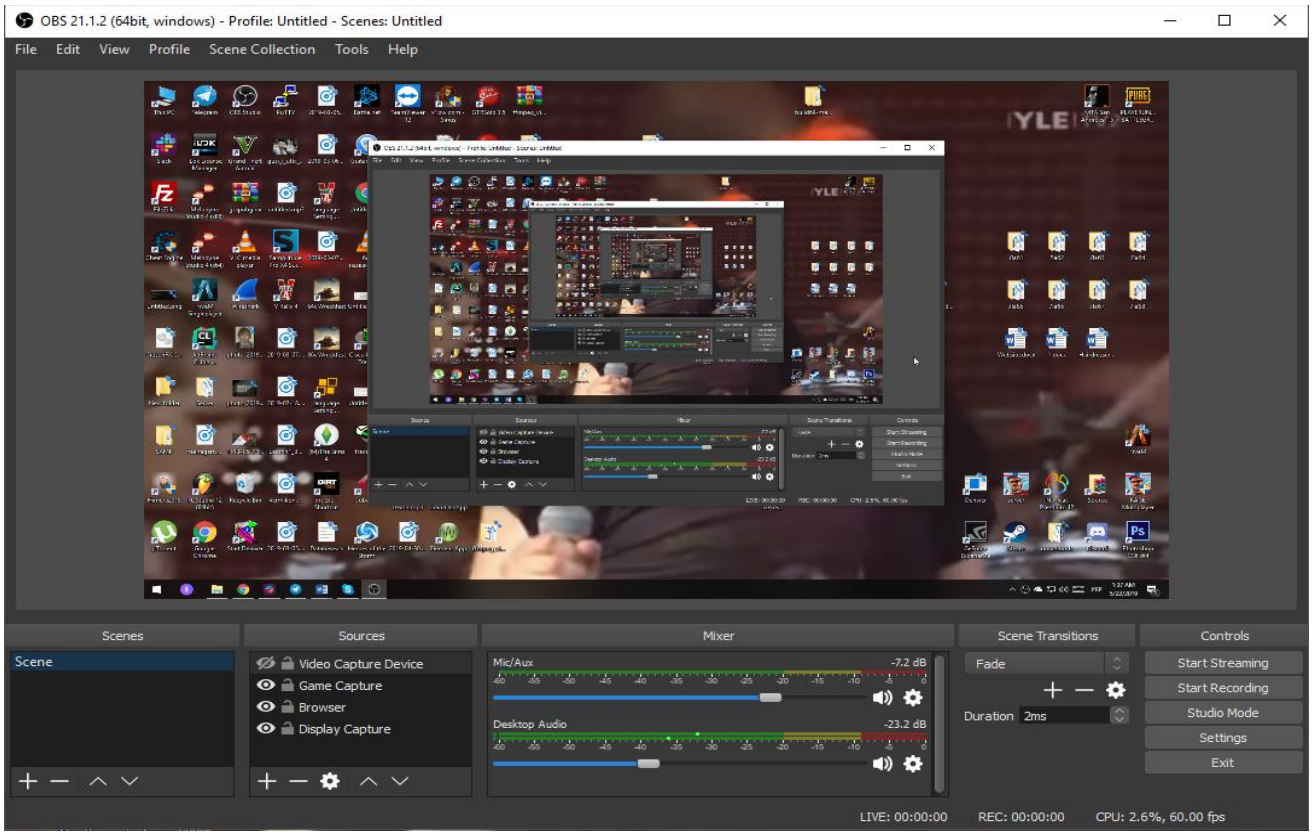

Рисунок 1.2 — Режим налаштування OBS Studio

Так, як програма запрограмована створювати «своє» вихідне зображення базуючись на сценах, які створив користувач, вона потребує хороших сучасних технічних характеристик комп’ютера та високошвидкісного інтернет з’єднання.

1.2.1 Огляд VLC-програвача

VLC mediaplayer (див. рисунок 1.3) — безкоштовний кросплатформниймедіапрогравач, 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. 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.

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

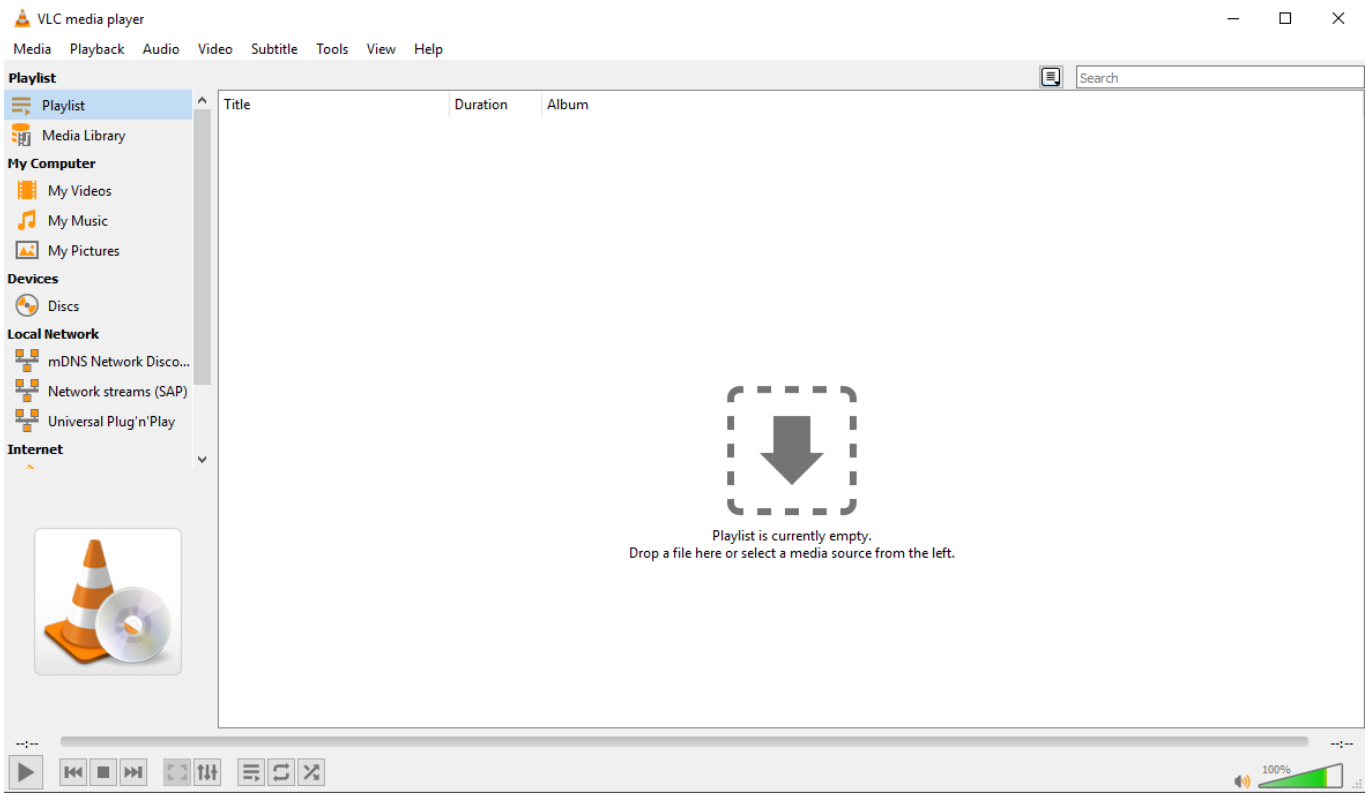

Рисунок 1.3 — VLC програвач

Для створення відеопотоку потрібно натиснути меню Media та обрати пункт Stream. Далі потрібно обрати, які дані мають транслюватись. Всього є чотири різних типів транслювання — це транслювання файлу, транслювання DVD диску, транслювання готової трансляції, та транслювання веб-камери і мікрофону. Після вибору даних для транслювання обирається протокол, по якому будуть транслюватись дані. Їх можна транслювати через RTSP, HTTP, RTMP, TCP, UDP,

MPEG, RTP або записувати в файл. Далі потрібно обрати компресор (кодек), зазвичай це h264 або MJPEG, налаштувати вихідний розмір зображення, частоту відправки key-frame'ів, бітрейт та інше. Після цього можна запускати трансляцію та підключатись до неї з іншого пристрою.

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.

VLC програвач можна використовувати не лише як клієнтську або серверну частину, досить часто його використовують як конвертор, за допомогою якого можливо змінити структуру файлу, конвертувати його у другий тип кодування, змінити його розширення або видалити деякі потоки даних.

VLC написаний мовами C, Objective-C та C++, що гарантують швидкість та незалежність його роботи від фреймворків та операційних систем. Спочатку інтерфейс програми базувався на wxWidgets, проте через багаточисельних проблем розробниками було прийняте рішення перейти на використання Qt 4 в якості графічного інтерфейсу. А для користувачів терміналу реалізований інтерфейс на Ncurses. Також проект є у відкритому доступі та розповсюджує своє API, що допомагає іншим розробникам користуватись, змінювати та створювати нові продукти.

1.2.2 Огляд Milestone VMS

Milestone VMS — платформа для запису та відтворення відеотранслювань. Цей медіаплеєр не лише відтворює відеопотік, але і водночас записує його. Програма має дуже великий обсяг можливостей для створення повноцінної системи відеоспостереження. Його використовують найчастіше у системах безпеки, тому-що програвач надає можливість транслювати одночасно понад 200 камер та зберігати їх

у той же час. У будь який момент можна переглянути будь який фрагмент записаного відеоматеріалу.

Його основна функція — це відтворення відеоданих, тому він може виступати лише у якості клієнтської частини, але спектр його можливостей надзвичайно великий і у декілька разів переважає своїх конкурентів. Можливість створення сітки віртуальних моніторів, які дозволяють водночас слідкувати за декількома камерами, Ретрансляція віртуальних моніторів, захоплення руху на відеоданих камер лише маленька частина всіх можливостей цього програвача.

Milestone VMS (див. рисунок 1.4) був створений компанією Microsoft та розповсюджується у платних та безкоштовних ліцензіях. Найбільшим недоліком програми є її залежність від операційної системи Windows, але у той же час цю програму використовують майже всі великі компанії, офіси, магазини, тощо. Офіційно в продаж програмне забезпечення з’явилось лише в 2017 році, а розроблявся проект понад 6 років. Програма дозволяє дуже гнучко налаштувати відображення трансляції.

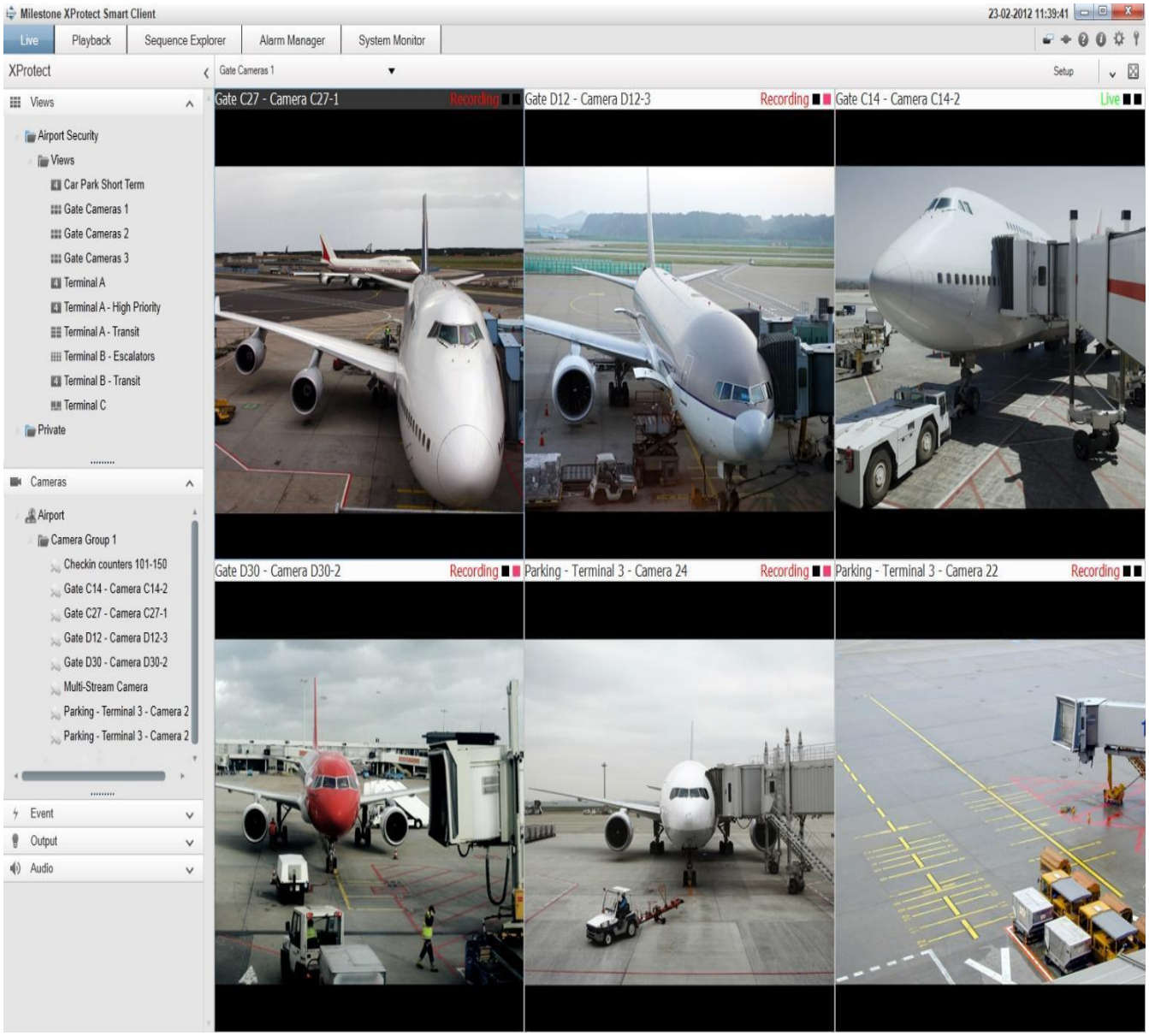

Рисунок 1.4 —Mircosoft VMS Milestone

Для налаштування програми потрібно мати камери, які транслюють дані по протоколу RTSP. Для підключення камер в програвач потрібно відкрити головні налаштування програвача, створити віртуальний сервер запису, а потім добавити камери та налаштувати їх з’єднання. Після того, як камери були добавлені потрібно створити віртуальну таблицю моніторів та вказати максимальну кількість камер, які можуть бути на сітці. Після цього потрібно відкрити програвач, увійти до локального облікового запису та прикріпити сітку. Після того, як переконались, що всі камери працюють та налаштовані правильно, можемо добавляти камери на сітку. Для збереження такої сітки і послідовності камер потрібно увімкнути режим «Встановлення».

Хоч і продукт є близьким до ідеалу, але і у нього є ряд технічних складностей. Зі збільшенням кількості одночасних трансляцій збільшується і технічне навантаження на процесор спричинене великим обсягом декодування даних. Тому підтримується продукт лише на сучасних комп'ютерах.

1.2.3 Огляд GStreamer

GStreamer— це набір бібліотек для побудови компонентів, що обробляють носії. Програми, які він підтримує, варіюються від простого відтворення файлів Ogg / Vorbis, потокової передачі аудіо / відео до складної обробки аудіо (змішування) та відео (нелінійне редагування). Був розроблений мовою C з прототипним ООП GObject. Є одним із найпоширеніших фреймворків для створення відеоредакторів, потокових серверів, програвачів, конвертерів та багато іншого. Працює на багатьох мажорних та мінорних операційних системах, таких як: Windows, Unix-подібні системи, BSD, Symbian, Android, iOS, macOS, Solaris та підтримує безліч компіляторів включаючи архітектури x86, x64, ARM, MIPS, PowerPC, SPARC. Gstreamer був портований на мови програмування C#, Python, Ruby, Java, C++, Go та інші.

Проєкт був заснований у 1999 році Еріком Волтінсеном. Згодом до нього приєднався ВімТейменс і вплинув на подальшу розробку системи. 11 січня 2001 року відбувся перший реліз та проєкт отримав інвестора і спонсора "РіджРан".

Розповсюджується GStreamer з ліцензіями GPL та має відкритий вихідний код, що дозволив багатьом розробникам внести свій вклад в технологію також. Проєкт розміщений на сервері freedesktop.org, за призначенням який повинен стандартизувати технології POSIX.

Базові можливості GStreamer:

— відеокодеки h.263, h.264, MJPEG, MPEG2, MPEG4, VP8, Dirac, WMV/VC1, Theora та інші;

— аудіокодекиSpeex, AAC, WavPack, MP3, FLAC, DolbyDigital (AC3), WMA, Vorbis, DTS/DCA, AMR NB/WB та інші;

— контейнери WebM, 3GPP, Ogg, MP4, Matroska, Quicktime, WAV, RealMedia, AVI, FLV, ASF, MPEG PS/TS та інші;

— протоколи RTP, RTCP, RTSP, RTMP, MMS, HTTP, SIP та інші;

— автоматизоване розпізнавання кодеків та контейнерів;

— методи отримання метаданих;

— субтитри з синхронізацією по часу;

— методи для відтворення звуків;

— підтримка перемикання субтитрів та звукових доріжок в реальному часі;

— перехід на будь-яку часову позицію потоку;

— перемотування, уповільнення та прискорення програвання потоку;

— встановлення кольорового формату та зміна розміру вихідного зображення, ресемплінг звукової доріжки;

— методи ущільнення звукової доріжки без втрати якості;

— методи рендерингу та підтримка графічних бібліотек.

1.2.4 Огляд фреймворку FFmpeg

FFmpeg — набір вільних бібліотек з відкритим вихідним кодом, які дозволяють записувати, конвертувати і передавати цифрові аудіо- та відеозаписи в різних форматах. Він включає libavcodec - бібліотеку кодування і декодування аудіо та відео і libavformat — бібліотеку мультиплексування і демультиплексування в медіаконтейнер. Назва походить від назви експертної групи MPEG і FF, що означає «fastforward». Проект заснував ФабрісБеллар (під псевдонімом ЖерарЛанто (фр. GerardLantau)) і до січня 2011 року керівництво їм здійснював МіхаельНідермайєр (нім. MichaelNiedermayer). Багато розробників FFmpeg брали участь в проекті MPlayer, і FFmpeg розташовувався на сервері MPlayer. 18 січня група розробників раптово відсторонила Нідермайєра від керівництва і, пояснюючи захоплення необхідністю боротьби з інфляцією і розбіжностями, спробувала взяти управління проектом в свої руки. Але в підсумку влада все одно залишилася в руках Нідермайєра, а результатом бурхливих суперечок стало створення окремої гілки проекту, в якій брали участь незадоволені станом речей розробники. Пізніше, через

триваючі розбіжностей з основною групою, вони заснували новий проект під назвою Libav.

FFmpeg розроблений під ОС на основі Linux, однак може бути скомпільовано під багато інших операційні системи. Поширюється під ліцензіями GNU LGPL або GNU GPL.

1.2.5 Огляд протоколу RTSP

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. 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. 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, і в той час як більшість керуючих повідомлень RTSP відправляються клієнтом на сервер, деякі команди відправляються в іншому напрямку (тобто від сервера до клієнта). Деякі типові запити HTTP, як наприклад OPTIONS, також доступні. За замовчуванням, номер порту транспортного рівня для RTSP — 554.

1.2.6 Огляд протоколу RTP

RTP був розроблений як протокол реального часу, з кінця в кінець (end-to-end), для передачі потокових даних. В протокол закладена можливість компенсації джиттера і виявлення порушення послідовності пакетів даних — типових подій при передачі через IP-мережі. RTP підтримує передачу даних для декількох адресатів через Multicast. RTP розглядається як основний стандарт для передачі голосу і відео в IP-мережах і спільно з кодеками.

Додатки, що формують потоки реального часу, вимагають своєчасної доставки інформації і для досягнення цієї мети можуть допустити деяку втрату пакетів. Наприклад, втрата пакета в аудіо-додатку може призвести до частки секунди тиші, яка може бути непомітна при використанні відповідних алгоритмів приховування помилок. Протокол TCP, хоча і стандартизований для передачі RTP, [3] як правило не використовується в RTP-додатках, так як надійність передачі в TCP формує тимчасові затримки. Замість цього, більшість реалізацій RTP базується на UDP. Крім цього, існують інші специфікації для транспортних протоколів SCTP і DCCP, але вони мало поширені.

Специфікація RTP описує два підпротокола:

— протокол передачі даних, RTP, який взаємодіє з передачею даних реального часу;

— протокол контролю, RTCP, який використовується для визначення якості обслуговування (QoS), зворотного зв'язку;

— синхронізація між медіа-потоками, займана смуга пропускання RTCP мала в порівнянні з RTP, зазвичай близько 5%.

+ Біти	0-1	2	3	4-7	8	9-15	16-31
0	Ver.	P	X	CC	M	PT	Порядковий номер
32	Мітка часу						
64	SSRC-ідентифікатор						
96	... CSRC-ідентифікатори ...						
96+(CC×32)	Додатковий заголовок (необов'язковий), містить довжину блоку даних — «AHL»						
96+(CC×32) + (X×(AHL+16))	Дані						

Рисунок 1.5 — Структура пакета

Ver (два біти) — це версія використаного протоколу. Актуальною на 2020 рік є версія під номером 2. P (один біт) —паддинг (Padding) на випадок, якщо RTP пакет заповнений нулями в кінці. X (один біт) —містить в собі розширення протоколу, якщо такий використовувався в RTP пакетах. CC (чотири біти) — містить в собі кількість CSRC-ідентифікаторів. M (один біт) —потрібен для обробки на боці програмного додатку, якщо це поле встановлено, воно ідентифікує про особливість пакету.

1.3 Варіативний вибір засобів для створення системи відеоспостереження

Найсучаснішим вибором фреймворку для створення системи відеоспостереження на даний момент часу є серверна сторона GStreamer та FFmpeg і клієнтська готова клієнтська сторона VMS Milestone. Для реалізації задач дипломної роботи було обрано саме їх, тому-щоGStreamer та FFmpeg є одними із найбільш масштабних фреймворків для реалізації системи відеоспостереження. У цьому фреймворку можна з легкістю створити процес транскодування даних, обробку клієнтів та динамічно змінювати його архітектуру його роботи. FFmpeg дозволяє нам працювати з відеоданими будь якого типу, тому-що у ньому реалізована підтримка майже всіх сучасних алгоритмів шифрування. Також це буде корисно для сбору статистики з відеопотоку.

2 РОЗРОБКА АЛГОРИТМУ РОБОТИ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ

2.1 Розробка методу обробки даних у системі відеоспостереження

Система автоматизації обробки даних поділятиметься на кілька складових:

— серверна частина забезпечує потокове відеотранслювання;

— алгоритм обробки даних забезпечує мінімізацію втрат даних та обробку їх якості;

— веб-додатокзабезпечує взаємодію системи та користувача;

— модуль налаштувань системи забезпечує зміну та збереження налаштувань роботи алгоритму в реальному часі;

— клієнтська частина забезпечує отримання потокового відео та подальшого відображення та/або запису у файл.

Схему взаємодії складових системи наведено на рисунку 2.1

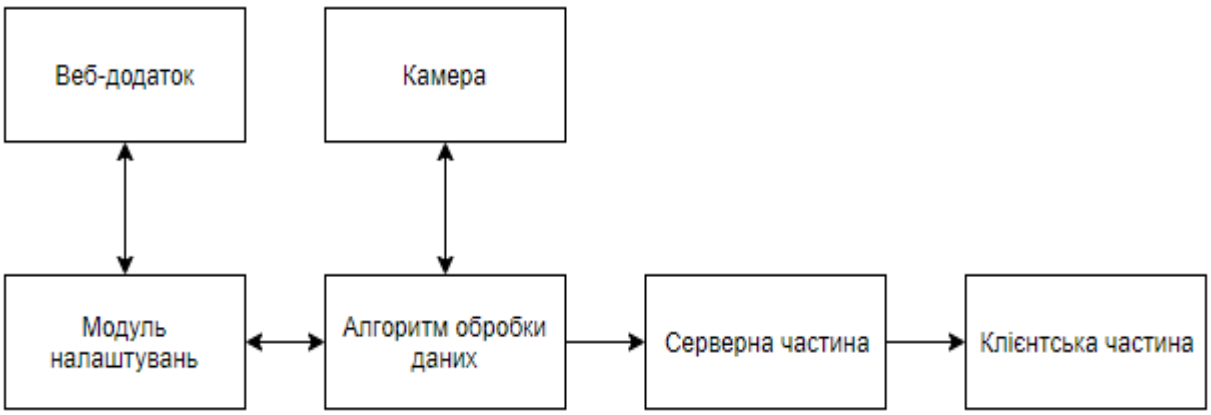

Рисунок 2.1 — Архітектура системи

Серверна частина буде розроблена за допомогою кросплатформних бібліотек GStreamer та FFMpeg. Бібліотека FFMpeg відповідає за зчитування та декодування даних із відеокамери, а бібліотека GStreamer забезпечує безперебійну та швидку роботу RTSP сервера.

Алгоритм обробки даних виконує дві функції. Перша з них, це математичне передбачення якості вихідного зображення та максимальна допустима швидкість

передачі вихідних даних (bitrate). А друга функція — це безпосередньо затримка або прискорення передачі даних, тимчасове збереження кадрів у оперативній пам’яті та мінімізація втрати даних.

Веб-додаток — важлива складова системи, яка відображає статистику роботи камер. З його допомогою користувач може легко змінювати налаштування алгоритму обробки даних та спостерігати за станом камер.

За допомогою модулю налаштувань користувач може вмикати та вимикати роботу алгоритму та змінювати його налаштування. Це сприяє гнучкості в роботі системи та швидкій зміні роботи режимів.

Клієнтська частина допомагає користувачу отримати, розкодувати та відобразити, або ж записати відеотрансляцію. Оскільки у жодному з існуючих програвачів не передбачена можливість синхронізації відеопотоку через презентаційну мітку кадру, буде створена також і клієнтська частина.

2.2 Розробка структури сервера

Для розробки серверної частини використаємо бібліотеку GStreamer. Вона забезпечить безперебійну роботу системи та передачу даних по протоколу RTSP. Основною частиною сервера є запровадження зв’язку з клієнтами та обмін даними між ними.

Основна концепція роботи цього модулю є відтворення pipeline’у, який вказується під час його запуску. Це допомагає серверу відтворити певний порядок дій у вказаній послідовності. На рисунку 2.2 зображена схема роботи сервера.

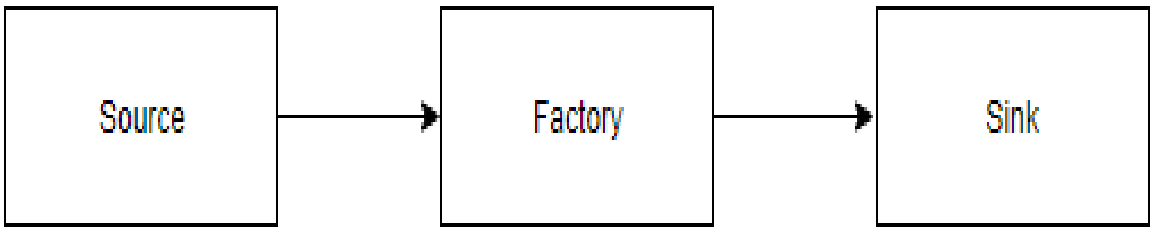

Рисунок 2.2 — схема роботи сервера

API бібліотеки GStreamer має дуже обширені можливості у його використанні. Вже створено майже всі можливі комбінації вхідних даних, обробки та вихідних даних. Також бібліотека забезпечує можливістю кодування/транскодування даних, відтворенням різних протоколів, але їх використання не підходить для розроблюваної архітектури, тому кодування та транскодування даних буде виконуватись перед передачою даних на сервер.

Source — це блок, який відповідає за отримання даних на сервер для послідуючого розповсюдження даних клієнтам. Цей блок містить свої буфери зберігання даних та регулюється за допомогою пайплайну. Бібліотека містить свій стандартний комплект цих блоків. Один з них, а саме appsrc, забезпечить запис даних у внутрішні буфери безпосередньо з розроблюваного алгоритму, а не з камери.

Factory — фабрика, що допомагає підключати та налаштовувати роботу з клієнтами, створює SDP пакет, для передачі інформації про відеопотік. За допомогою фабрики легко відслідити кількість користувачів та їх трафік.

Sink — блок, що відповідає за відправлення вихідних даних, роботу протоколу та його локального налаштування для кожного з користувачів. Він формує вихідні пакети та забезпечує обробку протокольних запитів.

Оскільки потрібно забезпечити стабільність та взаємозалежність окремих структур, сервер буде інтегрований в ядро програми окремим модулем, що дозволить з легкістю керувати ним та передбачати небажані ситуації.

Налаштування GStreamer бібліотеки проходить у декілька етапів:

— завантаження найновішої версії з офіційного репозиторію Github;

— встановлення всіх необхідних залежностей;

— встановлення плагінів;

— підготовка проекту до компіляції та безпосередня компіляція.

Використання бібліотеки FFMpeg дозволить зчитувати дані з камери та отримувати статистику про розмір кадрів, вхідний бітрейт, кількість кадрів за секунду, презентаційні мітки та інше. Бібліотека забезпечить конвертування даних у

потрібний формат для блоку source. Але спершу дані потрібно помістити у буфер алгоритму для збору статистики та мінімізації втрат.

2.3 Розробка структури алгоритму

Основною метою алгоритму є математичне передбачення максимально допустимого вихідного бітрейту і, у разі перевантаження мережі, його зниження для мінімізації втрат даних у системі відеоспостереження. Алгоритм повинен враховувати дані з усіх камер та розподіляти максимальний вихідний бітрейт на всіх. Для його реалізації використовуватиметься статистика вхідних даних з камер, таких як:

— вхідний бітрейт;
— upload швидкість;
— кількість камер у системі.
— максимальне пропусне «вікно» даних;
— кількість кадрів в секунду;
— режим роботи камери;
— розмір буферу алгоритму;
— кількість кадрів у буфері.

В основі математичних обчислень буде статистика попередніх вихідних даних. Для цього необхідно рахувати середні значення кожного елементу зі списку за останніх N секунд (кількість секунд буде задаватись користувачом).

Для обчислення середнього значення елементів потрібно створити модуль, який буде накопичувати статистику останніх отримуваних даних. Алгоритм повинен мати наступну структуру:

— обчислення середнього значення усіх важливих елементів;
— знаходження відношення максимального UploadRate камери до інших камер;
— врахування пропускної вихідної швидкості камери;

— обрахунки максимально допустимої швидкості вихідних даних з урахуванням попередньої статистики вхідних та вихідних швидкостей.

На рисунку 2.3 зображено алгоритм роботи передбачення максимально допустимої швидкості передачі вихідних даних.

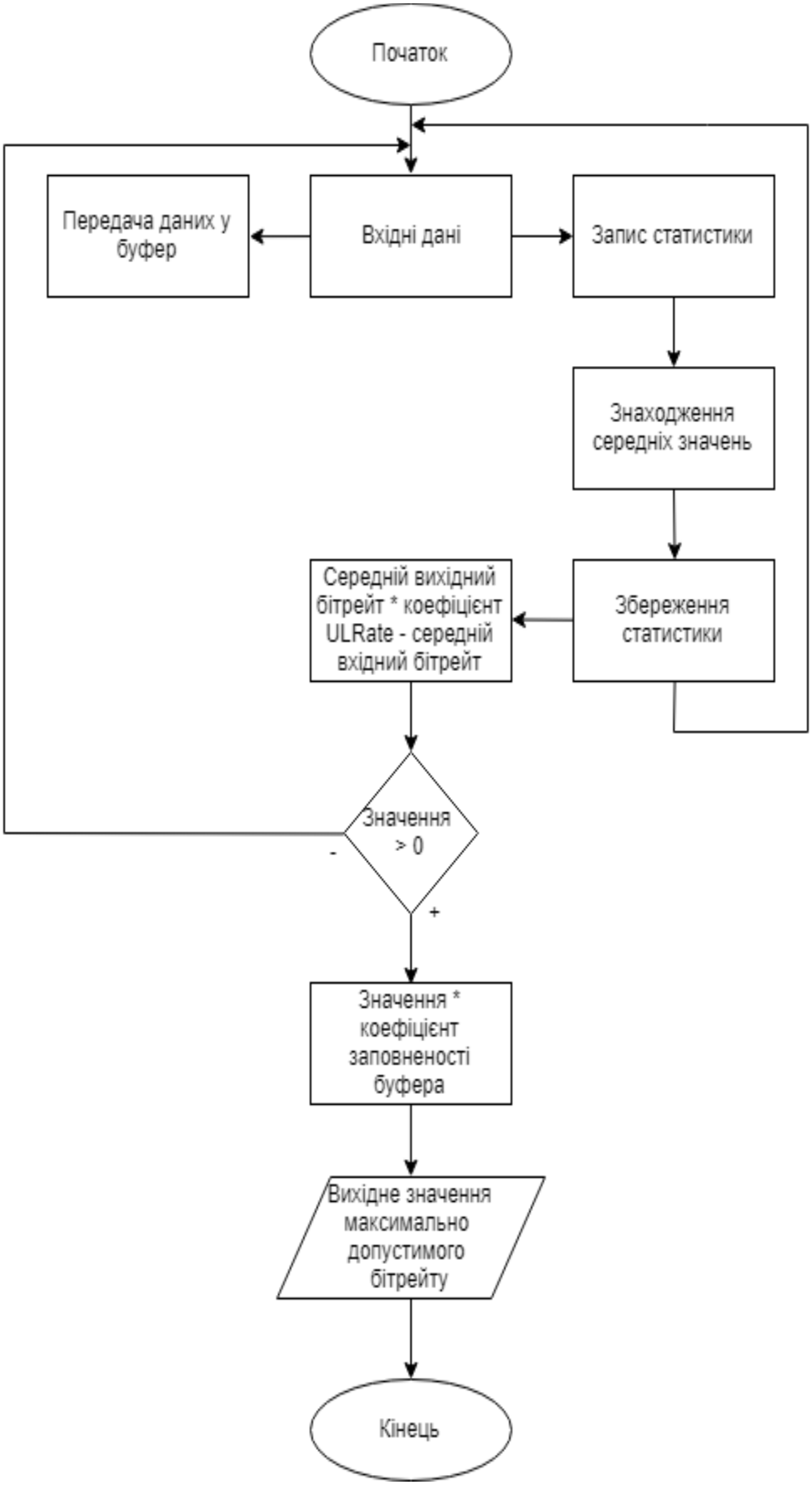

Рисунок 2.3 — Алгоритм математичного передбачення

Режим роботи алгоритму (увімкнутий або вимкнутий) повинен задаватись користувачем. Дуже важлива швидкість роботи алгоритму, тому-що обчислення будуть виконуватись щосекундно.

2.4 Розробка структури клієнтського програвача

Нажаль жоден з існуючих програвачів не має підтримки такого режиму роботи, тому необхідно розробити клієнтську програму, що дозволить «вирівнювати» кадри, які надсилає сервер. Основна ідея такої клієнтської частини заключається в тому, що вона буде під’єднуватись до серверу, отримувати кадри, «вирівнювати» їх відображення та створювати свій локальний сервер, до якого вже може підключитись будь-який відеопрогравач для систем відеоспостереження.

Перш за все клієнтська частина повинна буферизувати декілька секунд відеопотоку, для подальшого вирівнювання відображення. Програма повинна бути гнучкою у використанні, тому для її відтворення потрібні агрументи при запуску програми, а саме:

— RTSP URL потокового відеотранслювання;
— порт, на якому буде створено локальний сервер;
— кількість секунд затримки, для буферизації та вирівнювання даних;
— вибір мережевого протоколу підключення до серверу (TCP або UDP).

Такий набір аргументів забезпечить гнучкість роботи програми.

Синхронізація відео буде здійснюватись за допомогою часової презинтаційної мітки (PTS), мітки декодування (DTS) та тривалості відображення кадру (Duration). Кадри, які передаються через протокол RTP, зберігають лише тривалість відображення, відносно одиниці часу timebase. Зазвичай одиниця часу на потоках відеотранслювання дорівнює 1/90000, але це не завжди так, тому ці передача цих даних передбачена у протоколі RTSP. Для синхронізації буде використовуватись дельта часу між попереднім кадром і теперішнім. Оскільки протокол RTP підтримує лише передачу тривалості відображення та timebase потрібно почати рахунок PTS кадрів від нуля та щоразу змінювати за формулою «попереднє значення PTS + Duration». Таким чином кожен кадр буде набувати своєї корректної часової

призентаційної мітки. Тоді будуть готові всі необхідні дані для синхронізації та «згладжування» відеопотоку. Програма не потребує графічного інтерфейсу, тому-що працює у режимі закадрових обчислень.

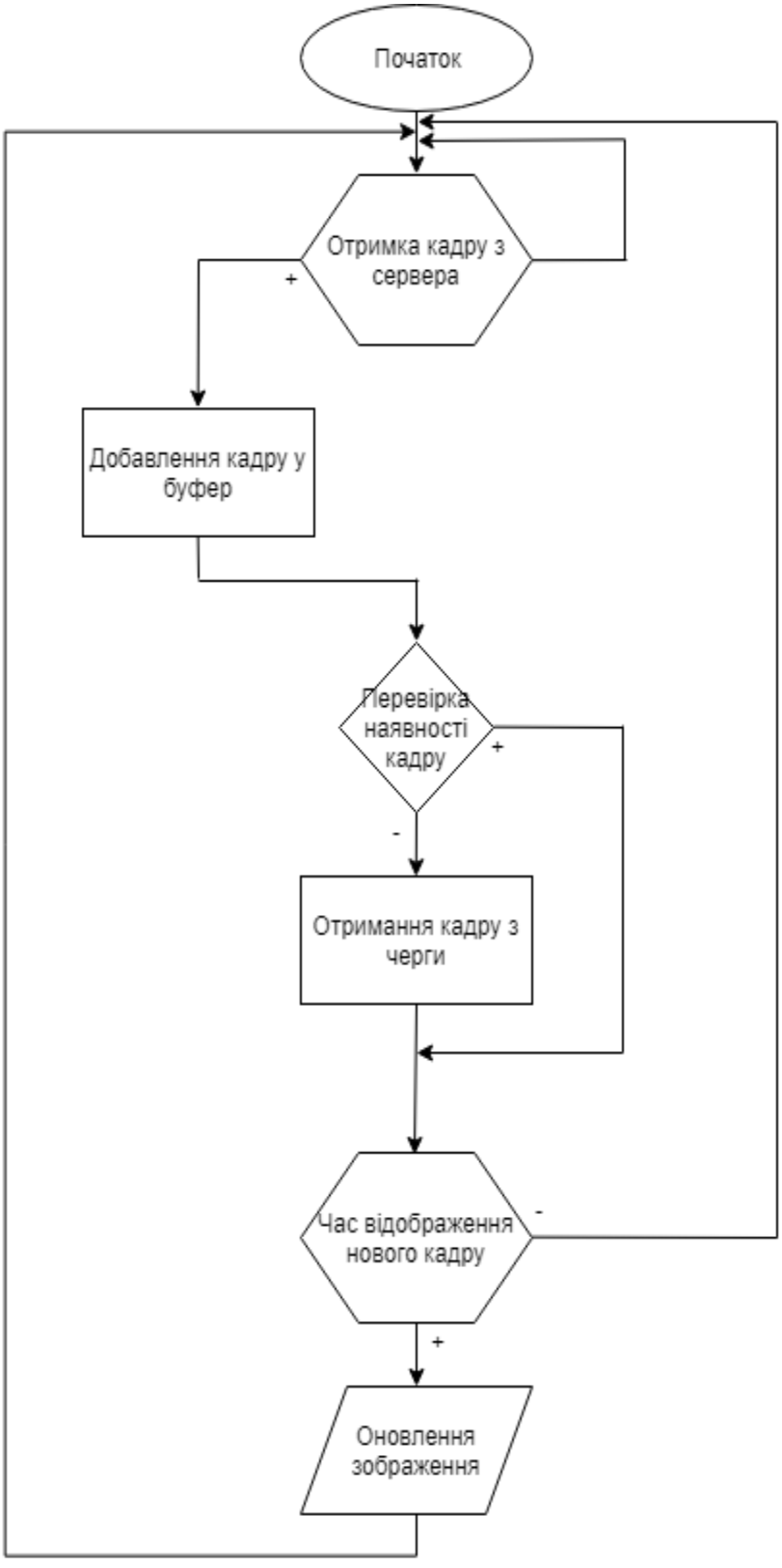

Рисунок 2.4 — Алгоритм «вирівнювання» фреймів

Також необхідно забезпечити автоматичне перепідключення клієнтської частини до основного серверу, у разі втрати зв’язку, та автоматичну обробку підключень локальних відеопрогравачів. Це дозволить програмі працювати у фоновому режимі безперебійно, швидко та не заважатиме користувачу спостерігати за основним програвачем.

2.5 Розробка користувацького інтерфейсу платформи

Для розроблюваної платформи необхідно розробити наступні сторінки:

— головна стартова сторінка, з якої кожен користувач розпочинає роботу з сайтом;

— реєстрація сторінка, за допомогою якої відбувається реєстрація нового користувача;

— вхід у систему сторінка, за допомогою якої виконується аутентифікація користувача;

— менеджер сторінка, за допомогою якої виконується управління камерами та алгоритмом;

— статистика сторінка, на якій зображена основна статистика вхідних та вихідних даних з камери;

— панель управління сторінка, через яку буде проходити користувацьке управління роботи алгоритмом та камер;

— створення камери сторінка, за допомогою якої буде доповнюватись основний список камер;

— графи сторінка, на якій будуть зображені графи вхідних та вихідних даних з камер;

— системні налаштування сторінка, за допомогою якої можна переглядати або змінювати налаштування відображення даних;

— адміністративна панель сторінка адміністратора, який може переглядати користувачів, їх останні дії та надавати права юзеру;

— панель управління камерою сторінка для управління камерою;

— перегляд інформації про камеру сторінка, за допомогою якої можна переглянути стан камери (лише для окремих моделей камер);

— перегляд потокової відеотрансляції сторінка, на якій можливо переглядати зображення камери;

— кодування сторінка, за допомогою якої встановлюються налаштування кодування зображення на камері;

— режим роботи сторінка, на якій задається режим роботи алгоритму;

— місцезнаходження сторінка з картою, яка відображає місцезнаходження камер.

Кожна сторінка поділятиметься на три частини:

— шапка (header);

— тіло (body);

— підвал (footer).

Взаємне розміщення вищевказаних частин утворюватиме шаблон, наведений на рисунку 2.5.

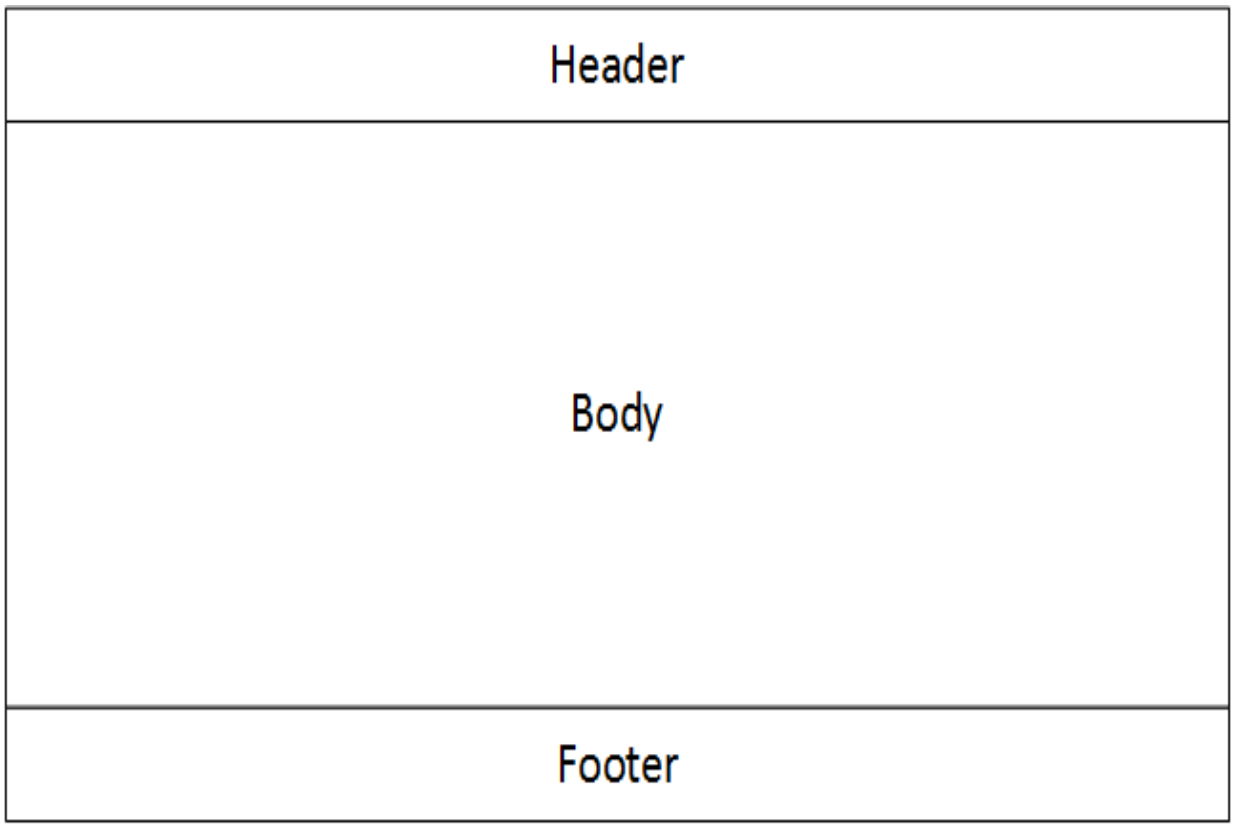

Рисунок 2.5 — Шаблон сторінки веб-додатку

Блок «Header» міститиме основні посилання для навігації по сайту. Блок матиме однакову внутрішню структуру та поведінку на усіх сторінках сайту. Він міститиме наступні елементи:

— посилання на сторінку «Головна», суміщене з логотипом сайту;

— посилання на сторінку «Менеджер»;

— посилання на сторінку «Панель Управління»;

— розділ «Камери», при наведенні на який відображатиметься список працюючих камер;

— кнопка «Вхід»/«Вихід», що відображається відповідно до статусу аккаунту.

Таким чином, зовнішній вигляд шапки сайту відрізнятиметься залежно від того, чи увійшов користувач у систему. Схему шапки сайту наведено на рисунку 2.6.

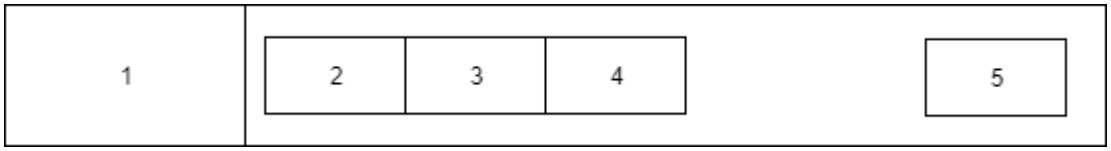

Рисунок 2.6 — Структурна схема шапки сайту

Блок «Footer» міститиме лише контактні дані розробника. Унікальний контент на кожній сторінці сайту розміщуватиметься у блоці «Body».

Розглянемо головну сторінку сайту, вона повинна відображати загальну статистику та діаграми роботи камер. В елементі body головної сторінки буде відображена google-карта з місцеположенням камер. За замовчуванням на графах повинна відображатись статистика усіх камер одночасно, тому до кожної камери буде прикріплений унікальний колір. Необхідна можливість переключення між камерами та вивід даних на графи виключно для однієї камери. Схема головної сторінки повинна мати набір таких елементів:

— карта з камерами з можливістю масштабування;

— кнопка для переключення режима роботи алгоритму;

— графи;

— меню вибору камери;

— статистика даних камери.

Для реалізації такої структури потрібно зберігати інформацію за методом витісняючої черги. Сторінка повинна відображати та оновлювати дані в реальному часі. Такий підхід дозволить користувачу максимально комфортно працювати з веб-менеджером. Також необхідно створити автоматизоване масштабування системи графів, що дозволить користувачу переглядати їх максимально детально. Пункт статистики даних повинен відображати відношення реальних даних до даних, обрахованих алгоритмом. Структурна схема тіла головної сторінки зображена на рисунку 2.7.

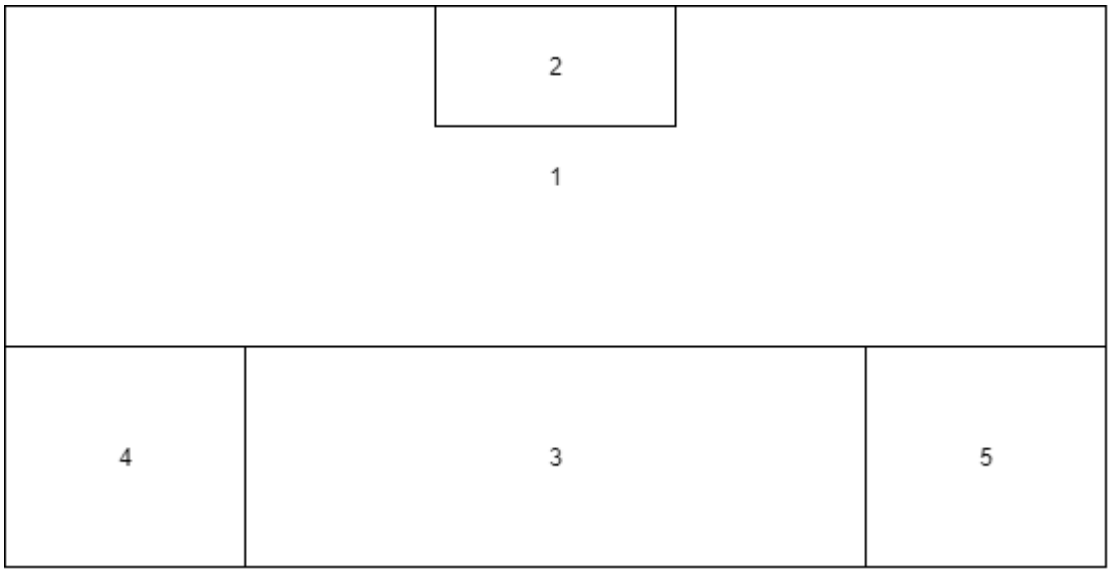

Рисунок 2.7 — Структурна схема головної сторінки

Сторінка «Менеджер» повинна містити перелік усіх камер та статистику, яку вони надсилають. Однією з можливостей цієї сторінки є конфігурування окремих модулей алгоритму та камер. Кожна з камер повинна мати вспливаюче меню, за допомогою якого можна перезавантажувати, видаляти та оновлювати камеру. На цій сторінці адміністратор повинен мати змогу додавати нові камери та маніпулювати над існуючими, переглядати детальну статистику та їх статус. Необхідно створити декілька меню, що дозволять налаштовувати алгоритм, модуль перевірки швидкості

з’єднання камер та відправку оновлених даних на камери, щоб ті оновили свої внутрішні дані для подальшої корректної роботи. Менеджер повинен відображати наступну інформацію:

— налаштування алгоритму;
— налаштування системи;
— налаштування графів головної сторінки;
— налаштування модулю перевірки швидкості камери;
— список камер.

Схема тіла сторінки менеджера зображена на рисунку 2.8.1.

Модулі налаштування повинні встановлювати «рукостискання» з базовим HTTP-сервером кожної з камер для подальшого обміну інформацією, налаштувань та статистики. Задля безпеки системи менеджер повинен створювати токен автентифікації з кожною камерою, це допоможе уникнути небажаного втручання третіх особ. Для передачі даних буде використовуватись метод POST.

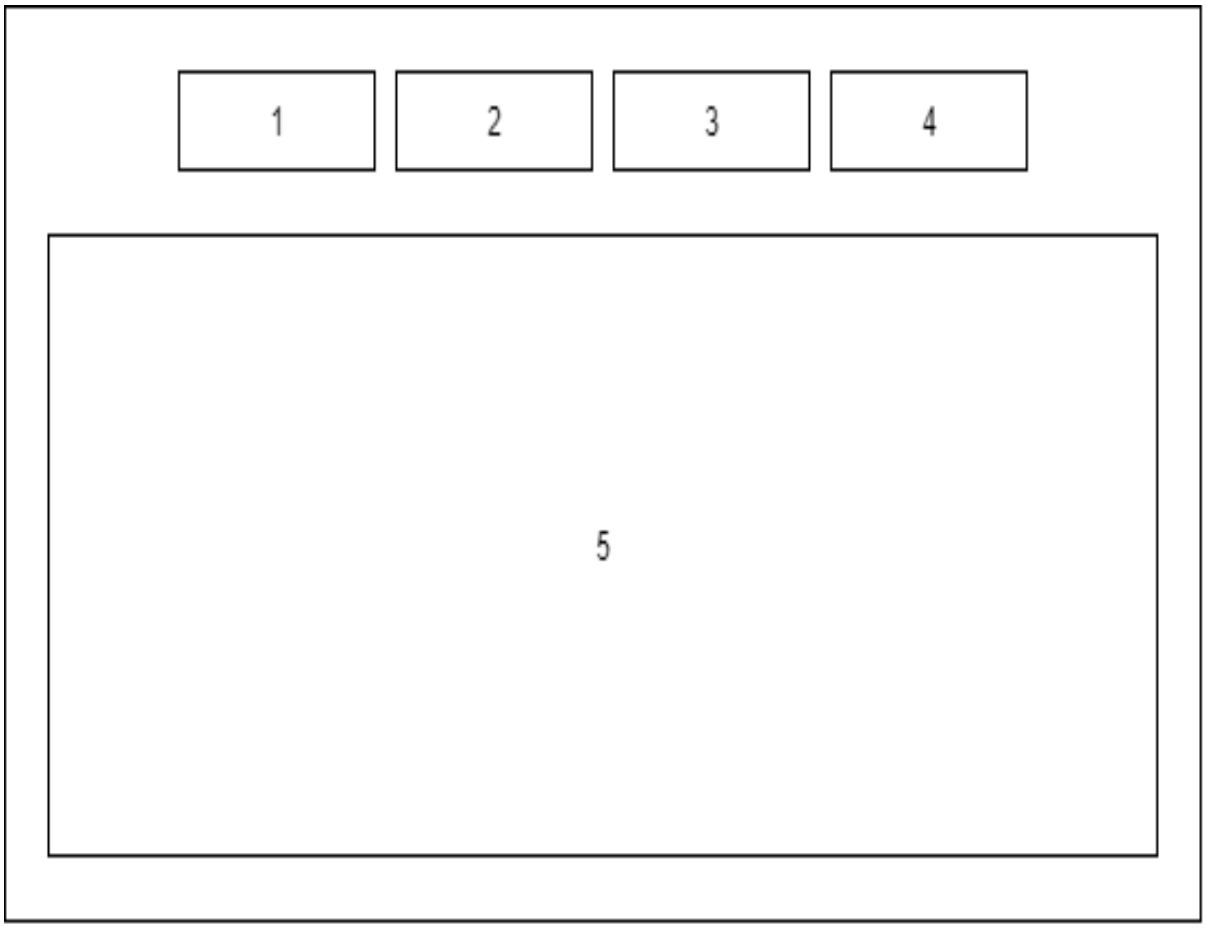

Рисунок 2.8.1 — Тіло сторінки менеджеру

Кожен модуль налаштувань повинен мати свій унікальний список змінних, але модулі повинні наслідуватись від одного інтерфейсу. Схема інтерфейсу модулів налаштувань зображена на рисунку 2.8.2.

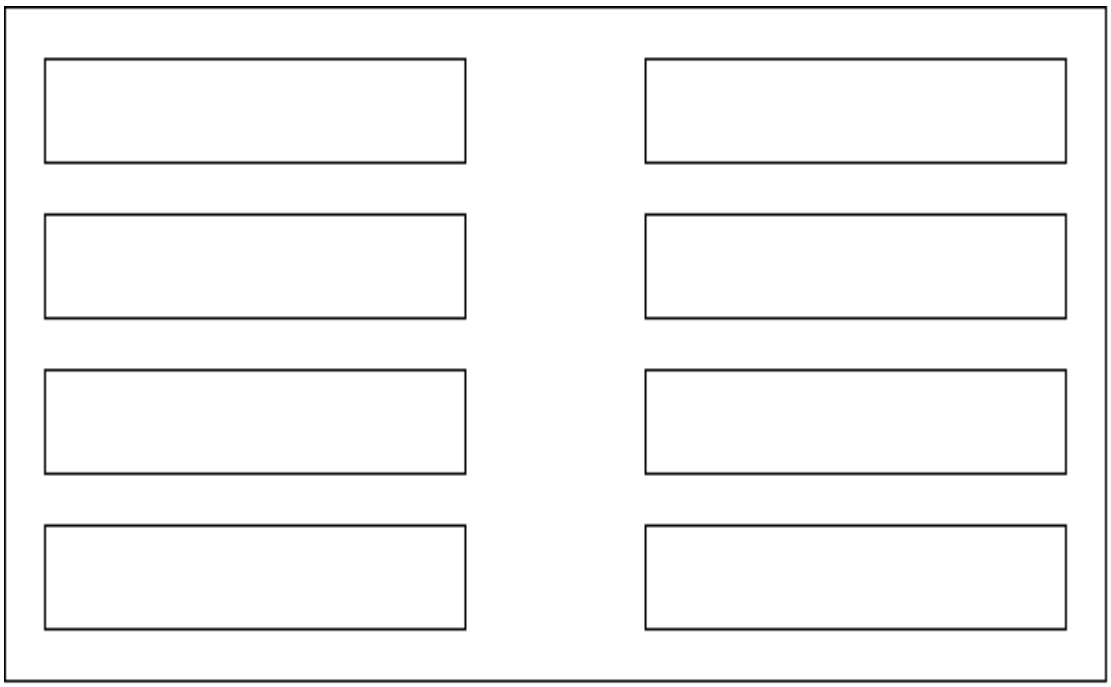

Рисунок 2.8.2 — Інтерфейс модулів налаштувань

Список камер є найголовнішою частиною сторінки. Кожна з камер повинна містити в собі статичну та динамічну інформацію і оновлюватись у режимі реального часу. Потрібно розробити 2 режими перегляду списку. Перший з них має відображати статичні дані безпосередньо кожної з камер, а саме:

— IP-адреса камери;
— ім’я камери, задається при добавленні у список;
— версія ядра, на якому в даний момент працює камера;
— URL потокової відеотрансляції;
— статус камери;
— UL швидкість;
— останнє оновлення статистики.

Такий набір елементів дозволить користувачу швидко дізнатись головну інформацію про камеру та статус її роботи (рисунок 2.8.3).

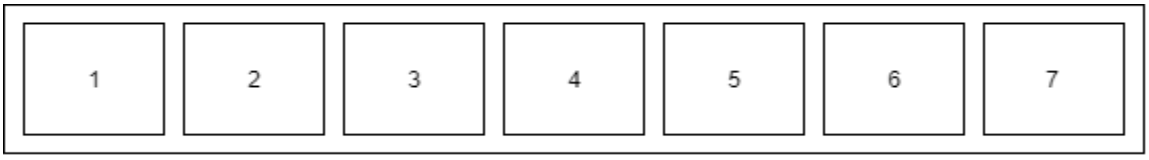

Рисунок 2.8.3 — Схема відображення інформації статичних даних камери

Другий режим повинен доповнювати цей список розгорнутою інформацією даних, які надсилає камера. Перелік цих даних є наступним:

— часова мітка формування даних;
— вхідний бітрейт;
— вихідний бітрейт;
— розмір буферу;
— кількість кадрів в буфері;
— середній розмір кадру;
— режим роботи камери;
— налаштування роботи транскодера;
— меню управління камерою.

Також розгорнуте меню камери має дозволяти користувачу регулювати режим роботи камери. Схематичне зображення розгорнутого списку на рисунку 2.8.4.

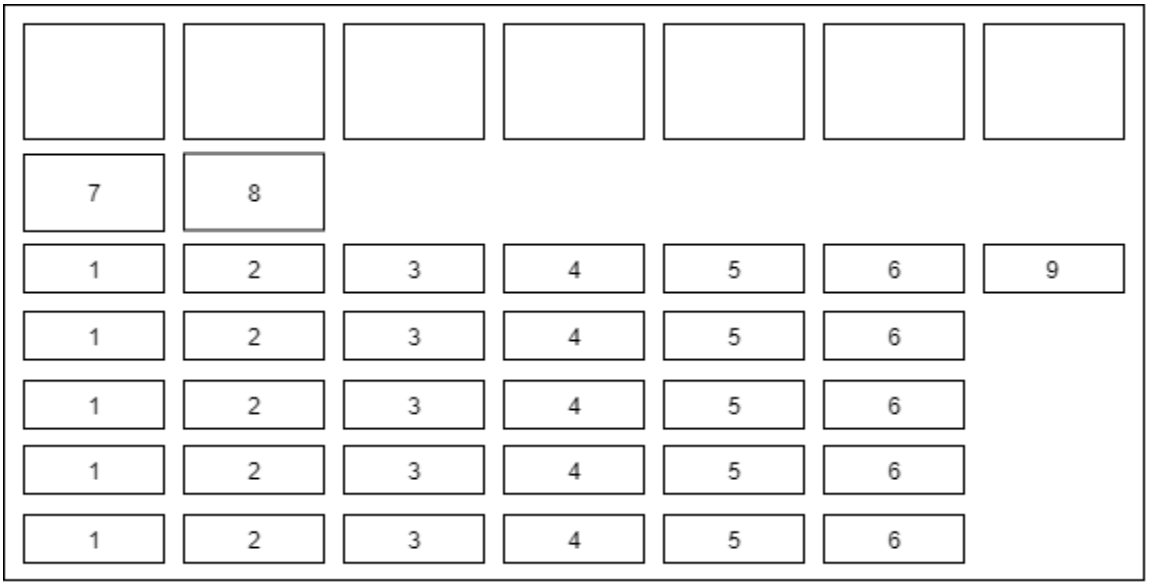

Рисунок 2.8.4 — Схематичне зображення розгорнутого списку надісланих даних камери

Такий список повинен надати користувачу всю необхідну інформацію про роботу камери. По ньому дуже просто буде відстежити всі кроки роботи ядра камери.

Всі інші сторінки будуть виконані за стандартними шаблонами. Тому, що вони відповідають принципу роботи цього веб-додатку.

Панель користувача міститиме в собі інформацію про останні дії, які здійснював сам користувач, зміну паролю, інформацію про користувача, налаштування авторизації, прав доступу та інше.

Панель управління дозволить надавати адміністративних прав користувачам та видаляти їх із локальної бази даних. Доступ до панелі управління буде здійснюватись по унікальному логіну та паролю суперадміністратора.

Набір таких елементів робить загальну систему дуже гнучкою для управління нею через веб-додаток. Кожен із компонентів системи доповнює один одного — це дозволяє користувачу швидко зорієнтуватись у принципі роботи всього проекту, а ручне налаштування роботи алгоритму дозволить швидко змінювати принцип роботи основного ядра системи. Також такий підхід допоможе користувачу відстежити помилку, у разі несправності системи. UI веб-додатку був максимально стиснутий, для швидкої навігації по ньому.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПЛАТФОРМИ МОВАМИ C++ ТА PYTHON

3.1 Варіантний аналіз і обґрунтування вибору програмних засобів

Перед початком розробки програмного коду платформи необхідно дослідити доступні програмні засоби та відібрати такі, що найкраще відповідають розробленому додатку.

Вибір програмних засобів буде відбуватися для таких складових платформи:

— ядро камери;

— алгоритм обчислення пропускної швидкості;

— веб-додаток.

Для розробки ядра, алгоритму обчислення та веб-додатку буде проведений аналіз мов програмування, а саме C++, C#, Java та Python.

C++ — компільована, статично типізована мова програмування загального призначення. Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає в себе поширені контейнери і алгоритми, введення-виведення, регулярні вирази, підтримку багатопоточності і інші можливості. C++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником мовою C, найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування. C++ широко використовується для розробки програмного забезпечення, будучи одним з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також ігор. Існує безліч реалізацій мови C++, як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder і інші. C++ зробив величезний вплив на інші мови програмування, в першу чергу на Java і C#. Синтаксис C++ успадкований від мови C. Одним з принципів розробки було збереження сумісності з C. Проте

безліч програм, які можуть однаково успішно транслюватися як компіляторами C, так і компіляторами C ++ не включає всі можливі програми на C.

C# (вимовляється Сі Шарп) — об'єктно-орієнтована мова програмування. Розроблена в 2001 роках групою інженерів компанії Microsoft під керівництвом АндерсаХейлсберг і Скотта Вільтаумота як мову розробки додатків для платформи Microsoft .NET Framework. Згодом був стандартизований як ECMA-334 і ISO / IEC 23270. C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, змінні, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі в форматі XML. Перейнявши багато від своїх попередників мов C++, Delphi, Модула, Smalltalk і, особливо, Java C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинне успадкування класів (між тим допускається множинна реалізація інтерфейсів).

Java — строго типізована об'єктно-орієнтована мова програмування загального призначення, розроблена компанією SunMicrosystems (в подальшому придбаної компанією Oracle). Розробка ведеться співтовариством, організованим через JavaCommunityProcess; мова і основні модулі реалізують його технології і поширюються за ліцензією GPL. Права на торговельну марку належать корпорації Oracle.

Python — високорівнева мова програмування загального призначення, орієнтований на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий набір корисних функцій. Активно розвивається мова програмування, нові версії з додаванням / зміною мовних властивостей виходять приблизно раз в два з половиною роки. Язик не піддавався офіційній стандартизації, роль стандарту де-факто виконує CPython, що розробляється під контролем автора мови. На даний

момент Pythonзаймає друге місце в рейтингу TIOBE з показником 12,12%. Результати порівняльного аналізу наведено у таблиці 3.1.

Таблиця 3.1 — Порівняльний аналіз мов програмування

Критерій	C++	C#	Python	Java
Підтримка об’єктно-орієнтованого програмування	+ (1.0)	+ (1.5)	+ (1.6)	+ (1.5)
Автоматичне керування пам’яттю	+ (0.3)	+ (0.5)	+ (1.0)	+ (0.5)
Кросплатформність	- (0.0)	+ (0.2)	+ (0.5)	+ (0.7)
Висока швидкодія	+ (1.0)	+/- (0.5)	+/- (0.5)	+/- (0.3)
Підтримка бібліотек GStreamer та FFMpeg	+ (1.0)	- (0.0)	- (0.0)	- (0.0)
Наявність та доступність сторонніх бібліотек	+ (1.0)	+ (1.0)	+ (1.0)	+ (0.0)
Сума	4.3	3.7	4.7	3

У результаті проведеного аналізу для розробки було обрано мови C++, на якій буде створюватись ядро системи та Python, за допомогою якої буде реалізована внутрішня логіка роботи менеджера.

Оскільки мова програмування не є кросплатформною та залежить від операційної системи, було вирішено використовувати віртуалізацію програми за технологією Docker, що дозволить запускати ядро на будь-якій системі, яка підтримує цю технологію.

Docker — програмне забезпечення для автоматизації розгортання і управління додатками в середовищах з підтримкою контейнеризації. Дозволяє «упакувати»

додаток з усім його оточенням і залежностями в контейнер, який може бути перенесений на операційну систему Linux або Windows, а також надає середовище з управління контейнерами.

3.2 Вибір середовища розробки

IDE (IntegratedDevelopmentEnvironment) — це інтегроване, єдине середовище розробки, яка використовується розробниками для створення різного програмного забезпечення. IDE являє собою комплекс з декількох інструментів, а саме: текстового редактора, компілятора або інтерпретатора, засобів автоматизації збирання і відладчика. Крім цього, IDE може містити інструменти для інтеграції з системами управління версіями і інші корисні утиліти. Є IDE, які призначені для роботи тільки з одним мовою програмування, проте більшість сучасних IDE дозволяє працювати відразу з декількома.

Для порівняння середовищ розробки було обрано наступні IDE: JetBrainsCLion, VisualStudioCode та PyCharm.

VisualStudioCode — текстовий редактор коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки веб і хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git репозиторіями, підсвічування синтаксису, IntelliSense і засоби для рефакторинга.

Оскільки VS Code є текстовим редактором, який підтримує безліч плагінів, які з легкістю можливо інтегрувати в нього, він ідеально підходить для написання коду мовою Python, тому що мова не потребує компіляції.

CLion — інтегроване середовище розробки для мов програмування C та C++, що розробляється компанією JetBrains. Підходить для операційних систем «Windows», «macOS», і «Linux». Середовище CLion має все необхідне для швидкої та зручної розробки системи для операційної системи Linux та в декілька разів перевершує своїх аналогів, саме тому було обрано це середовище для розробки коду на мові C++.

3.3 Схема класів та компоненти програмного модуля

Схема класів — діаграма, що демонструє класи системи, їх атрибути, методи і взаємозв'язки між ними [16]. Входить в UML. Схема класів є ключовим елементом об'єктно-орієнтованого моделювання. На схемі класи представлені в рамках, що містять три компоненти:

— в верхній частині написано ім'я класу,ім'я класу вирівнюється по центру і пишеться напівжирним шрифтом,імена класів починаються з великої літери,якщо клас абстрактний, то його ім'я пишеться напівжирним курсивом;

— посередині розташовуються поля (атрибути) класу,вони вирівняні по лівому краю і починаються з маленької літери.

Мова UML надає механізми для представлення членів класу, наприклад, атрибутів і методів, а також додаткової інформації про них.

Для позначення видимості членів класу (тобто, будь-який атрибут чи метод) ці позначення повинні бути розташовані перед ім'ям учасника.

— публічний (public) знак «+»;

— приватний (private) знак «-».

В UML представлені наступні види відношень:

— залежністьпозначає 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.

На рисунку 3.1 наведено схему класів програмного модуля, а саме наслідування класів FFMpegSource, CFileSource, CLiveSource від класу ISource. Та на рисунку 3.2 наслідування класів FFMpegSink, FileSink, RTSPServerSink, StreamSink та NullSink від класу ISink. Також інтегрований модуль логування та Jsonпарсер. Створений абстрактний клас Runnable, який дозволяє швидко створити новий потік, для обробки певної інформації. Наслідування від цього класу мають

наступні класи: ISink, CUploadRate та Net. Розроблений модуль для перевірки upload швидкості на сервер, де працює веб-додаток.

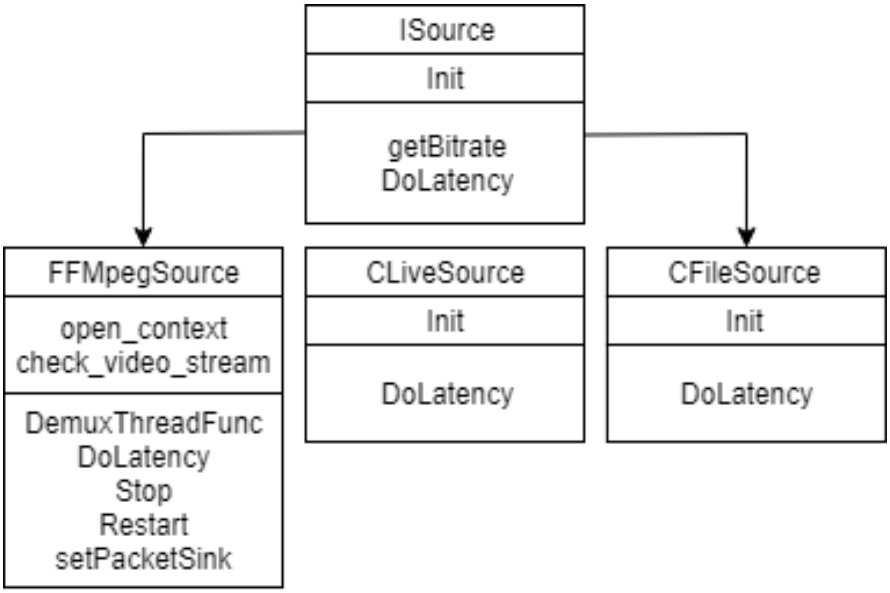

Рисунок 3.1 — Схема класів програмного модуля Source

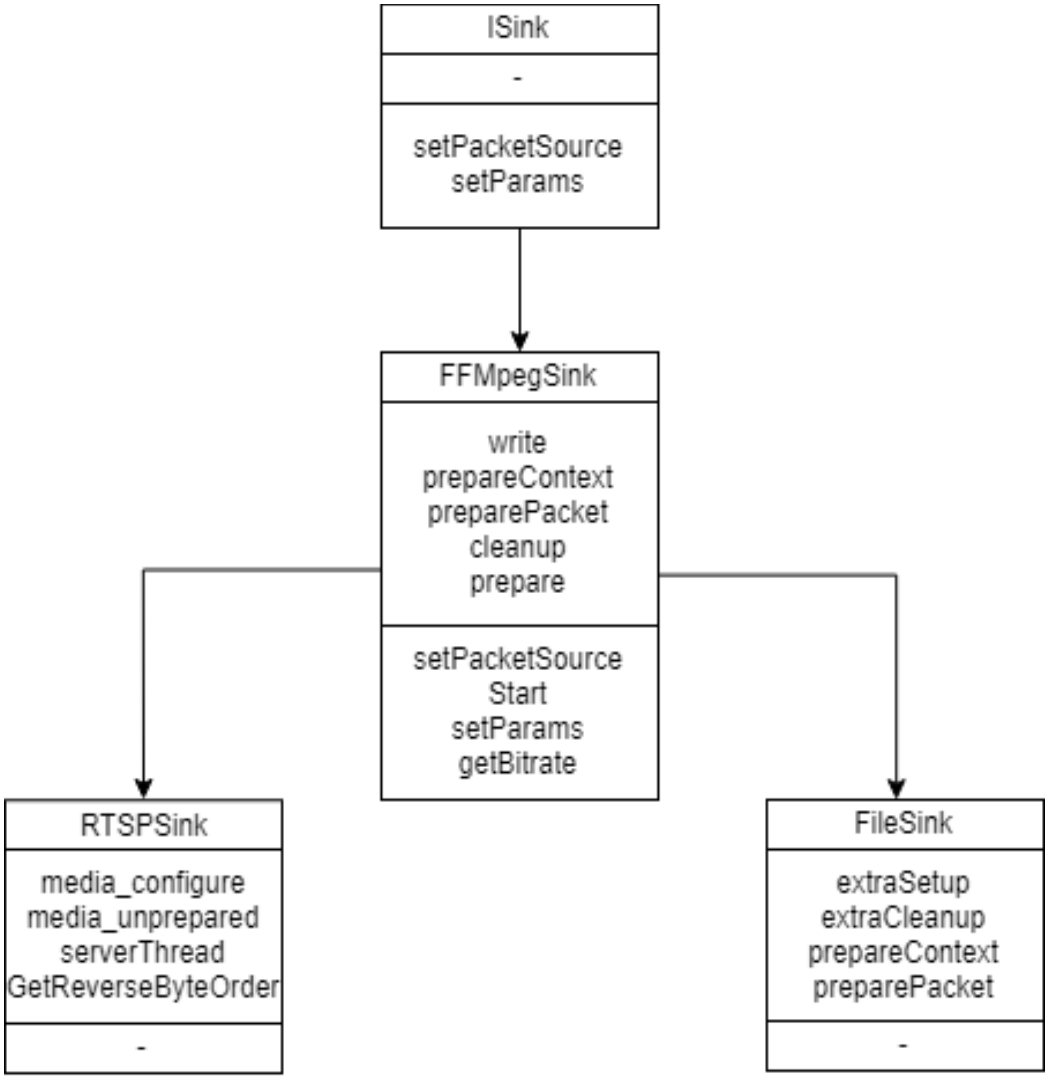

Рисунок 3.2 — Схема класів програмного модуля Sink

Були розроблені схеми класів модулейSource та Sink для ядра камери. Вони відображають основні взаємозв’язки між ними. Ці модулі відповідають за прийняття інформації з камери та передачу її користувачу.

3.4 Розробка програмних модулів системи

Для реалізації програмного забезпечення необхідно поділити розробку на декілька етапів. Спочатку необхідно створити мало функціональне, але працююче ядро камери. Це дозволить розпаралелізувати розробку усіх компонентів і позбавить їх від залежності один від одного. Потім відразу потрібно створити повнофункціональний веб-менеджер, для подальшого вдосконалення ядра.

Створення ядра камери відбуватиметься у наступному порядку:

— підключення до камери;

— отримання даних з камери;

— збір статистики по вхідним даним;

— обробка даних у алгоритмі пропускної можливості;

— відправка даних користувачам;

— збір статистики по вихідним даним;

— відправка статистики на менеджер.

Перший етап реалізований у модулі ISource. Функція Init() дозволяє обрати файл або лінк, на якому йде транслювання відеопотоку.

```
voidCFFmpegSource::Init(conststd::string&path)
{
   //----
this->path = path;
open_context();
check_video_stream_is_avalible();
   //----
demux_thread = std::thread(&CFFmpegSource::DemuxThreadFunc, this);
...
}
```

У цьому фрагменті коду описане підключення до трансляції, яка передається через link (наприклад “rtsp://127.0.0.1:554/”) або файл (наприклад /dev/video0). Після успішного підключення модулю до девайсу створюється новий потік, який буде постійно зчитувати оновлені дані, які надходять з камери.

```
voidCFFmpegSource::DemuxThreadFunc()
{
TPacketcurrentPacket;
while(true)
   {
if(thread_quit)
```

```
break;
        //---
if (currentPacket)
sink->push(move(currentPacket));
        //---
currentPacket.allocate();
autopacket = currentPacket.get();
        //---
intret = av_read_frame(format_context, packet);
if(ret == 0) {
stats.Push(static_cast<size_t>(packet->size) * 8);
DoLatency();
        }
    }
}
```

У цьому потоці програма постійно намагається отримати нові дані з камери та, в разі успіху, записує їх у загальний буфер і збирає статистику кожного нового кадру.

Після запису даних у буфер програма очікує відповідь алгоритму, який рахує максимальну пропускну можливість у даний момент та віддає кадр у модуль Sink, якщо не перевищено ліміт, або ж чекає, коли вихідний бітрейт буде нижче, ніж кінцевий результат алгоритму.

```
TPacketSimpleBuffer::pull() {
        if (!bufferPullAllowed)
return {};
      //---
        lock_guard<mutex>lock(queueMutex);
        if (queue.empty())
return {};
        autocurSize = queue.front().get()->size * 8;
        autocumulativeStats = throttleStats.getStatsByWindow().sum;
```

```
        autonewSize = cumulativeStats + curSize;
        autothreshold = (int)throttleValue*throttleStats.getMeasurementWindow()/1000;
if (throttleEnabled&&newSize>threshold) {
return {};
    //---
        }
    //---
        autopacket = move(queue.front());
        queue.pop_front();
    //---
        throttleStats.Push(curSize);
        frameSizeStats.Push(curSize);
        returnpacket;
}
```

Цей фрагмент коду відповідає за «зтягування» кадрів із загального буфера. Він спрацює лише у тому випадку, коли обчислення алгоритму поверне нам більше результат, ніж вихідний бітрейт.

Після того, як кадр був успішно отриманий, він попадає у модуль ISink, що відповідає за розповсюдження кадрів глядачам.

```
voidRTSPServerSink::work() {
autopacket = Packet.get();
stats.Push(packet->size * 8);
buffer = gst_buffer_new_allocate(nullptr, (gsize)packet->size, nullptr);
auto first4bytes = (uint32_t *)packet->data;
   *first4bytes = this->GetReverseByteOrder(packet->size - 4));
gst_buffer_fill(buffer, 0, packet->data, (gsize)packet->size);
   //---
   GST_BUFFER_PTS (buffer) =
av_rescale_q(packet->pts, time_base, {1, GST_SECOND}));
```

```
   GST_BUFFER_DURATION (buffer) =
av_rescale_q(packet->duration, time_base, {1, GST_SECOND}));
   //---
autoelement = gst_rtsp_media_get_element (media);
autoappsrc = gst_bin_get_by_name_recurse_up (GST_BIN (element), "mysrc");
g_signal_emit_by_name (appsrc, "push-buffer", buffer, &ret);
}
```

У фрагменті методу work класу RTSPServerSink описана логіка передачі зображення глядачам та збору статистики вихідного бітрейту. Ця частина коду відповідає за відправку отриманого кадру з буферу всім підключеним користувачам.

Після створення робочої версії серверної частини необхідно розробити веб-додаток, для подальшої інтеграції між ними. Створення системної логіки менеджера буде відбуватись на мові Python. Етапи розробки будуть такими:

— логіка «рукостискання» між камерою та менеджером;

— прийом та опублікування статистики в режимі реального часу;

— управління камерами, відправка команд через POST метод у вигляді JSON форматування;

— створення сторінки авторизації;

— створення головної сторінки;

— створення панелі управління.

Для створення менеджеру використаємо готовий Docker-образ Nginx та трішки його доповнимо. Це дозволить нам запустити свій http-сервер і опрацьовувати всі запити клієнтів.

Перш за все потрібно після додавання камери у базу створити «рукостискання» між менеджером і ядром камери.

```
defhandshake(ip, _id):
res = False
try:
js = {
        'type': 2,
```

```
            'body': {
                'manager_ip': settings.MY_IP,
                'id': _id,
            }
        }
        #----
resp = requests.post(
f'http://{ip}:8000/api/handshake',
json=js,
auth=HTTPBasicAuth('admin', settings.BASICAUTH_USERS['admin']),
timeout=2
        )
ifresp.ok:
res = True
exceptExceptionasex:
logger.error(str(ex))
returnres
```

Цей фрагмент коду відповідає за створення «рукостискання» між менеджером та ядром камери. Після цих дій камера запам’ятовує куди повинна відправляти дані і починає їх відправляти.

Отримання та парсинг даних з камери відбувається у класі CameraSerializer.

```
classCameraSerializer(serializers.ModelSerializer):
model = Camera
fields = (
    'id',
    'ip',
    'name',
    'handshake',
    'status',
    'added',
```

```
    'last_update',
    'url',
    'owner',
    'upload_rate',
    'color',
    'tag')
defget_tag(self, camera: Camera):
try:
stat = Stat.objects.filter(camera=camera).latest()
returnstat.tag
exceptStat.DoesNotExist:
returnNone
```

У цьому фрагменті коду описана логіка отримання даних з камери у форматі JSON та конвертування їх у змінну Dictionary.

Розробка клієнтської частини базується на чотирьох етапах:

— підключення до RTSP відеотранслювання та отримання даних;

— синхронізація кадрів;

— створення локального RTSP серверу або запис у файл.

Перший етап розроблений за допомогою FFMpeg бібліотеки та ідентичний до модуля ISource, який був описаний вище.

Другий етап заключається в вирівнювані кадрів та подальшої рівномірної передачі їх у файл або сервер.

```
{
fCurrentTime = CTimer::GetCurrentTime();
intcount = 0;
for(auto&client : ClientMediaList)
{
GstBuffer *buffer;
buffer = gst_buffer_new_allocate(nullptr, packet->size, nullptr);
if (client.second.startTime == -1)
```

```
client.second.startTime = fCurrentTime;
if (client.second.first_pts_value == AV_NOPTS_VALUE)
client.second.first_pts_value = Decoder->GetPTSSeconds(packet->pts);
        autovalue = fCurrentTime - client.second.startTime +
        client.second.first_pts_value;
autopts_value = Decoder->GetPTSSeconds(packet->pts);
if (value>= pts_value)
    {
gst_buffer_fill(buffer, 0, packet->data, packet->size);
...
}
```

Цей фрагмент описує логіку синхронізації кадрів в момент передачі їх на сервер. Завдяки тому, що RTSP протокол надсилає інформацію про тривалість кадру, програма може чітко і вчасно керувати їх подальшим розповсюдженням. Тепер система цілком робоча, всі її модулі були налаштовані і приведені до робочого стану.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу наукової розробки

Для визначення комерційного потенціалу дослідження необхідно залучити 3-х незалежних експертів. У нашому випадку такими експертами можуть бути:

— к.т.н. , доцент Богомолов С.В.;

— к.т.н., доцент Войцеховська О.В..;

— к.т.н., доцент Захарченко С.М..;

Оцінювання комерційного потенціалу дослідження здійснюється за 12-ма критеріями , що наведені в таблиці 4.1:

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 роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	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	Експлуатацій-ні витрати значно вищі, ніж в аналогів	Експлуатацій-ні 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

Продовження таблиці 4.1

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкуренції немає
Практична 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 і з 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
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

Кінець таблиці 4.1

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 регламентовані обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу дослідження заносимо в таблицю 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу дослідження

Критерії	Прізвище, ініціали, посада експерта		
	Богомолов С.В.	Войцеховська О.В.	Захарченко С.М.
	Бали, виставлені експертами:		
1	3	2	5
2	4	2	3
3	2	3	4
4	2	3	3
5	3	2	2
6	4	4	4
7	3	3	2
8	4	4	3
9	2	2	2
10	3	2	3
11	4	3	4
12	3	4	3
Сума балів	$СБ_1 = 37$	$СБ_2 = 34$	$СБ_3 = 38$
Середньоарифметична сума балів $\underline{СБ}$	$\underline{СБ} = 36{,}33 \approx 36$		

Таблиця 4.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
від 0 до 10	Низький
від 11 до 20	Нижче середнього
від 21 до 30	Середній
від 31 до 40	Вище середнього
від 41 до 48	Високий

Висновок: згідно отриманих даних в таблиці 4.2. за результатом оцінювання комерційного потенціалу дослідження експертами рівень комерційного потенціалу дослідження є вище середнього.

Успіх реалізації комерційної ідеї залежить від того, яку мету ставить перед собою розробник та якими ресурсами він володіє. При досить високому рівні комерційного потенціалу доцільним є, як створення спільного підприємства, так, і залучення коштів інвесторів. Доцільним, в даному випадку для розробника, буде складання обґрунтованого бізнес-плану для досягнення поставлених задач. Важливим завданням при залученні потенційних інвесторів, має бути доведена перспективність та корисність даної розробки.

4.2Прогнозування витрат на виконання науково-дослідної, (дослідно-конструкторської) та конструкторсько технологічної роботи

Прогнозування витрат на виконання даних робіт може складатись з таких етапів:

— розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

— розрахунок загальних витрат на виконання даної роботи;

— прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

1-й етап: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу рботи, можна здійснити за такими статтями та формулами:

Основна заробітна плата кожного із розробників (дослідників) $З_о$, якщо вони працюють в наукових установах бюджетної сфери:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн.,} \qquad (4.1)$$

де М — місячний посадовий оклад конкретного розробника, грн.

$Т_р$ — число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні;

t — число робочих днів роботи розробника (дослідника).

Таблиця 4.4 — Результат прогнозування витрат:

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
1.Науковий керівник	7500	340,9	20	6856,36
2.Інженер-конструктор	5500	250	14	3313,5
3.Інженер-програміст	10500	477,27	22	10452
Всього				20441,8

Основна заробітна плата робітників $З_р$, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$З_р = \sum_{1}^{n} t_i \cdot C_i \text{ грн.}, \qquad (4.2)$$

де t_i — норма часу (трудомісткість) на виконання конкретної роботи, годин;

n — число робіт по видах та розрядах;

C_i — погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i = \frac{М_м \cdot К_{мкс} \cdot К_с}{Т_р \cdot Т_{зм}}, \qquad (4.3)$$

де $М_м$ — розмір мінімальної заробітної плати за місяць у 2020 році, 5000 грн.;

$К_{мкс}$ — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (див. таблицю 3.1);

$Т_р$ — число робочих днів в місяці; приблизно $Т_р$ = 22 дні;

$Т_{зм}$ — тривалість зміни, зазвичай $Т_{зм}$ = 8 годин.

Таблиця 4.5 — Основна заробітна плата робітників:

Найменування робіт	Трудоміст-кість, н.-годин	Розряд роботи	Погодинна тарифна ставка, грн.	Величина оплати, грн.
1. Монтажні	15	5	30,9	463,5
2. Налагоджувальні	8	3	24.54	196,32
3.Складальні	22	2	20	440
Всього				1099,82

У випадку, коли робітники беруть участь у виконанні даного етапу роботи, працюючи при цьому на підприємствах промисловості, то погодинна тарифна ставка C_i робітника відповідного розряду розраховується за формулою

де $M_м$ — розмір мінімальної заробітної плати за місяць, грн./міс.;

$К_{мкс}$ — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (див. таблицю 4.6);

Таблиця 4.6 — Міжкваліфікаційні співвідношення для встановлення тарифних ставок робітникам (для цього випадку)

Розряд	1	2	3	4	5	6	7	8
$К_{МКС}$	1.0	1.1	1.35	1.5	1.7	2.0	2.2	2.4

3) Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = (0{,}1...0{,}12) \cdot З_о + Зр. \qquad (4.4)$$

Для нашого випадку:

$З_д$ = 0.1 × (20 441,8+1099,82)= 2154,17 грн.

Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$H_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \qquad (4.5)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування у 2020 році складає 22%, %.

$$Н_{зп}= (20\ 441{,}8 + 1099{,}82+2154{,}17)\times 0.22 = 5213{,}07 \text{грн.}$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час (чи для) виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн.}, \qquad (4.6)$$

де Ц — загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

$Н_а$ — річна норма амортизаційних відрахувань. Для нашого випадку можна прийняти, що $Н_а$ = (10...25)%;

Т — термін, використання обладнання, приміщень тощо, місяці.

Таблиця 4.7 — Амортизація обладнання:

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
Комп'ютер	12000	20	3	600

Приміщення лабораторії	78000	20	4	5200
Мікрокомп'ютер	1150	20	3	57,5
Програмне забезпечення PyCharm та CLion	18500	20	3	925
Камера відеоспостереження	1100	20	4	73,33
Всього				6855,83

Витрати на матеріали М, що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$M = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad \text{грн.,} \qquad (4.7)$$

де $Н_i$ — витрати матеріалу *і*-го найменування, кг;

$Ц_i$ — вартість матеріалу *і*-го найменування, грн./кг.;

$К_i$ — коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

$В_i$ — маса відходів матеріалу *і*-го найменування, кг;

$Ц_в$— ціна відходів матеріалу *і*-го найменування, грн/кг;

n — кількість видів матеріалів.

Таблиця 4.8 — Витрати на матеріал:

Найменування матеріалу, марка, тип, сорт	Ціна за одиниц, грн.	Витрачено, шт	Величина відходів	Ціна відходів, грн/шт	Вартість витраченого матеріалу, грн.	Примітка

Камери відеоспостереження марок AXIS, HikVision, BOSCH	2095,1	1	-	-	2095,1	
Мікрокомп'ютери Fitlet	6000	1	-	-	6000	
Всього					8095,1	

Витрати на послуги., що були використані під час виготовлення дослідного зразка

Таблиця 4.9

Найменування робіт (послуг)	Кількість, шт	Ціна за штуку, грн.	Сума, грн.
AWS S3 Bucket	4	450	1800
Камери відеоспостереження	3	7000	21000
Всього			22800

Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ грн,} \qquad (4.9)$$

де В — вартість 1 кВт-год. електроенергії, в 2020 р. В ≈ 2,50 грн./кВт;

П — установлена потужність обладнання, кВт;

Ф — фактична кількість годин роботи обладнання, годин,

$К_п$ — коефіцієнт використання потужності; $К_п < 1$.

$$В_е = 2.50 \times 1.38 \times 128 \times 0.85 = 375,36 \text{ грн.}$$

Інші витрати $В_{ін}$ охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3)* (З_о+З_р). \quad (4.10)$$

$$В_{ін} = 1 \times 21541,62 = 21541,62 \text{ грн.}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи — В.

В=20441,8+1099,8+2154+5213+6855,8+8095,1+22800+375+21541,6=88576,67г.

2-й етап: розрахунок загальних витрат на виконання даної роботи. Розрахунок загальних витрат здійснюється у тому випадку, коли дипломник виконує тільки певну частину даної роботи. У подальшому ця наукова робота буде продовжена.

Тоді загальна вартість всієї наукової роботи визначається за $В_{заг}$ формулою:

$$B_{заг} = \frac{B}{\alpha}, \quad (4.11)$$

де α — частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях, $\alpha = 0,9$

$$В_{заг} = \frac{88576,67}{0,9} = 98418,52 \text{ грн}$$

3-й етап: прогнозування загальних витрат на виконання та впровадження результатів виконаної наукової роботи. У подальшому ця наукова робота буде продовжена. Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (4.12)$$

де β — коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Так, якщо розробка знаходиться:

— на стадії науково-дослідних робіт, то $\beta \approx 0,1$;

— на стадії технічного проектування, то $\beta \approx 0,2$;

— на стадії розробки конструкторської документації, то $\beta \approx 0,3$;

— на стадії розробки технологій, то $\beta \approx 0,4$;

— на стадії розробки дослідного зразка, то $\beta \approx 0,5$;

— на стадії розробки промислового зразка, $\beta \approx 0,7$;

— на стадії впровадження, то $\beta \approx 0,9$;

— на стадії продажу, $\beta \approx 1$.

Оскільки дана розробка знаходиться на стадії науково-дослідних робіт, то $\beta = 0,2$

$$ЗВ = \frac{98418,67}{0,2} = 504139,5 \text{ грн}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Прогнозування комерційних ефектів здійснюється у двох основних випадках:

— коли можна прямо оцінити зростання чистого прибутку підприємства від впровадження результатів наукової розробки;

— коли неможливо прямо оцінити зростання чистого прибутку підприємства від впровадження результатів наукової розробки.

Дана розробка підпадає під другий випадок, оскільки вона знаходиться на стадії науково-дослідних робіт, До впровадження методів підвищення ефективності функціонування системи WIMAX кількість користувачів становила 250500. Після впровадження результатів наукової розробки кількість користувачів зростала кожного року на 50000 користувачів. Протягом трьох років. розрахунок загальних

витрат на виконання даної роботи. Розрахунок загальних витрат здійснюється у тому випадку, коли дипломник виконує тільки певну частину даної роботи.

Тоді загальна вартість всієї наукової роботи визначається. Ціна до впровадження розробки складала 180 грн з абонента,після впровадження розробки збільшилась на 250 грн.

Розрахунки здійснюватимемо за формулою 3.14:

$$\Delta П_i = \sum_1^n (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.13)$$

де $\Delta Ц_o$ — покращення основного оціночного показника від впровадження результатів розробки у даному році;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки, N= 250500 абонентів.;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки, ΔN=50000 аб. ;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_0$=250 грн;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки, n=3;

λ — коефіцієнт, який враховує сплату податку на додану вартість $\lambda = 0{,}8547$;

ρ — коефіцієнт, який враховує рентабельність продукту, $\rho = 0{,}25$;

υ — ставка податку на прибуток, υ = 18%.

$$\Delta П_1 = [250 \cdot 250500 + (250 + 180) \cdot 50000] \cdot 0{,}8547 \cdot 0{,}25 \cdot (1 - \frac{18}{100}) = 14400096$$

$$\Delta П_2 = [250 \cdot 250500 + (250 + 180) \cdot (50000 + 50000)] \cdot 0{,}8547 \cdot 0{,}25 \cdot (1 - \frac{18}{100}) = 17956250$$

$$\Delta П_3 = [250 \cdot 250500 + (250 + 180) \cdot (50000 + 50000 + 50000)] \cdot 0{,}8547 \cdot 0{,}25 \cdot (1 - \frac{18}{100}) = 21611250$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

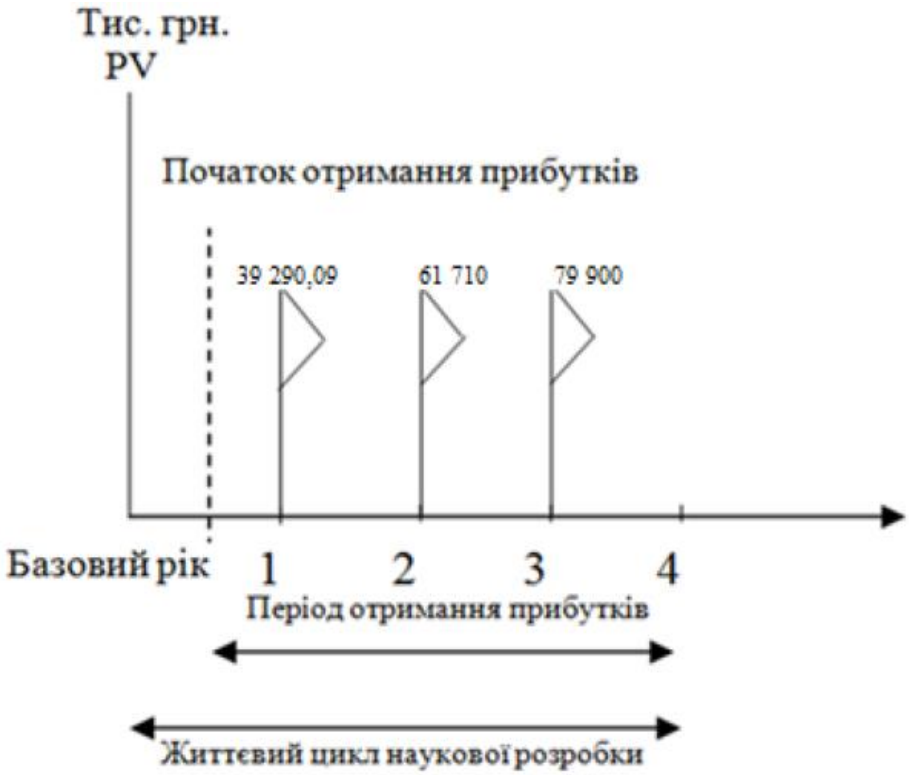

Рисунок 4.1 — Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$.

Для цього використаємо формулою:

$$E_{абс} = (ПП – PV), \quad (4.14)$$

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.; PV — теперішня вартість інвестицій PV = ЗВ, грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{m} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.15)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т — період часу, протягом якого виявляються результати впровадженої НДДКР, т=3;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; τ=0,1;

t — період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{14400096}{(1+0,1)^1} + \frac{17956250}{(1+0,1)^2} + \frac{21611250}{(1+0,1)^3} = 44428009$$

$$Е_{абс} = ПП - ЗВ = 44428009 - 504139,5 = 43923870$$

5-й крок. Розраховують відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього користуються формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.16)$$

де $Е_{абс}$ — абсолютна ефективність вкладених інвестицій, грн; PV — теперішня вартість інвестицій PV = ЗВ, грн; $Т_ж$ — життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[4]{1 + \frac{44428009}{504139,5}} - 1 = 2,07$$

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \qquad (4.17)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; d = 0,16;

f — показник, що характеризує ризикованість вкладень; зазвичай, f = 0,08.

$$\tau = 0{,}16 + 0{,}08 = 0{,}24$$

Оскільки $E_в > \tau$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Термін окупності вкладених у реалізацію наукового пректу інвестицій $T_{ок}$ можна розрахувати за формулою:

$$T_{ок} = \frac{1}{E_в}. \qquad (4.18)$$

Оскільки $T_{ок} < 3$ років, то це свідчить про доцільність фінансування такої розробки.

Основна заробітна плата розробників (дослідників) склала 20441,8 грн. Основна заробітна плата робітників — 1099,82 грн. Додаткова заробітна плата всіх розробників та робітників складає 2154,17 грн. Нарахування на заробітну плату розробників — 5213,07 грн. Амортизація обладнання склала 6855,83 грн. Також розраховані витрати на матеріали, послуги, на силову електроенергію, інші витрати та витрати всіх попередніх статей — 8095,1 грн., 22800 грн., 375,36 грн., 21541,62 грн., і 88576,67 грн. відповідно. На другому етапі розраховано суму загальних витрат — 504139,5 грн.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було розглянуто важливість та проблему створення якісної системи відеоспостереження. Чому було обрано саме такі фреймворки? GStreamer дозволяє нам дуже гнучко налаштовувати відеопотік, а FFmpeg має дуже обширені можливості у кодуванні і декодуванні відеоданих. Оскільки GStreamer та FFmpeg були створені мовою C, то розробка системи відбувалась мовою C++. Використання Milestone в якості клієнта дозволило нам максимально точно та синхронізовано відтворювати дані, які надходять з серверу.

Було розроблено структуру ядра камери, модулю налаштувань, алгоритм обчислення пропускної швидкості вихідного бітрейту, користувацький інтерфейс веб-додатку, структуру веб-менеджменту камер та збір статистики по принципу витісняючої черги. Було виконано варіантний аналіз та обґрунтування вибору програмних засобів при розробці платформи, а саме: вибір мови програмування C++ та Python, засобів розробки клієнтської частини додатку, вибір інтегрованого середовища розробки CLion та текстового редактора VisualStudioCode. Побудовано схему класів та схему компонентів веб-додатку; описано програмну реалізацію системи. Були розроблені головне ядро камери, веб-додаток та менеджер у ньому, для зручного керування алгоритмом та камерами, перегляд статистики, логіку «рукостискання» між камерою та менеджером за допомогою мов C++ та Python, а також Docker-технології. Був створений клієнтський додаток, що дозволяє користувачу нормалізувати поток відео який надходить з сервера.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Stroustrup B. The C++ ProgrammingLanguage, 4th edition / B. Stroustrup— Addison-Wesley, 2013 — 1368 ст.
2. Основывидеотрансляции— [Електронний ресурс]. — Режим доступу: https://habr.com/ru/company/ivideon/blog/230687/
3. Кузнецов С.Д. Основы баз данных, 2-е издание / С.Д. Кузнецов — Москва: «БИНОМ», 2007 — 484 ст.
4. MetroUserInterface: Описание и примеры Веб Дизайна— [Електронний ресурс]. — Режим доступу: https://habrahabr.ru/post/156625/
5. A TransportProtocolforReal-TimeApplications— [Електронний ресурс]. - https://tools.ietf.org/html/rfc3550
6. Фримен А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов, 4-е издание / А. Фримен — Москва: «Вильямс», 2013 — 688 ст.
7. Бизли Д. М. ЯзыкпрограммированияPython : справочник : пер. с англ. / Д. М. Бизли. — Киев : ДиаСофт, 2000.
8. Лесса А. Python. Руководстворазработчика : пер. с англ. / А. Лесса. — СПб. ДиасофтЮП, 2001.
9. Самоучитель по Java — [Електронний ресурс]. — Режим доступу: http://kit.znu.edu.ua/iLec/9sem/OOP/e-book/Язык_программирования_Java.pdf
10. Кузнецов С.Д. Основы баз данных, 2-е издание / С.Д. Кузнецов — Москва: «БИНОМ», 2007 — 484 ст.
11. FFMpegdocumentation — [Електронний ресурс]. — Режим доступу: https://ffmpeg.org/ffmpeg.html
12. Gstreamerdocumentation — [Електронний ресурс]. — Режим доступу: https://gstreamer.freedesktop.org/documentation/
13. H.264 кодированиевидео — [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/H.264

14. Дин Леффингуэлл, Дон Уидриг. Принципыработы с требованиями к программномуобеспечению. Унифицированныйподход. — М.Вильямс, 2002.
15. UML-схемыклассов— [Електронний ресурс]. — Режим доступу: https://msdn.microsoft.com/ru-ru/library/dd409437.aspx
16. ЧтотакоеТестированиеЧерного Ящика (Blackboxtesting)? — [Електронний ресурс]. — Режим доступу: http://software-testing.org/testing/testirovanie-chernogo-yaschika-black-box-testing.html
17. RTSP-протокол — [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/RTSP

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖЕНО
Завідувач кафедри ОТ
_____проф., д.т.н., О.Д. Азаров
«___»_____2020 р.

Технічне завдання
до магістерської кваліфікаційної роботи на тему:
«Програмний засіб для обробки даних та автоматизованого управління в системі відеоспостереження»
08-23.МКР.001.00.000 ТЗ

Керівник роботи:
_____ д. т. н., проф. Азаров О. Д.
«___»_____2020 р.
Виконав: студент гр. 1КІ-19м
Бабій Вадим Ігорович
«___»_____2020 р.

Вінниця 2020

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

а) актуальність досліджень;

б) наказ про затвердження теми дипломної роботи.

2 Мета і призначення МКР

а) мета — вдосконалення методу передачі відео-трафіку та мінімізація втрат зображень із використанням автоматичної системи управління;

б) призначення розробки — процес написання автоматизованої системи управління.

3 Вихідні дані для виконання МКР

— технічний опис програмного застосунку;

— мови програмування C++ та Python;

— середовище розробки Clion;

4 Вимоги до виконання МКР

— огляд і аналіз методів розробки;

— аналіз та порівняння існуючих систем відеоспостереження;

— розробка програмного забезпечення.

5 Етапи МКР та очікувані результати наведено у таблиці А.1

Таблиця А.1— Етапи розробки МКР

№ Етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Пошук та огляд інформаційних джерел	15.09.20р.	01.10.20р.	Розділ 1
2	Програмна реалізація додатку	02.10.20р.	25.10.20р.	Розділ 2
3	Тестування розробленого програмного засобу	26.10.20р.	30.10.20р.	Розділ 3

4	Економічна частина	15.11.20р.	18.11.20р.	Розділ 4

6 Матеріали, що подаються до захисту МКР

Пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, 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 Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

Вимоги викладені в МЕТОДИЧНИХ ВКАЗІВКАХ до дипломного проектування, ДСТУ 3008-95, ДСТУ 3974-2000 «Правила виконання дослідно-конструкторських робіт. Загальні положення» та діючого ГОСТ 2.114-95 ЕСКД.

9 Вимоги щодо технічного захисту інформації в МКР з обмеженим доступом відсутні.

ДОДАТОК Б

Лістинг програми ядра

```
#include "CFFmpegSource.h"
#include "utils.h"
#include "Settings.h"
#include "CLogger.h"
extern "C"
{
#include<libavformat/avformat.h>
}
constintCFFmpegSource::seconds_before_restart = 5;
constintCFFmpegSource::max_error_count = 5;
intCFFmpegSource::defaultFPS = 30;
CFFmpegSource::~CFFmpegSource()
{
Stop();
}
voidCFFmpegSource::Init(conststd::string&path)
{
this->path = path;
open_context();
check_video_stream_is_avalible();
av_dump_format(format_context, 0, "", 0);
demux_thread = std::thread(&CFFmpegSource::DemuxThreadFunc, this);
}
voidCFFmpegSource::open_context()
{
format_context = avformat_alloc_context();

AVDictionary *opts = nullptr;
if (Settings::getSettings()->is_enabled_STimeout())
av_dict_set(&opts, "stimeout", "10", 0);
intret = avformat_open_input(&format_context, path.c_str(), nullptr, &opts);
if(ret != AVERROR_SUCCESS)
throwstd::runtime_error("Formatcontexterror");
}
voidCFFmpegSource::check_video_stream_is_avalible()
{
```

```
intret = avformat_find_stream_info(format_context, nullptr);
if(ret != AVERROR_SUCCESS || !find_stream_index(AVMEDIA_TYPE_VIDEO))
throwstd::runtime_error("Notfoundvideostream");
get_frame_duration_ms();
}
boolCFFmpegSource::find_stream_index(AVMediaTypetype)
{
for(unsignedintiterator = 0; iterator<format_context->nb_streams; ++iterator)
   {
if(format_context->streams[iterator]->codecpar->codec_type == type)
      {
stream_index = iterator;
returntrue;
      }
   }
returnfalse;
}
voidCFFmpegSource::DemuxThreadFunc()  //Demultiplexerthread
{
assert(sink != nullptr);
interrors_count = 0;
CLogger *logger = CLogger::Instance();

TPacketcurrentPacket;
while(true)
   {
if(thread_quit)
break;
if (currentPacket)
sink->push(move(currentPacket));
currentPacket.allocate();
autopacket = currentPacket.get();
intret = av_read_frame(format_context, packet);
if(ret == 0) {
stats.Push(static_cast<size_t>(packet->size) * 8);
errors_count = 0;
         // TODO: useafterfreeondestructor
DoLatency();
      }
```

```
else
        {
            // TODO: Extractmethodrestart
            ++errors_count;
logger->Log(log4cpp::Priority::WARN, "Can'treadframe, errorcounteris %d!",
errors_count);
if(errors_count>= max_error_count) {
logger->Log(log4cpp::Priority::INFO, "Restartconnection!");
std::threadreinit(&CFFmpegSource::Restart, this); // TODO: fixthis...
reinit.detach();
stats. Clear();
return;
            }
        }
    }
logger->Log(log4cpp::Priority::INFO, "Demuxexited");
}
voidCFFmpegSource::Stop()
{
thread_quit = true;
if (demux_thread.joinable())
demux_thread.join();

if (format_context)
avformat_close_input(&format_context);
}
voidCFFmpegSource::Restart()
{
Stop();
intcount = 0;
try
    {
Init(path);
    }
catch(constchar *msg)
    {
        ++count;
if(count>= max_error_count)
        {
```

```
            //exit(1);
return;
        }
        std::this_thread::sleep_for(std::chrono::seconds(seconds_before_restart));
    }
}
voidCFFmpegSource::get_frame_duration_ms()
{
intframes_per_second = 0;
auto&stream = format_context->streams[stream_index];
if (stream->avg_frame_rate.den != 0)
frames_per_second = stream->avg_frame_rate.num / stream->avg_frame_rate.den;
elseif (stream->r_frame_rate.den != 0)
frames_per_second = stream->r_frame_rate.num / stream->r_frame_rate.den;
if (frames_per_second == 0)
frames_per_second = defaultFPS;
else
defaultFPS = frames_per_second;
frame_duration = 1000/frames_per_second;
}
voidCFFmpegSource::setPacketSink(IPacketSink *sink) {
this->sink = sink;
}
void *CFFmpegSource::getParams() {
assert(stream_index != -1);
returnformat_context->streams[stream_index]->codecpar;
}
void *CFFmpegSource::getTimeBase() {
assert(stream_index != -1);
return&format_context->streams[stream_index]->time_base;
}
intCFFmpegSource::getBitrate() {
returnstats.getStatsByWindow().sum;
}
voidCFFmpegSource::onNotify(Eventevent) {
CLogger::Instance()->Log(
            log4cpp::Priority::NOTICE,
            "CFFmpegSource: buffer " + EventHelper::eventNames[event] + "
changesignalreceived"); // TODO: taskName
```

```
switch (event) {
caseEvent::MeasureTime: { //TODO: irrelevanteventfornow
CLogger::Instance()->Log(
                log4cpp::Priority::NOTICE,
                "CFFmpegSource: applynewbuffermeasureTime: %d",
Settings::getSettings()->getMeasureTime());
break;
        }
caseEvent::MeasureWindow: {
          stats.setMeasurementWindow(Settings::getSettings()->getMeasureWindow());
CLogger::Instance()->Log(
                log4cpp::Priority::NOTICE,
                "CFFmpegSource: applynewbufferthrottleWindow: %d",
stats.getMeasurementWindow());
break;
        }
default: {
CLogger::Instance()->Log(
                log4cpp::Priority::NOTICE,
                "CFFmpegSource: UnknownEventType: %s",
EventHelper::eventNames[event]);
cerr<< "UnknownEventType: " <<EventHelper::eventNames[event] <<endl;
break;
        }
    }
}
intCFFmpegSource::GetVideoFPS() {
returndefaultFPS;
}
void *CFFmpegSource::getStream() {
returnformat_context->streams[stream_index];
}
```

ДОДАТОК В

Лістинг програми слухача

```
fromflaskimportFlask
fromflask_httpauthimportHTTPBasicAuth
fromflaskimportrequest, render_template, jsonify
fromapiimportConnectionApi
importjson
app = Flask(__name__)
auth = HTTPBasicAuth()
withopen('config.json', 'rb') as file:
users = json.load(file)['users']
@auth.get_password
defget_pw(username):
ifusernameinusers:
returnusers.get(username)
returnNone
@app.route("/")
@auth.login_required
defhello():
return "Hello " + auth.username()
@app.route('/api/')
@auth.login_required
defapi():
action = request.args.get('action')
returnrender_template('api.html', user=auth.username(), action=action)
@app.route("/status/")
@auth.login_required
defstatus():
connection_api = ConnectionApi()
resp = connection_api.status()
print(resp)
returnrender_template('status.html', status=resp)
@app.route("/api/handshake", methods=['POST'])
@auth.login_required
defhandshake():
js = json.loads(request.data)
print('Handshakefrom {}'.format(js['body']['manager_ip']))
connection_api = ConnectionApi()
```

```
resp = connection_api.handshake(js)
ifrespisNone:
return 'Coreisnotrunning', 404
returnjsonify(resp), 204 ifresp['success'] else 500
classTest:
def __init__(self):
        # self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM,
socket.IPPROTO_UDP)
self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM,
socket.IPPROTO_TCP)
self.sock.connect(('127.0.0.1', 8585))
def send_2s_1l(self, s1, s2, l1):
fmt = '<hhhhl'
        s = struct.pack(fmt, struct.calcsize(fmt), s1, s2, 0, l1)
crc = zlib.crc32(s) & 0xffffffff
print(struct.calcsize(fmt))
print(crc)
crc = struct.pack('<L', crc)
data = s + crc
print(data)
        # self.sock.sendto(data, ('127.0.0.1', 8585))
self.sock.send(data)
class Test2:
def __init__(self):
        # self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM,
socket.IPPROTO_UDP)
self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM,
socket.IPPROTO_TCP)
self.sock.connect(('127.0.0.1', 8585))
defsend(self, tupe, str):
        s = str.encode()
fmt = f'<BBH{len(s)}s'
data = struct.pack(fmt, tupe, 0, len(s), s)
print(data)
self.sock.send(data)
```

ДОДАТОК Г

Лістинг програми менеджера

```
fromrest_frameworkimportserializers
fromcollectionsimportOrderedDict
fromapi.backendimportsetter
fromtypingimportIterable, List
importdatetime
importpytz
importrandom
if __name__ == '__main__':
importdjango
if 'setup' indir(django):
django.setup()
fromapi.modelsimportStat, Camera, Control, AlgorithmValue
fromapi.tasksimportapply_after_turn_off
fromdjango.contrib.auth.modelsimportUser
classTimestampField(serializers.Field):
    # defto_native(self, value):
    #     epoch = datetime.datetime(1970,1,1)
    #     returnint((value - epoch).total_seconds())
defto_representation(self, value: datetime.datetime):
returnint(value.timestamp()) * 1000 + value.microsecond // 1000
defto_internal_value(self, data):
        # TODO: checktimezoneonclients
returndatetime.datetime.utcfromtimestamp(data // 1000).replace(microsecond=data %
1000 * 1000, tzinfo=pytz.UTC)
# classStatReadSerializer(serializers.ModelSerializer):
#     classMeta:
#         model = Stat
#         fields = ('order', 'title', 'duration')
classStatSerializer(serializers.ModelSerializer):
camera_id = serializers.PrimaryKeyRelatedField(queryset=Camera.objects.all(),
source='camera')
timestamp = TimestampField()
    # tag = serializers.CharField(required=False, write_only=True)
classMeta:
model = Stat
fields = (
```

```
            'id',
            'timestamp',
            'inputBitrate',
            'outputBitrate',
            'camera_id',
            'maxFrameCount',
            'bufferSize',
            'frameCount',
            'averageFrameSize',
            'throttleEnabled',
            'bufferPullAllowed',
            'throttleValue',
            'tag',
        )
    # defcreate(self, validated_data: dict):
    #     tag = validated_data.pop('tag', None)
    #     camera_id = validated_data.get('camera_id', -1)
    #     super().create(validated_data)
    # defcreate(self, validated_data):
    #     # camera = validated_data.pop('id')
    #     # print(camera, validated_data)
    #     # returnStat.objects.create(camera=camera, **validated_data)
    #     returnStat.objects.create(**validated_data)
classChoicesSerializerField(serializers.SerializerMethodField):
defto_representation(self, value):
method_name = 'get_{field_name}_display'.format(field_name=self.field_name)
method = getattr(value, method_name)
returnmethod()
classCameraSerializer(serializers.ModelSerializer):
    # stat_set = StatSerializer(many=True, read_only=True)
    # stats = serializers.SerializerMethodField()
    # ip = serializers.CharField(max_length=200, validators=[validate_ipv4_address], un)
owner = serializers.ReadOnlyField(default=None, source='owner.pk')
handshake = serializers.BooleanField(read_only=True)
last_update = serializers.DateTimeField(read_only=True)
status = ChoicesSerializerField()
upload_rate = serializers.IntegerField(read_only=True)
color = serializers.CharField(read_only=True)
tag = serializers.SerializerMethodField(read_only=True)
```

```
    # serializers.ChoiceField
classMeta:
model = Camera
fields = (
            'id',
            'ip',
            'name',
            'handshake',
            'status',
            'added',
            'last_update',
            'url',
            'owner',
            'upload_rate',
            'color',
            'tag'
        )
defget_tag(self, camera: Camera):
try:
stat = Stat.objects.filter(camera=camera).latest()
returnstat.tag
exceptStat.DoesNotExist:
returnNone
    # defget_stats(self, camera: Camera):
    #     query = Stat.objects.filter(camera=camera)[:10]
    #     serializer = StatSerializer(query, many=True)
    #     returnserializer.data
classMeterSerializer(serializers.Serializer):
current_value = serializers.SerializerMethodField()
critical_value = serializers.SerializerMethodField()
classMeta:
fields = ('current_value', 'critical_value')
defget_current_value(self, cameras: List[Camera]):
alg_value = AlgorithmValue.load()
ifself.context.get('mode', '') == 'bandwidth_usage':
returnalg_value.total_bw_usage  # random.randint(0, 100)
elifself.context.get('mode', '') == 'suported_cameras':
returnlen(cameras)
defget_critical_value(self, cameras: List[Camera]):
```

```
alg_value = AlgorithmValue.load()
ifself.context.get('mode', '') == 'bandwidth_usage':
returnalg_value.total_allowed_bw_usage  # 85
elifself.context.get('mode', '') == 'suported_cameras':
returnalg_value.supported_cameras_telicomm  # 15
    # defget_max_value(self, cameras: List[Camera]):
    #     ifself.context.get('mode', '') == 'bandwidth_usage':
    #         return 100
    #     elifself.context.get('mode', '') == 'suported_cameras':
    #         return 50
    #
    # defget_min_value(self, cameras: List[Camera]):
    #     ifself.context.get('mode', '') == 'bandwidth_usage':
    #         return 0
    #     elifself.context.get('mode', '') == 'suported_cameras':
    #         return 0
classGraphStatSerializer(serializers.ModelSerializer):
timestamp = serializers.SerializerMethodField()
input_rate = serializers.SerializerMethodField()
output_rate = serializers.SerializerMethodField()
classMeta:
model = Camera
fields = ('timestamp', 'input_rate', 'output_rate', 'actual_bw_usage',
'unprocessed_bw_usage')
defget_input_rate(self, camera: Camera):
try:
stat = camera.stat_set.latest()
returnstat.inputBitrate
exceptStat.DoesNotExist:
return 0
defget_output_rate(self, camera: Camera):
try:
stat = camera.stat_set.latest()
returnstat.outputBitrate
exceptStat.DoesNotExist:
return 0
defget_timestamp(self, camera: Camera):
returnTimestampField().to_representation(datetime.datetime.now())
classGraphSerializer(serializers.ModelSerializer):
```

```
stat = serializers.SerializerMethodField()
defget_stat(self, camera: Camera):
serializer = GraphStatSerializer(camera)
returnserializer.data
classMeta:
model = Camera
fields = ('id', 'stat')
classOnlyOneField(object):
def __init__(self):
self.fields = []
self.serializer = None
defset_context(self, serializer):
self.fields = [fieldforfieldinserializer.fieldsifnotserializer.fields[field].required]
self.serializer = serializer
def __call__(self, value: OrderedDict, *args, **kwargs):
already_found = False
forfieldinself.fields:
iffieldinvalueandvalue[field] isnotNone:
ifalready_found:
raiseserializers.ValidationError({"message": f"Onlyonefield {str(self.fields)} isrequired"})
self.serializer.field_name = field
already_found = True
ifnotalready_found:
raiseserializers.ValidationError({"detail": f"Onefield {str(self.fields)} isrequired"})
classCameraSetStatSerializer(serializers.ModelSerializer):
maxFrameCount = serializers.IntegerField(required=False, write_only=True,
allow_null=True)
throttleEnabled = serializers.BooleanField(required=False, write_only=True,
default=None, allow_null=True)
bufferPullAllowed = serializers.BooleanField(required=False, write_only=True,
default=None, allow_null=True)
throttleValue = serializers.IntegerField(required=False, write_only=True,
allow_null=True)
camera_id = serializers.PrimaryKeyRelatedField(queryset=Camera.objects.all(),
source='camera', write_only=True)
classMeta:
model = Camera
validators = (OnlyOneField(), )
```

```
fields = ('camera_id', 'maxFrameCount', 'throttleEnabled', 'bufferPullAllowed',
'throttleValue')
defsave(self):
ifself.is_valid(raise_exception=True):
self.result = setter(self.validated_data['camera'].ip, self.field_name,
self.validated_data[self.field_name])
classControlSerializer(serializers.ModelSerializer):
classMeta:
model = Control
validators = (OnlyOneField(), )
fields = ('telicomm_mode', )
defsave(self):
ifself.is_valid(raise_exception=True):
old = Control.load()
ifold.telicomm_mode == Trueandself.validated_data.get('telicomm_mode', None) ==
False:
apply_after_turn_off.delay()
self.update(old, self.validated_data)
classUserSerializer(serializers.ModelSerializer):
cameras = serializers.PrimaryKeyRelatedField(many=True,
queryset=Camera.objects.all())
classMeta:
model = User
fields = ('id', 'username', 'cameras')
deftest():
frompprintimportpprint
importjson
fromdjangorestframework_camel_case.parserimportCamelCaseJSONParser
fromdjangorestframework_camel_case.renderimportCamelCaseJSONRenderer
fromrest_framework.parsersimportJSONParser
importio
stream = io.BytesIO(asd)
data = JSONParser().parse(stream)
    s = StatSerializer(data=data)
print(s.is_valid(raise_exception=True))
print(s.validated_data)
s.save()
    c = CameraSerializer(instance=Camera.objects.filter(pk=1), many=True)
    print(json.loads(CamelCaseJSONRenderer().render(c.data).decode()))
```

```
exit(0)
    #
    #
    # stream = io.BytesIO(asd)
    # data = JSONParser().parse(stream)
    # s = StatSerializer(data=data)
    #
    # print(s.is_valid(raise_exception=True))
    # print(s.validated_data)
    # s.save()
gs = GraphSerializer(instance=Camera.objects.all(), many=True)
    pprint(json.loads(CamelCaseJSONRenderer().render(gs.data).decode()))
exit(0)
    # ids = [camera.pk forcamerainCamera.objects.all()]
    # latest_stats = [Stat.objects.get() foridinids]
res = Stat.objects.raw(
        "SELECT * FROM (SELECT camera_id, id FROM api_stat ORDER BY timestamp
DESC) t GROUP BY t.camera_id, id")
    #res = Stat.objects.order_by('-timestamp').distinct('camera_id')
ms = MeterSerializer(instance=res, context={'mode': 'suported_cameras'})
pprint(CamelCaseJSONRenderer().render(ms.data))
exit(0)
cn_cl = ControlSerializer(instance=Control.load())
data = CamelCaseJSONRenderer().render(cn_cl.data)
print(data)
data = CamelCaseJSONParser().parse(io.BytesIO(b'{"telicommMode":true}'))
cn_cl = ControlSerializer(data=data)
print(cn_cl.is_valid(raise_exception=True))
print(cn_cl.validated_data)
cn_cl.save()
exit(0)
    # for i inrange(100):
    a = Camera.objects.all().first()  # type: Camera
print("dsf")
a.status = Camera.ONLINE
    ccc = CameraSerializer(instance=a)
fromdjangorestframework_camel_case.renderimportCamelCaseJSONRenderer
fromrest_framework.renderersimportJSONRenderer
pprint(JSONRenderer().render(ccc.data))
```

```
print(repr(ccc))
print(a.status)
stream = io.BytesIO(asd)
data = JSONParser().parse(stream)
    s = StatSerializer(data=data)
print(s.is_valid(raise_exception=True))
print(s.validated_data)
s.save()
    c = CameraSerializer(instance=Camera.objects.filter(pk=3), many=True)
    pprint(json.loads(CamelCaseJSONRenderer().render(c.data).decode()))
csss = CameraSetStatSerializer(data={
        'camera_id': 4,
        # 'maxFrameCount': 20,
        'throttleValue': 546461
    })
    # print(csss.is_valid(raise_exception=True))
try:
data = csss.save()
exceptserializers.ValidationErrorasex:
pprint(CamelCaseJSONRenderer().render(data=ex.detail).decode())  # STATUS 400
else:
pprint(CamelCaseJSONRenderer().render(data=data).decode())  # STATUS 200
if __name__ == '__main__':
test()
```

ДОДАТОК Д

Користувацький інтерфейс

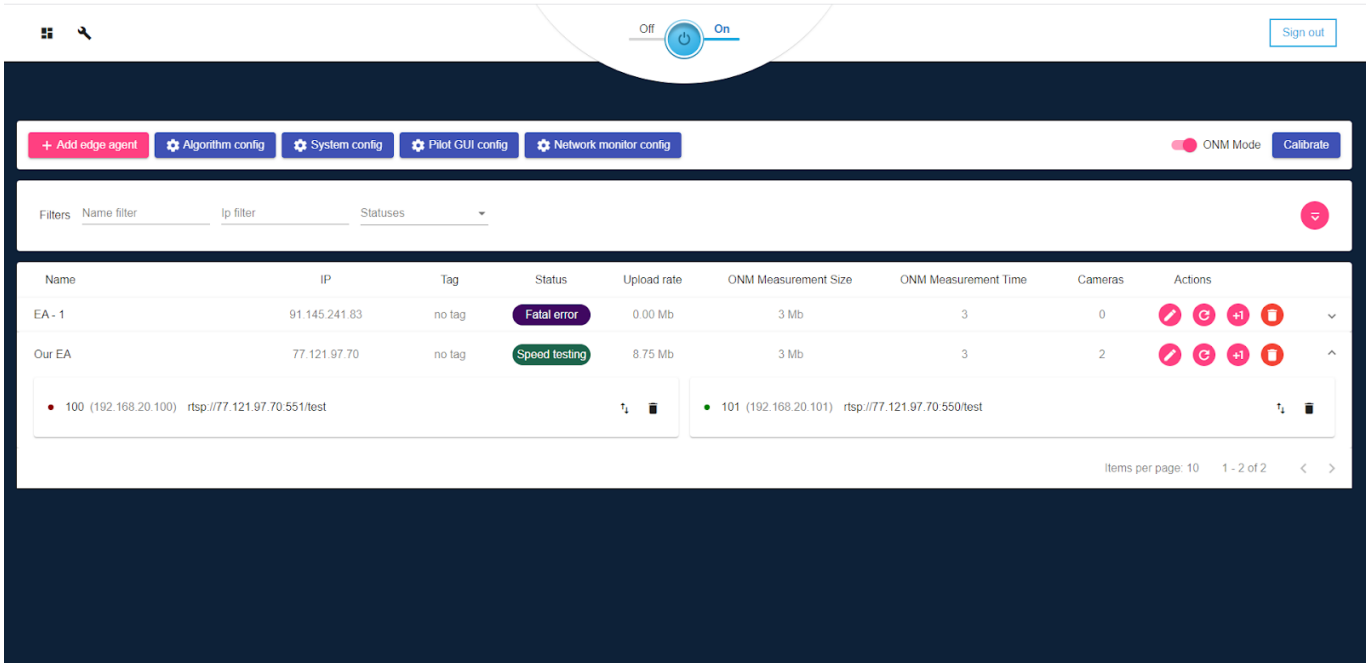

Рисунок Д.1 — Головне вікно керування