Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри,
канд. техн. наук, доцент
_____ Я. М. Крайник
« __ » _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Програмно-апаратний комплекс дистанційного керування у системах транспортної логістики

Спеціальність 123 Комп'ютерна інженерія

123 – КР.1 – 405.21810516

Студент: _____ М. М. Матюшок
« __ »_____ 2022 р.

Керівник: старший викладач _____ І. С. Бурлаченко
« __ » _____ 2022 р.

Миколаїв 2022

ЗАВДАННЯ

на виконання бакалаврської роботи

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 1

ця сторінка після друку буде замінена

ЗАВДАННЯ

на виконання бакалаврської роботи

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 2

ця сторінка після друку буде замінена

АНОТАЦІЯ

1 сторінка !!!!

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 1

ця сторінка після друку буде замінена

ABSTRACT

1 сторінка !!!!

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 2

ця сторінка після друку буде замінена

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	Integrated development environment
CSI	Camera Serial Interface
SSH	Secure Shell
RTMP	Real Time Messaging Protocol
RTSP	Real Time Streaming Protocol
ЄС	Європейський Союз
з/п	за порядком
ОС	операційна система
РМ	робоче місце
АПЗ	апаратно-програмне забезпечення

ВСТУП

В наш час служби доставки стали дуже популярними та затребуваними, особливо доставка продуктів чи їжі з різних кафе, ресторанів та магазинів. Люди замовляють кур'єрські доставки додому, на місце роботи, та навіть під час прогулянок замовляють доставку. У великих містах, таких як Київ, Львів, Одеса, Харків служби доставки розвинені найбільше та мають покриття доставки по всій території міста. Інакше кажучи, можна замовити доставку з будь-якої точки міста.

Існує досить багато служб доставки, наприклад найбільш популярні компанії Raketa, Glovo, Uber Eats.

Raketa– це класичний представник служби доставки. Компанія була заснована ще в 2018 році, до приходу на ринок Glovo та Uber Eats. За весь час свого існування були відпрацьовані всі нюанси та проблеми, крім того це український проект, на відміну від двох останніх.

Glovo – це міжнародний проект служби доставки, який з'явився на нашому ринку з 2018 року, проте трішки пізніше ніж Raketa.

Uber Eats – це також міжнародна служба доставки, яка зайшла на український ринок в 2019 році.

До всіх цих сервісів доставки підключені різноманітні ресторани та кафе. Крім цього, кожен сервіс має мобільний додаток для легкості та зручності користування, достатньо декількох тапів по екрану смартфона і кур'єр буде везти замовлення.

Проте, під час пандемії Covid – 19, були введені необхідні обмеження та відповідні санітарні норми, що негативно вплинуло на всі служби доставки.

Під час карантину почали стрімко розвиватись різні проекти та стартапи по створенню роботів-кур'єрів та їх подальше впровадження в процес доставки. Це б замінило кур'єрів на роботів та зробило б процес доставки більш автономним. Серед таких проектів найбільш відомі – Естонський

стартап Starship Technologies, стартап Refraction AI заснований в Мічиганському університеті та Xiaomanlv від Damo Academy.

Дана робота присвячена розробці апаратно-програмного рішення для навігації робота-кур’єра, а саме передача відеозображення в режимі реального часу.

Мета дослідження: розробити апаратно-програмний комплекс для для передачі відеозображення в режимі реального часу .

Об’єкт дослідження: методи передачі відео та керуючих команд в режимі реального часу.

Предмет дослідження: програмно-апаратний комплекс передачі відеозображення для дистанційного керування у системах транспортної логістики на базі мікрокомп’ютера Raspberry Pi Model B 3+.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

- аналіз існуючих технологій та систем потокової передачі відео;
- підібрати необхідне апаратне забезпечення;
- підготовити та налаштувати пристрій для подальшої розробки;
- встановити необхідне програмне забезпечення зі стеку технологій необхідного для реалізації проєкту, а саме, операційну систему для Raspberry Pi, Node.JS сервер та Express JS;
- реалізувати зв’язок апаратної частини з сервером;
- реалізувати передачу відеозображення в режимі реального часу;
- реалізувати передачу керуючих команд в режимі реального часу;
- розробити питання з охорони праці та безпеки життєдіяльності.

Для досягнення поставленої мети, було використано наступні **методи дослідження**:

- експеримент (розробка серверної частини);
- тестування (робота програмно-апаратного комплексу).

Практичне значення отриманих результатів: розроблений програмно-апаратний комплекс може бути використаний у службах доставки, наприклад

Glovo або Uber Eats, також може бути використаний різними гіпермаркетами чи кафе. Можливе удосконалення додатковим апаратним обладнанням і програмним забезпеченням, може використовуватися при розробці стартапу.

Апробація Матюшок М.М., Бурлаченко І. С. Системи навігації автономних безпілотних наземних транспортних засобів. «Ольвійський форум-2022: Стратегії країн Причорноморського регіону в геополітичному просторі»: зб. тез доп. XVI Міжнародної наукової конференції. м. Миколаїв, 23–26 червня 2022 р. Миколаїв, 2022.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМ ПЕРЕДАЧІ ВІДЕО В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Дана робота ставить за мету розробку програмно-апаратного комплексу, апаратну частину якого складає міні комп’ютер з відеокамерою, а програмну – клієнт серверна частина. Розглянемо деякі з існуючих систем передачі відео в режимі реального часу для керування дроном.

1.1. Аналіз стрімінгових систем

В наш час використання різних систем для онлайн трансляцій стало надзвичайно популярним. Такі системи використовуються майже скрізь, наприклад в рекламному бізнесі, ведення прямих ефірів або в більш складних комплексних проєктах, таких як організація дистанційного керування за допомогою передачі відео в режимі реального часу.

Слід зазначити, що організація стрімінгових систем [1] є доволі складним процесом, так як вимагає різних математичних обчислень.

Перш за все, потрібно підібрати відповідну технічну базу, проаналізувати наявні системні засоби та підібрати потрібні інструменти для розробки.

1.1.1 Налаштування прямої трансляції.

Налаштування прямої трансляції зводиться до декількох кроків:

− підключення відео та аудіо джерел, які записують вміст прямої трансляції до потокового пристрою, зазвичай в ролі такого пристрою виступає комп’ютер або ноутбук. Щоб передати зображення на канал прямої трансляції потрібно підключити ці джерела до апаратного кодувальника або слід скористатися картою відео захоплення, якщо використовується програмний енкодер;

− підключення відео та аудіо джерел, які записують вміст прямої трансляції до потокового пристрою, зазвичай в ролі такого пристрою виступає комп’ютер або ноутбук. Щоб передати зображення на канал прямої трансляції потрібно підключити ці джерела до апаратного кодувальника або слід скористатися картою відео захоплення, якщо використовується програмний енкодер;

− наступний крок – це налаштування енкодеру, пристрій, що перетворює відео та аудіо контент у потокові файли, які вже готові для використання на стрімінгових платформах. Під час налаштування енкодеру слід пам’ятати про деякі аспекти, а саме, мінімальна роздільна здатність має бути 1280 x 720, крім цього, мінімальний бітрейт має бути 3000 Кбіт/с, та мінімальна частота кадрів повинна бути 30 кадрів;

− підключення енкодеру до стрімінгової платформи за допомогою ключа та URL-адреси, що надає стрімінгова платформа (лише у випадку використання стрімінгової платформи);

− перевірка інтернет з’єднання;

− запуск відеостріму.

Інакше кажучи, процес прямої трансляції включає в себе запис відео та аудіо контенту, його кодування, та використання на стрімінгових платформах.

Для організації прямих трансляцій використовується досить багато обладнання та програмного забезпечення. Тож потрібно як слід розібратися з підбором необхідних комплектуючих та їх налаштуванням.

1.1.2 Обладнання для організації прямої трансляції

Для організації прямої трансляції необхідне наступне обладнання:

− відеокамера;

− мікрофон;

− карта відео захоплення;

− енкодер.

Використання комплектуючих та програмного забезпечення залежить від мети, яка зазначена в проєкті. Наприклад, для ведення звичайного відео стріму, що в наш час є досить популярним, потрібні всі перелічені вище пристрої, проте можна вести пряму трансляцію використовуючи лише смартфон, в ньому вже вбудоване все необхідне обладнання, потрібно лише встановити програмне забезпечення.

Важливим компонентом кожної стрімінгової системи є карта відеозахоплення. Це проміжний електричний пристрій, який перетворює аналоговий сигнал в цифровий відеопотік. Зазвичай такі карти мають PCI-e інтерфейс, проте на ринку наявні і карти з USB інтерфейсом, але їх значно менше. Використовувати карту відеозахоплення слід у випадку використання програмного енкодеру. Коли використовується апаратний енкодер, використання карти відеозахоплення є недоцільним.

1.1.3 Відео енкодери

Поговоримо про енкодери. Щоб поділитися своїм відео та аудіо потоком зі світом, вам знадобиться енкодер.

Енкодер – це пристрій, який перетворює відеофайли з одного формату в інший. Іншими словами, він бере необроблений канал з камери та передає його у доступний для перегляду вміст на вашій головній трансляції в прямому ефірі.

Існує два типи енкодерів: апаратні енкодери та програмні енкодери.

Ось короткий опис апаратних та програмних енкодерів.

Апаратні енкодери:

- спеціально розроблені для прямої трансляції. Вони звільняють ваш комп’ютер для виконання інших завдань;
- не вимагають карти захоплення;
- менш доступні і складні для оновлення.

Оскільки апаратні енкодери [2] звільняють ваш комп’ютер для зосередження на інших завданнях, вони чудово підходять для більш

професійних прямих трансляцій. Таким чином, ПК може зосередитися на прямій трансляції, поки енкодер працює у фоновому режимі. Схема роботи апаратного енкодеру показана на рисунку 1.1.

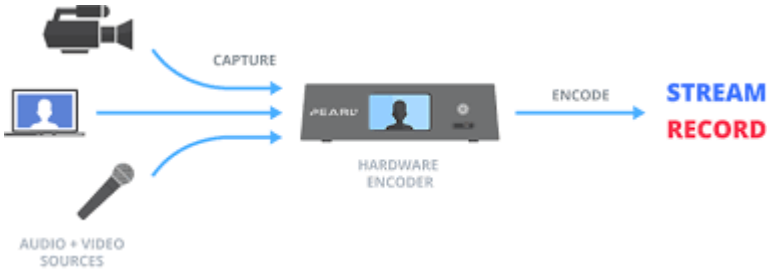

Рисунок 1.1 – Схема роботи апаратного енкодеру

Програмні кодери:

– чудовий варіант для тих, хто хоче уникнути зайвих налаштувань. Слід пам’ятати, що вони залежать від обчислювальної потужності вашого комп’ютера;

– може знадобитися карта відео захоплення;

– різноманітність цін і варіантів оновлення;

– перевагою програмних енкодерів є те, що вони працюють самі по собі, не вимагаючи додаткових підключень та налаштувань.

Існує досить багато програмних енкодерів, проте серед них є декілька найкращих, а саме: Livestream Studio 6, OBS Studio, Streamlabs OBS. . Схема роботи апаратного енкодеру показана на рисунку 1.2.

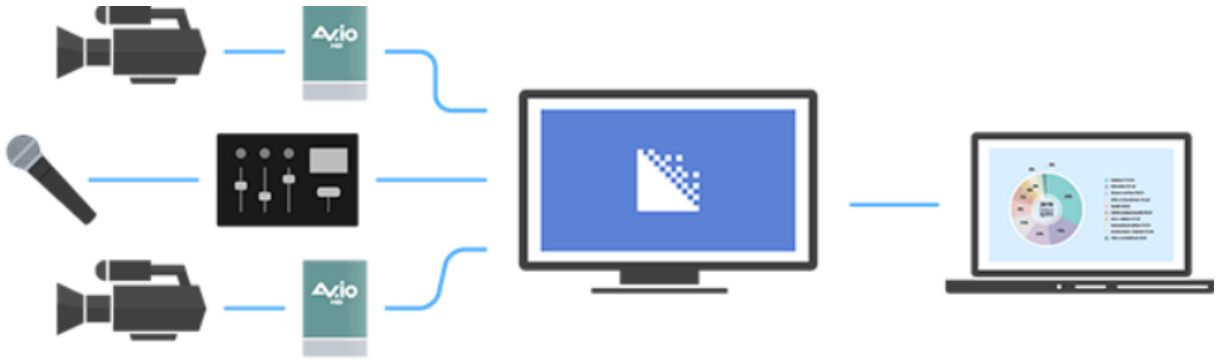

Рисунок 1.2 – Схема роботи програмного енкодеру

Стисле порівняння характеристик апаратного та програмного енкодеру наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння характеристик характеристик апаратного та програмного енкодеру

Назва характеристики	**Апаратний енкодер**	**Програмний енкодер**
Призначення	Для професійних прямих трансляцій та інтернет-мовлення	Для ведення прямих ефірів, наприклад Twitch
Переваги	Не навантажує комп’ютер додатковими процесами; Великий діапазон відео-форматів;	Не потребує додаткових налаштувань та підключень; Легко оновлюється;
Недоліки	Складність в підключені; Обмеження в оновлені;	Використовує ресурси комп’ютера

Як видно із порівняльної таблиці 1.1 кожен вид енкодеру має свої переваги та недоліки. Перш за все, кожен вид має різну сферу застосування. Який вид енкодеру підходить найбільше залежить від проєкту.

1.1.4 Протоколи потокової передачі відео

Перш за все потрібно вказати, що RTMP [3] та RTSP [4] – це протоколи потокової передачі даних, в свою чергу це означає, що вони представляють собою набори певних правил інтерфейсу логічного рівня, який визначає метод передачі відео, аудіо та інших типів даних між різними платформами, системами та пристроями. Ці правила потрібні щоб задавати стандарт передачі інформації та обробку помилок, що необхідно для нормальної роботи будь-якої відео трансляції.

RTMP – це стандартизований протокол передачі мільтимедійних даних через інтернет. Ця технологія була розроблена Macromedia. Як протокол зв’язку технологія RTMP забезпечує стабільну і плавну передачу даних, необхідних для передачі та прийому відео в реальному часі. Це досягається методом фрагментування потоку даних на невеликі однакові частини, для

відеоданих – це 128 байт та їх послідовна передача на приймаючий пристрій, який знову збирає їх в відеопотік. Зараз RTMP існує в п'яти варіантах.

Таблиця 1.2 – Варіанти RTMP

Назва	Опис
RTMP	Базовий протокол на основі TCP
RTMPS	Використовує додаткове шифрування TSL або SSL, щоб гарантувати, що потік не буде перехоплений через інтернет. Підходить для корпоративних зустрічей і закритих відеоконференцій.
RTMPE	Використовує шифрування безпеки Adobe і є більш легким шаром шифрування, ніж RTMPS
RTMPT	Інкапсульований з HTTP для обходу міжмережевих екранів та фільтрації корпоративного трафіку
RTMPFP	Використовує UDP замість TCP

Пряма трансляція відео працює наступним чином: камера знімає відео та відправляє його на сервер відеоплатформи через кодувальник, цей етап називається «перша миля». Потім сервер опрацьовує цей потік та передає його далі через мережу доставки контенту CDN для розповсюдження відеопотока на пристрої користувачів, цей етап називається «остання миля». На рисунку 1.3 наведена діаграма роботи RTMP.

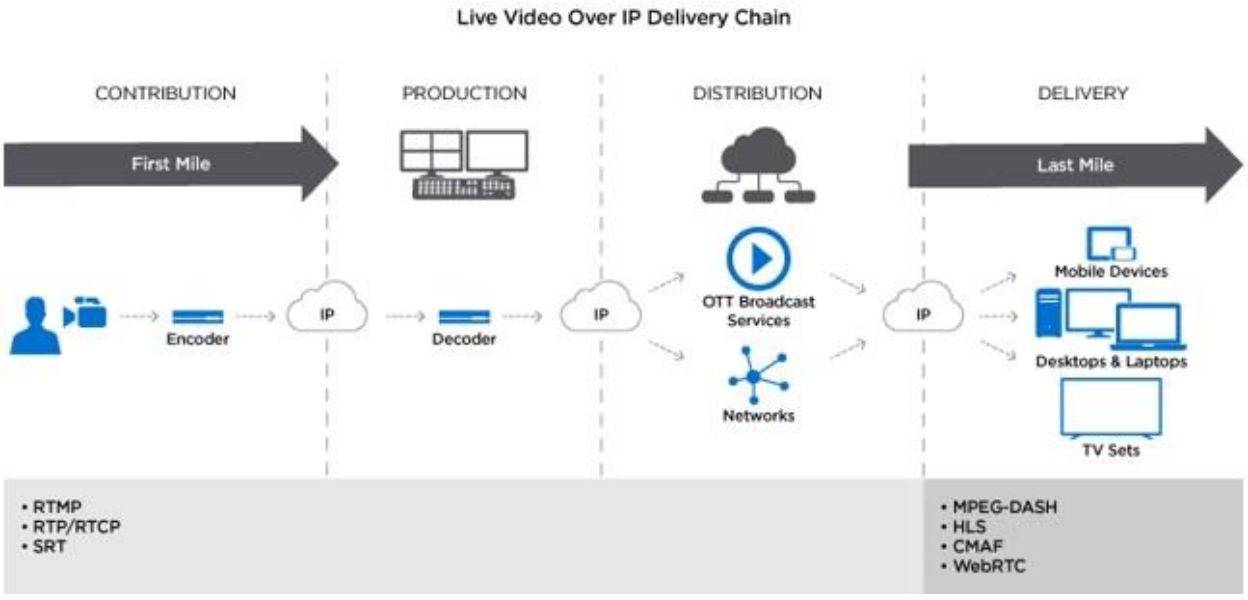

Рисунок 1.3 – Діаграма роботи RTMP

На діаграмі показані чотири етапи доставки відео в реальному часі. Перший етап зазвичай називають «перша миля» - передача відео від камери до сервера, а четвертий етап – «остання миля», відео доставляється на екран користувача.

Доставку пакетів RTMP виконує за декілька секунд в три етапи:

- клієнт та сервер обмінюються інформацією;
- встановлення зв’язку. Клієнт та сервер встановлюють всі параметри і специфікації з’єднання;
- передача потоку.

RTSP – це менш відомий, ніж RTMP набір правил для потокової передачі відео за допомогою інтернету.

Для забезпечення плавної та узгодженої потокової передачі даних RTSP використовує два інші мережеві протоколи зв'язку — TCP для видачі та прийому команд управління (наприклад, запиту на відтворення або зупинку) та UDP (протокол користувальницьких датаграм) для доставки аудіо, відео та даних. Завдяки цьому клієнт може почати відтворювати поток RTSP під час завантаження потоку.

Хоча RTSP можна використовувати як для прямих трансляцій, так і для передачі відео на запит, зараз його зазвичай використовують на останній милі для передачі відеопотоку з хмари в програвач пристрою користувача, оскільки даний протокол дозволяє глядачеві відтворювати, зупиняти і перемотувати відео. Крім того, RTSP також популярний у системах, де потрібно передати відеосигнал з IP камер. На рисунку 1.4 показана схема роботи RTSP.

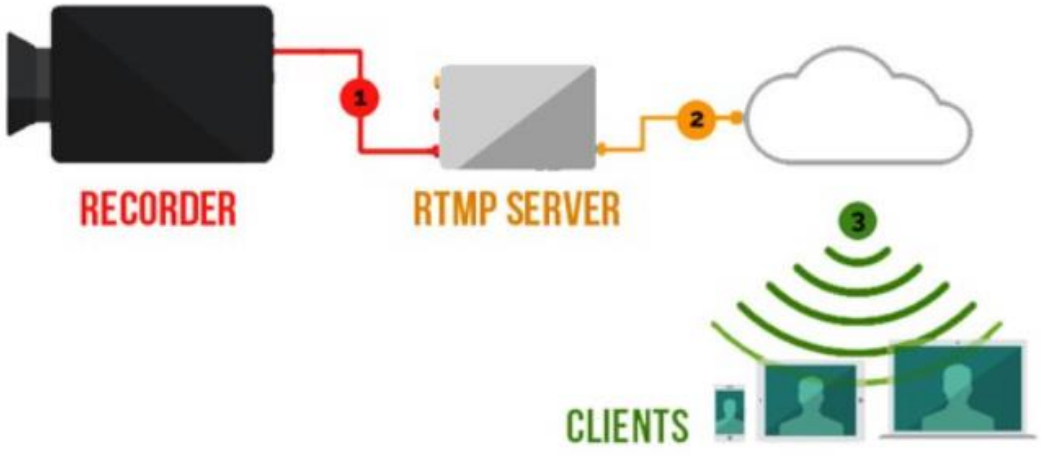

Рисунок 1.4 – Схема роботи RTSP

Як протокол зв'язку RTSP працює в такий спосіб. Коли користувач (програма, програма або камера) хоче передати відеосигнал з віддаленого джерела, пристрій відправляє RTSP-запит на спеціальний сервер (або платформу потокового відео), щоб визначити доступні параметри, такі як відтворення, пауза, перемотування та запис. Потім сервер повертає на пристрій сигнал зі списком запитів, які може приймати через RTSP.

Як тільки пристрій (плеєр) дізнається список команд і як зробити запит, він передає запит опису відеоконтенту сервер потокової передачі, і сервер відповідає описом цього мультимедіа. Далі пристрій відправляє запит на завантаження, а сервер відповідає інформацією про транспортний механізм, і далі ініціюється процес потокової передачі через зазначений механізм.

Оскільки RTSP залежить від виділеного сервера та покладається на RTP, цей протокол не підтримує шифрування відеоконтенту або повторну передачу втрачених пакетів. Крім того, RTSP не сумісний з HTTP, отже, він не дозволяє безпосередньо відтворювати відеопотік у веб-браузері. Для цього потрібно конвертувати відеопотік через спеціальний проміжний сервер.

Стисле порівняння характеристик RTMP та RTSP наведено в таблиці 1.3.

Таблиця 1.3 – Порівняння характеристик RTMP та RTSP

Назва характеристики	**RTMP**	**RTSP**

Затримка	3-30 секунд	2-5 секунди
Аудіокодеки	AAC, AAC-LC + v1, v2, MP3, Speex, Opus, Vorbis	AAC, AAC-LC + v1, v2, MP3, Speex, Opus, Vorbis
Відеокодеки	H.264, VP8, VP6. Sorenson Spark, Screen Video v1/v2	H.265, H.264, VP9, VP8
Сумісність відтворення	Flash Player, Adobe AIR, RTMP-сумісні плеєри	Qucktime Player, RTSP/RTP-сумісні плеєри
Переваги	Низька затримка; Низька буферізація; Стабільність;	Низька затримка; Підтримка IP-камер; Сегментована потокова передача; Можливості налаштування;
Недоліки	Несумісність з HTML5; Несумісність з HTTP; Можливі проблеми з масштабуванням; Вразливість смуг пропускання;	Несумісність з HTTP; Низька популярність;
Варіанти форматів	RTMP, RTMPE RTMPS RTMPFP	Весь стек RTP, RTCP (протокол керування в режимі реального часу), RTSP

1.1.5 Відеокодеки

Відеокодек [5] – це програма або алгоритм стиснення та відновлення стиснених відеоданих. Кодек – це пристрій або програма, яка визначає яким чином потрібно стиснути відео контент.

Одним із головних аспектів під час організації передачі відеозображення є вибір кодеку для стиснення відеосигналу. Цей програмний елемент більшою мірою впливає на ціну обладнання, яке підтримує ті чи інші кодеки.

Найпоширеніші відеокодеки в наш час це: H.265, H.264, MJPEG, MPEG-4.

MJPEG (Motion JPEG) [6] – цей кодек використовує покадрову компресіюю. Це доволі старий алгоритм стиснення зображення. Принцип роботи цього алгоритму в тому що кожен кадр відео стискається окремо. Проте головна перевага цього підходу, в простоті алгоритму стиснення зображення, в свою чергу це не вимагає високопродуктивного процесора. Кодек MJPEG характеризується не раціональним використанням пропускного каналу зв'язку, так як при стисненні кадрів не враховується зміна в послідовності кадрів, що в свою чергу приводить до відправлення зайвої

(однакової) інформації. Таким чином даний кодек, актуальний лише для бюджетних систем.

MPEG-4 – міжнародний стандарт, який використовується для стиснення цифрового відео та аудіо сигналів. Переважно використовується для організації потокового відео. Кодек MPEG-4 як і H.264 стискає послідовність зображень. Головною відмінністю від MJPEG є те, що вони відправляють лише оновлення, тобто те що змінилось в кадрі. Такий алгоритм стиснення має назву зовнішньокадрова компресія. За рахунок використання цього алгоритму знижається навантаження на смугу пропускання. В порівнянні з MJPEG при однакових показниках якості зображення кодек H.264 [7] може зменшити розмір відео файлу більш, ніж на 80%, а у порівнянні з кодеком MPEG-4 кодек H.264 виграє приблизно на 50%. Даний кодек використовує складні алгоритми стиснення відео, та вимагає більшою продуктивності від системи, проте існує думка, що це кодек повністю витіснить MJPEG та MPEG-4.

H.265 [8] – це новий стандарт компресії відео, який потребує в два рази мешу смугу попускання каналу, що дозволяє працювати в мережах з меншою пропускною можливістю каналу. Порівняння з кодеком H.264 показано на рисунку 1.5.

Рисунок 1.5 – Порівняння смуг пропускання двох кодеків

При однаковій якості відео H.265 економить до 80% ресурсів мережі. Це добре видно на рисунку 1.6.

Resolution	Minimum Upload Speed*	
	H.264	H.265
480p	1.5 mbps	0.75 mbps
720p	3 mbps	1.5 mbps
1080p	6 mbps	3 mbps
4K	32 mbps	15 mbps

Рисунок 1.6 – Порівняння навантажень на мережу двох кодеків

Замість макроблоків, які використовувалися в H.264, в H.265 використовуються блоки з деревоподібною структурою кодування. Перевага кодека H.265 – у використанні блоків більшого розміру [9]. Це було показано в тестах PSNR з моделю кодеру HM-8.0, де порівнювались результати кодування з різними розмірами блоків. Порівняння кодеків H.264 та H.265 показані на рисунку 1.7.

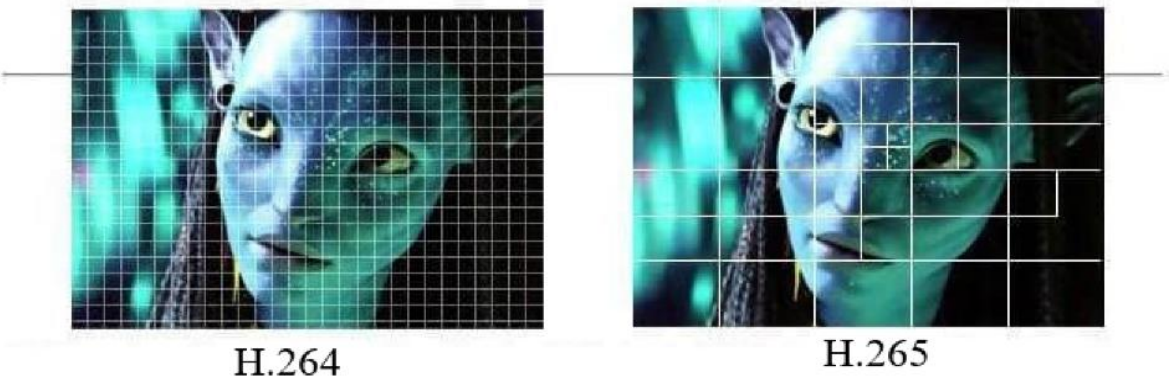

Рисунок 1.7 – Порівняння кодеків H.264 та H.265

Тести показали, що використання блоків великого розміру більш ефективно при кодуванні відео з високим розширенням. Також використання таких блоків зменшує час для декодування.

Таким чином, можна зробити висновок, що даний кодек необхідний у пристроях, які підтримують розширення 4K (Ultra HD).

1.2. Аналіз потокового відео на базі дрону з використанням UgCS.

1.1.6 Огляд системи UgCS

У даній роботі, досліджується система потокового відео від UgCS [10].

UgCS забезпечує надійну пряму відеотрансляцію з низькою затримкою від дронів у будь-яке місце штаб, офіс, операційний центр через WiFi або LTE. Підсистема UgCS Video створена за стандартом MISB для метаданих відео, яка підтримує координати GPS, висоту, інформацію про положення камери, а також відеозапис.

Відеоспостереження на базі дронів може виконувати кілька цілей:

- забезпечення високоякісного відеоспостереження для;
- транслювання відео від дрону в кімнату відеомоніторингу.

Дуже часто пілот дрона зосереджується на пілотуванні і, таким чином, не може контролювати відеотрансляцію. У таких випадках моніторинг може здійснювати окремий відеоспостерігач. Однією з переваг такої установки є те, що відеоспостерігач може залишатися в автомобілі або наметі, захищеному від негоди, і при цьому виконувати дуже точний аналіз відео, поки дрон знаходиться в повітрі.

Налаштування польового відеоспостерігача може бути особливо корисним для:

- промисловий огляд вертикальних стін, дамб, ліній електропередач, мостів;
- пошуково-рятувальні роботи.

1.1.7 Потокове відео на основі дронів DJI: серії Mavic, серії Matrice, Phantom, Inspire

Налаштування потокового відео за допомогою дронів DJI складається з наступних компонентів: планшет на базі Android, з встановленим програмним забезпеченням UgCS для DJI, комп'ютер з операційною системою Windows з

UdCS Enterprise. Якщо мобільний пристрій та комп’ютер знаходяться в одній мережі Wi-Fi, вони підключаються автоматично. У деяких випадках може знадобитися ручне налаштування.

Щоб переглянути відеозапис, використовується UgCS Video Player. Схема підключених компонентів наведена на рисунку 1.8. Крім того, система UgCS може використовувати RTSP протокол.

Рисунок 1.8 – Схема підключених компонентів

1.1.8 Трансляція з дрону в кімнату відеомоніторингу

Потокове відео на основі дронів з UgCS ENTERPRISE можна розгорнути в розподіленій конфігурації з пілотами на місцях і спостерігачами в кімнаті відеоконтролю. Схема передачі потокового відео від дрону до камери відеомоніторингу показана на рисунку 1.9.

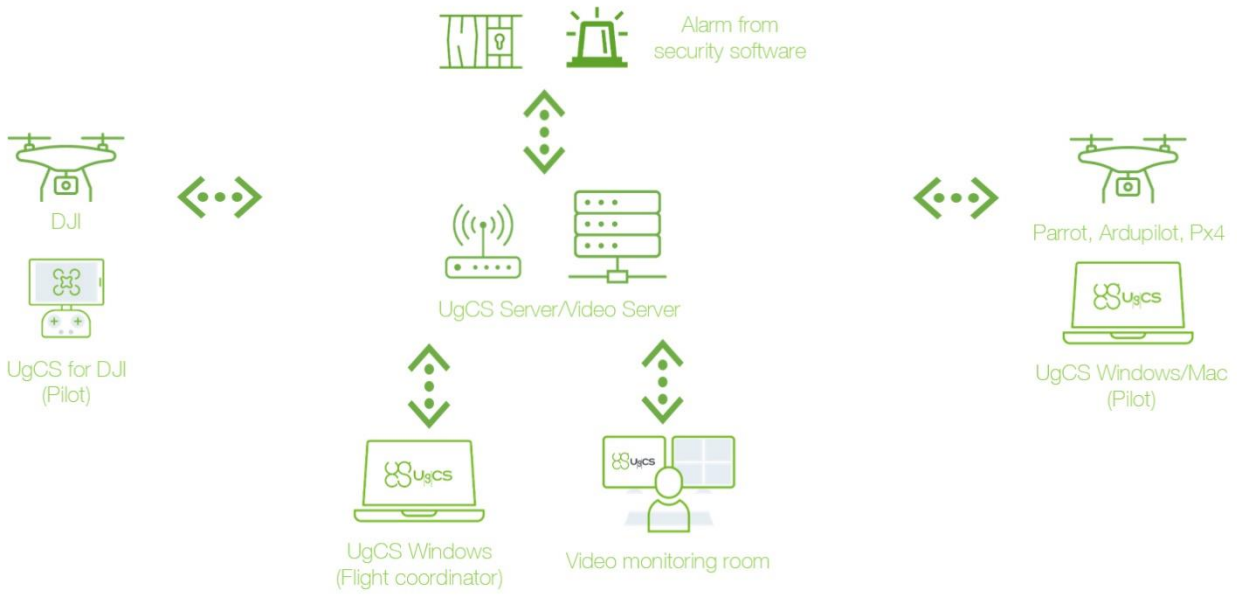

Рисунок 1.9 – Схема передачі відео з дрону

Відеосервер UgCS необхідно запустити в кімнаті відеомоніторингу на комп’ютері з загальнодоступною IP-адресою. Щоб транслювати відео з дрону, пілот повинен підключити програму до відеосервера UgCS.

1.1.9 Сумісність з MISB

Система побудована на основі стандарту MISB. Цей стандарт визначає формат потокового медіа, який підтримує передачу метаданих між відео- та аудіопотоками. Транспортний потік базується на стандарті MPEG-TS, тоді як відеопотік кодується кодеком H.264, а потік метаданих кодується кодеком KLV рисунок 1.10.

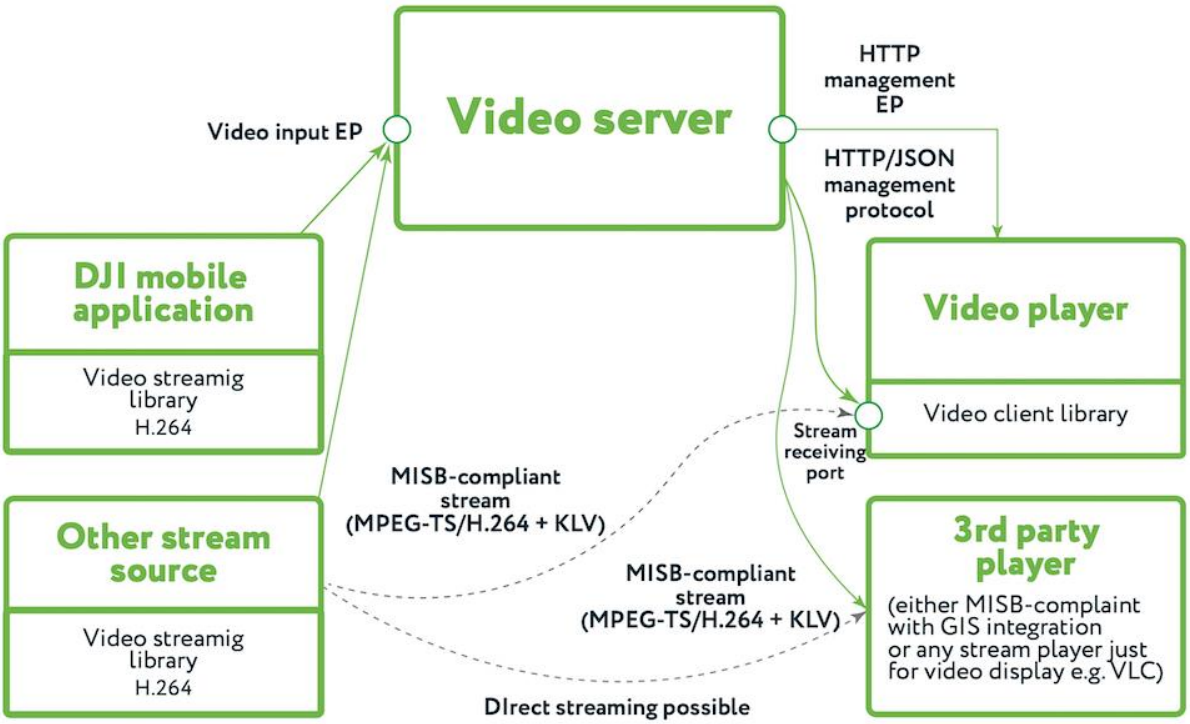

Рисунок 1.10 – Налаштування відео потоку на базі дрону з використанням UgCS

Відеосервер використовується для мультиплексування потоків між кількома клієнтами, наприклад для запису відео. Формат введення та виведення потоку ідентичні, хоча протокол потокової передачі може відрізнятися (див. нижче). Тому це може бути додатковим компонентом для відтворення потоку безпосередньо з джерела. Однак у такому випадку аудиторія буде обмежена одним клієнтом (якщо не використовується багатоадресна передача UDP).

Потік можна відтворювати за допомогою більшості сучасних медіа-програвачів (наприклад, VLC), але потік KLV з метаданими не відображатиметься з такими програвачами. Відео також можна відтворювати за допомогою будь-якого MISB-сумісного програвача, який зазвичай інтегрований в деяку ГІС, щоб відео миттєво пов'язувалося з місцем на карті.

Відеосервер підтримує MPEG-TS з транспортом на основі UDP [11].

1.3. Висновки до розділу 1

В розділі 1 проаналізовано технології для побудови систем потокового відео. Розглянуті енкодери та різні кодеки, які можна використовувати в своїх проєктах. Крім цього, були розглянуті протоколи для передачі потокового відео. Також був проведений аналіз готової системи потокового відео на базі дрону з використанням системи UgCS.

Спираючись на аналіз сучасних технологій, використовуючи які, можна організувати систему потокової передачі відео, а саме аналіз сучасних кодеків та їх порівняння, можна дійти висновку, що для майбутньої розробки найкраще підходить кодек MJPEG. Головною перевагою даного кодеку є те що він не вимагає високопродуктивного апаратного забезпечення.

РОЗДІЛ 2
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДИСТАНЦІЙНОГО КЕРУВАНЯ У СИСТЕМАХ ТРАНСПОРТНОЇ ЛОГІСТИКИ

В якості апаратної частини комплексу дистанційного керування у системах транспортної логістики обрано мін-комп’ютер Raspberry Pi та Raspberry Pi Camera. Такий вибір зумовлено невеликою вартістю конструкції, та широкими можливостями для подальшого вдосконалення проекту.

2.1 Конкурентні системи потокової передачі відео

В наш час тема потокової передачі відео є досить популярною, через це існує велика різноманітність реалізацій таких систем. При виборі апаратної частини розробники спираються на співвідношення продуктивності та ціни. Крім цього, підбір комплектуючих залежить від мети яку розробники хочуть досягти.

2.1.1 Система потокової передачі відео на базі ESP32-CAM

ESP32 – це серія недорогих мікросхем з низьким енергоспоживанням, розроблена компанією Espressif Systems. Представляють собою систему на кристалі з інтегрованими контролерами радіозв’язку Wi-Fi, Bluetooth та Thread. В серіях ESP32 та ESP32-S використовуються ядра з архітектурою компанії Tensilica, а в серіях ESP32-C та ESP32-H – ядра з відкритою архітектурою RISC-V.

ESP32-CAM – це модуль який поєднує в собі плату для розробки ESP-32 і камеру OV2640 2MP. Особливість плати є компактні розміри та низький рівень енергоспоживання. Завдяки тому, що в плату вбудована камера, даний пристрій використовується в різний цілях, наприклад в якості камери відеоспостереження, системи розпізнавання обличчя та жестів, сканування QR кодів та системи потокової передачі відео. ESP32-CAM зображено на рисунку 2.1.

Рисунок 2.1 – Модуль ESP32-CAM

Основні характеристики ESP32-CAM [12] наведено в таблиці 2.1.

Таблиця 2.1 – Основні характеристики ESP32-CAM

Характеристика	**Значення**
Процесор	Tensilica Xtensa LX6 32 bit Dual-Core 160/240 MHz
SPI Flash пам’ять	32 Мбіт
SRAM	520 КБ
PSRAM	4 МБ
Напруга живлення	2.2 – 3.6 В
Споживання	80 mA
Бездротові модулі зв’язку	Wi-Fi 802.11 b/g/n, Bluetooth 4.2 BR/EDR + BLE
Потужність передавання	802.11b : 17±2 дБм (11Мбіт/с); 802.11g: 14±2 дБм (54 Мбіт/с); 802.11n: 13±2 дБм (@MCS7)
Робоча температура	0 - 40° C
Підтримка інтерфейсів	UART, SPI, I2C, PWM
GPIO-порти	9 шт.
Підтримка карт пам'яті	до 4 ГБ
Модуль камери	OV2640
Розміри	40.5 мм x 27 мм x 4.5 мм

Функціональна блокова діаграма мікроконтролера наведена на рисунку 2.2.

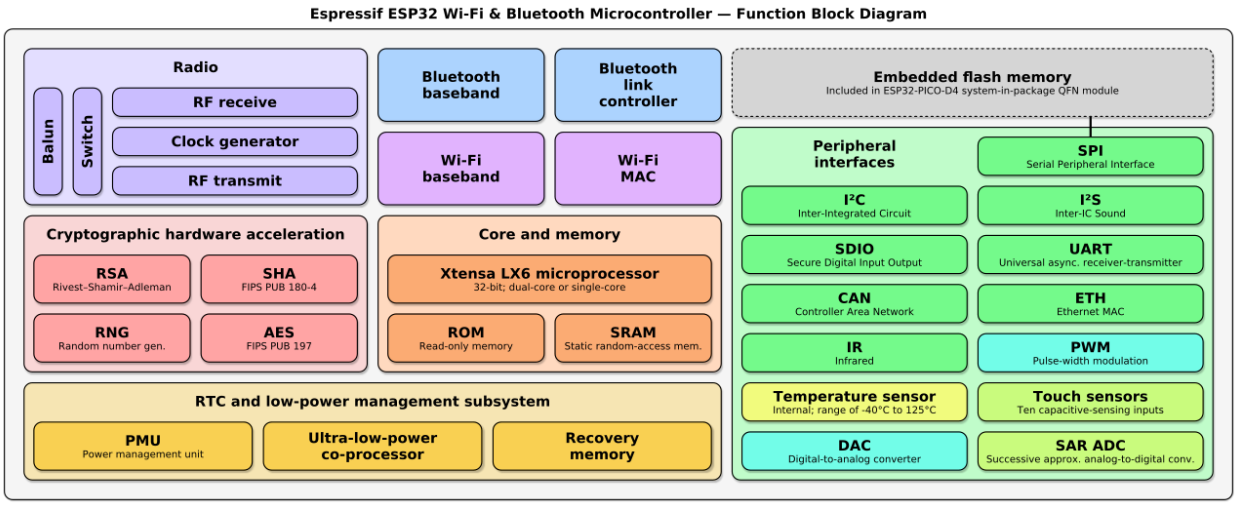

Рисунок 2.2 - Функціональна блокова діаграма мікроконтролера ESP32

Незважаючи на те що задача потокової трансляції відео потребує більшої обчислювальної потужності, ніж та, що може бути отримана з 8-бітними мікроконтролерами, в результаті порівняльних тестувань було виявлено що модуль ESP32-CAM здатен транслювати відео в режимі реального часу зі швидкістю, не набагато меншою за швидкість передачі відеозображення мікрокомп’ютерами Raspberry Pi.

На популярність цього модуля вплинула відкритість програмного забезпечення ESP32 та підтримка розробки багатьох середовищ програмування, в тому числі і фреймворку для розробки програм для мікроконтролерів Arduino.

2.2 Проектування системи потокової передачі відео

Після аналізу системи потокової передачі відео на базі ESP32-CAM, описаної у попередньому розділі, було вирішено використати мікрокомп’ютер Raspberry Pi та додатковий модуль Raspberry Pi Camera для подальшої розробки проєкту.

Використання даної апаратної платформи значно збільшує ціну проєкту, проте має багато переваг. Перш за все, Raspberry Pi – це потужний мікрокомп’ютер, на відміну від мікроконтролера ESP32. Що в свою чергу

означає можливість подальшого удосконалення та розширення функціоналу, з використанням додаткових модулів.

2.2.1 Опис апаратної платформи

Попередньо провівши аналіз ринку та порівняння різних моделей мікрокомп’ютерів Raspberry Pi, в якості апаратної платформи, для реалізації проєкту було обрано Raspberry Pi 3 Model B+.

Raspberry Pi 3 Model B+ - це нова оптимізована модель лінійки мікрокомп’ютерів третього покоління. Виробник покращив характеристики нової моделі Raspberry Pi, тепер дана модель працює на оновленому процесорі, з підвищеною частотою, також слід відмітити, що в новій версії Raspberry Pi 3 B+ з’явився Gigabit Ethernet.

Такий вибір зумовлено перш за все, співвідношенням ціни та обчислювальної потужності даної моделі, крім цього, на вибір моделі мікрокомп’ютера вплинула обмеженість бюджету проєкту, наявність функцій, які необхідні для розробки проєкту та можливість його подальшого удосконалення. В даному випадку, до апартної платформи Raspberry Pi підключається додатковий модуль камери Raspberry Pi Camera, через CSI порт[13], розгортається клієнтська частина системи потокової передачі відео та виконується трансляція відео в режимі реального часу до серверної частини застосунку.

Camera Serial Interface (CSI) – це специфікація Mobile Industry Processor Interface (MIPI) Alliance, направлена на зниження витрат на дисплейну підсистему в мобільних пристроях. Специфікація визначає послідовну шину та протокол зв’язку між хостом (джерелом зображення) та пристроєм (отримувачем зображення). Останніми активними специфікаціями інтерфейсу є: CSI-2 v3.0, CSI-2 v1.1. Схема передачі даних по CSI наведена на рисунку 2.3.

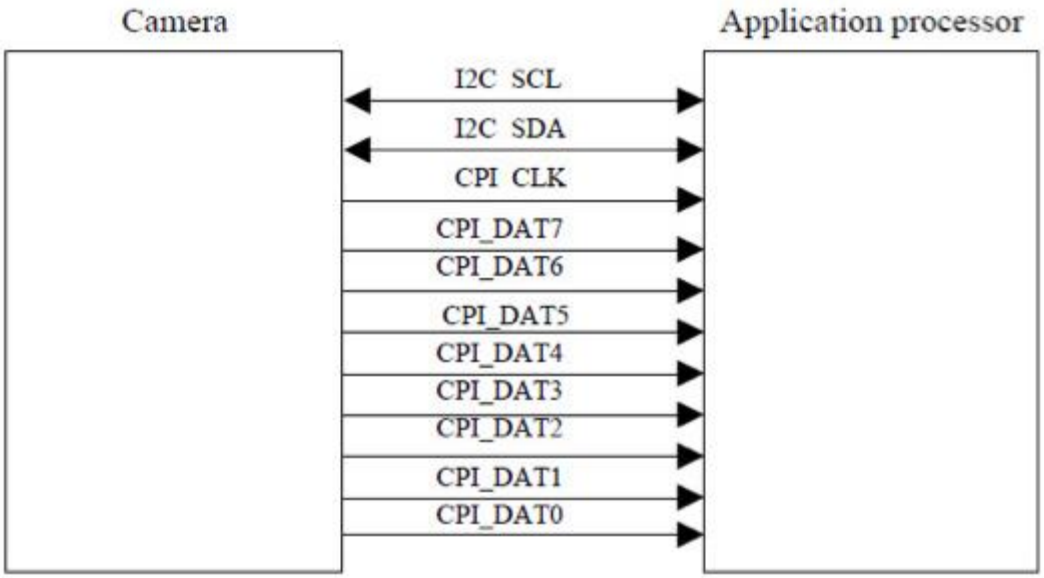

Рисунок 2.3 – Схема передачі даних по CSI

Обрана платформа Raspberry Pi 3 Model B+ зображена на рисунку 2.4.

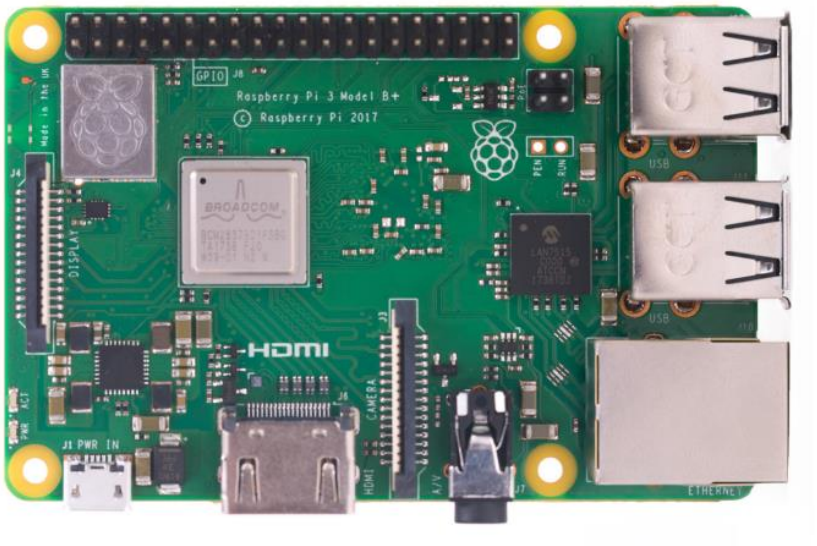

Рисунок 2.4 – Платформа Raspberry Pi 3 Model B+

Основні характеристики Raspberry Pi 3 Model B+ [14] наведено в таблиці 2.2.

Таблиця 2.2 – Основні характеристики Raspberry Pi 3 Model B+

Характеристика	**Значення**
Процесор	Broadcom BCM2837B0 64-bit, Cortex-A53
Робоча частота	1,4ГГц
Графічний процесор	VideoCore 4 3D

Оперативна пам'ять	1 ГБ
Gigabit Ethernet	Передача даних через USB шину до 300Mbps
Пам'ять для даних	Роз'єм для MicroSD
Бездротові інтерфейси	Wi-Fi (2.4GHz / 5GHz) IEEE 802.11 b/g/n/ac і Bluetooth 4.2 LE (на базі чіпа Cypress CYW43455)
Відео вихід	HDMI
Аудіо вихід	4-контактний роз'єм 3. мм з підтримкою композитного відео-виводу
GPIO	40 контактів
Роз'єм для камери	CSI
Роз'єм для дисплею	Raw LCD (DSI)
Живлення	5V 2.1A
Підтримка ОС	Raspbian, Windows 10 IoT Core, OpenELEC, OSMC, Pidora, Arch Linux, RISC OS

Комунікація з сервером, що обробляє відеопотік, відбувається через локальну мережу, до якої підключений комп’ютер та робоча плата з відеокамерою.

2.2.2 Опис використаної відеокамери для зйомки відеопотоку

Для розробки проєкту була підібрана камера яка підключається до апаратної платформи за допомогою порту CSI. Провівши аналіз ринку, була обрана камера Raspberry Pi 5mp. Слід відмітити, що існує досить велика кількість різних модулів камер для Raspberry Pi, проте через обмеження бюджету проєкту, була обрана звичайна камера без додаткових функцій та фільтрів.

Основні характеристики камери Raspberry Pi 5mp [15] наведено в таблиці 2.3.

Таблиця 2.3 – Основні характеристики камери Raspberry Pi 5mp

Характеристика	**Значення**
Роздільна здатність	5mp, 2592x1944 px
Матриця	¼ дюйма
Відео формати	1080p (30fps), 720p (60fps), VGA (60-90fps)

Габарити	24x24x9 мм
Шлейф	150 мм
Інтерфейс	CSI 15-pin
Сумісніть	Всі моделі A та B
Вага	3 гр

Основні характеристики сервопривода TowerPro MG90S наведено в таблиці 2.9.

Зовнішній вигляд Raspbery Pi 5mp наведено на рисунку 2.5.

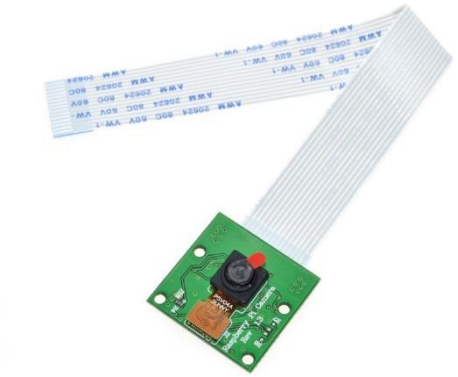

Рисунок 2.5 – Зовнішній вигляд Raspbery Pi 5mp

2.2.3 Живлення системи

Так як до Raspberry Pi 3+ не підключені додаткові модулі, котрі потребують живлення, був підібраний AC/DC адаптер, який відповідає вимогам живлення апаратної платформи, а саме 5 В та 3 А.

Блок живлення наведений на рисунку 2.6.

Рисунок 2.6 – Блок живлення для Raspberry Pi 3+

2.3 Тести продуктивності апаратної платформи

Після випуску мікрокомп’ютера Raspberry Pi 3 Model B+ були проведені різні тести продуктивності щоб порівняти цю модель з попередніми, крім цього були досліджені особливості температурного режиму та продуктивність під час нагрівання [16].

2.3.1 Нагрівання центрального процесору мікрокомп’ютера

В попередній моделі, а саме Raspberry Pi 3, були проблеми через нагрівання центрального процесору, особливо під час запуску програми з інтенсивними процесами. На рисунку 2.7 показано нагрівання апаратної платформи на SoC чіпі.

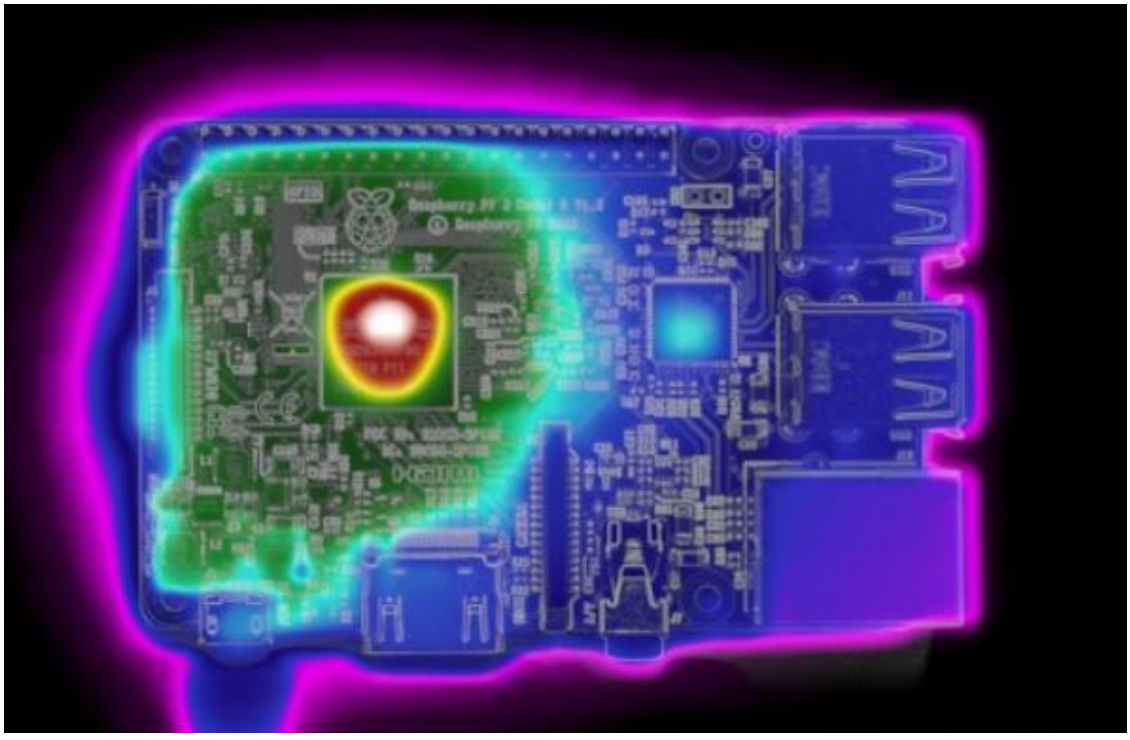

Рисунок 2.7 – Нагрівання чіпу Raspberry Pi 3

В порівнянні з Raspberry Pi 3 мікрокомп’ютер Raspberry Pi 3 Model B+ отримав покращену конструкції друкованої плати, в наслідок цього тепло розподіляється більш рівномірно, по всьому корпусу SoC, це означає що пікова температура буде нижча. При нагріванні тепло поширюється майже

через всю плату, на відміну від одностороннього нагрівання моделі Raspberry Pi 3. На рисунку 2.8 показано розподілення тепла по всьому SoC корпусу моделі Raspberry Pi 3 Model B+.

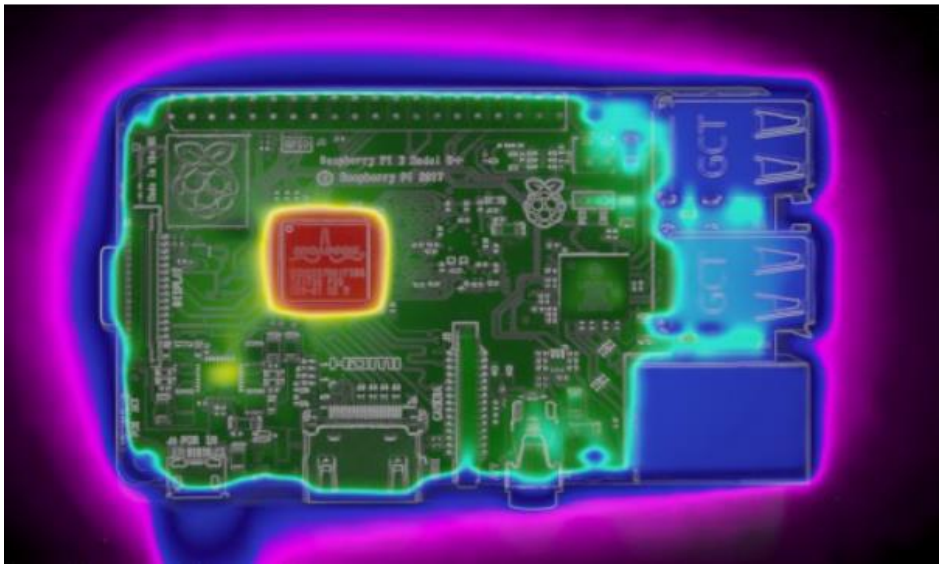

Рисунок 2.8 – Нагрівання чіпу Raspberry Pi 3 Model B+

2.3.2 Whetstone тест

Whetstone тест [17] вимірює продуктивність процесора в інструкціях Whetstone за секунду WIPS. Під час виміру продуктивність подається в мільйонах WIPS або MWIPS. Спираючись на отримані результати після проведення цього тесту, можна дійти висновку, що покращення архітектури попередньої моделі, а саме Raspberry Pi 2, дає значний приріст продуктивності, а додаткові 200 МГц, дає перевагу перед Raspberry Pi 3. Результати тесту наведені на рисунку 2.9.

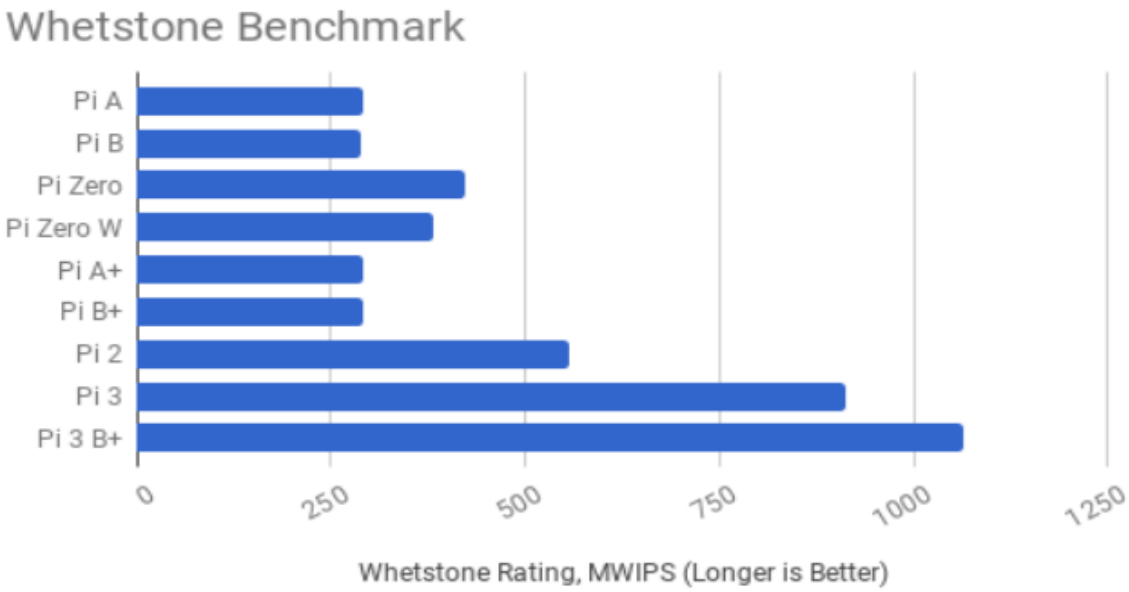

Рисунок 2.9 – Результат Whetstone тесту

2.3.3 Dhrystone тест

Вимірювання проводиться в мільйонах інструкцій за секунду або MIPS. Результати Dhrystone тесту наведені на рисунку 2.10.

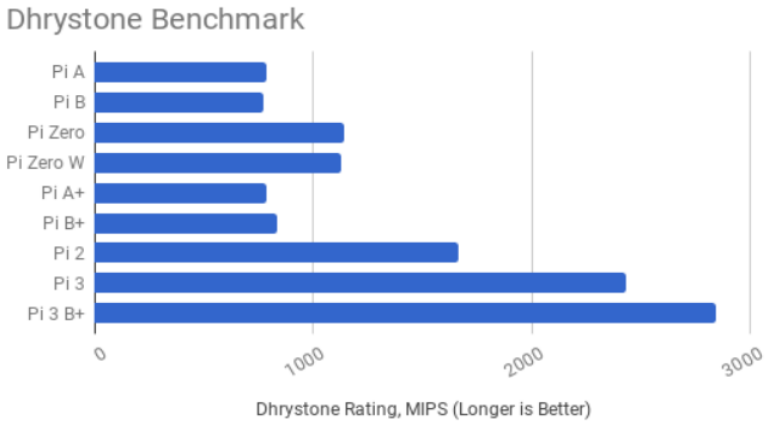

Рисунок 2.10 – Результати Dhrystone тесту

Raspberry Pi 3 B+ займає лідируючу позицію серед інших моделей мікрокомп’ютера.

2.3.4 SysBench CPU тест

Додаток SysBench [18] пропонує ряд синтетичних тестів, а тест продуктивності центрального процесору демонструє різницю між однопотоковим та багатопотоковому режимом роботи. На рисунку 2.11

наведені результати тестів, а саме кількість секунд необхідних для виконання тесту в однопотокових і багатопотокових режимах, причому багатопотоковий режим доступний лише для чотирьохядерних моделей Pi 2, Pi 3, Pi 3 B+.

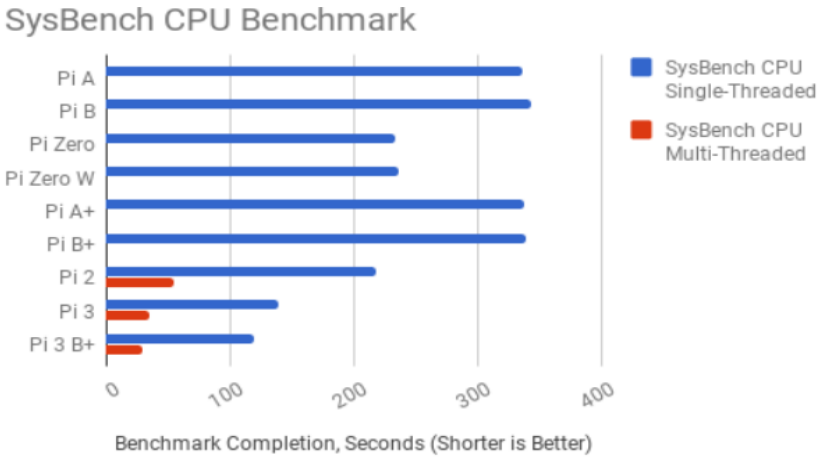

Рисунок 2.11 – Результат тестів SysBench

Результати проведення тесту демонструють що паралельність може означати значно більше ніж чисту продуктивність. Незважаючи на старіший дизайн SoC, використання всіх чотирьох ядер Pi 2, значно продуктивніше ніж використання одного ядра Pi 3 B+.

2.3.3 Thermal Throttling тест

Коли тепловий датчик на SoC досягає попередньо налаштованої максимальної температури, мікрокомп’ютер автоматично зменшує тактову частоту, щоб запобігти тепловому пошкодженню. Це відоме як «термічний дроселінг». І є сферою, в якій в минулому боролися користувачі Pi 3. На рисунку 2.12 наведені результати тесту.

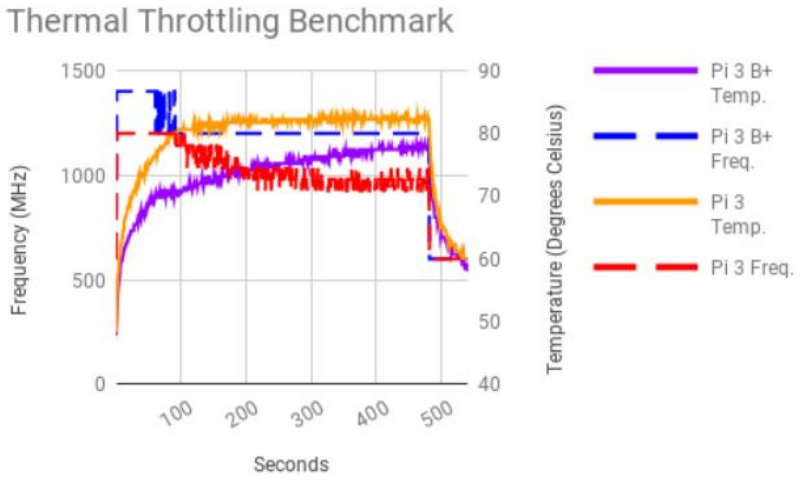

Рисунок 2.12 – Результати тесту

На графіку наведено порівняння Pi 3 та Pi 3 B+ у двох аспектах: тактова частота, з якою працює процесор, зображена вздовж лівої осі та виміряна м МГц, а також температуру – вздовж правої осі, виміряна в градусах по Цельсію.

Пунктирні лінії відображають тактову частоту, спочатку тактова частота підвищується до свого максимуму. Суцільні лінії показують підвищення температури та падіння тактової частоти. Pi 3 B+ працює більш стабільно, окрім моменту коли частота падає з 1.4 ГГц до 1.2 ГГц, після цього частота залишається приблизно на одному рівні, а саме 1.2 ГГц.

2.4 Висновки до розділу 2

В розділі 2 проаналізована система потокової передачі відео на базі мікроконтролера ESP32-CAM. Із переваг слід відзначити бюджетність такої системи, проте вагомий недолік низька обчислювальна потужність.

Спираючись на технічні характеристики мікроконтролера ESP32-CAM, на 32 бітну архітектуру процесору, та на низьку тактову частоту 240 МГц були підібрані відповідні комплектуючі для подальшої розробки проекту, а саме мікрокомп’ютер Raspberry Pi Model B 3+ та окремий модуль камери Raspberry Pi Camera.

Щодо переваг цієї апаратної платформи, перш за все, достатня обчислювальна потужність для реалізації проєкту, адже даний

мікрокомп’ютер має 64 бітну архітектуру процесору, це означає, що мікропроцесор може обробляти 8 байтів даних за один цикл команди, на відміну від 32 бітних мікропроцесорів, вони обробляють лише 4 байти даних за один цикл команди. Також вагомою перевагою мікрокомп’ютеру Raspberry Pi Model B 3+ є можливість подальшого удосконалення проєкту та додавання нових модулів до нього.

Для роботи даної апаратної платформи необхідне живлення 5 В 3 А.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДИСТАНЦІЙНОГО КЕРУВАННЯ У СИСТЕМАХ ТРАНСПОРТНОЇ ЛОГІСТИКИ

Розроблений програмний комплекс призначений для потокової передачі відео та передачі керуючих команд в режимі реального часу. Модуль камери надсилає відеопотік до апартної платформи, на якій розгорнута клієнтська частина застосунку, після чого відеопотік передається до серверної частини, яка розгорнута на окремому комп’ютері.

Програмний комплекс реалізовано мовою програмування JavaScript. Потокова передача відео реалізована засобами бібліотеки pi-camera-connect.

3.1 Попереднє налаштування Raspberry Pi 3 B+

Перед початком розробки програмного комплексу необхідно встановити операційну систему для мікрокомп’ютера Raspberry Pi та налаштувати. Для встановлення операційної системи необхідно мати карту пам’яті microSD щонайменше Class10. Завантажити операційну систему можна на офіційному сайті [19]. Після завантаження операційної системи потрібно записати її на карту пам’яті, для цього потрібно використати додатковий застосунок, завантажений з офіційного сайту. В якості операційної системи була обрана Raspbian [20].

Raspbian – це безкоштовна операційна система на базі Debian, оптимізована для мікрокомп’ютера Raspberry Pi. Вона надає користувачеві до більш, ніж 10000 бінарних пакетів Debian, які оптимізовані для найкращої сумісності з Raspberry Pi.

Слід зазначити що в останній версії Raspbian сервіс SSH [21] вимкнений за замовченням. Проте, щоб отримати віддалений доступ до мікрокомп’ютера необхідно ввімкнути цей сервіс. Це можна зробити декількома шляхами. Перший – це через графічний інтерфейс мікрокомп’ютера, якщо є можливість підключити монітор, клавіатуру та мишу. Якщо ж такої можливості не має, то

перед запуском операційної системи необхідно до папки boot додати пустий файл з назвою «SSH», до речі так само можна зробити для того щоб ввімкнути Wi-Fi модуль та підключитися до існуючої мережі, проте у файлі має буди відповідна конфігурація. Необхідна конфігурація наведена на рисунку 3.1.

```
country=UA
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="NETWORK-NAME"
    psk="NETWORK-PASSWORD"
}
```

Рисунок 3.1 – Конфігурація Wi-Fi

Після встановлення операційної системи та проведення необхідних налаштувань мікрокомп’ютер готовий до наступних кроків розробки, а саме для підключення віддаленого керування за допомогою застосунку Putty та VNC viewer.

Крім цього, на Raspberry Pi необхідно встановити NodeJS server для подальшої роботи. Для цього слід зайти на офіційний сайт NodeJS [22] та вибрати версію для операційної системи linux.

3.2 Короткий опис бібліотеки pi-camera-connect

Pi-camera-connect [23] – це бібліотека для захоплення та потокової передачі даних камери Raspberry Pi безпосередньо на сервер NodeJS.

Переваги цієї бібліотеки наступні:

- зображення формату JPEG можна зробити за ~33 мс за допомогою вбудованого аналізатора MJPEG
- зображення та відеопотоки передаються безпосередньо в NodeJS як буфер зберігаючи всі дані в пам’яті;
- робота бібліотеки перевірена за допомогою Jest;
- використовуються найсучасніші методи розробки;
- сумісність з мовою програмування TypeScript.

3.2.1 Діаграма класів

Діаграму класів наведено на рисунку 3.2.

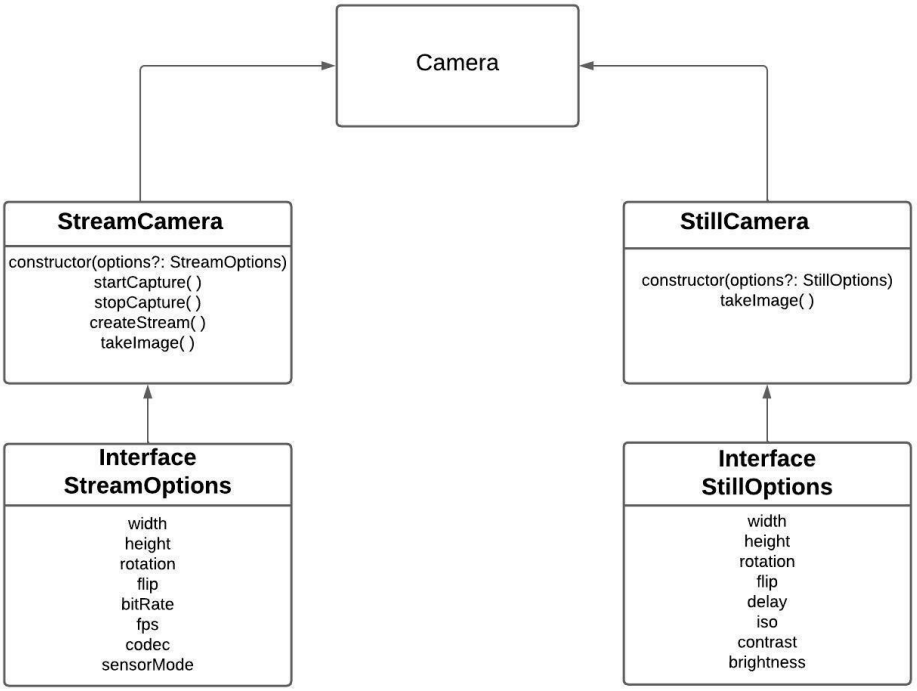

Рисунок 3.2 – Діаграма класів pi-camera-connect

Як видно з даної діаграми, наявні два класи зі своїми методами, та два інтерфейси з налаштуваннями.

3.2.2 Короткий опис методів класу StreamCamera

Клас StreamCamera відповідає за потокову передачу відео та налаштування зображення.

- startCapture( ) – початок захвату відео;
- stopCapture( ) – закінчення захвату відео;
- takeImage( ) – зробити знімок;
- createStream( ) – створення нового відео потоку;
- constructor (options?: StreamOptions) – конструктор який встановлює налаштування перед початком потокової передачі відео, слід зазначити що налаштування опціональні, тобто якщо їх не вказати, то потокова передача відео почнеться з налаштуваннями за замовченням. Також слід зазначити, що

мікрокомп'ютер Raspberry Pi підтримує лише деякі кодеки, а саме MJPEG та H264.

3.3 Розробка серверної частини NodeJS

Серверна частина була розроблена мовою програмування JavaScript та з використанням бібліотеки SocketIO [24] та фреймворку Express [25]. До переваг NodeJS слід віднести:

- висока швидкість роботи застосунків;
- велика кількість безкоштовних інтсрументів та бібліотек;
- багатоплатформність;
- детальна документація.

На серверній частині програмного комплексу відбувається отримання та відображення відеопотоку, який надсилається з клієнтської частини. Крім цього, з вікна відображення відеопотоку можлива відправка команд які певним чином будуть оброблятися на стороні сервера. Код серверної частини та вікна відображення відеопотоку наведений у додатку А та Б відповідно.

3.3.1 Короткий опис серверної частини

Після запуску серверу NodeJS, ініціалізуються дві змінні до яких будуть додаватися підключені пристрої та встановлюється кімната до якої будуть додаватися адреса та сам сокет. На рисунку 3.3 показана ініціалізація цих змінних.

```
let iotDevices = new Map();//<string, SocketIO.Socket>();
let rooms = new Map();//<string, <string, SocketIO.Socket>()>();
```

Рисунок 3.3 – Ініціалізація змінних

Наступний крок – це встановлення з'єднання між серверною частиною та клієнтською. На рисунку 3.4 показано встановлення з'єднання між сервером та клієнтом.

```
io.of("streaming").on("connection", socket => {
    let address = socket.handshake.address;
    console.log("New client connection established...", address);
```

Рисунок 3.4 – Встановлення з'єднання

Після успішного встановлення з'єднання відбувається ініціалізація камери та починається потокова передача відео. На рисунку 3.5 наведений фрагмент коду початку потокової передачі відео.

```
socket.on("pi-video-stream", (data, res) => {
    let roomName = "room" + data;
    socket.to(roomName).emit("consumer-receive-feed", res);
});
```

Рисунок 3.5 – Початок передачі відеопотоку

Також на стороні серверу відбувається отримання та обробка керуючих команд. Керуючі команди наведені на рисунку 3.6.

```
app.post('/straight', (req, res) => {
  res.send('Straight');
  console.log('Straight');
});

app.post('/left', (req, res) => {
  res.send('Left');
  console.log('Left');
})

app.post('/right', (req, res) => {
  res.send('Right')
  console.log('Right');
})

app.post('/back', (req, res) => {
  res.send('Back')
  console.log('Back');
})
```

Рисунок 3.6 – Керуючі команди на стороні серверу

Після виконання команди в консоль передається яка саме команда була виконана, проте в подальшому логіка обробки команд може бути удосконалена.

Доцільно перевірити роботу керуючих команд які відправляються до серверу. На рисунку 3.7 показано виконання керуючих команд.

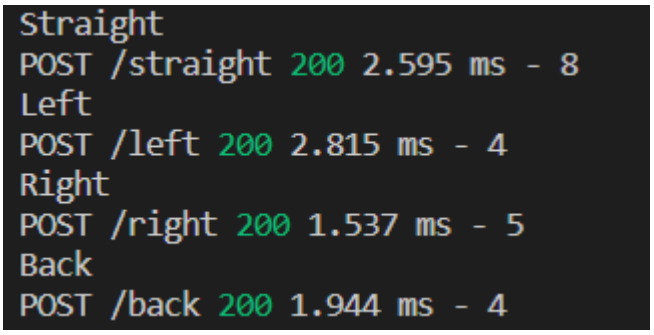

Рисунок 3.7 – Результат виконання керуючих команд

3.3.2 Короткий опис вікна для перегляду потокового відео

На стороні сервера знаходиться вікно для перегляду потокового відео, яке надходить від клієнтської частини. Це файл з розширенням html, в якому знаходиться html розмітка сторінки та javascript код, для реалізації відображення відеопотоку на сторінці. Крім цього, на даній сторінці знаходиться користувацький інтерфейс керуючих команд.

Спочатку відбувається під’єднання до сокету за відповідною адресою, потім створюється пустий буфер, в який записується потокове відео, адже потокова передача відео в бібліотеці pi-camera-connect відбувається за допомогою буферу.

Перегляд відеопотоку можливий лише коли камера під’єднана до сокету, інакше користувач буде бачити пусту сторінку.

3.4 Розробка клієнтської частини

Клієнтська частина також написана мовою програмування JavaScript та з використанням бібліотеки SocketIO. На стороні клієнта відбувається запис та передача відеопотоку на сторону серверу. Код клієнтської частини наведений у додатку В.

3.4.1 Короткий опис клієнтської частини

Після запуску клієнтської частини відбувається налаштування камери, а саме роздільна здатність та кодек, який буде використовуватися. Налаштування камери наведені на рисунку 3.8.

```
const streamCamera = new StreamCamera({
    width: 1280,
    height: 720,
    codec: Codec.MJPEG,
    flip: Flip.Vertical,
    sensorMode: SensorMode.Mode6
});
```

Рисунок 3.8 – Налаштування камери

Наступний крок – це з'єднання з сервером, ініціалізація камери та відправлення буферу з відеопотоком на сторону серверу. На рисунку 3.9 наведений фрагмент коду який реалізує відправку буферу на сторону сервера.

```
socket.on('connect', () => {
    socket.sendBuffer = [];

    socket.emit("pi-cam-init", "Cam-1");

    console.log("Connected to the server!" + socket.id);
})
```

Рисунок 3.9 – Відправка буферу з відеопотоком на сторону сервера

Також присутня перевірка чи користувач під'єднався для перегляду потокового відео чи ні. В кожному із випадків в консолі друкується повідомлення. На рисунку 3.10 показана перевірка присутності користувача.

```
socket.on('new-consumer', (data) => {
    console.log(data + ' has join the stream');
});

socket.on('consumer-left', (data) => {
    console.log(data + ' has left the stream');
});
```

Рисунок 3.10 – Перевірка присутності користувача

Крім цього наявні функції які відповідають за початок та зупинку потокової передачі відео. Ці функції наведені на рисунку 3.11.

```
async function cameraStartCapture() {
    await streamCamera.startCapture();
}

async function cameraStopCapture() {
    await streamCamera.stopCapture();
}
```

Рисунок 3.11 – Функції керування потоковою передачею відео

3.4.2 Варіанти використання готового програмного комплексу

Оператор може під'єднатися до перегляду потокового відео лише після запуску сервера. У випадку переривання роботи сервера або вимкнення камери, програма також достроково завершує свою роботу. Незалежно від того чи приєднався оператор для перегляду відеопотоку чи ні, клієнтська частина продовжуватиме запис, використовуючи камеру, та буде надсилати потокове відео до серверу якщо ж камера буде ввімкнена. В інакшому випадку відеозапис та надсилання потокового відео достроково завершиться.

Діаграму варіантів використання програми наведено на рисунку 3.12.

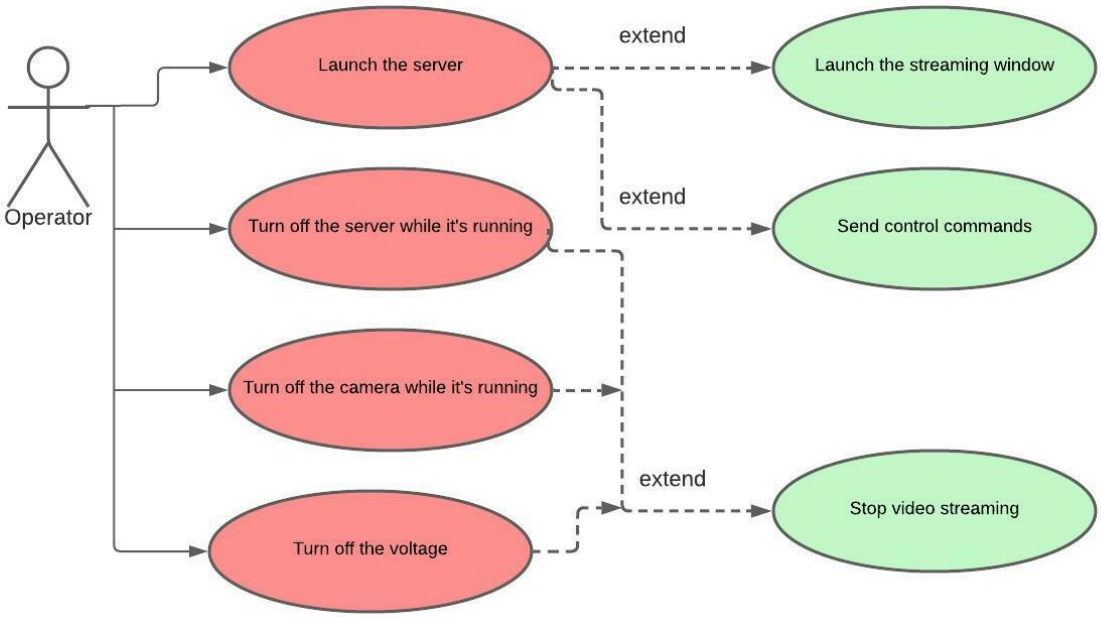

Рисунок 3.12 – Діаграма використання програмного комплексу

Оператору, який використовує програму, необхідно слідкувати, щоб камера була постійно ввімкнена, та на апаратну платформу постійно було подано струм.

3.5 Тестування комплексу

Після розробки програмно-апаратного комплексу необхідно протестувати його роботу, а саме підключення клієнтської частини до серверної, перевірка потокової передачі відео та перевірка виконання керуючих команд.

Після того як запускається сервер відбувається запуск клієнтської частини, в консолі друкується відповідне сповіщення. На рисунку 3.13 показано сповіщення про приєднання клієнтської частини до серверу та початок потокової передачі відео.

```
Connected to the server!/streaming#4Qx4IVG5Spv7Fvq-AAAC
Camera is now capturing
```

Рисунок 3.13 – Сповіщення про початок потокової передачі відео

Тепер оператор може приєднатися до серверу та переглядати відеопотік.

На рисунку 3.14 показано сповіщення про приєднання оператора до сервера.

```
Socket connected:  true        (index):18
New watcher for Cam 1          (index):22
```

Рисунок 3.14 – Сповіщення про приєднання оператора

Після приєднання оператор може переглядати відеопотік та надсилати керуючі команди до серверу. На рисунку 3.15 показаний користувальницький інтерфейс оператора.

Рисунок 3.15 – Користувальницький інтерфейс оператора

В нижньому лівому кутку знаходяться чотири керуючі команди, які відправляються до серверу та обробляються відповідним чином. На рисунку 3.16 показано сповіщення про відправку керуючих команд.

Рисунок 3.16 – Перевірка виконання керуючих команд

3.6 Висновки до розділу 3

В розділі 3 описано розроблення програмного комплексу, який б забезпечував потокову передачу відео та відправлення керуючих команд в режимі реального часу. Програмний комплекс написаний мовою програмування JavaScript та з використанням бібліотеки SocketIO. Для реалізації потокової передачі відео використовуються засоби бібліотеки pi-camera-connect.

Розроблений програмний комплекс забезпечує з'єднання клієнтської частини із серверною, потокову передачу відео та відправлення керуючих команд в режимі реального часу.

Створений програмний комплекс при роботі, а саме потоковій передачі відео має деякі затримки та потребує оптимізації в майбутньому.

ВИСНОВКИ

В ході виконання бакалаврської роботи розроблено програмно-апаратний комплекс дистанційного керування у системах транспортної логістики, що складається з апаратної частини та програмного комплексу. В якості апаратної платформи є мікрокомп’ютер Rasberry Pi 3 B+ з підключеним модулем камери через порт CSI. Апаратна платформа має достатню обчислювальну потужність для коректної роботи програмно-апаратного комплексу. Програмний комплекс забезпечує потокову передачу відео та передачу керуючих команд в режимі реального часу, проте потребує оптимізації для мінімізації затримки при потоковій передачі відео.

Однією з основних переваг розробленого програмно-апаратного комплексу є його вартість. Крім цього, наявність великої кількості безкоштовних бібліотек для розробки та широкий вибір мови програмування.

Основний недолік розробленого комплексу – порівняно низька якість потокового відео 640x480p60 та затримка при потоковій передачі відео, при стабільній швидкості інтернету 150 Mbps затримка в середньому становить 3 секунди, проте якщо інтернет нестабільний та навантаження на центральний процесор комп’ютера на якому розгорнутий сервер досягають 100%, то і затримка значно збільшується і становить в середньому 15 секунд. Покращення якості відео може бути досягнена заміною модуля камери на більш кращий аналог, який здатен вести запис в якості 720p60 або ж 1080p60. В майбутньому провівши оптимізацію коду можна мінімізувати затримки при потоковій передачі відео. Під час подальшого розвитку проекту, планується додати логіку обробки керуючих команд, щоб за допомогою них керувати пристроєм на який буде встановлений програмно-апаратний комплекс дистанційного керування.

Результати роботи можуть бути використані у службах доставки, наприклад Glovo або Uber Eats, також може бути використаний різними гіпермаркетами чи кафе. Можливе удосконалення додатковим апаратним

обладнанням і програмним забезпеченням, може використовуватися при розробці стартапу

В процесі виконання роботи також виконано спеціальну частину з охорони праці. Були проаналізовано вимоги щодо охорони праці на підприємствах та установах, а також роботу при користуванні електронними пристроями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tyler Akidau Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing 2018 200p.

2. Hardware encoder, [Online]. Available: https://docs.microsoft.com/en-us/microsoftteams/hardware-decoders-and-encoders. [Accessed 20 Червень 2022].

3. Real Time Messaging Protocol (RTMP), [Online]. Available: https://rtmp.veriskope.com/docs/spec/ [Accessed 20 Червень 2022].

4. Real Time Streaming Protocol (RTSP), [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2326 [Accessed 20 Червень 2022].

5. Iain Richardson Video Codec Design: Developing Image and Video Compression Systems 2002 328p.

6. MJPEG Documentation, [Online]. Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000063.shtml [Accessed 20 Червень 2022].

7. MJPEG And H.264 Comparison, [Online]. Available: https://ipvm.com/reports/h264-mjpeg-bandwidth-quality-test [Accessed 20 Червень 2022].

8. H.265/HEVC Video Encoder, [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/medfound/h-265---hevc-video-encoder [Accessed 20 Червень 2022].

9. Comparison Of The Coding Efficiency Of Video Coding Standards, [Online]. Available: https://iphome.hhi.de/wiegand/assets/pdfs/2012_12_IEEE-HEVC-Performance.pdf [Accessed 20 Червень 2022].

10. UgCS System Documentation, [Online]. Available: https://www.ugcs.com/files/GPR/GPR_integrated_system_article-FIN.pdf [Accessed 20 Червень 2022].

11. Ilya Grigorik High Performance Browser Networking 2013 398p.

12. ESP32 Series Datasheet, [Online]. Available: https://www.ugcs.com/files/GPR/GPR_integrated_system_article-FIN.pdf [Accessed 20 Червень 2022].

13. Camera Serial Interface, [Online]. Available: https://www.mipi.org/sites/default/files/Camera%20Serial%20Interface_CSI2_CSI3_Overview.pdf [Accessed 20 Червень 2022].

14. Raspberry Pi 3 Model B+ Datasheet, [Online]. Available: https://raspberrypi.in.ua/wp-content/uploads/2018/03/datasheet_raspberry_pi_3_model_b.pdf [Accessed 20 Червень 2022].

15. Raspberry Pi Camera Board v1.3 Datasheet, [Online]. Available: https://uk.pi-supply.com/products/raspberry-pi-camera-board-v1-3-5mp-1080p [Accessed 20 Червень 2022].

16. Raspberry Pi 3B+ 32 bit and 64 bit Benchmarks and Stress Tests, [Online]. Available: https://www.researchgate.net/publication/327467963_Raspberry_Pi_3B_32_bit_and_64_bit_Benchmarks_and_Stress_Tests [Accessed 20 Червень 2022].

17. Whetstone Benchmark, [Online]. Available: https://www.researchgate.net/publication/318761123_Whetstone_Benchmark_Detailed_Later_Results [Accessed 20 Червень 2022].

18. SysBench CPU Benchmark, [Online]. Available: https://imysql.com/wp-content/uploads/2014/10/sysbench-manual.pdf [Accessed 20 Червень 2022].

19. Raspberry Pi Official Site, [Online]. Available: https://www.raspberrypi.com/software/operating-systems/ [Accessed 20 Червень 2022].

20. Raspbian OS, [Online]. Available: http://www.raspbian.org/. [Accessed 18 Червень 2022].

21. SSH Documentation, [Online]. Available: https://man.openbsd.org/ssh. [Accessed 18 Червень 2022].

22. NodeJS Official Site,[Online]. Available: https://nodejs.org/en/. [Accessed 19 Червень 2022].

23. Pi Camera Connect For NodeJS, [Online]. Available: https://www.npmjs.com/package/pi-camera-connect. [Accessed 19 Червень 2022].

24. Tyson Cadenhead Socket.IO Cookbook 2015 184p.

25. Ethan Brown Web Development with Node and Express 2018 325p.

26. Sharon Carmel, "Network media streaming" United States Patent US6389473B1, 24 Березня 1998.

27. Soon-Heung Jung, "RTSP-based progressive streaming method" United States Patent US8214511B2, 24 Квітня 2006.

28. William May, " Real-time or near real-time streaming" United States Patent US11019309B2, 1 Квітня 2010.

29. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин, [Online]. Available: http://zakon3.rada.gov.ua/laws/show/z0382-99. [Accessed 18 Червень 2022].

30. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, [Online]. Available: https://mozdocs.kiev.ua/view.php?id=2445. [Accessed 18 Червень 2022].

ДОДАТОК А
КОД СЕРВЕРНОЇ ЧАСТИНИ

```
module.exports = startupSocketIO = server => {
    const io = require('socket.io') (server);

    let iotDevices = new Map();//<string, SocketIO.Socket>();
    let rooms = new Map();//<string, <string, SocketIO.Socket>()>();

    /**
     * Listening on io.
     */
    io.of("streaming").on("connection", socket => {
        let address = socket.handshake.address;
        console.log("New client connection established...", address);


        // Initialize

        socket.on("pi-cam-init", (data) => {
                console.log("Camera " + data + " is now online!");

                if(!iotDevices.has(data)) {
                // Camera socket will join a room given by the id
                let roomName = "room" + data;
                socket.join(roomName);

                    // Add camera client to a room map for easier maintaining
                    if (rooms.has(roomName)) {
                        rooms.get(roomName).set(socket.id, socket);
                    } else {
                    rooms.set(roomName, new Map().set(socket.id, socket));
                    }

                    // Add camera client to a map for easier maintaining
                    iotDevices.set(data, socket);

                } else if(iotDevices.get(data) !== socket) {
                    console.log("Camera socket different from map, Adding the new socket into the map.");
                    let roomName = "room" + data;
```

```
                    iotDevices.get(data).leave(roomName);
                    socket.join(roomName);
                    iotDevices.set(data, socket);
                }
            });

        //================================================
        // Pi Camera Streaming
        //================================================
        socket.on("pi-video-stream", (data, res) => {
            let roomName = "room" + data;
            socket.to(roomName).emit("consumer-receive-feed", res);
        });

        //================================================
// Pi Camera Disconnect
//================================================
        socket.on("pi-disconnect", (data, res) => {
    console.log("Disconnect (socket) from pi camera " + address);
    let roomName = "room" + data;

    // Check room connected clients and remove the all clients
    if(rooms.has(roomName)) {
      rooms.delete(roomName);
    }

    // Broadcast offline status to all clients
    socket.to(roomName).emit("pi-terminate-broadcast");

    res("Pi camera disconnected from server.")
        });
```

```
    //================================================
    //Join to Start Watching Stream
    //================================================
        socket.on("consumer-start-viewing", (data, res) => {
        console.log("Start stream from client " + address + " on pi camera ", data);
        let roomName = "room" + data;

        let camera;
        if(iotDevices.has(data)) {
            socket.join(roomName);

            // Add web client to a room map for easier maintaining
            if(rooms.has(roomName)) {
                rooms.get(roomName).set(socket.id, socket);
            } else {
                rooms.set(roomName, new Map().set(socket.id, socket));
            }

            camera = iotDevices.get(data);
            camera.emit("new-consumer", socket.id, () => {
                console.log("New Consumer has joined " + data + " stream");
            });

            res("Connect to " + camera.id + " steam");

        } else {
            res("Camera is not online");
        }
        });

});
```

```
    io.sockets.on("error", e => console.log(e));

    return io;
}
```

ДОДАТОК Б

КОД ВІКНА ВІДОБРАЖЕННЯ ПОТОКОВОГО ВІДЕО

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <title>Preview Stream</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.4.0/socket.io.js" crossorigin="anonymous"></script>
  <script>
    const socket = io.connect('http://localhost:3000/streaming');

    socket.on("connect_error", (err) => {
      console.log(`connect_error due to ${err.message}`);
    });

    socket.on('connect', () => {
      console.log('Socket connected: ', socket.connected);
      socket.sendBuffer = [];

      socket.emit('consumer-start-viewing', 'Cam-1', () => {
        console.log("New watcher for Cam 1 ");
      });

      socket.on('consumer-receive-feed', (data, res) => {

        $('#play').attr('src',data);
        $('#log').text(data);
      });
    })

  </script>
</head>
```

```
<body>
<img id="play" style="width: 1280px; height: 700px;">
</br>
<div>
  <button id="straight">Straight</button>
  <button id="left">Left</button>
  <button id="right">Right</button>
  <button id="back">Back</button>
</div>


</body>
<script>
  document.querySelector("#straight").addEventListener("click", async () => {
    let response = await fetch('/straight', {
      method: 'POST'
    });
    console.log('straight');
  })
  document.querySelector("#left").addEventListener("click", async () => {
    let response = await fetch('/left', {
      method: 'POST'
    });
    console.log('left');

  })
  document.querySelector("#right").addEventListener("click", async () => {
    let response = await fetch('/right', {
      method: 'POST'
    });
    console.log('right');

  })
```

```
  document.querySelector("#back").addEventListener("click", async () => {
    let response = await fetch('/back', {
      method: 'POST'
    });
    console.log('back');
  })
</script>
</html>
```

ДОДАТОК В
КОД КЛІЄНТСЬКОЇ ЧАСТИНИ

```
const io = require('socket.io-client');
const { StreamCamera, Codec, Flip, SensorMode } = require('pi-camera-connect');
const socket = io.connect('http://192.168.0.103:3000/streaming');

const streamCamera = new StreamCamera({
    width: 1280,
    height: 720,
    codec: Codec.MJPEG,
    flip: Flip.Vertical,
    sensorMode: SensorMode.Mode6
});


socket.on('connect', () => {
    socket.sendBuffer = [];

    socket.emit("pi-cam-init", "Cam-1");

    console.log("Connected to the server!" + socket.id);
})
```

```
socket.on('new-consumer', (data) => {
    console.log(data + ' has join the stream');
});

socket.on('consumer-left', (data) => {
    console.log(data + ' has left the stream');
});

streamCamera.on('frame', (data) => {
    socket.binary(true).emit('pi-video-stream', 'Cam-1', "" + data.toString("base64"));
});


async function cameraStartCapture() {
    await streamCamera.startCapture();
}

async function cameraStopCapture() {
    await streamCamera.stopCapture();
}

cameraStartCapture().then(() => {
    console.log('Camera is now capturing');
});
```