Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 22028.22.03.11 ПЗ

Виконав здобувач IV курсу, група КІ2-22-3 _____ Артур КАРАСЕВИЧ
Підпис Ініціали, прізвище

Керівник _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ПЕРШИЙ (БАКАЛАВРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Карасевичу Артуру Івановичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI
Керівник проекту (роботи) Лисенко Сергій Миколайович, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз сучасних протоколів мережевого відеонагляду та обґрунтування вибору стандарту ONVIF для створення розумної камери

Проєктування архітектури системи трансляції та обробки відеоданих на базі мікрокомп'ютера Raspberry PI

Програмно-апаратна реалізація та тестування функціональних можливостей ONVIF-сервера з підтримкою Profile S

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Архітектура ПЗ проєкту для систем відеонагляду ONVIF

Апаратне забезпечення проєкту

Структурна схема обробки та маршрутизації медіаданих

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «_10_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – вибір та обґрунтування апаратно-програмних компонентів для розробки розумної камери	01.04.2026	виконано
5	Робота над розділом 3 – проєктування архітектури та програмна реалізація ONVIF-сервера для мережевого відео нагляду	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	23.05.2026	виконано
7	Попередній захист ВКР	25.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач_____ Артур КАРАСЕВИЧ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи_____ Сергій ЛИСЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 22028.22.03.11 ПЗ	Пояснювальна записка	56		
			Графічні матеріали			
2		КвРКІ 22028.22.03.11 Е8	Архітектура ПЗ проєкту	1		
			для системи відеонагляду			
			ONVIF			
3		КвРКІ 22028.22.03.11 Е8	Апаратне забезпечення	1		
			проєкту			
4		КвРКІ 22028.22.03.11 Е8	Структурна схема	1		
			обробки та маршрутизації			
			медіаданих			
5		КвРКІ 22028.22.03.11 Е8	Лістинг коду програми	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 22028.22.03.11 ВП		
Розробив		Карасевич		01.06	Програмно-технічний засіб реалізації Onfvif сервера на базі Raspberry PI Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Лисенко		01.06		У	1	1
						ХНУ, КІ2-22-3		
Н. контр.		Лисенко		01.06				
Затв.		Павлова		01.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI».

Автор роботи: Артур КАРАСЕВИЧ.

Керівник роботи: Сергій ЛИСЕНКО.

Пояснювальна записка: 56 с., 22 рис., 9 табл., 4 дод., 55 джерел.

Графічна частина: 3 креслення.

ВІДЕОНАГЛЯД, МЕРЕЖЕВИЙ ПРОТОКОЛ, МІКРОКОМП'ЮТЕР, РОЗУМНА КАМЕРА, ONVIF, PYTHON, RASPBERRY PI.

Кваліфікаційна робота бакалавра присвячена розробці та дослідженню програмно-технічного засобу реалізації Onvif-сервера на базі мікрокомп'ютера Raspberry PI. Актуальність теми зумовлена стрімким розвитком систем цифрового відеоспостереження та необхідністю забезпечення сумісності між обладнанням різних виробників. Використання міжнародного стандарту ONVIF дозволяє інтегрувати бюджетні та спеціалізовані камери в єдину інфраструктуру безпеки, що підвищує гнучкість та масштабованість сучасних систем моніторингу.

Метою роботи є проєктування, програмна реалізація та тестування серверного рішення, яка забезпечує трансляцію відеопотоку, управління, параметрами камери та метаданими згідно згідно зі специфікаціями ONVIF. Для досягнення поставленої мети було виконано аналіз протоколів передавання медіа даних таких як RTSP і HTTP, обрано оптимальне програмне середовище для Raspberry PI, розроблено архітектуру серверної частини та налаштовано засоби взаємодії з клієнтським програмним забезпеченням для керування відео пристроями в реальному часі.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

Стрімкий розвиток цифрових технологій та глобальна цифровізація призвели до значного зростання попиту на ефективні системи технічної безпеки. В сучасних умовах системи мережевого відеоспостереження інтегруються в критичну інфраструктуру підприємств, "розумні" будинки та міські системи моніторингу. Проте швидке зростання ринку призвело до появи великої кількості обладнання від різних виробників, що часто працює на закритих протоколах, створюючи проблему несумісності технічних засобів.

І тема переддипломної практики а саме: Програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI, стала доволі актуальною і це зумовлено необхідністю створення гнучких, компактних та економічно вигідних рішень для керувань відео потоками в межах мережевої інфраструктури. Використання в проекті міжнародного стандарту ONVIF(Open Network Video Interface Forum) дозволяє забезпечити сумісність IP-камер із різними системами управління, що є критично важливим для систем відеоспостереження.

Використання в завданні мікрокомп’ютера Raspberry PI, відкриває нові можливості для створення компактних, автономних та бюджетних серверних рішень. Реалізація ONVIF на базі цього мікрокомп’ютера Raspberry PI дозволить не лише збирати та транслювати відео потоки, а й гнучко налаштовувати логіку роботи системи під конкретні потреби замовника.

1 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ТА ЗАСОБІВ РЕАЛІЗАЦІЇ

1.1 Розбір предметної області та виявлення наявних проблем і формулювання завдання

На сьогоднішній день технології IP-відеоспостереження та мережевої безпеки є дуже актуальними оскільки цифровізація міст вимагає постійного відео нагляду. А саме відео нагляд є необхідним в багатьох міських та приватних системах до прикладу на дорогах для фіксації порушень правил дорожнього руху, в громадських місцях для фіксації порушення санітарних правил по типу людина кинула сміття на землю замість смітника, на підприємствах для безпеки або нагляду за небезпечними процесами і таких прикладів відео нагляду в наш час є надзвичайно багато. Прикладом міста з добре розвиненою системою відео нагляду може стати Сінгапур в якому надзвичайно велика за масштабами система відео нагляду буквально за всім містом що значно покращило чистоту вулиць міста оскільки якщо людина викине в сміття не в смітник то камери зразу це побачать і людина отримає штраф. Також і за розвиненої системи відео нагляду в Сінгапурі дуже добра безпекова ситуація на вулицях. І дивлячись на приклад Сінгапуру видно що дана система покращила життя міста в багатьох аспектах і тепер є розуміння наскільки є важливою така система[1-10].

Тепер поговоримо про проблеми які можуть виникнути при реалізації таких систем. А саме колись раніше такі системи відео нагляду були аналоговими а зараз цифрові але і в одних і в других була одна проблема а саме що кожен виробник мав свій протокол і це створювало такі проблеми: «зоопарк» обладнання, камера одного бренду не працює з реєстратором іншого бренду без сторонніх «костилів»; закриті екосистеми, навмисне обмеження сумісності пристроїв виробником щоб клієнти купували лише товари їхнього бренду; вартість серверів, повноцінний сервер лише для однієї або двох камер це дорого і енергозатратно.

І постає таке завдання а саме створити універсальний міст-сервер який дозволить кожній камері або потоку доступ до будь-якої системи з єдиним стандартом. Таким містком в завданні стане ONVIF-сервер та міні пк з низьким енергоспоживанням Raspberry Pi. На рисунку 2.1 можна побачити типову схему IP-відеоспостереження[11-17].

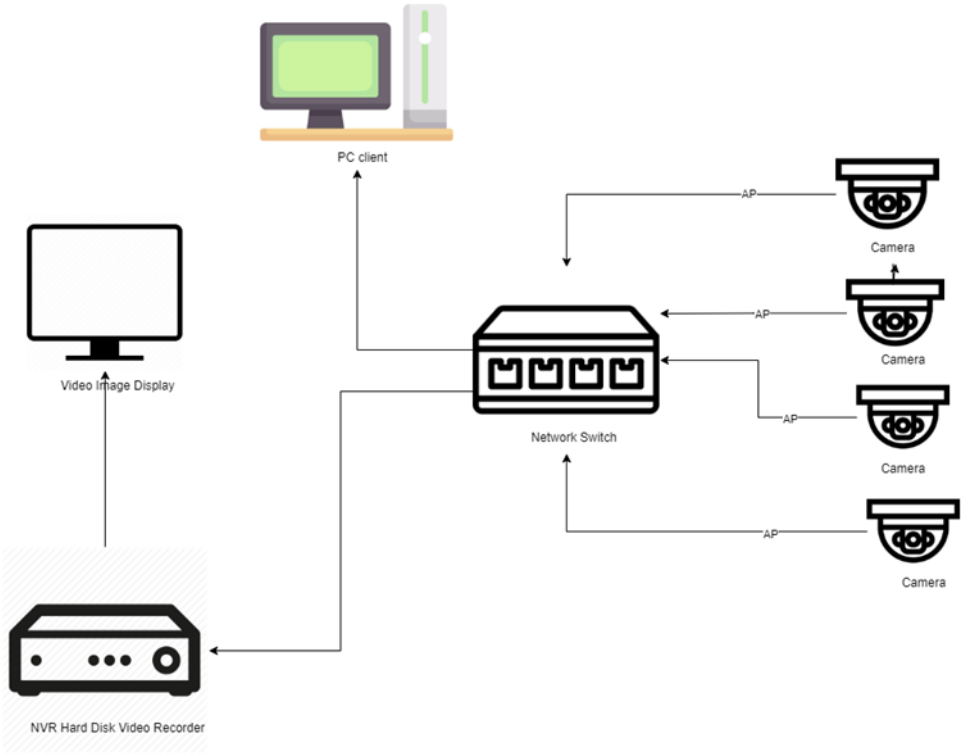


Рисунок 2.1 – Типова схема IP-відеоспостереження[51]

1.2 Порівняльний огляд переваг та недоліків наявних рішень

Існують три основних підходи для створення сервера відеоспостереження. І перший підхід це готові NVR мережеві відео реєстратори які виділяються хорошою надійністю але вони є закритими, тобто в них не можна доставити своє програмне забезпечення що обмежує функціональність. На рисунку 2.1 де була зображена типова схема IP-відеоспостереження в ній якраз і використовується NVR відео реєстратор. Зображення такого пристрою можна побачити нижче на рисунку 2.2 [18-20].

Рисунок 2.2 – Зображення NVR відео реєстратора[52]

Другим підходом у реалізації сервера відеоспостереження це повноцінний сервер на базі персонального комп'ютера з операційними системами Windows або Linux. Переваг в такого рішення як і в NVR теж не багато а точніше з вагомих лише одна а саме це висока потужність такого виду серверів. Недоліків ж у такого сервера є декілька а саме: велика ціна, займає багато місця, велике споживання електроенергії. На рисунку 2.3 показано зображення типового сервера відео спостереження на базі персонального комп'ютера[21-25].

Рисунок 2.3 – Типовий сервер відеоспостереження на базі пк[53]

Третім рішенням для створення сервера є мікрокомп’ютер наприклад Raspberry PI який обрано для завдання. Такий тип реалізації сервера має вже набагато більше плюсів а саме на відміну від двох попередніх варіантів в нього зазвичай низька ціна, в нього дуже низьке енергоспоживання десь приблизно 5-10 Вт, гнучкість тобто в нього відкрите програмне забезпечення і також на відміну від попередніх варіантів він має дуже компактні розміри в порівнянні з NVR і персональним комп’ютером. Зображення мікрокомп’ютера Raspberry Pi 4 model B показано на рисунку 2.4[26-28].

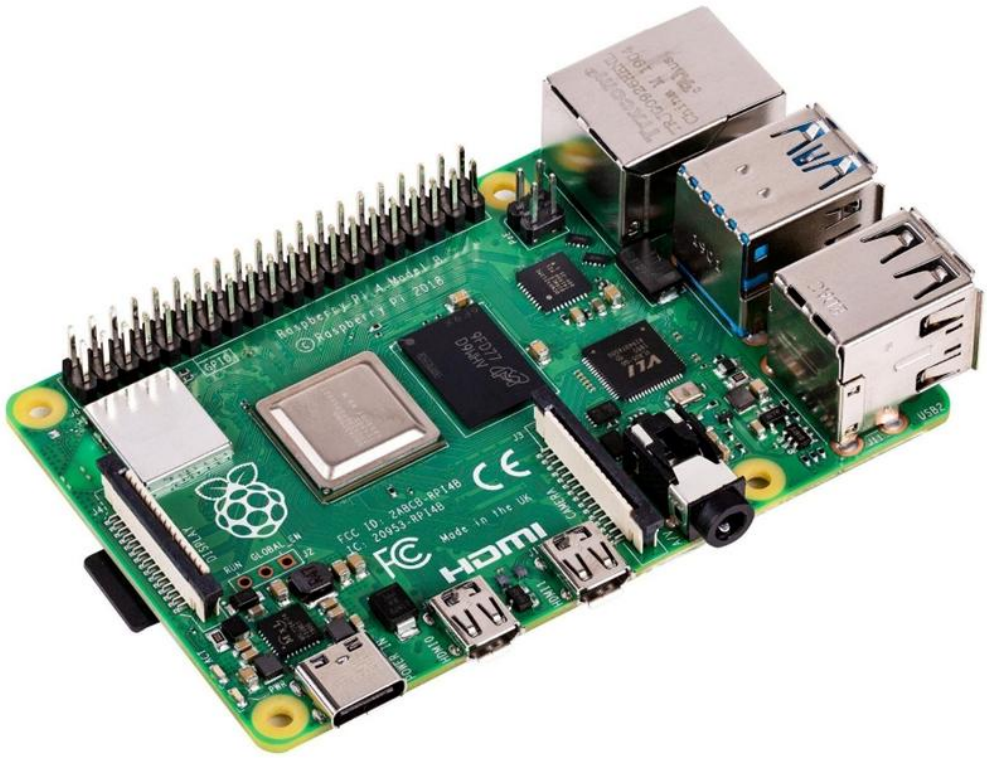


Рисунок 2.4 – Мікрокомп’ютер Raspberry Pi 4 model B[54]

Тепер розглянемо більш детально і наглядно порівняння трьох варіантів реалізації серверів відео нагляду. Розглянемо детально порівняння їх основних характеристик а також їх ціни та гнучкість в плані того наскільки в них відкрите програмне забезпечення. Дане порівняння показано нижче в табличці 2.1[29-32].

Таблиця 1.1 – Порівняння реалізацій серверів відео нагляду.

Назва	Ціна	Енергоспоживання	Гнучкість	Габарити
Готовий NVR	Середня/висока	Середнє(15-30Вт)	Майже відсутня	Великі
Сервер на базі ПК	Дуже висока	Високе(100-300Вт)	Повна	Дуже великі
Raspberry Pi	Низька	Дуже низьке(5-10Вт)	Повна(відкрите ПЗ)	Компактні

Тепер після порівняння цих трьох реалізацій ми бачимо наскільки великі переваги в рішення на базі мікрокомп'ютера Raspberry Pi і саме і за цього дане рішення є актуальним на сьогоднішній день [33-35].

1.3 Аналіз архітектури стандарту ONVIF та мережевих протоколів передачі медіа даних

Тепер розглянемо фундамент реалізації завдання а саме розглянемо два ключових аспекти проекту. І перше з чим потрібно розібратися це сам ONVIF сервер який буде зв'язувати всю систему відео нагляду. Детальніше про сам ONVIF сервер а саме він собою являє набір протоколів які базуються на: XML,SOAP,RTP/RTSP. Завдяки цим протоколам в ONVIF сервері реалізується автоматичне виявлення пристроїв, керування поворотом камери PTZ, отримувати звук та відео. Тепер необхідно розглянути кожний із базових протоколів ONVIF детальніше щоб зрозуміти що кожен з них виконує і так в нас буде розуміння як їх використовувати.

Протокол XML eXtensible Markup Language він є мовою розмітки яка використовується для структурування даних. Його роль полягає в тому щоб описувати всі параметри камери: назву, роздільну здатність відео, налаштування мережі [36-38].

Принцип його роботи такий: коли клієнт запитує інформацію, сервер відправляє XML файл,де тегами наприклад:<Resolution>1920x1080</Resolution> вказує чітко прописані всі дані. І саме це дозволяє пристроям різних брендів розуміти структуру даних один одного.

Протокол SOAP Simple Object Access Protocol слугує для обміну структурованими повідомленнями в мережі. А саме він працює таким чином: використовується в ONVIF для керування пристроєм: надіслати команду повернути камеру, зупинити запис або дати посилання на відео потік. Це працює саме так: клієнт відправляє SOAP-запит який в середині містить XML-код, потім сервер його обробляє і надсилає SOAP-відповідь. І так працює більшість функцій керування в ONVIF саме через SOAP поверх HTTP [39-40].

Далі розглянемо протокол RTSP Real Time Streaming Protocol який являє собою прикладний протокол, призначений для керування доставкою даних у реальному часі що є надзвичайно важливим в системах відеоспостереження. Його роль полягає в тому що він є пультом керування відео потоку, а саме він не передає саме відео, але він каже серверу такі команди: грай, пауза, зупини. Логіка його роботи така: коли натискається Play у програмі для перегляду камер, вона надсилає команду DESCRIBE або Play через RTSP.Тоді в свою чергу сервер дає відповідь що готується віддавати відеодані.

Протокол RTP Real-time Transport Protocol його роль полягає в безпосередній передачі медіа даних кадрів відео від сервера до клієнта. Це працює таким чином: RTP розбиває відео на маленькі пакети, нумерує їх і додає часові мітки timestamps, потім на іншому кінці плеєр міг зібрати їх у правильному порядку і без затримок.

Тепер розглянемо сам алгоритм роботи всіх цих протоколів в нашому проекті: пошук, клієнт шукає наш мікрокомп’ютер Raspberry Pi в мережі; обмін даними SOAP/XML,клієнт питає: які в тебе потоки ?. Raspberry Pi через SOAP в XM відповідає:у мене є потік за адресою;

керування потоком RTSP,клієнт каже: добре, почни передачу play за цією адресую; передача відео RTP, Raspberry Pi починає сипати пакетами відео через RTP ,і ми бачимо картинку на екрані.

Також важливо розглянути таке поняття в ONVIF як PROFILES від яких залежить як саме буде працювати наш програмно-технічний засіб. А саме ці профілі це готові набори інструкцій, які визначають, що саме пристрій вміє робити. І вибір потрібного профіля є необхідною дією тому що стандарт ONVIF настільки великий, що жоден пристрій не може підтримувати все одразу. Тому і за цього розробники розділили функції на групи-профілі. Розглянемо такі основні профілі: profile S для потокового відео він призначений для передачі відео потоку через мережу. В його функції входить: налаштування мережевих параметрів, отримання відео потоку, керування поворотом та нахилом камери PTZ, підтримка аудіо. Саме більшість IP-камер та відео реєстраторів NVR працюють саме за цим профілем ;profile T для сучасного відео,він є вдосконаленою версією Profile S, випущений в 2018 році. Призначений він для роботи з сучасними кодеками та розширеними функціями відео. У його функції входить підтримка форматів стиснення H.264 та H.265, двосторонній аудіо зв’язок, керування детекція руху , виявлення обличь, передача метаданих. Цей профіль забезпечує набагато кращу якість картинки при меншому навантаженні на мережу; profile G для зберігання та пошуку, даний профіль призначений для керування записом на диск та відтворення архіву. Його функціями є налаштування розкладу запису, пошуку відео у сховищі, отримання та відтворення архівних записів. Він є критично важливим для відео реєстраторів NVR, оскільки дозволяє переглядати відео, яке було записане раніше; profile Q для швидкого встановлення, він призначений для спрощення першого налаштування пристроїв які тільки вводяться в експлуатацію. Він виконує функції швидкого виявлення пристроїв в мережі discovery та забезпечує безпечне керування ключами доступу [41].

Далі другим ключовим аспектом в реалізації проекту стає сам мікрокомп’ютер Raspberry Pi а саме його апаратна база і чому саме його обрано для цього проекту. І саме завдяки цим характеристикам він є вдалим рішенням для реалізації ONVIF сервера а саме це те що в нього є апаратне прискорення відео форматів H.264/H/265, також дуже важливо що в нього низьке тепловиділення при роботі і за чого він не потребує додаткових витрат на додаткові системи охолодження що здешевлює проект. І саме головне це те що він може спокійно без зайвого шуму працювати 24 на 7 без шуму що надзвичайно важливо для систем відео спостереження адже зазвичай такі системи повинні працювати 24 на 7 безперебійно.

Також ще одним важливим аспектом реалізації проекту це є програмні підходи використані при розробці. До таких програмних підходів входить використання готових бібліотек наприклад на Python або Node.js. Також потрібно використовувати прошарок Media Server, який «згортає» звичайний потік у формат зрозумілий для ONVIF.

1.4 Постановка задачі до виконання завдання по темі Програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI

Щоб виконати завдання по розробці даного програмно-технічного засобу реалізації Onvif сервера на базі Raspberry PI потрібно чітко сформулювати дії які потрібно виконати в ході виконання цього завдання [42].

І першим що потрібно зробити це обрати оптимальну модель мікрокомп’ютера Raspberry PI яких є доволі багато моделей і вибір саме необхідної моделі є дуже важливим тому що саме мікрокомп’ютер є основою проекта від чого залежить стабільність роботи всієї системи відео спостереження. І для кращого розуміння яку саме модель мікрокомп’ютера Raspberry Pi потрібно детально розглянути декілька моделей схожих між собою для вибору саме тої яка ідеально піддійте нашому проекту.

Першою моделлю мікрокомп’ютера яку детально розглянуто буде Raspberry Pi 4 model B. Ця модель мікрокомп’ютера була випущена в 2019 році і вона є досі популярна в серверних рішеннях. Вона має достатньо потужності для обробки декількох відео потоків в форматі Full HD. Зображення даного мікрокомп’ютера можна побачити на рисунку 2.4 в підрозділі 2.2. Тепер безпосередньо розглянемо детально характеристики Raspberry Pi 4 model B наведені в табличці 2.2 нижче [43].

Таблиця 1.2 – Характеристики мікрокомп’ютера Raspberry Pi 4 model B

Компонент	Специфікація
Процесор	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC
Тактова частота процесора	1.5 ГГц (базова)/1.8 ГГц (для новіших ревізій v1.4+).
Пам’ять (RAM)	1 ГБ, 2 ГБ, 4 ГБ або 8 ГБ LPDDR4-3200 SDRAM (залежно від моделі)
Живлення	5 В постійного струму через роз'єм USB-C (мінімум 3 А*) 5 В постійного струму через роз'єм GPIO (мінімум 3 А*) Живлення через Ethernet (PoE) увімкнено (потрібна окрема точка доступу PoE HAT)
Відео	1 × micro-HDMI (4K), апаратне декодування H.265 (4kp60) 2-смуговий дисплейний порт MIPI DSI, 2-смуговий порт камери MIPI CSI
Мережа	Gigabit Ethernet, Wi-Fi IEEE 802.11ac 2.4/5.0 GHz, Bluetooth 5.0
Порти	2 × USB 3.0, 2 × USB 2.0, 4-полюсний стереоаудіо- та композитний відеопорт, слот для карт microSD для завантаження операційної системи та зберігання даних

Другою моделлю яку детально розглянемо стане Raspberry Pi 5 яка вийшла наприкінці 2023 року і яка являє собою нове покоління продуктивності. Дана модель у 2-3 рази потужніша за попередню Pi4 і це вже дозволяє для ONVIF сервера працювати вже з 4K відео потоками та швидше виконувати аналітику наприклад розпізнавання обличь. Розглянемо детальні характеристики даної моделі мікрокомп'ютера які наведені в табличці 2.3 нижче [44].

Таблиця 1.3 – Характеристики мікрокомп'ютера Raspberry Pi 5

Компонент	Специфікація
Процесор	Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with cryptography extensions, 512KB per-core L2 caches and a 2MB shared L3 cache
Пам'ять (RAM)	LPDDR4X-4267 SDRAM (1 ГБ, 2 ГБ, 4 ГБ, 8 ГБ та 16 ГБ)
Живлення	Живлення 5 В/5 А постійного струму через USB-C з підтримкою Power Delivery.Стандартний 40-контактний роз'єм Raspberry Pi.Годинник реального часу (RTC), що живиться від зовнішньої батареї.Кнопка живлення
Відео	Графічний процесор VideoCore VII з підтримкою OpenGL ES 3.1, Vulkan 1.3.Два виходи для дисплея 4Kp60 HDMI® з підтримкою HDR.Декодер 4Kp60 HEVC
Мережа	Дводіапазонний Wi-Fi® 802.11ac.Bluetooth 5.0 / Bluetooth з низьким енергоспоживанням (BLE). Гігабітний Ethernet з підтримкою PoE+ (потрібен окремий PoE+ HAT). 2 × 4-смугові приймачі-передавачі камери/дисплея MIPI
Порти	Слот для картки microSD з підтримкою високошвидкісного режиму SDR104.2 порти USB 3.0, що підтримують одночасну роботу зі швидкістю 5 Гбіт/с.2 порти USB 2.0. Інтерфейс PCIe 2.0 x1 для швидкої периферії (потрібен окремий M.2 HAT або інший адаптер)

Наступним пунктом в постановці задач для виконання завдання це є обґрунтування вибору та розгортання операційної системи. Тобто тут треба обрати оптимальну операційну систему для мікрокомп'ютера щоб він працював з нею якомога ефективніше та в основних аспектах описати розгортання вибраної операційної системи. Така ОС повинна бути з низьким споживанням ресурсів та мати широку підтримку драйверів для Raspberry Pi. І на цю роль було обрано дві операційні системи які найбільше можуть підійти а саме це Raspberry Pi OS та чистий Debian. Але хоча Raspberry Pi OS базується на Debian все таки вони мають суттєві відмінності в контексті вбудованих рішень і за чого вони потребують детального порівняння щоб таки визначити яка ж все таки з цих двох операційних систем найкраще нам підходить. Детальне порівняння цих двох ОС наведено нижче в таблиці 2.4 [47].

Таблиця 1.4 – Порівняння Raspberry Pi OS та чистого Debian

Критерії порівнювання	Raspberry Pi OS (64-bit)	Debian (AArch64)
Ядро (Kernel)	Оптимізоване спеціально для чіпів Broadcom	Стандартне універсальне ядро Linux
Підтримка GPU	Повна підтримка апаратного прискорення відео	Може потребувати ручного встановлення драйверів
Налаштування	Наявність утиліти raspi-config для GUI/CLI	Тільки ручне редагування конфіг-файлів
Програмна база	Репозиторії Debian + власні репозиторії RPi	Тільки офіційні стабільні репозиторії Debian
Стабільність	Висока (орієнтована на мультимедіа та освіту)	Максимальна (серверний стандарт)

Отже розглянувши детально порівняння цих двох операційних систем і зваживши всі плюси і мінуси вибір падає саме на Raspberry Pi OS 64-bit оскільки має доволі багато готових інструментів для роботи саме з Raspberry Pi.

Тепер потрібно розглянути сам основний алгоритм розгортання операційної системи на Raspberry Pi щоб встановити всі необхідні інструменти і налаштування для реалізації в подальшому ONVIF сервера для системи відео спостереження. Отже алгоритм такий:

1. Підготовка носія інформації,тут потрібно обрати відповідну карту пам’яті яка буде відповідати вимогам які потрібні для розгортання сервера а саме тут буде доцільне використання карт пам’яті таких класів як Class 10 A1 або A2 тому що в них висока швидкість випадкового читання/запису.

2. Першочергове налаштування через термінал,на цьому кроці необхідно виконати після першого завантаження ОС базову конфігурацію відповідних команд таки як: sudo apt update && sudo apt upgrade -y яка відповідає за оновлення системних репозиторіїв та пакетів що є важливим для безпеки; sudo raspi-config ця команда потрібна щоб налаштувати об’єм виділеної пам’яті GPU і її важливо виконати адже це важливий параметр для відео сервера,тут рекомендується виділити 128 або 256 МБ.

3. Встановлення системних залежностей, це крок потребує встановлення необхідних бібліотек для роботи ONVIF а саме таких як: FFmpeg, Python3-pip, libv4l-dev. Встановлення цих інструментів і інших є важливим в нашому завданні адже наприклад якщо взяти FFmpeg то він використовується як основний інструмент для перекодування та трансляції потоків у формат RTSP [49].

Далі згідно постановки задачі потрібно виконати встановлення та налаштувати необхідне програмне забезпечення для ONVIF серверу, а саме тут потрібно встановити і налаштувати такі інструменти як: python-onvif, FFmpeg, MediaMTX. Саме встановлення і налаштування такого роду інструментів допоможе створити стабільне рішення з низькою затримкою передачі даних, яке повністю відповідає вимогам специфікації ONVIF.

Наступним кроком потрібно забезпечити трансляцію відео потоку через протокол RTSP всередині ONVIF-контейнера. А саме після налаштування програмного середовища основним завданням стає організація безпосередньої трансляції. В ONVIF це означає, що сервер повинен не просто віддавати відео, а загорнути посилання на цей потік у відповідну XML-структуру, яку зрозуміє клієнт.

Саме завдяки використанню RTSP як внутрішнього транспорту всередині ONVIF-структури, досягається універсальність а саме відео з Raspberry Pi можна буде переглянути як у професійних програмах перегляду таких як: Milestone, Hikcentral, так і у звичайному VLC-плеєрі.

І останнім кроком в постановці задачі є тестування доступу до сервера через стандартні клієнти такі як ONVIF Device Manager. На цьому завершальному етапі реалізації проекту потрібно провести верифікацію створеного ONVIF-сервера. А саме метою даного тестування є перевірка того, чи розпізнається Raspberry Pi стороннім програмним забезпеченням як повноцінна IP-камера.

Отже якщо підсумувати всі кроки у постановці задачі для виконання поставленого завдання отримаємо такий основний алгоритм дій:

1. Обрати оптимальну модель мікрокомп’ютера Raspberry Pi.
2. Обрати ти розгорнути операційну систему.
3. Встановити та налаштувати програмне забезпечення для ONVIF.
4. Забезпечити трансляцію відео потоку через протокол RTSP. всередині ONVIF-контейнера.
5. Протестувати доступ до сервера через стандартні клієнти.

Виконавши цих п’ять кроків із алгоритму дій вийде реалізувати повноцінний програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI[50].

1.5 Висновки по першого розділу

В межах першого розділу кваліфікаційної роботи було проведено комплексне теоретичне дослідження методів та засобів реалізації мережевого сервера відеоспостереження на базі мікрокомп'ютера Raspberry PI.За результатами проведеного аналізу предметної області та порівняння існуючих рішень отримаємо такі ряд ключових підсумків по кожному пункту першого розділу.

Було обґрунтовано актуальність розробки а саме, сучасний стан розвитку систем цифрової безпеки до відкритих стандартів вказує на критичну необхідність переходу від закритих пропрієтарних систем до відкритих стандартів таких як ONVIF.Аналіз досвіду розгортання масштабних систем моніторингу на прикладі Сінгапуру показав, що ефективність відеоспостереження прямо залежить від можливості інтеграції різних технічних засобів у єдину інформаційну мережу. Було виявлено, що основною перешкодою для масштабування таких систем є проблема несумісності обладнання різних виробників, що робить розробку універсального ONVIF-сервера вкрай затребуваним завданням.

Наступним було здійснено порівняльний аналіз апаратних рішень. У ході дослідження було розглянуто три основні підходи до побудови серверної інфраструктури: готові NVR-реєстратори, сервери на базі персональних комп'ютерів та мікрокомп'ютерні системи. Встановлено, що використання мікрокомп'ютера Raspberry Pi є найбільш раціональним з точки зору енергоефективності а саме споживання 5-10 Вт проти 100-300 Вт у ПК, габаритних розмірів та вартості реалізації. Це дозволяє створювати автономні розумні камери, здатні виконувати роль повноцінного мережевого вузла без використання дорогого серверного обладнання.

Також досліджено стекову структуру стандарту ONVIF. А саме детальний розбір протоколів XML, SOAP, RTSP та RTP показав, що стандарт ONVIF

забезпечує повний цикл взаємодії між пристроями: від опису характеристик камери в XML-структурах до безпосередньої трансляції медіаданих через RTP-пакети. Визначено, що для реалізації поставленого завдання найбільш доцільним є впровадження Profile S, оскільки він охоплює критично важливий функціонал а саме налаштування мережевих параметрів та передачу потокового відео в реальному часі.

Було також визначено оптимальну апаратну та програмну платформу для розробки цього проекту. Порівняння характеристик Raspberry Pi 4 та Raspberry Pi 5 продемонструвало, що нова архітектура на базі процесора Broadcom BCM2712 забезпечує необхідну продуктивність для обробки відеопотоків високої чіткості 4K та майбутнього впровадження алгоритмів відеоаналітики. Для програмної реалізації обрано 64-бітну операційну систему Raspberry Pi OS, яка поєднує стабільність ядра Debian з глибокою оптимізацією під апаратні прискорювачі відео форматів H.264/H.265.

І останнім було проведено чітку постановку задачі. А саме на основі теоретичного аналізу було розроблено покроковий алгоритм дій, що включає підготовку носія інформації, розгортання системного середовища, встановлення медіа-сервера MediaMTX та інструментів кодування FFmpeg, а також кінцеву верифікацію через стандартні клієнти Happytime Onvif Client. Це закладає фундамент для наступних етапів роботи-проєктування архітектури ПЗ та практичної реалізації серверного рішення.

Отже таким чином, проведений аналіз підтвердив технічну можливість та економічну доцільність створення програмно-технічного засобу реалізації Onvif-сервера на базі мікрокомп’ютера Raspberry PI.

2 РОЗРОБКА АРХІТЕКТУРИ ТА ТЕХНІЧНИХ РІШЕНЬ РОЗУМНОЇ КАМЕРИ НА БАЗІ СТАНДАРТУ ONVIF

2.1 Визначення апаратних і програмних підсистем програмно-технічного засобу

Для реалізації функціонального ONVIF-сервера на базі мікрокомп’ютера Raspberry pi 5 необхідно чітко розмежувати зони відповідальності між апаратною і програмною частиною проєкту. Систему можна розділити на три головні рівні а саме, на апаратний вузол, системне середовище та прикладні сервіси.

Отже розпочнемо з апаратної підсистеми а саме як вже говорилось основою пристрою є одноплатний мікрокомп’ютер Raspberry PI 5 який виступає в ролі центрального обчислювального модуля.

І перше що потрібно розглянути це центральний обчислювальний модуль який базується на 64 бітному чотири ядерному процесорі Broadcom BCM2712 з архітектурою Arm Cortex-A76, що працює на тактовій частоті 2,4 ГГц. Дане обчислювальне ядро відповідає за виконання системних викликів операційної системи, керування розподілом оперативної пам'яті між сервісами трансляції та забезпечення стабільної роботи мережевого стека при високих навантаженнях. Наявність вбудованого криптографічного розширення та оптимізованих L2 і L3 кешів дозволяє мінімізувати затримки при обробці SOAP-запитів ONVIF-сервера.

Далі з одних найважливіших модулів в архітектурі пристрою це є модуль захоплення зображення представлений цифровою камерою, яка інтегрується з обчислювальним модулем через спеціалізований високошвидкісний послідовний інтерфейс MIPI CSI Camera Serial Interface.

Використання двох 4 смугових приймачів-передавачів MIPI дозволяє передавати сирі відеодані безпосередньо до графічного процесора VideoCore VII,

що суттєво розвантажує основний процесор та знижує загальне енергоспоживання пристрою.

І мережевий інтерфейс системи забезпечує двосторонній зв'язок між розумною камерою та клієнтським програмним забезпеченням наприклад, Happytime ONVIF Client. Для забезпечення максимальної пропускної здатності та стабільності відеопотоку у форматі Full HD/4K використовується контролер Gigabit Ethernet. Водночас, наявність вбудованого дводіапазонного модуля Wi-Fi 802.11ac дозволяє розгортати систему у місцях з відсутньою провідною інфраструктурою, підтримуючи при цьому необхідну швидкість передачі даних для RTSP-трансляції.

Отже комплексна взаємодія цих компонентів дозволяє сформувати завершений апаратний стек, здатний функціонувати у цілодобовому режимі, що є критичною вимогою для сучасних систем відонагляду. На рисунку 2.1 можна побачити структурну схему апаратних інтерфейсів розумної камери на базі Raspberry PI 5.

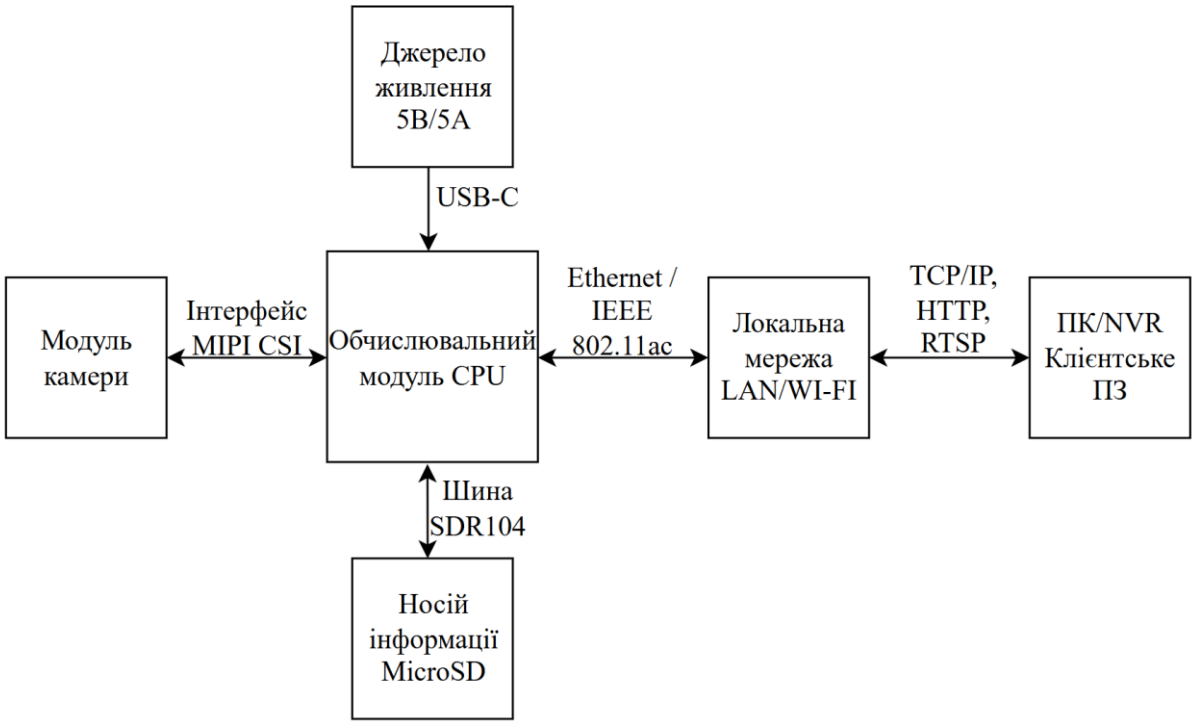


Рисунок 2.1 – Структунрна схема апаратних компонентів розумної камери

Як показано на рис. 2.1, архітектура апаратного забезпечення побудована за радіально-вузловим принципом, де центральним елементом є обчислювальний модуль CPU мікрокомп'ютера. Використання інтерфейсу MIPI CSI для підключення модуля камери дозволяє забезпечити пряму передачу відеоданих до графічного процесора, що мінімізує затримки. Мережева взаємодія реалізується через гігабітний контролер Ethernet, що є критично важливим для стабільної трансляції потоків з високим бітрейтом. Живлення системи через інтерфейс USB-C з підтримкою Power Delivery гарантує стабільну роботу процесора Cortex-A76 навіть при пікових навантаженнях під час апаратного кодування відео.

Тепер перейдемо до розгляду програмної підсистеми а саме, програмне забезпечення розроблюваного засобу реалізовано за модульним принципом, що забезпечує високу відмовостійкість та можливість гнучкого налаштування окремих компонентів. Логічну структуру ПЗ можна розділити на декілька взаємозалежних рівнів, кожен з яких виконує специфічні функції згідно зі специфікацією ONVIF Profile S.

Рівень операційної системи та драйверів базується на 64-бітній версії Raspberry Pi OS на ядрі Debian. Основним завданням цього рівня є забезпечення низькорівневого доступу до апаратних прискорювачів відео та керування мережевими інтерфейсами. Для взаємодії з камерою використовуються драйвери стека libcamera, які дозволяють отримувати відеопотік з мінімальною затримкою.

Підсистема медіа-сервісів реалізована за допомогою двох потужних інструментів таких як: FFmpeg який виступає в ролі мультимедійного «двигуна», який здійснює захоплення відеопотоку з пристрою /dev/video0, виконує його апаратне кодування у формат H.264 використовуючи кодек h264_v4l2m2m та інкапсулює дані для передачі, і другий інструмент це MediaMTX який виконує роль високопродуктивного RTSP-сервера.

Він створює точку доступу endpoint, до якої підключаються зовнішні клієнти для отримання живого відео.

Керуюча підсистема ONVIF Server Logic побудована на мові програмування Python з використанням мікрофреймворку Flask. Цей модуль є інтелектуальним центром розумної камери. Він відповідає за: слухання вхідних HTTP-з'єднань на порту 8080, за парсинг вхідних XML-повідомлень, оформлених за стандартом SOAP і за генерацію динамічних відповідей, що містять метадані пристрою, параметри відеопрофілів та URI-посилання на медіа-потік.

Така архітектура дозволяє відокремити важкі процеси обробки відео FFmpeg від логіки управління Python, що забезпечує стабільну роботу системи навіть при одночасному підключенні декількох користувачів. В табличці 2.1 можна побачити в короткому форматі технічні характеристики кожного з вище перерахованих програмних модулів і за що кожен відповідає.

Далі потрібно розглянути взаємодію цих підсистем бо ефективність функціонування розроблюваної розумної камери залежить не лише від технічних характеристик окремих компонентів, а й від оптимізованої взаємодії між апаратною та програмною підсистемами.

Цей процес інтеграції System Integration забезпечує безперебійний та синхронізований рух інформації від фізичного сенсора до екрана кінцевого користувача.

Для забезпечення високої надійності архітектура взаємодії розділена на дві незалежні магістралі: площину даних Data Plane та площину управління Control Plane.

Площина даних для обробки та маршрутизації відеопотоку. Фізичний світловий сигнал, захоплений модулем камери, перетворюється на цифровий масив даних і передається через апаратний інтерфейс MIPI CSI до ядра операційної системи Raspberry Pi OS, де він ідентифікується як системний пристрій наприклад, /dev/video0. На цьому етапі до роботи підключається

мультимедійний фреймворк FFmpeg. Він здійснює безперервне захоплення сирого відеопотоку і, використовуючи апаратні прискорювачі процесора BCM2712, виконує його компресію за стандартом H.264.

Стиснений медіапотік локально перенаправляється на внутрішній медіа-сервер MediaMTX. Останній створює активну RTSP-сесію, буферизує кадри та робить їх доступними для трансляції у локальну мережу через стандартний порт 8554.

Таблиця – 2.1Технічні характеристики програмних модулів

Назва модуля	Версія і тип	Основна функція у системі	Використовуваний порт
MediaMTX	v1.9.0 Binary	Публікація та ретрансляція відеопотоку	8554 RTSP
FFmpeg	v6.1 Library	Апаратне кодування відео h264_v4l2m2m	-
Python Flask	v3.0 Framework	Обробка запитів GetCapabilities, GetProfiles	8080 HTTP/SOAP
Python-onvif	v0.1.3 Library	Генерація XML-структур згідно зі стандартом	-

Площина управління для логіки взаємодії за протоколом ONVIF. Паралельно з трансляцією відеоданих функціонує керуюча підсистема, реалізована за допомогою скриптів на мові Python фреймворком Flask. Цей програмний модуль працює як незалежний фоновий процес демон і прослуховує мережевий порт 8080 на наявність вхідних HTTP з'єднань. Коли клієнтське

програмне забезпечення Happytime ONVIF Client ініціює підключення, воно надсилає запити у форматі XML/SOAP наприклад, GetCapabilities або GetProfiles.

Головним завданням Python сервера на цьому етапі є синтаксичний аналіз запиту та динамічне формування валідної XML-відповіді. Зокрема, при отриманні запиту GetStreamUri, керуюча підсистема повинна передати клієнту точне мережеве посилання URI на RTSP потік, який генерується сервером MediaMTX.

Розподіл цих процесів на рівні операційної системи дозволяє уникнути блокування системних ресурсів. Завдяки багатозадачності Raspberry Pi OS, навіть інтенсивний обмін керуючими SOAP повідомленнями не впливає на стабільність, затримку та частоту кадрів основного відеопотоку, що повністю відповідає вимогам до професійних систем мережевого відеонагляду.

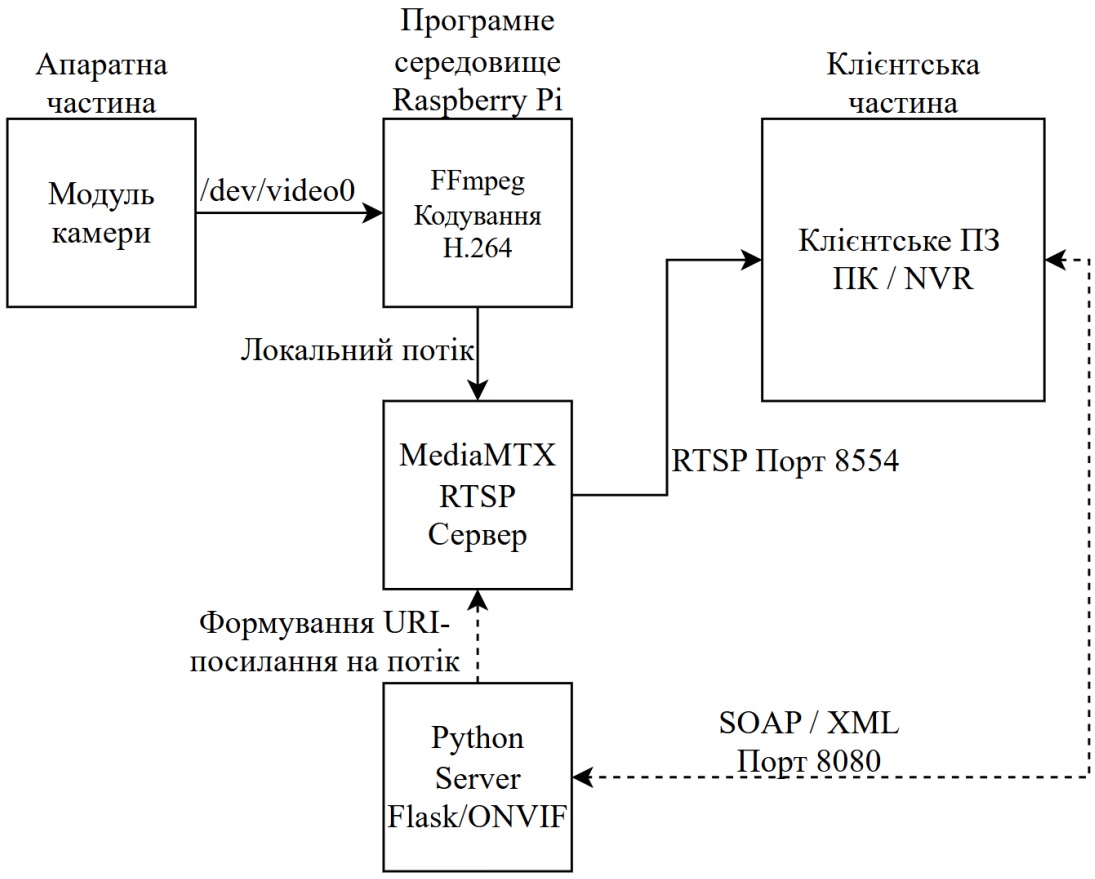


Рисунок 2.2 – Діаграма потоків даних підсистеми розумної камери

2.2 Проєктування структурної схеми трансляції відеоданих

Процес формування та передачі медіаданих є найбільш ресурсоємною задачею розроблюваної системи. Для забезпечення трансляції відео високої роздільної здатності у реальному часі необхідно спроєктувати оптимальний конвеєр обробки даних Video Pipeline, який мінімізує затримки latency на кожному етапі шляху від фізичної матриці камери до мережевого порту мікрокомп'ютера.

Архітектура конвеєра трансляції будується на послідовній передачі даних між апаратними та програмними буферами. Цей процес можна розділити на три ключові стадії. І першою стадією є захоплення та первинна обробка Capture Stage а саме це світловий потік, що потрапляє на оптичну систему модуля камери, фокусується на CMOS-матриці, де відбувається фотоелектричне перетворення. Згенерований цифровий сигнал передається через шлейф MIPI CSI-2 безпосередньо до підсистеми ISP Image Signal Processor мікрокомп'ютера Raspberry Pi 5. На цьому етапі відбувається апаратна корекція кольору, балансу білого та придушення шумів, після чого сформовані «сирі» кадри поміщаються у віртуальний пристрій /dev/video0.

Далі другою стадією йде апаратне стиснення Encoding Stage і оскільки передача нестисненого відео вимагає надвисокої пропускної здатності понад 1 Гбіт/с для формату 1080p, критичним етапом є компресія даних. Для цього використовується утиліта FFmpeg, яка звертається до апаратного кодека h264_v4l2m2m. Використання саме апаратного прискорювача VideoCore VII замість програмного стиснення силами центрального процесора дозволяє знижити навантаження на CPU з 80-90% до 10-15%, що є ключовим фактором для стабільної роботи системи охолодження.

І останьою третьою стадією є інкапсуляція та мережева трансляція Streaming Stage а саме стиснений у формат H.264 потік не може бути переданий у мережу напряму, він потребує пакування у транспортні контейнери. FFmpeg

передає стиснені кадри до локального медіа-сервера MediaMTX. Останній виконує роль мультиплексора: він упаковує кадри у пакети протоколу RTP (Real-time Transport Protocol), додає часові мітки для синхронізації та керує сесіями підключення користувачів через протокол RTSP.

Таким чином, спроєктована структурна схема гарантує, що важкі операції з відео виконуються на рівні спеціалізованих апаратних блоків мікрокомп'ютера, залишаючи ресурси центрального процесора вільними для обробки мережевих протоколів управління.

На рисунку 2.3 можна побачити як саме влаштований конвеєр обробки та трансляції відеоданих.

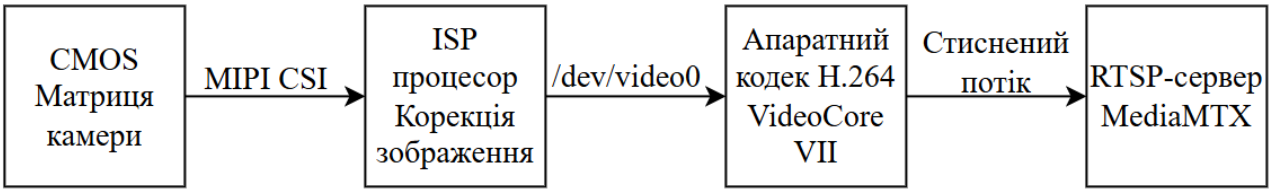


Рисунок 2.3 – Конвеєр обробки та трансляції відеоданих

2.3 Обґрунтування вибору параметрів кодування та розрахунок мережевого навантаження

Проєктування системи трансляції відеоданих вимагає знаходження оптимального балансу між якістю зображення, навантаженням на апаратну частину мікрокомп'ютера та пропускною здатністю локальної мережі.

Для забезпечення стабільної роботи ONVIF-сервера необхідно обґрунтувати вибір роздільної здатності, частоти кадрів FPS та формату стиснення.

В якості основного стандарту компресії для розроблюваної системи обрано кодек H.264 Advanced Video Coding. Хоча сучасний H.265 забезпечує кращий ступінь стиснення, саме H.264 є базовим стандартом для специфікації ONVIF Profile S. Його використання гарантує 100% сумісність розумної камери з будь-

якими апаратними NVR-реєстраторами та клієнтським програмним забезпеченням без необхідності додаткового транскодування на стороні клієнта.

Модуль камери, підключений до Raspberry Pi 5, здатен захоплювати відео у форматі до 4K. Проте, для задач базового відеомоніторингу трансляція 4K-потоку є надлишковою і призводить до невиправданого забивання мережевого каналу.

Оптимальним вибором для даної системи є формат Full HD 1920x1080 пікселів при частоті 30 кадрів на секунду. Це забезпечує достатню деталізацію для розпізнавання об'єктів та плавності рухів, зберігаючи при цьому помірне навантаження на мережу.

Для передачі відео у мережу використовується змінний бітрейт VBR, який адаптується до динаміки в кадрі, але обмежується верхньою межею для запобігання перевантаженню роутера. Для формату 1080p і 30fps з кодеком H.264 рекомендований цільовий бітрейт становить близько 4 Мбіт/с.

Для обґрунтування необхідності використання апаратного стиснення розрахуємо обсяг даних нестисненого відеопотоку Raw Video для формату 1080p. Розрахунок виконується за наступною формулою:

$$R = W \cdot H \cdot F \cdot c \cdot d, \tag{2.1}$$

де R – бітрейт нестисненого відеопотоку біт/с;

W – ширина кадру у пікселях 1920;

H – висота кадру у пікселях 1080;

F – частота кадрів $30 кадрів/с;

c – кількість кольорових каналів для моделі RGB c = 3

d – глибина кольору на один канал стандартно 8 біт.

Підставивши значення, отримаємо: $R = 1920 \cdot 1080 \cdot 30 \cdot 3 \cdot 8 \approx 1.49$.

Як видно з розрахунку, передача нестисненого відео вимагає пропускної здатності майже 1.5 Гбіт/с, що перевищує можливості більшості локальних мереж і є абсолютно неприйнятним. Саме тому застосовується кодек H.264. Коефіцієнт компресії K_c визначається як відношення нестисненого бітрейту до цільового стисненого $B_c = 4$ Мбіт/с:

$$K_c = \frac{R}{B_C} = \frac{1492.99}{4} \approx 373, \quad (2.2)$$

де K_c – коефіцієнт компресії відеоданих;

R – бітрейт нестисненого відеопотоку, Мбіт/с;

B_C – цільовий бітрейт стисненого відеопотоку Мбіт/с.

Крім того, при розрахунку навантаження на мережу необхідно враховувати мережеві заголовки overhead протоколів RTP, UDP та IP. Реальна пропускна здатність мережі B_{net} розраховується з урахуванням коефіцієнта запасу k = 1.2 :

$$B_{net} = B_c \cdot k = 4 \cdot 1.2 = 4.8, \quad (2.3)$$

де B_{net} – реальна необхідна пропускна здатність локальної мережі;

B_c – цільовий бітрейт стисненого відеопотоку, Мбіт/с;

k – коефіцієнт запасу, що враховує мережеві накладні витрати (overhead) транспортних протоколів RTP, UDP та IP.

Дані виконанні розрахунки дозволяють зробити порівняльну таблицю вимог до мережі щоб наглядно побачити різницю.

З наведених розрахунків видно, що обраний формат 1080p генерує близько 1.8 ГБ трафіку на годину. Гігабітний мережевий інтерфейс Raspberry Pi 5, а також модуль Wi-Fi 802.11ac мають пропускну здатність, яка багаторазово перевищує ці вимоги. Це дозволяє в майбутньому масштабувати систему, запускаючи на

одному мікрокомп'ютері декілька паралельних RTSP-потоків без ризику мережевих затримок Drop Frames.

Таблиця 2.2 – Залежність мережевого трафіку від параметрів відеопотоку

Роздільна здатність	Частота кадрів	Цільовий бітрейт Мбіт/с	Необхідна пропускна здатність LAN	Обсяг даних за 1 годину ГБ
720p 1280x720	25	2.0	5 Мбіт/с	0.90
1080p 1920x1080	30	4.0	10 Мбіт/с	1.80
2K 2560x1440	30	6.0	15 Мбіт/с	2.70
4K 3840x2160	30	12.0	30 Мбіт/с	5.40

2.4 Проєктування підсистем енергоспоживання та теплообміну

Використання обчислювальної платформи Raspberry Pi 5 для задач обробки та трансляції відео в реальному часі висуває підвищені вимоги до системи стабілізації живлення та відведення тепла.

Оскільки процес кодування відео за стандартом H.264 є ресурсомістким, він призводить до значного навантаження на центральний процесор CPU та графічне ядро GPU, що, в свою чергу, збільшує споживання струму та генерацію теплової енергії.

Зробимо розрахунок енергетичного бюджету системи. Raspberry Pi 5 базується на архітектурі, що потребує живлення через інтерфейс USB-C за протоколом Power Delivery.

Для забезпечення стабільної роботи всіх компонентів системи, включаючи зовнішню USB-камеру та периферійні пристрої, необхідне джерело живлення з параметрами 5В / 5А і 25 Вт.

Загальна потужність, що споживається системою, розраховується за формулою 2.4:

$$P_{total} = P_{RPI} + P_{cam} + P_{usb} + P_{loss}, \quad (2.4)$$

де P_{RPI} – потужність, що споживається самою платою Raspberry Pi 5 при максимальному навантаженні;

P_{cam} – потужність, необхідна для роботи зовнішньої USB-камери;

P_{usb} – потужність, що споживається флеш-накопичувачем та іншою периферією;

P_{loss} – коефіцієнт втрат на перетворення та опір провідників приймається як 10-15%

Для аналізу режимів роботи системи складено таблицю 2.3 енергоспоживання при різних рівнях навантаження.

Таблиця 2.3 – Показники енергоспоживання Raspberry Pi 5

Режим роботи	Струм Ампер	Потужність Ват
Режим очікування Idle	0.6-0.8	3.0-4.0
Трансляція відео FFmpeg H.264	1.8-2.4	9.0-12.0
Максимальне навантаження Stress + Camera	3.2-3.8	16.0-19.0

Перейдемо до обґрунтування системи охолодження. Критичним фактором для Raspberry Pi 5 є температурний режим процесора Broadcom BCM2712. При досягненні температури 80°C система активує механізм тротлінгу тобто зниження тактової частоти, що призведе до затримок у відеопотоці drop frames та втрати стабільності ONVIF-сервера.

В даному проєкті передбачено використання активної системи охолодження Active Cooler, що складається з алюмінієвого радіатора та вентилятора з програмним керуванням PWM.

На рисунку 2.4 зображена схема розміщення елементів охолодження на Raspberry Pi 5

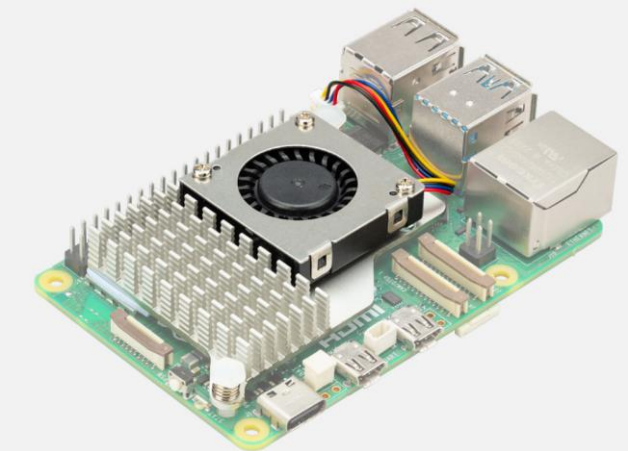


Рисунок 2.4 – Зображена схема розміщення елементів охолодження[55]

Алгоритм керування теплообміном базується на зміні швидкості обертання вентилятора залежно від показників температурних датчиків SoC.

На рисунку 2.5 можна побачити графік по якому видно залежності швидкості вентилятора від температури процесора.

Застосування ОС Raspberry Pi Lite без графічного інтерфейсу дозволяє знизити базову температуру системи на 5-8°C, звільняючи ресурси процесора виключно для роботи підсистеми відеоспостереження.

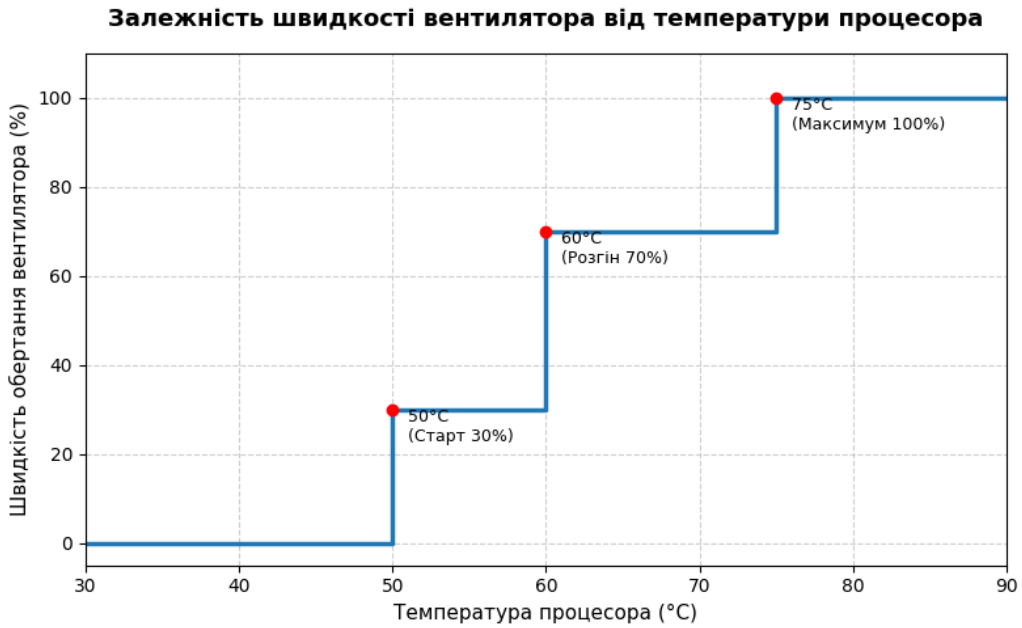


Рисунок 2.5 – Графік залежності швидкості вентилятора від температури процесора

2.5 Теоретичне проєктування системи мережевої безпеки

У сучасних системах відеоспостереження, побудованих на базі архітектури інтернету речей IoT, питання кібербезпеки є критичним. Відкриті трансляції та незахищені канали керування наприклад, стандартні реалізації RTSP та ONVIF є вразливими до перехоплення трафіку Man-in-the-Middle, підбору паролів Brute-force та несанкціонованого доступу до відеопотоку.

Для розробленого апаратно-програмного комплексу на базі Raspberry Pi 5 спроєктовано комплексну трирівневу модель безпеки, яка включає захист транспортного рівня відеопотоку, криптографічний захист каналу керування та фільтрацію мережевих пакетів на рівні операційної системи.

Розпочнемо з захисту танспортного рівня відеопотоку RTSP Authentication. Протокол транспортування медіаданих у реальному часі RTSP за замовчуванням передає інформацію у відкритому вигляді. Для захисту програмного медіасервера від несанкціонованих підключень у проєкті реалізовано механізм

автентифікації на основі дайджесту Digest Authentication, процедура якого регламентована міжнародним стандартом RFC 7616.

На відміну від базового методу автентифікації, який передає облікові дані у простому текстовому кодуванні що легко перехоплюється засобами аналізу трафіку, дайджест-автентифікація виключає передачу пароля мережею у відкритому вигляді. Замість цього сервер та клієнт обмінюються криптографічними хешами.

Процес встановлення захищеного з'єднання відбувається у кілька послідовних етапів. Спочатку клієнтська програма ініціює запит до камери на отримання параметрів потоку без передачі жодних авторизаційних даних. У відповідь на це серверна частина відхиляє запит із помилкою доступу та одразу генерує для клієнта унікальний одноразовий числовий рядок, який виконує функцію маркера сесії. Отримавши цей маркер, клієнт на своєму боці обчислює криптографічну хеш функцію, використовуючи власний секретний пароль, ідентифікатор користувача логін та наданий сервером маркер. На завершальному етапі клієнт повторює початковий запит, обов'язково додаючи до нього обчислений криптографічний хеш для остаточної перевірки та валідації на стороні сервера.

На рисунку 2.6 зображена така схема обміну повідомленнями при дайджест автентифікації.

Налаштування політик доступу здійснюється через головний конфігураційний файл медіасервера, де закріплюється низка критичних параметрів.

Зокрема, у системі жорстко прописуються облікові дані адміністратора ідентифікатор користувача та пароль для надання прав публікації відеопотоку від джерела до сервера.

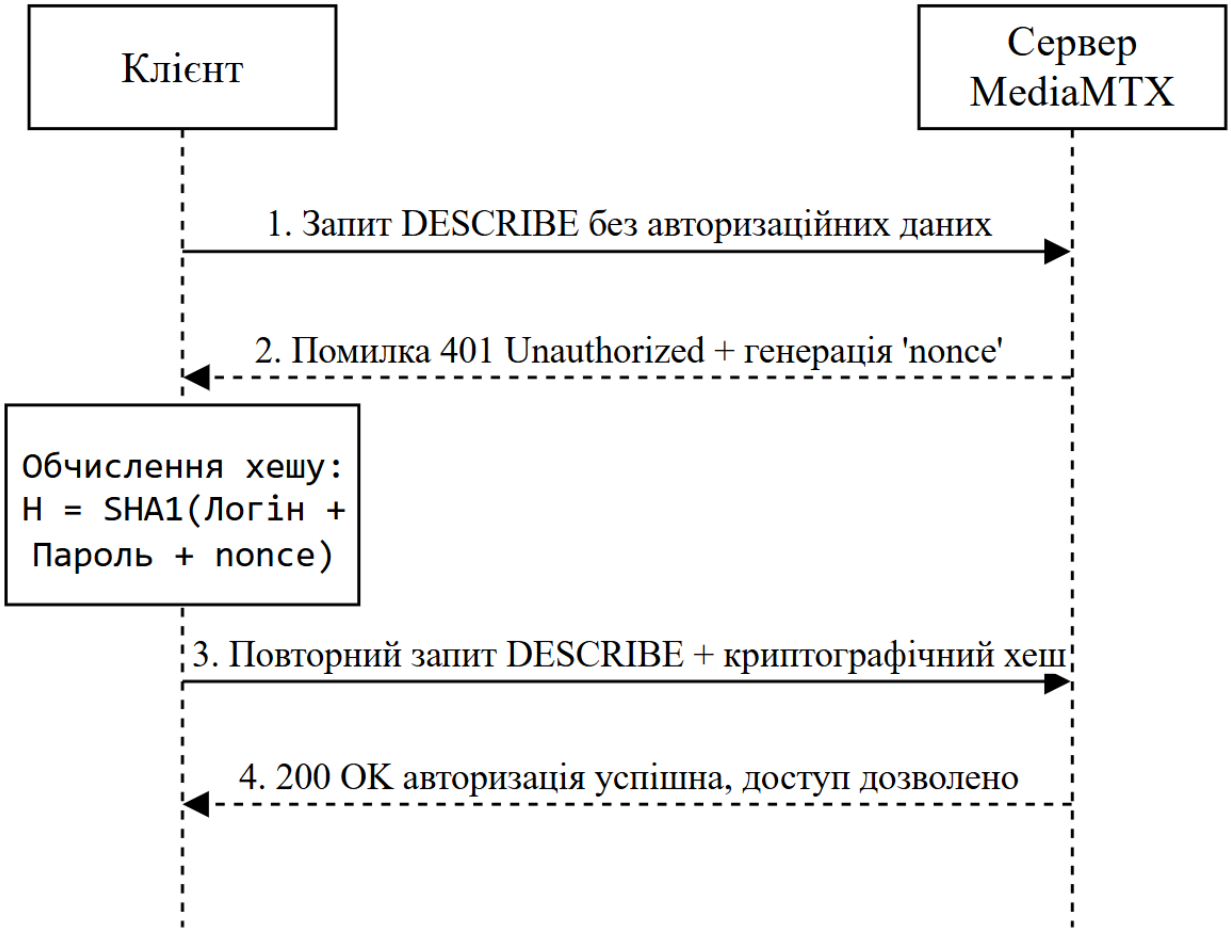


Рисунок 2.6 – Схема обміну повідомленнями при дайджест автентифікації

Окрім цього, регламентуються окремі облікові дані для прав читання Read rights, що на програмному рівні унеможливлює анонімне підключення будь-яких сторонніх глядачів. Також конфігуруються параметри доступу за IP-адресами, які в межах даної архітектури налаштовані на прийом підключень з будь-якого вузла мережі, але виключно за умови успішної криптографічної перевірки пароля клієнта.

Далі розглянемо криптографічний захист керуючого каналу ONVIF а саме WS-Security. Взаємодія між клієнтським терміналом та розробленим сервером керування відбувається за допомогою протоколу SOAP, структура якого базується на форматі XML. Для забезпечення конфіденційності, цілісності та автентичності команд керування камерою впроваджено специфікацію безпеки вебсервісів WS-Security. Згідно з профілем токенів користувача UsernameToken,

кожен запит керування повинен містити блок криптографічних даних у своєму заголовку.

Основою механізму безпеки є обчислення криптографічного дайджесту пароля, що математично описується наступним виразом:

$$H_{\text{пароля}} = Base64(SHA1(N_{\text{вип}} + T_{\text{ств}} + P_{\text{корист}})), \quad (2.5)$$

де $H_{\text{пароля}}$ – підсумковий криптографічний хеш дайджест пароля, що передається мережею;

$N_{\text{вип}}$ – криптографічне одноразове випадкове число, згенероване клієнтом для запобігання атакам повторного відтворення Replay attacks;

$T_{\text{ств}}$ – мітка часу створення запиту у форматі єдиного координаційного часу UTC, яка визначає термін валідності повідомлення;

$P_{\text{корист}}$ – секретний пароль користувача у відкритому вигляді відомий лише клієнту та серверу;

SHA1 та Base64 – стандартні алгоритми криптографічного хешування та текстового кодування бінарних даних відповідно.

Наявність часової мітки $T_{\text{ств}}$ вимагає жорсткої синхронізації системного часу між клієнтським пристроєм та сервером на базі Raspberry Pi 5 за протоколом точного часу NTP. У разі розбіжності часу понад встановлений ліміт наприклад, через спробу перехоплення та затримки пакетів, сервер автоматично відхиляє запит як скомпрометований або застарілий. Структуру захищеного XML повідомлення стандарту ONVIF можна побачити на рисунку 2.7.

Тепер приступимо до проєктування політик мережевого екрану. Третім, глобальним рівнем захисту є ізоляція мережевих портів безпосередньо на рівні ядра операційної системи. За замовчуванням більшість UNIX-подібних систем залишають відкритими порти активних служб, що створює потенційні вектори для кібератак. Для мінімізації поверхні атаки на розробленому пристрої

застосовано програмний мережевий екран, що керує підсистемою пакетної фільтрації. Спроєктована політика безпеки базується на класичному принципі заборонити все за замовчуванням Default Deny, згідно з яким відкриваються лише комунікаційні порти, критично необхідні для функціонування підсистеми відеоспостереження та віддаленого технічного обслуговування.

Процес ініціалізації та конфігурації мережевого екрану являє собою послідовність суворих налаштувань.

На початковому етапі виконується повне скидання всіх існуючих правил маршрутизації до заводських параметрів задля уникнення програмних конфліктів.

Далі встановлюється базова політика абсолютної заборони для всіх вхідних з'єднань, що фактично робить пристрій невидимим для несанкціонованого сканування мережі ззовні.

Після встановлення цієї глобальної заборони, система переходить до етапу точкового відкриття лише тих шлюзів, які забезпечують її цільове призначення. Зокрема, створюються винятки для протоколу TCP на порті 8080, який відповідає за обмін керуючими командами між клієнтом та ONVIF-сервером, а також для порту 8554, через який здійснюється безперервна трансляція медіаданих за протоколом RTSP.

Окремою політикою налаштовується обмежений доступ до порту 22 SSH для віддаленого технічного обслуговування пристрою адміністратором. Для цього порту додатково застосовується механізм захисту від атак типу Brute-force, який автоматично блокує IP-адресу зловмисника після кількох невдалих спроб авторизації підряд.

Також у правилах екрану вмикається системне логування відхилених пакетів, що дозволяє аналізувати спроби несанкціонованого доступу та вчасно виявляти потенційні кіберзагрози у мережі.

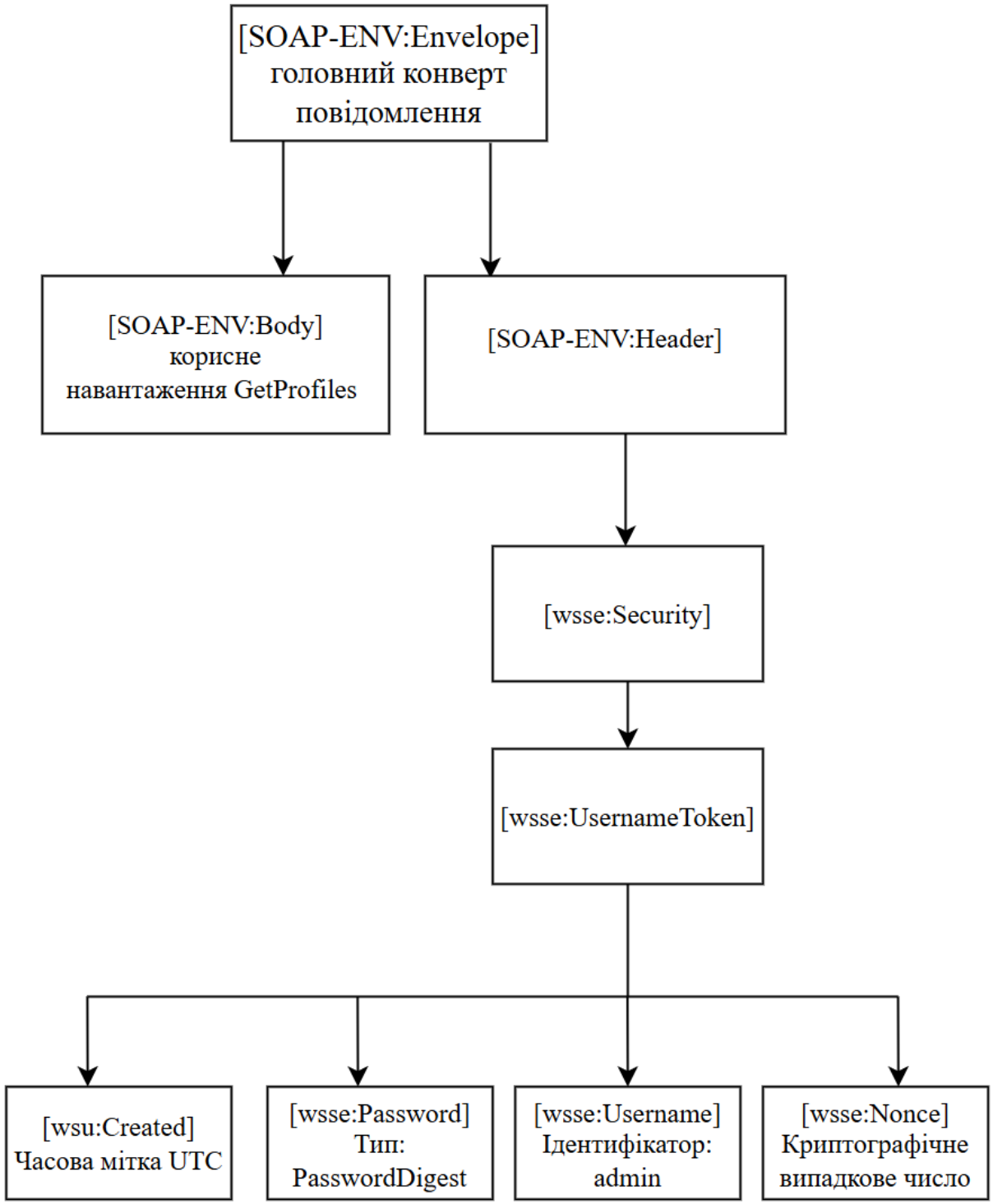


Рисунок 2.7 – Структура захищеного XML-повідомлення стандарту ONVIF

Після цього створюється спеціальне правило з обмеженням частоти підключень для порту віддаленого адміністрування стандартний порт 22, завдяки чому система здатна автоматично блокувати IP-адресу зловмисника після фіксації шести невдалих спроб авторизації протягом короткого проміжку часу.

Фінальним кроком є точкове відкриття портів керування пристроєм порт 8080 та трансляції медіаданих порт 8554, що гарантує стабільну роботу клієнтського програмного забезпечення.

Зведена структура спроєктованих правил фільтрації трафіку наведена у таблиці 2.4.

Таблиця 2.4 – Правила маршрутизації мережевого екрану

Мережевий порт та протокол	Цільова системна служба	Дія екрану	Обґрунтування застосування правила
Всі порти 1-65535	Будь-яка служба	Заборонити	Забезпечення базової політики абсолютної ізоляції вузла
8080/TCP	Сервер керування Python	Дозволити	Обробка SOAP-запитів стандарту ONVIF від клієнта
8554/TCP, UDP	Програмний медіасервер	Дозволити	Безперебійна трансляція RTSP відеопотоку
22/TCP	Служба віддаленого доступу	Обмежити	Захищений віддалений доступ для адміністратора із захистом від перебору паролів

2.6 Висновки до другого розділу

У другому розділі кваліфікаційної роботи було проведено комплексне проєктування апаратно-програмної архітектури вузла відеоспостереження, орієнтованого на використання в екосистемі інтернету речей IoT. Базовою обчислювальною платформою обрано мікрокомп'ютер Raspberry Pi 5, апаратна продуктивність якого дозволяє ефективно обробляти та транслювати відеопотік високої роздільної здатності в режимі реального часу.

Під час виконання роботи обґрунтовано вибір програмного стека для реалізації ключових функцій системи. Транспортування медіаданих спроєктовано на базі інтеграції утиліти FFmpeg, яка відповідає за захоплення та кодування відео у формат H.264, та спеціалізованого медіасервера MediaMTX для стабільної маршрутизації RTSP-потоку. Для забезпечення апаратної сумісності з сучасними клієнтськими системами керування відео VMS розроблено архітектуру програмного сервера мовою Python. Даний сервер реалізує базові специфікації міжнародного протоколу ONVIF Profile S шляхом обміну стандартизованими XML-повідомленнями.

Також у межах проєктування було детально проаналізовано мережеве навантаження та оптимізовано параметри трансляції. Математично обґрунтовано вибір роздільної здатності Full HD 1080p при частоті 30 кадрів на секунду. Проведені розрахунки коефіцієнтів компресії довели, що апаратне стиснення H.264 дозволяє зменшити обсяг переданих даних до цільового бітрейту у 4 Мбіт/с. Це гарантує плавну передачу зображення без критичного навантаження на локальну мережу LAN/Wi-Fi та усуває ризик виникнення ефекту недостатньої пропускної здатності при передачі відео високої чіткості.

Окрему увагу в процесі проєктування було приділено апаратному забезпеченню стабільності системи. На основі розрахунку енергетичного бюджету пристрою доведено необхідність використання джерела живлення потужністю 27 Вт із підтримкою протоколу Power Delivery. Зважаючи на

критичне теплове навантаження на центральний процесор під час безперервного кодування медіаданих, обґрунтовано інтеграцію активної системи охолодження з автоматизованим ШІМ-керуванням. Це рішення гарантує уникнення температурного тротлінгу та запобігає втраті кадрів під час трансляції.

З метою нівелювання вразливостей IoT-пристроїв розроблено трирівневу модель мережевої кібербезпеки. Модель включає ізоляцію некритичних портів за допомогою пакетного фільтра UFW політика Default Deny, захист керуючого каналу ONVIF із застосуванням криптографічної специфікації WS-Security, а також впровадження механізму дайджест-автентифікації Digest Authentication для обмеження доступу до транспортного відеопотоку. Такий комплексний підхід до безпеки повністю унеможливлює несанкціонований доступ до конфіденційної інформації, захищає систему від атак типу Man-in-the-Middle та перешкоджає використанню камери як елемента ботнет-мереж.

Важливою перевагою розробленої архітектури є її модульність та гнучкість. Завдяки чіткому розподілу процесів між апаратним рівнем, медіасервером та керуючим Python-скриптом, система здатна легко масштабуватися. Це створює надійне підґрунтя для майбутнього розширення функціоналу, наприклад, для інтеграції алгоритмів комп'ютерного зору, розпізнавання об'єктів чи підключення додаткових сенсорів без кардинальної зміни базового програмного коду.

Запропоновані у розділі проєктні та архітектурні рішення формують надійну, продуктивну та захищену базу пристрою відеоспостереження. Розроблена теоретична модель повністю підготовлена до етапу практичної реалізації та апаратного тестування, що розглядається у наступному розділі роботи.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ ONVIF-СЕРВЕРА НА БАЗІ RASPBERRY PI 5

3.1 Розгортання операційного середовища та конфігурація апаратних інтерфейсів

Першим етапом практичної реалізації проєкту є підготовка базового програмного середовища мікрокомп'ютера Raspberry Pi 5. Для запису образу системи на карту пам'яті формату microSD було використано офіційну утиліту розгортання Raspberry Pi Imager. Як базову операційну систему обрано дистрибутив Raspberry Pi OS Lite архітектури 32-bit. Вибір саме 32-бітної версії операційної системи без графічного інтерфейсу Lite є технічно обґрунтованим рішенням. Відсутність віконного менеджера та використання 32-бітної адресації пам'яті дозволяє суттєво знизити базове споживання оперативної пам'яті та ресурсів центрального процесора. Завдяки цьому вивільняється максимальна кількість обчислювальних потужностей для забезпечення безперебійної роботи ресурсомістких процесів захоплення та транскодування відеопотоку.На рисунку 3.1 показано процес запису завантажувальних файлів для вставнолення ОС на microSD карту.

Оскільки спроєктований вузол відеоспостереження функціонує як автономний серверний пристрій, його початкове налаштування здійснювалося у так званому безголовому режимі headless mode, що виключає необхідність підключення зовнішнього монітора та периферійних пристроїв введення.

Для забезпечення можливості віддаленого адміністрування ще на етапі запису образу було попередньо сконфігуровано параметри локальної бездротової мережі та активовано системну службу протоколу безпечного доступу SSH Secure Shell.

Після першого завантаження апаратної платформи та успішної мережевої авторизації через термінальний клієнт, проводилася базова системна конфігурація за допомогою вбудованої системної утиліти керування raspi-config.

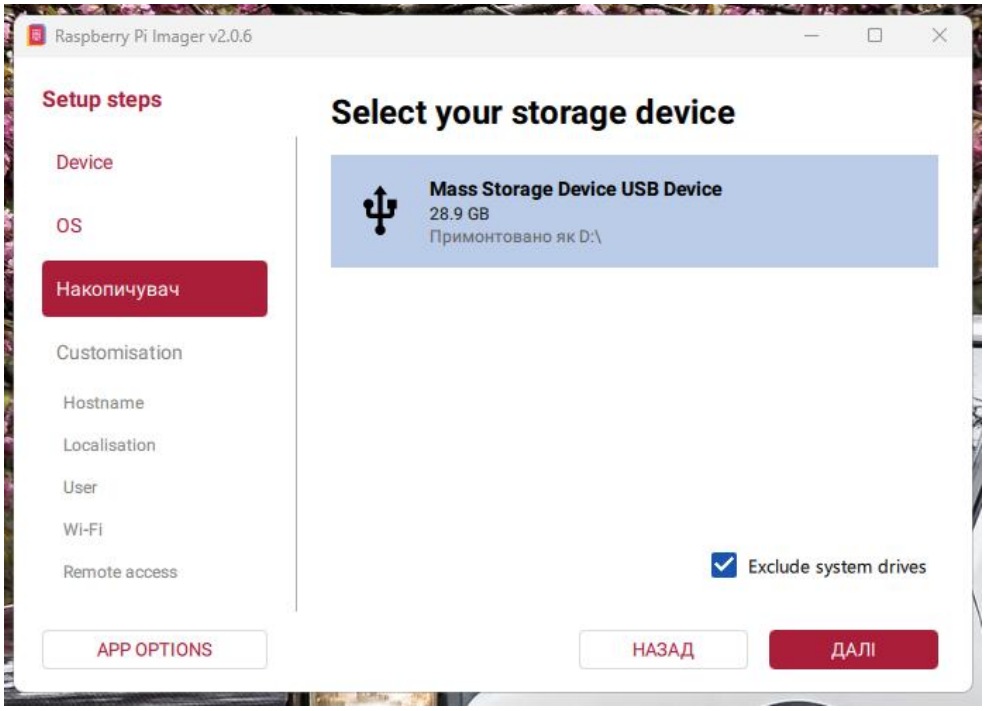


Рисунок 3.1 – Процес розгортання операційної системи запис в Raspberry Pi Imager

На цьому етапі було виконано розширення розділу файлової системи на весь доступний обсяг фізичного накопичувача, що необхідно для коректного збереження системних журналів та тимчасових файлів медіасервера.

Важливим кроком стало точне налаштування часового поясу та синхронізації системного годинника через інтернет, оскільки точний час є критично необхідним параметром для валідації часових міток при генерації криптографічних відповідей протоколу WS-Security.

Наступним кроком конфігурації стала фізична інтеграція сенсорного обладнання, а саме підключення веб-камери A4tech HD 720P PC Camera до порту USB мікрокомп'ютера.

Процес фізичного підключення веб-камери до мікрокомп’ютера в порт USB показаний нижче на рисунку 2.2.

Для верифікації успішної апаратної ініціалізації підключеного пристрою ядром Linux використовувалася системна команда lsusb, яка здійснює опитування шини та виводить ідентифікатори підключеного обладнання.

Рисунок 3.2 – Апаратна інтеграція веб-камери з мікрокомп’ютером

Додатково, для підтвердження того, що система коректно завантажила необхідні драйвери та розпізнала пристрій саме як джерело відеосигналу стандарту Video4Linux, здійснювалася перевірка наявності символьного файлу відео буфера за системним шляхом /dev/video0.

Успішна реєстрація цього файлу у системному каталозі пристроїв підтвердила повну готовність апаратної частини комплексу до захоплення медіаданих та подальшої передачі їх до програмного модуля кодування.

3.2 Розробка та програмна логіка ONVIF-серверу на мові Python

На початку перед тим як розбирати логіку роботи ONVIF серверу краще за все це побудувати блок-схему в якій буде винесена основна логіка роботи

серверу його основні елементи щоб надалі розуміти про що йде мова. Таку блок-схему зображено на рисунку 3.3 в додатку А.

Для програмної реалізації сервера керування, що відповідає базовим специфікаціям протоколу ONVIF Profile S, було обрано об'єктно орієнтовану мову програмування Python у поєднанні з мікрофреймворком Flask. Цей вибір обґрунтовується високою швидкістю розгортання HTTP сервера, відмінною підтримкою мережевих протоколів та мінімальним споживанням системних ресурсів мікрокомп'ютера.

Основою взаємодії між вузлом відеоспостереження та клієнтським програмним забезпеченням є обмін повідомленнями за протоколом SOAP, що базуються на форматі XML. Для забезпечення повної сумісності та правильної інтерпретації команд клієнтом, на початку програмного коду визначаються обов'язкові простори імен XML Namespaces. Вони ініціалізуються у строковій змінній:

```
NS = ('xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"...')
```

Дана зміна містить посилання на офіційні схеми консорціуму W3C та стандарти ONVIF. Ці простори імен динамічно підставляються у кожен XML-шаблон відповіді, що формується сервером.

Наступним кроком архітектурної реалізації є створення єдиної точки входу Endpoint для всіх керуючих команд. За допомогою вбудованого декоратора фреймворку створюється маршрут:

```
@app.route('/onvif/device_service', methods=['POST']),
```

який жорстко обмежений на прийом виключно POST запитів. Саме на цю мережеву адресу порт 8080 клієнтське програмне забезпечення відправляє свої інструкції.

Після отримання мережевого пакета, серверна функція device_service() декодує тіло запиту у текстовий формат utf-8 та розпочинає процес ідентифікації команди. Зважаючи на вимоги до швидкодії та відносну простоту реалізованих

функцій, логіка маршрутизації побудована не на складному парсингу XML-дерева, а на швидкому лексичному пошуку ключових ідентифікаторів методів безпосередньо у тексті запиту.

Наприклад, умовна конструкція:

```
if 'GetStreamUri' in xml_data
```

здійснює перевірку наявності запиту на отримання мережевого посилання для перегляду відео.

У разі позитивного збігу, сервер негайно формує відповідь, використовуючи заздалегідь підготовлений XML-шаблон, до якого інтегрована IP адреса Raspberry Pi та порт медіасервера 8554. Критично важливою вимогою HTTP транспорту для SOAP повідомлень є правильне зазначення типу контенту. Тому кожна згенерована відповідь обгортається у спеціальний об'єкт повернення даних із явно вказаним параметром:

```
return Response(STREAM_URI_XML,mimetype='application/soap+xml').
```

Для забезпечення безвідмовності роботи сервера передбачено обробку нетипових ситуацій. Якщо сервер отримує невідому команду або непідтримуваний метод ONVIF, спрацьовує базовий механізм захисту, який повертає клієнту валідний, але порожній SOAP конверт. Це архітектурне рішення дозволяє програмному забезпеченню клієнта коректно обробити виняток на своєму боці без критичного розриву TCP з'єднання з камерою. Фінальний запуск циклу очікування запитів ініціюється командою:

```
app.run(host='0.0.0.0', port=8080),
```

що наказує серверу приймати підключення з будь-яких зовнішніх IP адрес у межах локальної мережі.

3.3 Системна інтеграція медіасервера MediaMTX та інструментарію FFmpeg

MediaMTX і FFmpeg це дві різні програми, які спілкуються між собою в середині мікрокомп’ютера Raspberry PI, і для кращого розуміння їх роботи необхідно розгялнути їх схему потоку даних. Ця схема покаже, як відео йде від апаратної частини програмно-технічного засобу до мережі. Дана схема зображена на рисунку 3.4 в додатку В.

Як видно з наведеної схеми, початковим етапом обробки є безпосереднє захоплення відеосигналу з апаратного інтерфейсу камери. Цю функцію бере на себе інструментарій FFmpeg, який звертається до підсистеми драйверів ядра операційної системи для отримання сирих або частково стиснених медіаданих. Отримавши потік, FFmpeg виконує його транскодування у цільовий формат, найчастіше H.264, активно використовуючи апаратні блоки прискорення мікрокомп'ютера. Такий підхід дозволяє суттєво знизити навантаження на центральний процесор Raspberry Pi та уникнути затримок при підготовці відео високої роздільної здатності для подальшої трансляції.

Після завершення процесу кодування, FFmpeg виступає у ролі локального публікатора Publisher.

Для реалізації транспортного рівня відеоспостереження, який відповідає за безперебійну доставку медіаданих від камери до клієнтського програмного забезпечення, спроєктовано зв'язку з двох незалежних програмних компонентів: утиліти обробки відео FFmpeg та маршрутизатора потоків MediaMTX. Таке розділення обов'язків дозволяє оптимізувати навантаження на систему, оскільки кожен компонент виконує вузькоспеціалізовану задачу.

Першим етапом інтеграції стало розгортання та конфігурування сервера MediaMTX. Даний сервер розповсюджується у вигляді скомпільованого бінарного файлу, що не вимагає встановлення додаткових залежностей у системі. Управління поведінкою сервера здійснюється через головний конфігураційний

файл mediamtx.yml. У межах розробленого проєкту ключовим налаштуванням є декларація шляху трансляції paths: live:, який визначає кінцеву точку доступу Endpoint для підключення клієнтів. За замовчуванням сервер налаштовано на прийом вхідних потоків та їх подальшу ретрансляцію за протоколом RTSP на стандартному мережевому порту 8554.

Захоплення нестисненого відеосигналу з апаратного інтерфейсу камери та його подальше кодування реалізовано за допомогою кросплатформного мультимедійного фреймворку FFmpeg. Для автоматизації цього процесу розроблено керуючий Python скрипт. Логіка скрипта полягає у формуванні масиву аргументів командного рядка та ініціалізації системного підпроцесу через модуль subprocess. Щоб надалі краще розуміти логіку даного Python скрипта розгялнемо блок-схему на рисунку 3.5 в додатку А.

Особливу увагу під час розробки було приділено оптимізації параметрів кодування для мінімізації мережевої затримки latency, що є критичним показником для систем відеоспостереження реального часу. У конфігурації FFmpeg жорстко задано використання програмного кодека відео libx264. Для прискорення обробки кадрів на малопотужному процесорі застосовано спеціалізований профіль налаштувань preset ultrafast у комбінації з параметром tune zerolatency, який вимикає буферизацію кадрів. Формат пікселів примусово встановлено у значення yuv420p, що гарантує максимальну сумісність із застарілими версіями клієнтського програмного забезпечення.

Важливим архітектурним рішенням став вибір транспортного протоколу для передачі сформованого RTSP потоку до сервера MediaMTX. Замість стандарту UDP, який не гарантує доставку пакетів, у параметрі rtsp transport було явно вказано використання протоколу tcp. Це рішення забезпечує цілісність відеопотоку та запобігає появі графічних артефактів під час трансляції у завантажених локальних мережах. Запуск сформованої команди ініціює безперервний процес трансляції на локальну адресу медіасервера

rtsp://localhost:8554/live, після чого потік стає доступним для зовнішніх підключень за протоколом ONVIF.

3.4 Запуск проекту на мікрокомп’ютері Raspberry PI

Завершальним етапом розробки програмно-апаратного засобу є проведення запуску сервера для підтвердження працездатності спроєктованої архітектури. Метою тестування є перевірка стабільності роботи транспортного та керуючого рівнів, а також оцінка споживання системних ресурсів мікрокомп'ютера Raspberry Pi 5 під час активного транскодування медіаданих.

Експериментальне дослідження розпочалося з ініціалізації базового транспортного рівня, а саме із запуску маршрутизатора потоків MediaMTX. Запуск виконувався безпосередньо у терміналі операційної системи Raspberry Pi OS. Під час ініціалізації сервер успішно зчитав конфігураційний файл mediamtx.yml, після чого відкрив локальні мережеві порти для прийому вхідних з'єднань.

У консольному виведенні було зафіксовано повідомлення про готовність до роботи за протоколами RTSP та WebRTC, що підтвердило успішне розгортання медіасервера.

Окрім базової ініціалізації портів, під час первинного запуску здійснювався візуальний моніторинг системного журналу подій. Це дало змогу верифікувати коректність виділення оперативної пам'яті процесом та підтвердити відсутність конфліктів на рівні мережевих інтерфейсів і раніше налаштованого пакетного фільтра.

Слід зазначити, що безпомилковий старт маршрутизатора є критично важливою передумовою для наступних етапів тестування, оскільки він виконує роль центрального комутаційного вузла між джерелом відеоданих та кінцевими споживачами. Лише після переходу системи у стабільний режим очікування

вхідних сесій транспортний рівень вважається повністю готовим до роботи. Запуск MediaMTX можна побачити на рискнку 3.6.

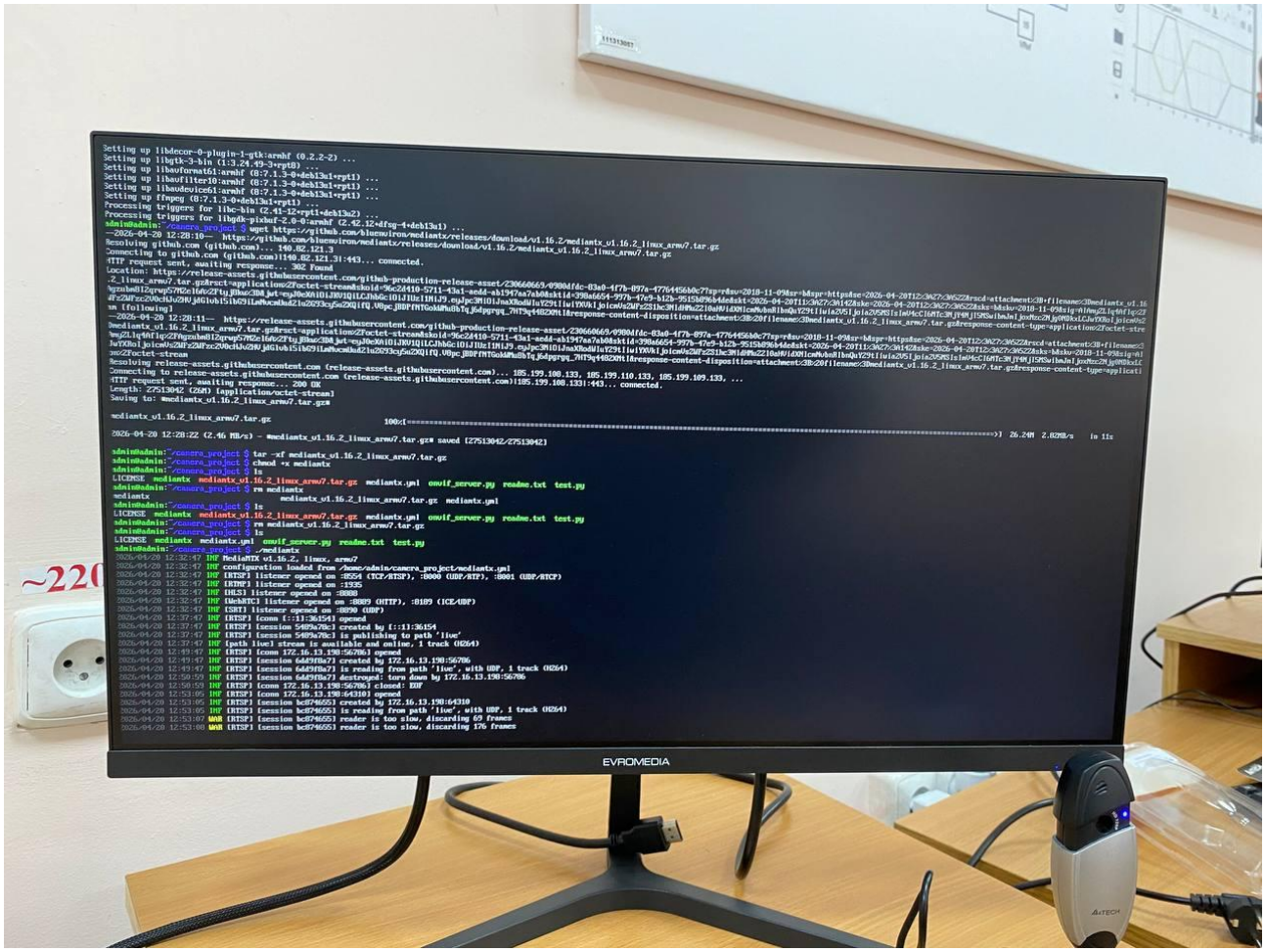


Рисунок 3.6 – Ініціалізація та запуск медіасервера MediaMTX у консолі Raspberry Pi

Після підготовки маршрутизатора було виконано запуск керуючого скрипта test.py, який відповідає за ініціалізацію захоплення відео з підключеної USB-камери A4tech.

Завдяки використанню модуля subprocess, скрипт успішно згенерував команду для утиліти FFmpeg та перенаправив її стандартне виведення у консоль для моніторингу. У системних логах було зафіксовано визначення апаратних характеристик камери, успішне застосування програмного кодека libx264 з профілем ultrafast та початок безперервної передачі кадрів на адресу локального медіасервера. Жодних помилок буферизації Buffer underflow або втрати кадрів

Frame drop на цьому етапі виявлено не було. Даний запуск показано на рисунку 3.7.

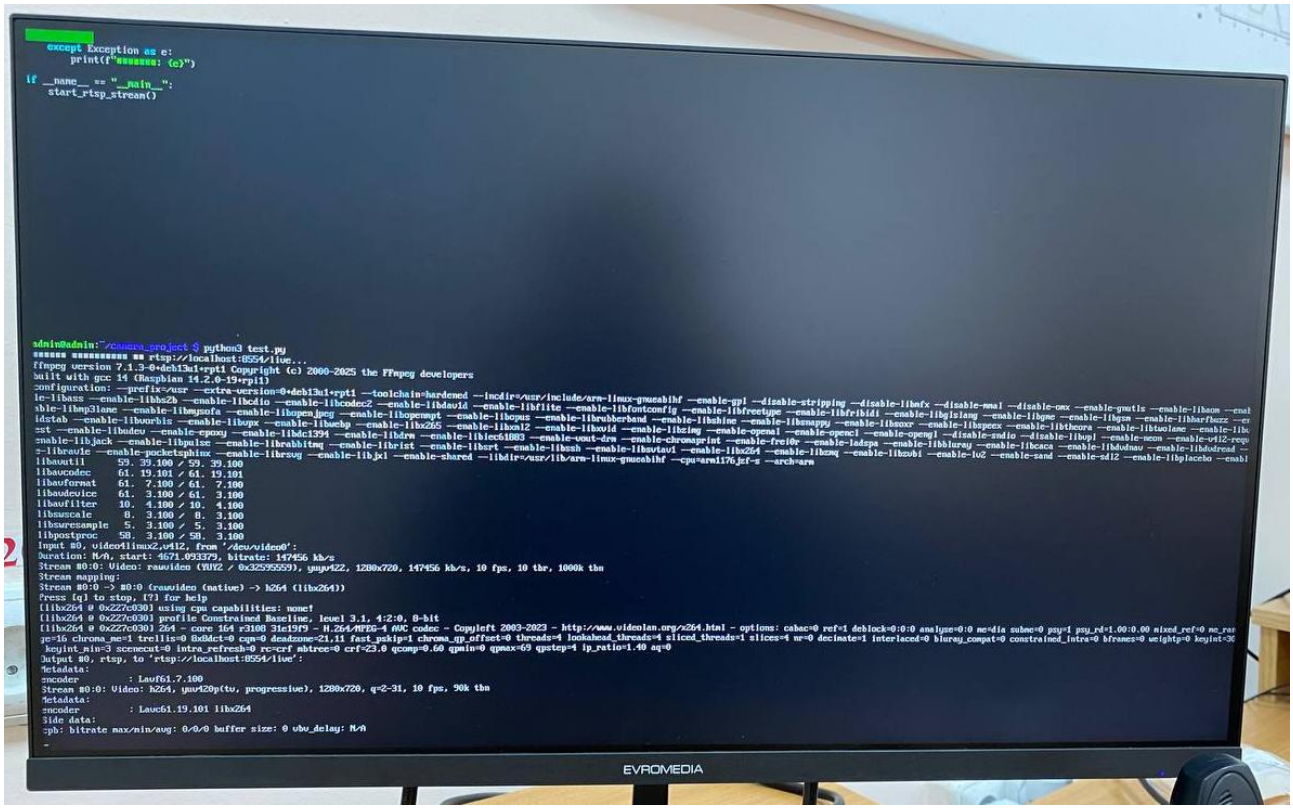


Рисунок 3.7 – Моніторинг процесу транскодування відеопотоку утилітою FFmpeg

Оскільки кодування відео високої роздільної здатності є надзвичайно ресурсомістким процесом, під час трансляції було проведено вимірювання навантаження на апаратну частину Raspberry Pi 5. Для моніторингу використовувалася вбудована системна утиліта htop. Результати вимірювань підтвердили правильність вибору 32-бітної операційної системи без графічного інтерфейсу. Основне навантаження лягло на центральний процесор, тоді як споживання оперативної пам'яті залишилося на стабільно низькому рівні. Зведені показники апаратного навантаження під час безперервної трансляції наведено у таблиці 3.1.

Таблиця 3.1 – Споживання апаратних ресурсів Raspberry Pi 5 під час трансляції

Процес/Служба	Завантаження CPU %	Споживання RAM МБ
Базова ОС фонові служби	1-2%	150 МБ
MediaMTX маршрутизація	2-4%	45 МБ
FFmpeg кодування H.264	45-60%	180 МБ

Наступним кроком експерименту стала перевірка керуючого рівня системи, який реалізує протокол ONVIF. У новому вікні термінала було запущено головний Python сервер onvif_server.py.

Сервер успішно прив'язався до мережевого порту 8080 та перейшов у режим очікування POST запитів. Для перевірки реакції системи з віддаленого персонального комп'ютера, що знаходився в одній локальній мережі з мікрокомп'ютером, було ініційовано процедуру пошуку пристроїв Device Discovery. Сервер миттєво відреагував на вхідні SOAP-повідомлення, що відобразилося у консолі у вигляді логування успішно оброблених методів, зокрема GetCapabilities та GetProfiles. Це підтвердило повну сумісність розробленої структури XML відповідей із загальноприйнятими клієнтськими стандартами.

На рисунку 3.8 показано результат запуску ONVIF сервера а саме логи SOAP запитів в його консолі.

Фінальною стадією лабораторного дослідження стала комплексна верифікація доставки відеосигналу до кінцевого споживача. За допомогою отриманого через протокол ONVIF мережевого посилання URI, на клієнтському персональному комп'ютері було відкрито RTSP потік за допомогою кросплатформного медіаплеєра VLC.

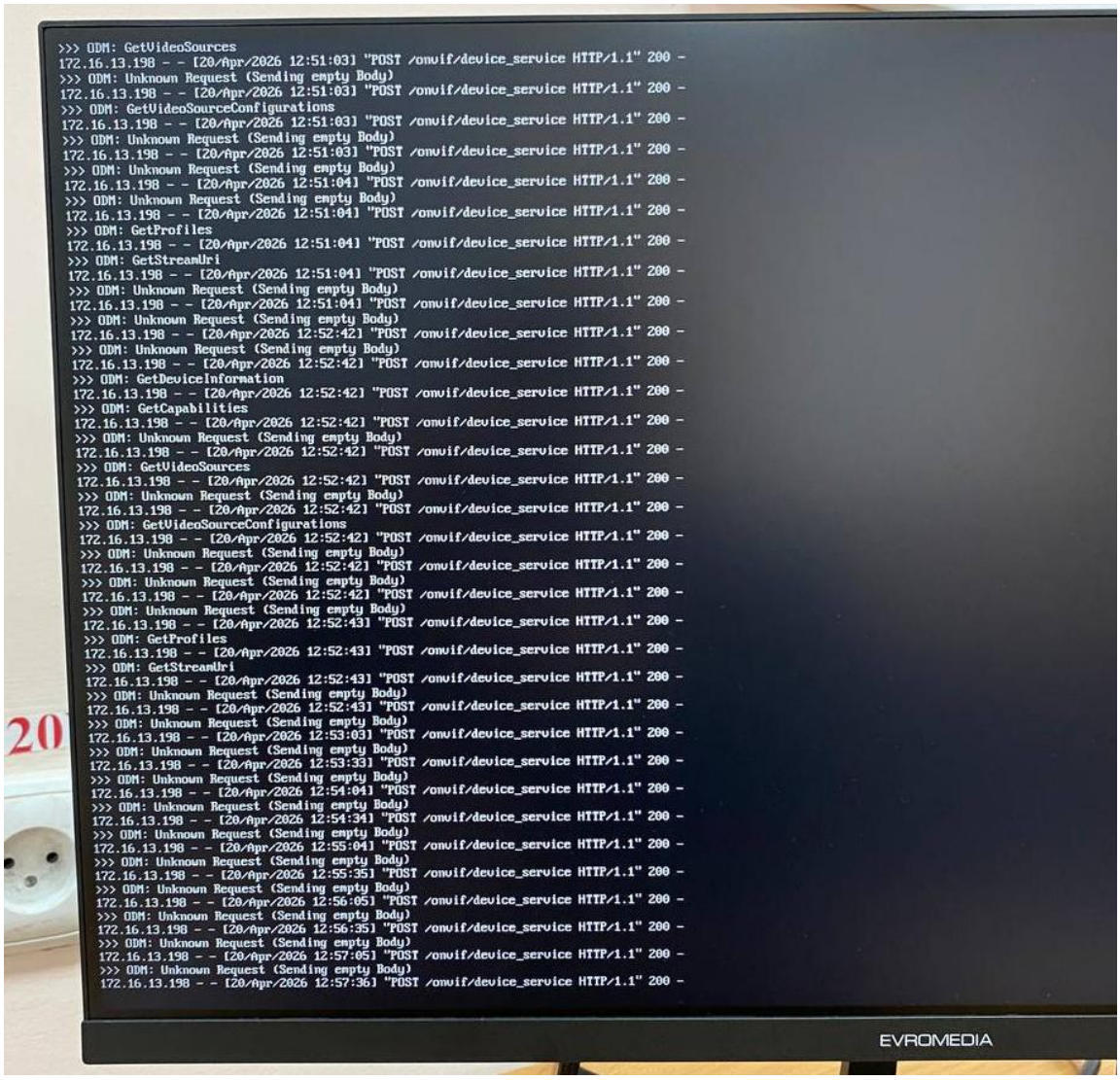


Рисунок 3.8 – Обробка вхідних SOAP запитів програмним ONVIF сервером

З'єднання було встановлено миттєво, після чого на екрані з'явилося живе зображення з камери A4tech. Суб'єктивна оцінка якості відеосигналу показала відсутність графічних артефактів, розсинхронізації чи пікселізації зображення. Завдяки застосуванню профілю нульової затримки zerolatency у налаштуваннях FFmpeg та використанню протоколу TCP, транспортна затримка відеопотоку End-to-End Latency у межах бездротової локальної мережі склала менше однієї секунди, що є відмінним показником для систем реального часу, побудованих на базі мікрокомп'ютерів. Процес налаштування і запуску трансляції відеопотоку в VLC показано на рисунку 3.9

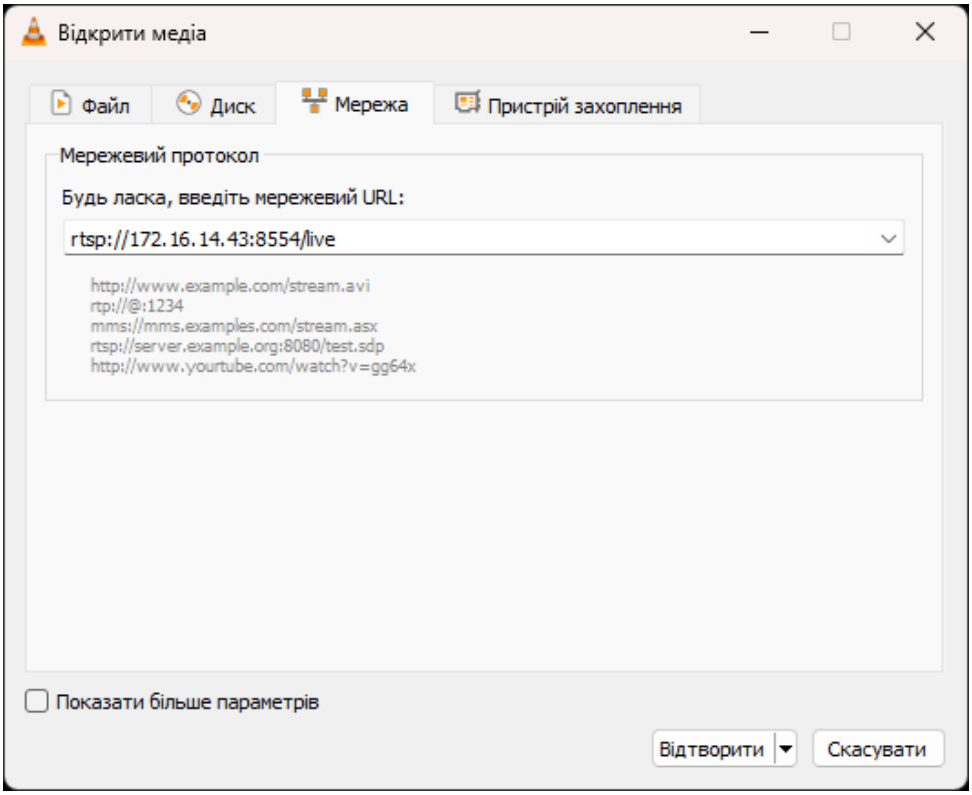


Рисунок 3.9 – Налаштування мережевого підключення до відеопотоку у медіаплеєрі VLC

І вже на рисунку 3.10 показано кінцевий результат а саме як інший пристрій клієнтський ноутбук підключився по мережі до камери підключеної до Raspberry PI.

Успішне проведення всіх етапів експерименту без критичних збоїв та програмних винятків повністю підтверджує життєздатність розробленої програмно-технічної архітектури. Комплекс готовий до експлуатації як базовий вузол у розширених системах відеоспостереження розумного дому або корпоративного сегмента.

Для підтвердження експлуатаційної надійності розробленого апаратно-програмного комплексу в умовах цілодобової роботи режим 24/7 було проведено п'ятнадцятихвилинний стрес-тест. Метою даного етапу експериментального дослідження була фіксація динаміки нагрівання кристала центрального процесора Raspberry Pi 5 під час безперервного транскодування відеопотоку

утилітою FFmpeg та перевірка ефективності активної системи охолодження, спроєктованої на попередніх етапах роботи.

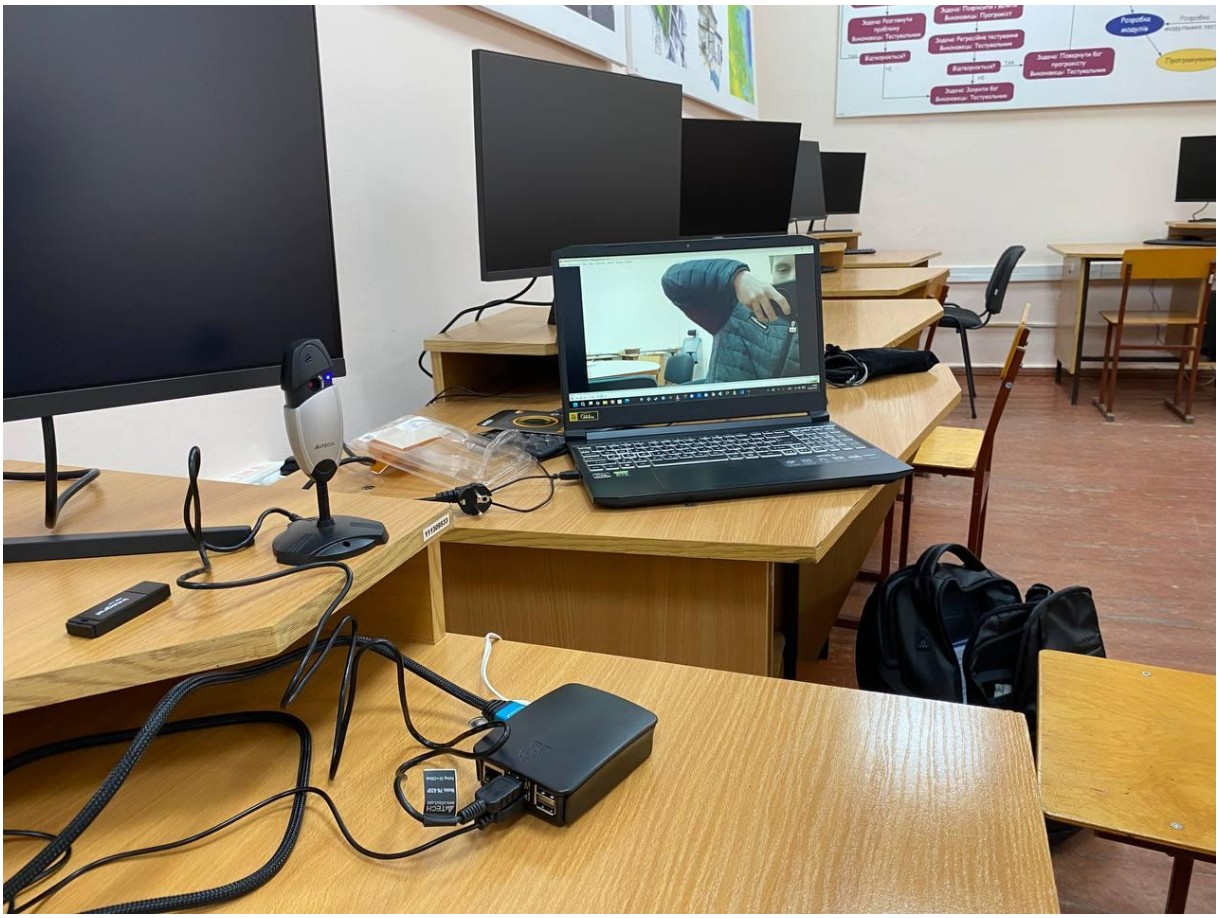


Рисунок 3.10 – Успішна трансляція відеопотоку у клієнтському медіаплеєрі VLC

Моніторинг температурних показників здійснювався за допомогою системної команди vcgencmd measure_temp, яка опитувала апаратні датчики мікрокомп'ютера з інтервалом в одну хвилину. Одночасно фіксувався загальний рівень завантаження процесорних ядер CPU Load. Результати вимірювань засвідчили, що у стані спокою базова температура процесора становила близько 45 градусів цельсія. Після ініціалізації процесу трансляції завантаження процесора стрімко зросло до стабільного діапазону 55-60%, що супроводжувалося прогнозованим підвищенням тепловиділення.

Протягом перших семи хвилин активної роботи температура кристала лінійно зростала, досягнувши позначки у 68 градусів цельсія. На цьому етапі спрацював апаратний широтно-імпульсний модулятор ШІМ, який автоматично підвищив оберти активного кулера. Завдяки цьому температурна крива стабілізувалася і протягом наступного часу стрес тесту не перевищувала 70 градусів. Динаміку зміни температурного режиму та навантаження процесора проілюстровано на графіку який зображено на рисунку3.11.

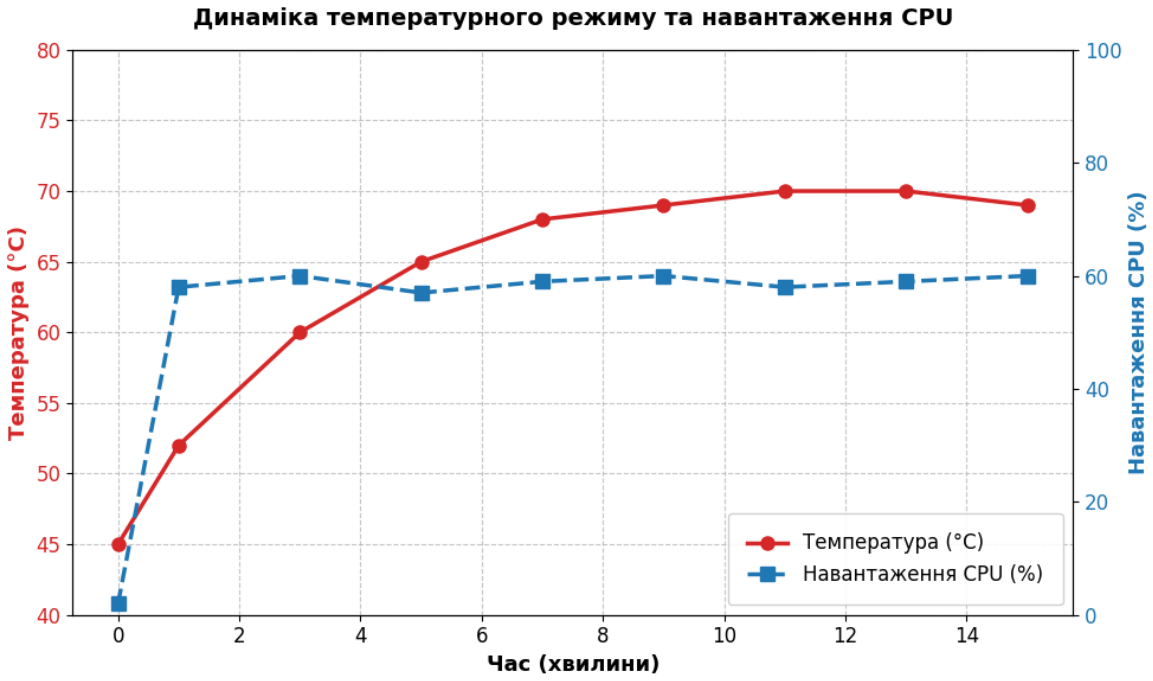


Рисунок 3.11 – Динаміка температурного режиму та навантаження CPU Raspberry Pi 5 під час стрес тесту

Отримані експериментальні дані переконливо доводять, що розроблена система не досягає критичної позначки температурного тротлінгу 85 градусів цельсія, при якій ядро автоматично знижує тактову частоту.

Відповідно, програмний кодувальник FFmpeg забезпечується стабільним ресурсом обчислювальної потужності, що виключає можливість пропуску кадрів frame drop або критичних затримок у транспортному відеопотоці навіть за умов тривалої експлуатації комплексу.

3.5 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи виконано практичну реалізацію та розгортання спроектованого програмно-апаратного комплексу відеоспостереження. Базовим середовищем для функціонування серверних модулів обрано операційну систему Raspberry Pi OS архітектура 32-bit, налаштування якої у безголовому режимі дозволило мінімізувати споживання апаратних ресурсів.

Розроблено та впроваджено керуючий програмний модуль мовою Python із використанням мікрофреймворку Flask. Модуль успішно реалізує обробку вхідних POST запитів за специфікацією ONVIF Profile S. Шляхом динамічної генерації XML відповідей у форматі протоколу SOAP забезпечено коректну ідентифікацію камери у мережі, передачу її апаратних характеристик та формування актуальних мережевих посилань URI для клієнтського програмного забезпечення.

Транспортний рівень системи успішно інтегровано на базі мультимедійного фреймворку FFmpeg та локального маршрутизатора MediaMTX. Доведено ефективність застосування програмного кодека H.264 із профілем ultrafast та примусовою маршрутизацією пакетів за протоколом TCP. Комплекс експериментальних досліджень підтвердив здатність мікрокомп'ютера стабільно захоплювати, транслювати та маршрутизувати відеопотік із затримкою Latency менше однієї секунди.

Результати стрес-тесту системи показали, що максимальне навантаження на центральний процесор під час активного транскодування не перевищує 60%, а максимальна температура кристала стабілізується на позначці 70 градусів цельсія завдяки системі активного охолодження. Це повністю виключає ризик теплового тротлінгу та підтверджує високу надійність створеного пристрою. Комплекс визнано функціонально завершеним і повністю придатним для використання у складі сучасних систем керування відеоспостереженням VMS.

ВИСНОВКИ

У кваліфікаційній роботі за результатами виконаних теоретичних та експериментальних досліджень було успішно вирішено актуальне практичне завдання а саме спроєктовано та програмно реалізовано автономний програмно-технічний засіб вузла відеоспостереження на базі мікрокомп’ютера Raspberry Pi 5 з підтримкою міжнародного стандарту ONVIF.

У першому розділі проведено комплексний аналіз сучасних технологій відеонагляду та принципів побудови архітектур інтернету речей IoT. Обґрунтовано доцільність відмови від закритих пропрієтарних рішень на користь відкритих міжнародних стандартів передачі медіаданих RTSP та мережевого керування пристроями ONVIF Profile S. На основі порівняльного аналізу апаратних платформ доведено, що використання мікрокомп'ютера Raspberry Pi 5 є оптимальним вибором для побудови кінцевого вузла. Ця платформа забезпечує необхідний баланс між високою обчислювальною потужністю для транскодування відео у реальному часі, енергоефективністю та широкими можливостями інтеграції зовнішньої периферії через інтерфейс USB.

У другому розділі розроблено загальну системну архітектуру та виконано детальне проєктування підсистем пристрою. На основі розрахунку енергетичного бюджету та аналізу тепловиділення обґрунтовано необхідність використання джерела живлення з підтримкою Power Delivery та впровадження активної системи охолодження з ШІМ керуванням. Це дозволило гарантувати стабільну роботу процесора без ризику виникнення температурного тротлінгу. Окрім апаратної частини, спроєктовано комплексну трирівневу модель мережевої кібербезпеки IoT пристрою. Вона включає ізоляцію портів засобами міжмережевого екрана UFW, криптографічний захист каналу керування на базі профілю WS Security та дайджест-автентифікацію клієнтів для доступу до транспортного відеопотоку. У третьому розділі здійснено безпосередню практичну програмно-технічну реалізацію розробленого комплексу. Для

оптимізації системних ресурсів розгорнуто 32 бітне середовище операційної системи Raspberry Pi OS Lite. На об'єктно-орієнтованій мові Python із використанням мікрофреймворку Flask створено функціональний керуючий сервер, здатний динамічно обробляти вхідні SOAP-запити.

Також в роботі налаштовано інтеграцію програмного кодувальника FFmpeg із локальним маршрутизатором MediaMTX, що забезпечило стабільну трансляцію H.264 відеопотоку за протоколом TCP. Проведені експериментальні дослідження та стрес тестування підтвердили високу надійність системи,транспортна затримка відео Latency склала менше однієї секунди, а максимальна температура процесора під постійним навантаженням стабілізувалася на безпечній позначці у 70 градусів цельсія.

Отримані результати підтверджують, що розроблений вузол відеоспостереження повністю відповідає поставленим технічним вимогам, є стійким до високих навантажень і може бути успішно інтегрований як у локальні системи розумного дому, так і в професійні системи керування відео VMS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Беверлі Д. Програмно-технічний засіб для домашньої автоматизованої системи відеоспостереження та сигналізації на базі одноплатного комп'ютера Raspberry Pi : кваліфікаційна робота. Хмельницький : Хмельницький національний університет, 2025.

2. Мандрик А. І. Програмно-технічний засіб охоронної сигналізації та нагляду в кіберфізичній системі «Розумний будинок» на платформі Raspberry Pi : кваліфікаційна робота. Хмельницький : Хмельницький національний університет, 2022.

3. Чайковська А. В. Програмно-технічні засоби підключення світлових приладів на основі одноплатної комп’ютерної системи Raspberry Pi4 : кваліфікаційна робота. Хмельницький : Хмельницький національний університет, 2025.

4. Коржова Д. Програмно-технічний засіб VPN сервера на основі одноплатної комп’ютерної системи Raspberry PI4 : кваліфікаційна робота. Хмельницький : Хмельницький національний університет, 2025.

5. Gopalakrishna V. Enhancing Pedestrian Safety and Traffic Analytics through ONVIF Metadata Processing and V2X Communication : Master's thesis. University of Cincinnati, 2024.

6. Pham D. T., Amin N. A., Yasutake D. [та ін.]. Development of plant phenotyping system using Pan Tilt Zoom camera and verification of its validity. *Computers and Electronics in Agriculture*. 2024. Vol. 227. 109579.

7. Élo G., Paller G. Enhanced floating plastic waste detecting on offsets of river Tisza, Hungary. *Chemical Engineering Transactions*. 2023. Vol. 107. P. 73–78.

8. Lee J., Lee S. Construction site safety management: a computer vision and deep learning approach. *Sensors*. 2023. Vol. 23, No. 2. P. 944.

9. Druta R., Gjuraj E., Savescu V. V., Nanu S. Enhancing real-time collaboration in Industrial Settings. *2025 25th International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2025. P. 115–121.

10. Bella G., Biondi P., Bognanni S., Esposito S. Petiot: Penetration testing the internet of things. *Internet of Things*. 2023. Vol. 22. 100707.

11. Valentini E. P., Franco T. A., Gottsfritz E. N. [та ін.]. Honeypot Embedded in Low-Cost IoT Hardware for Collecting and Analyzing Real-World Cyber Threats. *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing*. 2025. P. 1–6.

12. Ingle P., Kim Y. G. Integrated Interoperability Based Panoramic Video Synopsis Framework. *Proceedings Of The 39th ACM/SIGAPP Symposium On Applied Computing*. 2024. P. 584–591.

13. Mesa-Simón M., Escobar-Molero A., Sáez-Mingorance B. [та ін.]. Enabling live video provenance and authenticity: A C2PA-Based system with TPM-based security for livestreaming platforms. *Authorea Preprints*. 2025.

14. Aranzazu-Suescun C., Zapata-Rivera L. F. Methodology for Securing IoT IP-based Camera Systems. *IEEE Internet of Things Journal*. 2025.

15. Paller G., Élö G. Towards a Floating Plastic Waste Early Warning System. *SENSORNETS*. 2022. P. 45–50.

16. Khomenko Y., Babichev S. Modular IoT Architecture for Monitoring and Control of Office Environments Based on Home Assistant. *IoT*. 2025. Vol. 6, No. 4. P. 69.

17. Alcalde Bermenjo D. GITT. Automatización del hogar con un sistema domótico basado en Raspberry Pi. 2024.

18. Ardebili A. A., Zappatore M., Longo A., Ficarella A. Virtual Fencing for Safety-Critical Cyber-Physical Systems: Computer-Vision Enabled Digital Twins. *IEEE Access*. 2025.

19. Aras Y. E., Akpınar M., Kayaarma S. Y. Vagon İçi Yolcu Yoğunluğu Tespiti Passenger Density Detection in Railway Carriages. *2024 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 2024. P. 1–6.

20. Barros D. M. Development of a Hardware Module for Integrating Traffic Data Obtained from CCTV Cameras : Master's thesis. Universidade do Porto (Portugal), 2022.

21. Singh J., Dabas P., Bhati S. [та ін.]. A hybrid edge-cloud computing approach for energy-efficient surveillance using deep reinforcement learning. *2023 3rd International conference on technological advancements in computational sciences (ICTACS)*. IEEE, 2023. P. 432–438.

22. Barros D. M. Desenvolvimento de Um Módulo de Hardware Para Integração de Dados de Tráfego Obtidos a Partir de Câmaras CCTV. 2022.

23. Saputra I., Ashabi K. Application of ESP32-CAM for Cloud-Based Surveillance System in 3D Printing. *Jurnal Inovtek Seri Mesin Vol*. 2022. Vol. 3, No. 1. P. 1–6.

24. da Silva Cortez D. E., de Morais Barroca Filho I., Silva E. M. C., Girão G. Traffic control system development based on computer vision. *International Conference on Computational Science and Its Applications*. Cham : Springer International Publishing, 2022. P. 323–339.

25. Hassanizadeh P., Ebrahimi S., Dziembowski S., Szczepanski J. Trustless Delegation of Vector Commitment Construction in Resource-Constrained Settings. *Cryptology ePrint Archive*. 2025.

26. Pham D. T., Okayasu T., Yasutake D. [та ін.]. Enhancing Image Quality Assessment in Plant Phenotyping Robots. *Agricultural Information Research*. 2024. Vol. 33, No. 2. P. 97–108.

27. Bridova I., Brida P., Janovec M. System Approach of Smart Home Implementation with Cybersecurity Elements. *2025 11th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2025. Vol. 1. P. 1–6.

28. Vani R., Athilingam R., Krishna Naik K., Krishnaiah J. Hexapod for Remote Monitoring Using Internet of Things. *Doctoral Symposium on Computational Intelligence*. Singapore : Springer Nature Singapore, 2025. P. 171–181.

29. Singh D., Singh V., Singh S. A. Unlocking the Potential of Time Series IoT (Internet of Things) Data: The Transformative Role of AI and Machine Learning in Smart Cities. *2025 International Conference on Cognitive Computing in Engineering, Communications, Sciences and Biomedical Health Informatics (IC3ECSBHI)*. IEEE, 2025. P. 319–326.

30. Jinlong E., He L., Li Z., Liu Y. WiseCam: wisely tuning wireless pan-tilt cameras for cost-effective moving object tracking. *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023. P. 1–10.

31. Butihen K. A., Feliciano D. G., Pilapil R. [та ін.]. Thermal screening solution using infrared thermography and closed-circuit television. *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*. IEEE, 2022. P. 1–6.

32. Concepcion M. W. D. Design of a Wireless Mesh Network for Surveillance using IP Cameras in Barangay Camias, Magalang, Pampanga : Doctoral dissertation. Holy Angel University, 2024.

33. Chaudhary P. R., Christopher J., Maiti R. R. icaminspector: Classify video traffic and detect iot (spy) camera flows. *2024 Conference on Building a Secure & Empowered Cyberspace (BuildSEC)*. IEEE, 2024. P. 16–23.

34. Vaño R., Lacalle I., Sowiński P. [та ін.]. Cloud-native workload orchestration at the edge: A deployment review and future directions. *Sensors*. 2023. Vol. 23, No. 4. P. 2215.

35. Bhattacharya T., Ponaganti S., Vardhan A. P., Venkatesan Y. S. LAXMI: a novel AI-based system for smart, energy-efficient, and secure video footage processing. *The Journal of Supercomputing*. 2025. Vol. 81, No. 16. P. 1480.

36. Mathews J. Automated Event Recognition for Covert Cameras. South Dakota School of Mines and Technology, 2022.

37. Jinlong E., Han F., He L. [та ін.]. WiseCam: A systematic approach to intelligent pan-tilt cameras for moving object tracking. *IEEE Transactions on Mobile Computing*. 2024. Vol. 23, No. 12. P. 12330–12344.

38. Zannat A. A. 5G Networks in Port Operations: Case Study Pori : Doctoral dissertation. Tampere University, 2025.

39. Aueawatthanaphisut A., Trirattananurak C. A Unified Semantic Framework for Mitigating Device and Data Heterogeneity in IoT Systems. *2025 20th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. IEEE, 2025. P. 1–6.

40. Шапран Р. Проєктування інтелектуальної системи розумного дому. 2025.

41. Almazrouei O. S. M. B. H., Magalingam P., Hasan M. K., Shanmugam M. A review on attack graph analysis for iot vulnerability assessment: challenges, open issues, and future directions. *IEEE Access*. 2023. Vol. 11. P. 44350–44376.

42. Peñarando Martínez I. Aplicación para gestionar alarmas y cámaras domóticas. 2024.

43. Yu Y., Lu X., Gong J. An Internal Environment Anomaly Detection System for Secondary Water Supply Pump Houses Based on the Improved PatchCore Algorithm. *2024 China Automation Congress (CAC)*. IEEE, 2024. P. 983–988.

44. Györgyi C., Kecskeméti K., Vörös P. [та ін.]. In-network quality control of IP camera streams. *Proceedings of the Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 2023. P. 334–339.

45. Hussain A., Khan M., Ullah H. [та ін.]. Anomaly based camera prioritization in large scale surveillance networks. *Computers, Materials, & Continua*. 2022. Vol. 70, No. 2. P. 2171.

46. Gonçalves C., Dias T., Osório A. L., Camarinha-Matos L. M. A multi-supplier collaborative monitoring framework for informatics system of systems.

Working Conference on Virtual Enterprises. Cham : Springer International Publishing, 2022. P. 44–53.

47. Kramberger T., Cafuta B., Kramberger R. [та ін.]. SmartWasteCloud: an intelligent waste management system based on IoT and neural networks. *2023 International Conference on Artificial Intelligence Science and Applications in Industry and Society (CAISAIS)*. IEEE, 2023. P. 1–6.

48. Agrawal A., Maiti R. R. iTieProbe: How Vulnerable Your IoT Provisioning via Wi-Fi AP Mode or EZ Mode?. *IEEE Transactions on Information Forensics and Security*. 2024. Vol. 19. P. 10058–10070.

49. Iyengar A., Pearson J. Edge Computing Patterns for Solution Architects: Learn methods and principles of resilient distributed application architectures from hybrid cloud to far edge. Packt Publishing Ltd, 2024.

50. Purbaya S., Ariyanto E., Sudiharto D. W., Wijiutomo C. W. Improved image quality on surveillance embedded IP camera by reducing noises. *2017 3rd International Conference on Science in Information Technology (ICSITech)*. IEEE, 2017. P. 156–159.

51. Jin. Video Surveillance Control (CCTV) System Architecture. URL: https://bestuadjvl.click/product_details/33676631.html (дата звернення: 14.05.2026).

52. 8-портовий POE IP-відеореєстратор ASECAM 8CH NVR ONVIF 8 Мп (4K) Xmeye. URL: https://prom.ua/ua/p2014418443-portovyj-poe-videoregistrator.html (дата звернення: 14.05.2026).

53. OptoFrame™ W530. URL: https://www.xitrix.net/products/optoframe-w-530 (дата звернення: 14.05.2026).

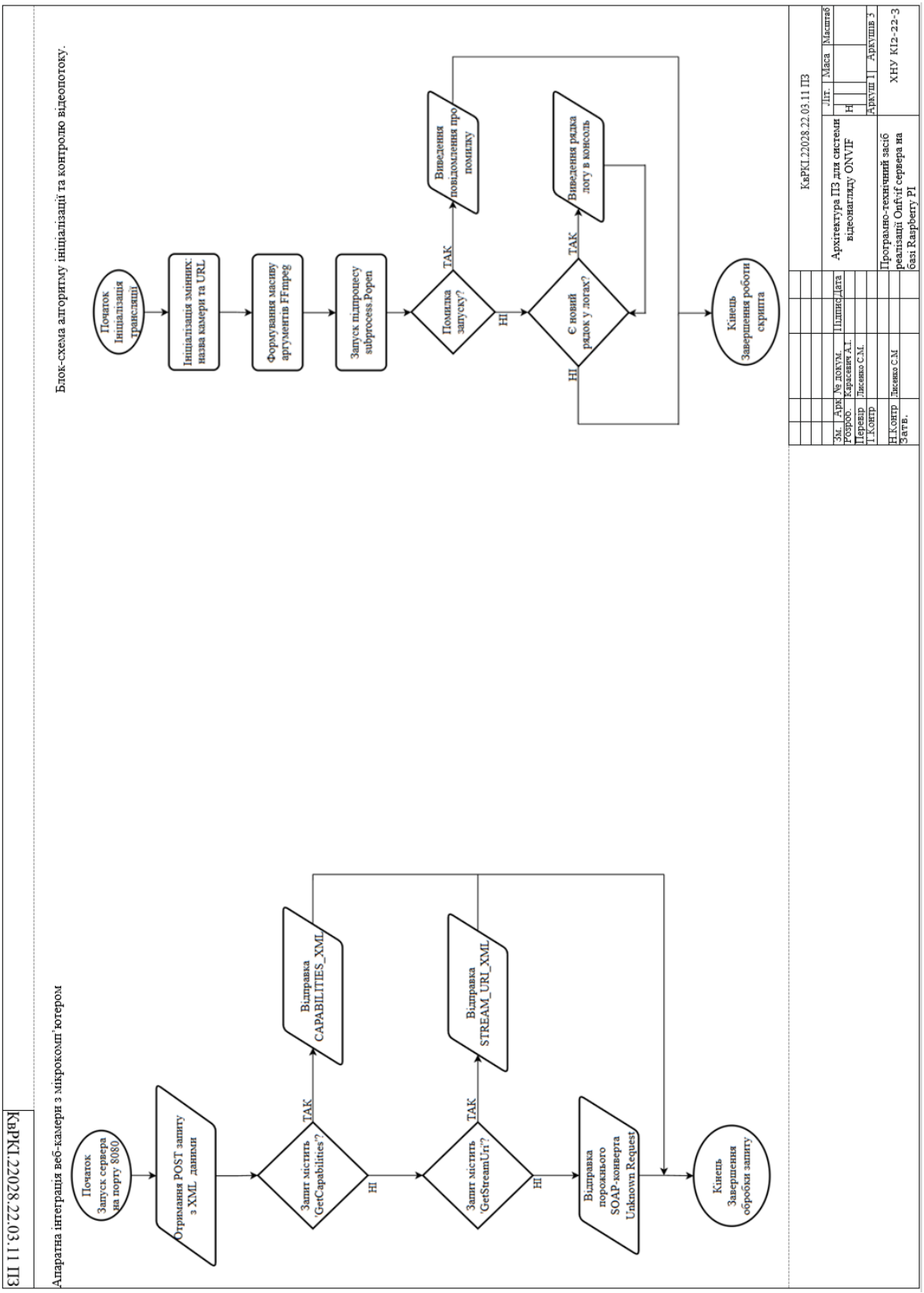
54. Мікрокомп'ютер Raspberry Pi 4 Model B 4GB. URL: https://evo.net.ua/mikrokomputer-raspberry-pi-4-model-b-4gb/ (дата звернення: 14.05.2026).

55. Офіційний радіатор з вентилятором для Raspberry Pi 5 Active Cooler. URL: https://rozetka.com.ua/ua/429246332/p429246332/ (дата звернення: 14.05.2026).

ДОДАТОК А

(обов’язковий)

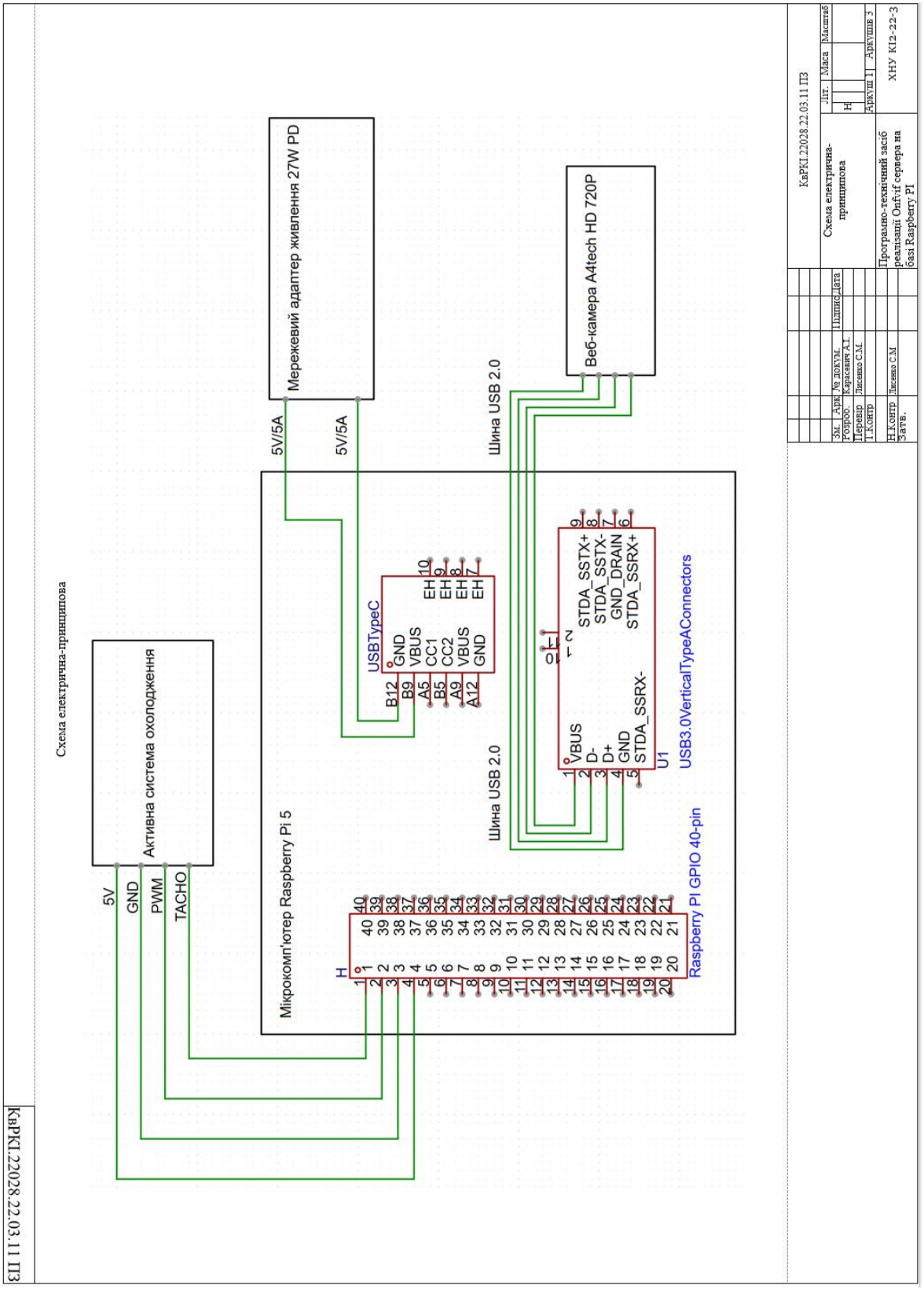
Копія креслення «Архітектура ПЗ для системи відеонагляду ONVIF»



ДОДАТОК Б

(обов’язковий)

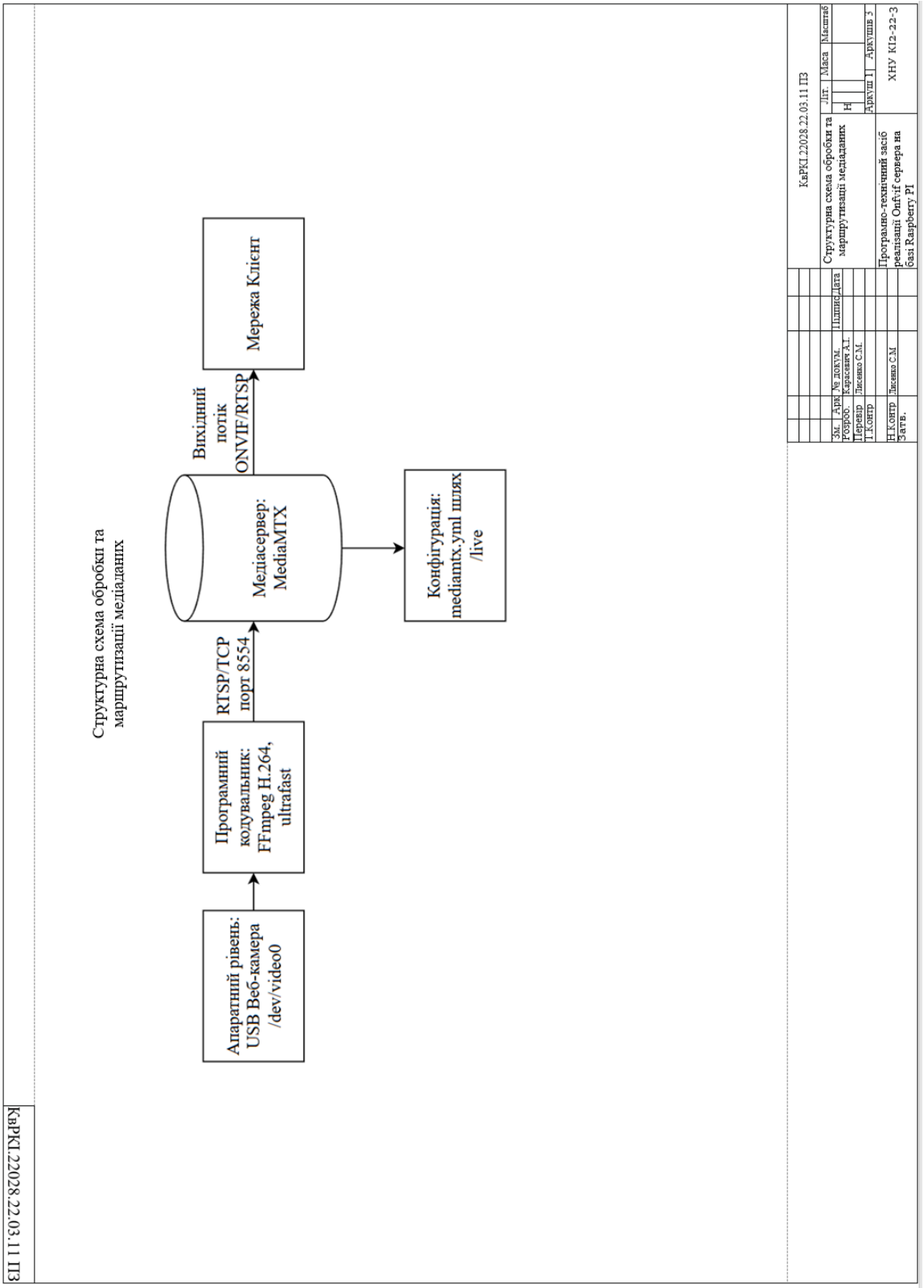
Копія креслення «Апаратне забезпечення проєкту»



ДОДАТОК В

(обов’язковий)

Копія креслення «Структурна схема обробки та маршрутизації медіаданих »



ДОДАТОК Г

(обов’язковий)

Копія креслення «Лістинг коду програми»

Лістинг коду до файлу Ffmpeg:

```
import subprocess
import os

def start_rtsp_stream():

camera_name = "dev/video0/"

rtsp_url = "rtsp://localhost:8554/live"

    command = [
 'ffmpeg',
 '-f', 'dshow',
 '-i', camera_name,
 '-vcodec', 'libx264',
 '-preset', 'ultrafast',
 '-tune', 'zerolatency',
 '-g', '30',
 '-bf', '0',
 '-pix_fmt', 'yuv420p',
 '-pkt_size', '1300',
 '-payload_type', '96',
 '-rtsp_transport', 'tcp',
 '-f', 'rtsp',
 rtsp_url
]

    print(f"Запуск трансляції на {rtsp_url}...")

    try:
```

```
            process = subprocess.Popen(command,
       stdout=subprocess.PIPE, stderr=subprocess.STDOUT)

       for line in iter(process.stdout.readline, b''):
        print(line.decode('utf-8').strip())

       except Exception as e:
       print(f"Помилка: {e}")

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

Лістинг коду до файлу ONVIF_server.py:

```
     from flask import Flask, Response, request

     app = Flask(__name__)

     NS = (
         'xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope" '
         'xmlns:tds="http://www.onvif.org/ver10/device/wsdl" '
         'xmlns:trt="http://www.onvif.org/ver10/media/wsdl" '
         'xmlns:tt="http://www.onvif.org/ver10/schema"'
     )

     # 1. Відповідь GetCapabilities
     CAPABILITIES_XML = f"""<?xml version="1.0" encoding="UTF-8"?>
     <SOAP-ENV:Envelope {NS}>
       <SOAP-ENV:Body>
         <tds:GetCapabilitiesResponse>
           <tds:Capabilities>
             <tt:Device>

<tt:XAddr>http://192.168.0.224:8080/onvif/device_service</tt:XAddr>
             </tt:Device>
```

```
            <tt:Media>

<tt:XAddr>http://192.168.0.224:8080/onvif/device_service</tt:XAddr>
              <tt:StreamingCapabilities>
                <tt:RTP_RTSP_TCP>true</tt:RTP_RTSP_TCP>
              </tt:StreamingCapabilities>
            </tt:Media>
          </tds:Capabilities>
        </tds:GetCapabilitiesResponse>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>"""

    # 2. Відповідь GetDeviceInformation
    DEVICE_INFO_XML = f"""<?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope {NS}>
      <SOAP-ENV:Body>
        <tds:GetDeviceInformationResponse>
          <tds:Manufacturer>Raspberry Pi</tds:Manufacturer>
          <tds:Model>Pi 5 Server</tds:Model>
          <tds:FirmwareVersion>1.0.0</tds:FirmwareVersion>
          <tds:SerialNumber>RPi5-2024-TEST</tds:SerialNumber>
          <tds:HardwareId>v5.0</tds:HardwareId>
        </tds:GetDeviceInformationResponse>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>"""

    # 3. GetProfiles
    PROFILES_XML = f"""<?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope {NS}>
      <SOAP-ENV:Body>
        <trt:GetProfilesResponse>
          <trt:Profiles token="MainProfile" fixed="true">
            <tt:Name>MainProfile</tt:Name>
            <tt:VideoSourceConfiguration token="VideoSourceConfig_0">
```

```
                <tt:Name>VSC_0</tt:Name>
                <tt:UseCount>1</tt:UseCount>
                <tt:SourceToken>VideoSource_0</tt:SourceToken>
                <tt:Bounds x="0" y="0" width="1280" height="720"/>
              </tt:VideoSourceConfiguration>
              <tt:VideoEncoderConfiguration token="VideoEncoderConfig_0">
                <tt:Name>VEC_0</tt:Name>
                <tt:UseCount>1</tt:UseCount>
                <tt:Encoding>H264</tt:Encoding>
                <tt:Resolution>
                  <tt:Width>1280</tt:Width>
                  <tt:Height>720</tt:Height>
                </tt:Resolution>
                <tt:Quality>4</tt:Quality>
                <tt:RateControl>
                  <tt:FrameRateLimit>30</tt:FrameRateLimit>
                  <tt:EncodingInterval>1</tt:EncodingInterval>
                  <tt:BitrateLimit>2048</tt:BitrateLimit>
                </tt:RateControl>
                <tt:H264>
                  <tt:GovLength>30</tt:GovLength>
                  <tt:H264Profile>Main</tt:H264Profile>
                </tt:H264>
                <tt:Multicast>

<tt:Address><tt:Type>IPv4</tt:Type><tt:IPv4Address>0.0.0.0</tt:IPv4Address>
</tt:Address>
                  <tt:Port>0</tt:Port>
                  <tt:TTL>1</tt:TTL>
                  <tt:AutoStart>false</tt:AutoStart>
                </tt:Multicast>
                <tt:SessionTimeout>PT1M</tt:SessionTimeout>
              </tt:VideoEncoderConfiguration>
            </trt:Profiles>
```

```
    </trt:GetProfilesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>"""

# 4. GetVideoSources
VIDEO_SOURCES_XML = f"""<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope {NS}>
  <SOAP-ENV:Body>
    <trt:GetVideoSourcesResponse>
      <trt:VideoSources token="VideoSource_0">
        <tt:Framerate>30</tt:Framerate>
        <tt:Resolution>
          <tt:Width>1280</tt:Width>
          <tt:Height>720</tt:Height>
        </tt:Resolution>
      </trt:VideoSources>
    </trt:GetVideoSourcesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>"""

# 5. GetStreamUri
STREAM_URI_XML = f"""<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope {NS}>
  <SOAP-ENV:Body>
    <trt:GetStreamUriResponse>
      <trt:MediaUri>
        <tt:Uri>rtsp://192.168.0.224:8554/live</tt:Uri>
        <tt:InvalidAfterConnect>false</tt:InvalidAfterConnect>
        <tt:InvalidAfterReboot>false</tt:InvalidAfterReboot>
        <tt:Timeout>PT1M</tt:Timeout>
      </trt:MediaUri>
    </trt:GetStreamUriResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>"""
```

```
# Додаткова відповідь для стабільності
VIDEO_CONFIGS_XML = f"""<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope {NS}>
  <SOAP-ENV:Body>
    <trt:GetVideoSourceConfigurationsResponse>
      <trt:Configurations token="VideoSourceConfig_0">
        <tt:Name>VSC_0</tt:Name>
        <tt:UseCount>1</tt:UseCount>
        <tt:SourceToken>VideoSource_0</tt:SourceToken>
        <tt:Bounds x="0" y="0" width="1280" height="720"/>
      </trt:Configurations>
    </trt:GetVideoSourceConfigurationsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>"""

@app.route('/onvif/device_service', methods=['POST'])
def device_service():
    xml_data = request.data.decode('utf-8')

    # Логування запиту для відладки
    if 'GetCapabilities' in xml_data:
        print(">>> ODM: GetCapabilities")
    return
Response(CAPABILITIES_XML, mimetype='application/soap+xml')

    if 'GetDeviceInformation' in xml_data:
        print(">>> ODM: GetDeviceInformation")
    return
Response(DEVICE_INFO_XML, mimetype='application/soap+xml')

    if 'GetProfiles' in xml_data:
        print(">>> ODM: GetProfiles")
        return
```

```
    Response(PROFILES_XML, mimetype='application/soap+xml')

        if 'GetVideoSources' in xml_data:
            print(">>> ODM: GetVideoSources")
       return
    Response(VIDEO_SOURCES_XML, mimetype='application/soap+xml')

        if 'GetVideoSourceConfigurations' in xml_data:
            print(">>> ODM: GetVideoSourceConfigurations")
       return
    Response(VIDEO_CONFIGS_XML, mimetype='application/soap+xml')

        if 'GetStreamUri' in xml_data:
            print(">>> ODM: GetStreamUri")
       return Response(STREAM_URI_XML, mimetype='application/soap+xml')

        # Відповідь за замовчуванням
        print(">>> ODM: Unknown Request (Sending empty Body)")
        return Response(
            f'<?xml   version="1.0"   encoding="UTF-8"?><SOAP-ENV:Envelope
{NS}><SOAP-ENV:Body/></SOAP-ENV:Envelope>',
            mimetype='application/soap+xml'
        )

    if __name__ == '__main__':

        print("ONVIF Server started on port 8080")
        app.run(host='0.0.0.0', port=8080, debug=False)
```

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Артур КАРАСЕВИЧ

Співавтор:

Назва: Програмно-технічний засіб реалізації Onfvif сервера на базі Raspberry PI

Експерт: Сергій ЛИСЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:5.02%

Коефіцієнт подібності 2:1.17%

Мікропробіли: 20

Заміна букв: 3

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-19 17:42:09.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-19 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 14%**

	Документ		Сумарний збіг по Базі Даних	
ID: 271736 Назва: БКР Програмно-технічний засіб реалізації Onfvif сервера на базі Raspberry PI Додано в БД: 2026-05-19 Автора: Артур КАРАСЕВИЧ Керівники: Сергій ЛИСЕНКО Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	82107	633	1473 (2%)	18 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Карасевич Артур Іванович

Тема: Програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___56___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи була розробка програмно-технічного засобу реалізації Onvif сервера на базі Raspberry PI

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено теоретичне дослідження методів та засобів реалізації мережевого сервера відеоспостереження на базі мікрокомп'ютера Raspberry PI. Зокрема, проаналізовано предметну область, здійснено порівняльний огляд наявних рішень, досліджено архітектуру стандарту ONVIF та мережеві протоколи передачі медіаданих XML, SOAP, RTSP, RTP, а також виконано постановку задачі. В другому розділі кваліфікаційної роботи проведено розробку архітектури та технічних рішень розумної камери на базі стандарту ONVIF. А саме: визначено апаратні та програмні підсистеми; спроєктовано структурну схему трансляції відеоданих; обґрунтовано вибір параметрів кодування стандарт стиснення H.264 та розраховано мережеве навантаження; спроєктовано підсистеми енергоспоживання та теплообміну з використанням активного охолодження; розроблено комплексну трирівневу модель системи мережевої безпеки. В третьому розділі кваліфікаційної роботи виконано програмно-апаратну реалізацію та впровадження ONVIF-сервера на базі Raspberry PI 5. Зокрема: розгорнуто операційне середовище Raspberry Pi OS Lite 32 біт; розроблено програмну логіку сервера мовою Python з використанням мікрофреймворку Flask;

виконано системну інтеграцію маршрутизатора MediaMTX та інструментарію FFmpeg; здійснено успішний запуск проєкту, перевірку трансляції та стрес-тестування температурних режимів системи.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатня увага аналізу предметної області

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: робота виконана на належному технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре (B / 86)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Стецюк Микола Васильович, PhD, ст. викладач, к. кібербезпеки

_____

"28" 05 2026 р. _____(підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Артур КАРАСЕВИЧ
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-22-3

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Програмно-технічний засіб реалізації Onvif сервера на базі Raspberry PI
Автор Артур КАРАСЕВИЧ
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д.т.н., професор Сергій ЛИСЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 5.02%; та системою Anti-Plagiarism складає 1.0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми _____ Підпис — Андрій НІЧЕПОРУК (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ Підпис — Сергій ЛИСЕНКО (Ім'я, ПРІЗВИЩЕ)