МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп’ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Система дистанційного керування БПЛА через розподілені ресурси

Спеціальність 123 Комп’ютерна інженерія

Освітня програма «Комп’ютерна інженерія»

Здобувач _____Олег БОНДАРЕНКО
підпис

«__» _____ 2026 р.

Керівник старший викладач

_____ Катерина ОБУХОВА
підпис

«__» _____ 2026 р.

Миколаїв – 2026

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Бондаренко Олег Андрійович
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Система дистанційного керування БПЛА через розподілені ресурси

Затверджена наказом по ЧНУ ім. Петра Могили від 25.11.2025 № 294.

2. Строк представлення кваліфікаційної роботи «____»_____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Створення прототипу апаратно-програмного комплексу системи дистанційного керування БПЛА з розподіленою обробкою відеоданих у режимі реального часу

4. Перелік питань, що підлягають розробці:

1) аналіз та обґрунтування архітектури системи з розподіленою обробкою відеоданих;
2) виконати моделювання системи та порівняльний аналіз продуктивності апаратних платформ Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B;
3) розробка апаратно-програмного забезпечення БПЛА та інтеграція алгоритмів OpenCV ;
4) тестування створеного прототипу системи.

5. Перелік графічних матеріалів
Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Катерина ОБУХОВА
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Олег БОНДАРЕНКО
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Система дистанційного керування БПЛА через розподілені ресурси

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	01.12.2025	10.12.2025	Виконано
2	Огляд літератури за темою роботи	11.12.2025	25.12.2025	Виконано
3	Складання календарного плану КБР	26.12.2025	30.12.2025	Виконано
4	Аналіз предметної області	12.01.2026	05.02.2026	Виконано
5	Розробка проєктних рішень	06.02.2026	03.03.2026	Виконано
6	Моделювання та конструювання АПЗ	04.03.2026	07.04.2026	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	08.04.2026	30.04.2026	Виконано
8	Оформлення КБР та презентації	01.05.2026	20.05.2026	Виконано
9	Перший передній захист КБР	22.05.2026	22.05.2026	Виконано
10	Другий передній захист КБР	05.06.2026	05.06.2026	Виконано
11	Завершення оформлення КБР та презентації	06.06.2026	09.06.2026	Виконано
12	Перевірка на академічний плагіат, фальсифікацію та списування	10.06.2026	11.06.2026	Виконано
13	Відгук керівника КБР	12.06.2026	12.06.2026	Виконано
14	Подання КБР рецензенту та рецензування КБР	13.06.2026	14.06.2026	Виконано
15	Подання КБР, її електронної копії та інших документів (відгуку, рецензії)	15.06.2026	17.06.2026	Виконано
16	Захист кваліфікаційної бакалаврської роботи	24.06.2026	24.06.2026	Виконано

Керівник роботи _____ Катерина ОБУХОВА

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Олег БОНДАРЕНКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 2026_ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Система дистанційного керування БПЛА через розподілені ресурси»
Здобувач: Бондаренко Олег Андрійович
Керівник: ст. викладач Обухова Катерина Олександрівна

Розвиток безпілотних літальних апаратів упродовж останніх років набуває стрімкого характеру завдяки розширенню доступних обчислювальних платформ та засобів комп'ютерного зору. Сучасні БПЛА дедалі частіше застосовуються для відеоспостереження, моніторингу територій, пошуково-рятувальних операцій та забезпечення безпеки. Одним із ключових завдань таких систем є аналіз відеоданих у реальному часі для подальшого прийняття керуючих рішень на основі результатів розпізнавання об'єктів.

Актуальність теми зумовлена тим, що ефективність роботи систем дистанційного керування БПЛА значною мірою залежить від обчислювальних ресурсів апаратної платформи. Питання ефективного використання доступних одноплатних комп'ютерів для задач комп'ютерного зору в системах керування БПЛА залишається недостатньо дослідженим, що підтверджує актуальність обраної теми.

Об'єктом дослідження є процес дистанційного керування БПЛА з використанням розподілених обчислювальних ресурсів.

Предметом дослідження є апаратно-програмне забезпечення системи обробки відеоданих у реальному часі на бортовому обчислювальному модулі БПЛА на базі одноплатних комп'ютерів Raspberry Pi.

Метою роботи є підвищення ефективності системи дистанційного керування БПЛА шляхом дослідження та порівняння можливостей різних бортових обчислювальних платформ для обробки відеопотоку в режимі реального часу з метою виявлення небезпечних об'єктів та перешкод.

Практична значимість роботи полягає у дослідженні та визначенні оптимального поєднання апаратних і програмних засобів для побудови системи дистанційного керування БПЛА з обробкою відеоданих у режимі реального часу. На основі порівняльного аналізу платформ Raspberry Pi із застосуванням алгоритмів бібліотеки OpenCV визначається найефективніша конфігурація бортового обчислювального модуля, здатного самостійно виявляти небезпечні об'єкти та перешкоди і своєчасно підказувати оператору подальші дії. Отримані результати знайдуть застосування у системах дистанційного керування БПЛА для відеоспостереження, моніторингу територій, пошуково-рятувальних операцій, а також є підґрунтям для подальшого розвитку розподілених архітектур керування роботизованими системами.

Апробація: Бондаренко О. А. (бакалаврант), Обухова К. О. (ст. викладач каф. комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Система дистанційного керування БПЛА через розподілені ресурси. Міжнародна науково-практична конференція «Інформаційні технології: теорія та практика», Харківський національний університет міського господарства імені О. М. Бекетова [2].

Пояснювальна записка кваліфікаційної бакалаврської роботи складається зі вступу, трьох розділів, висновків, переліку джерел посилання та додатків.

У першому розділі проведено системний аналіз предметної області систем дистанційного керування БПЛА з розподіленою обробкою відеоданих. Виконано огляд та порівняльний аналіз існуючих рішень і апаратних платформ для бортових обчислювальних модулів, обґрунтовано вибір платформ Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B як об'єктів порівняльного дослідження. Сформовано повну специфікацію вимог до апаратно-програмного забезпечення системи з урахуванням масогабаритних вимог БПЛА DJI Phantom 4.

У другому розділі розроблено покроковий алгоритм роботи системи обробки відеоданих у реальному часі та спроєктовано дворівневу розподілену архітектуру системи керування БПЛА. Обґрунтовано вибір усіх ключових апаратних та програмних компонентів, зокрема бібліотеки OpenCV як основного інструменту комп'ютерного зору та протоколу UDP для передачі даних між бортовим модулем та наземною станцією.

У третьому розділі реалізовано апаратну частину системи у двох конфігураціях на базі Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B, розроблено програмне забезпечення системи комп'ютерного зору мовою Python 3.14 з асинхронною багатопотоковою архітектурою, проведено порівняльне тестування продуктивності обох платформ за чотирма тестовими сценаріями при різних роздільних здатностях відеопотоку.

Загальний розмір роботи 73 сторінок (без додатків), 27 рисунків, 8 таблиць, 3 додатки, 21 джерел посилання.

Ключові слова: безпілотний літальний апарат, система дистанційного керування, комп'ютерний зір, Raspberry Pi, OpenCV, розподілена архітектура, обробка відеоданих у реальному часі, edge-обчислення.

ABSTRACT

of the Bachelor's Thesis

"A System for Remote Control of UAVs via Distributed Resources"

Applicant: Bondarenko Oleg

Supervisor: Senior Lecturer Obukhova Kateryna

The development of unmanned aerial vehicles has accelerated rapidly in recent years thanks to the expansion of available computing platforms and computer vision technologies. Modern UAVs are increasingly used for video surveillance, territory monitoring, search and rescue operations, and security. One of the key tasks of such systems is real-time video data analysis for subsequent decision-making based on object recognition results.

The relevance of the topic stems from the fact that the effectiveness of UAV remote control systems depends to a large extent on the computational resources of the hardware platform. The issue of effectively utilizing available single-board computers for computer vision tasks in UAV control systems remains under-researched, which confirms the relevance of the chosen topic.

The object of the study is the process of remote control of UAVs using distributed computing resources.

The subject of the study is the hardware and software of a real-time video data processing system on a UAV's onboard computing module based on Raspberry Pi single-board computers.

The aim of this work is to improve the efficiency of the UAV remote control system by researching and comparing the capabilities of various onboard computing platforms for real-time video stream processing to detect hazardous objects and obstacles.

The practical significance of this work lies in the study and determination of the optimal combination of hardware and software for building a remote-control system for UAVs with real-time video processing. Based on a comparative analysis of Raspberry Pi platforms using OpenCV library algorithms, the most effective configuration of an onboard computing module is determined, capable of independently detecting hazardous objects and obstacles and promptly suggesting further actions to the operator. The results obtained will find application in UAV remote control systems for video surveillance, territory monitoring, and search and rescue operations, and also serve as a foundation for the further development of distributed control architectures for robotic systems.

Presentation: Bondarenko O. A. (undergraduate student), Obukhova K. O. (senior lecturer, Department of Computer Engineering, Petro Mohyla Black Sea National University, Mykolaiv, Ukraine). Remote control system for UAVs via distributed resources. International Scientific and Practical Conference "Information Technologies: Theory and Practice," O. M. Beketov National University of Urban Economy, Kharkiv [2].

The explanatory note for the bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter presents a systematic analysis of the subject area of UAV remote control systems with distributed video data processing. It provides a review and comparative analysis of existing solutions and hardware platforms for onboard computing

modules, and justifies the selection of the Raspberry Pi 3 Model B+ and Raspberry Pi 4 Model B platforms as subjects for comparative study. A complete specification of hardware and software requirements for the system was developed, taking into account the size and weight requirements of the DJI Phantom 4 UAV.

In the second chapter, a step-by-step algorithm for the operation of the real-time video data processing system was developed, and a two-level distributed architecture for the UAV control system was designed. The selection of all key hardware and software components is justified, in particular the OpenCV library as the primary computer vision tool and the UDP protocol for data transmission between the onboard module and the ground station.

In the third chapter, the hardware part of the system was implemented in two configurations based on the Raspberry Pi 3 Model B+ and Raspberry Pi 4 Model B; the computer vision system software was developed in Python 3.14 with an asynchronous multithreaded architecture, and comparative performance testing of both platforms was conducted across four test scenarios at various video stream resolutions.

The total size of the work is 73 pages (excluding appendices), 27 figures, 8 tables, 3 appendices, 21 references.

Keywords: unmanned aerial vehicle, remote control system, computer vision, Raspberry Pi, OpenCV, distributed architecture, real-time video processing, edge computing.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПЗ – апаратно-програмне забезпечення

БПЛА – безпілотні літальні апарати

КБР – кваліфікаційна бакалаврська робота

ПЗ – програмне забезпечення

FPS – Frames per Second

GPU – Graphics Processing Unit

ВСТУП

Розвиток безпілотних літальних апаратів (БПЛА) упродовж останніх років набуває стрімкого характеру завдяки розширенню доступних обчислювальних платформ та засобів комп'ютерного зору. На сьогоднішній день сучасні БПЛА дедалі частіше застосовуються для відеоспостереження, моніторингу територій, пошуково-рятувальних операцій та забезпечення безпеки. Одним із ключових завдань таких систем є аналіз відеоданих у реальному часі для подальшого прийняття керуючих рішень на основі результатів розпізнавання об'єктів.

Актуальність теми зумовлена ефективністю роботи систем дистанційного керування БПЛА. значною мірою залежить від обчислювальних ресурсів апаратної платформи. У сучасних системах обробка відеоданих може виконуватися як безпосередньо на борту апарата (edge-обчислення), так і з використанням віддалених обчислювальних ресурсів. Такий підхід дозволяє реалізувати розподілену архітектуру, у якій обчислювальне навантаження розподіляється між бортовими та віддаленими серверами. Для невеликих автономних роботизованих пристроїв часто застосовуються одноплатні комп'ютери, які поєднують компактні розміри, низьке енергоспоживання та високу продуктивність. Тому можна сказати, що на сьогодні дослідження впливу апаратних характеристик обчислювальної платформи на швидкість та стабільність роботи алгоритмів комп'ютерного зору в системах керування БПЛА є актуальним.

Прикладне та практичне значення обраної теми полягає у дослідженні та визначенні оптимального поєднання апаратних і програмних засобів для побудови системи дистанційного керування БПЛА з обробкою відеоданих у режимі реального часу. На основі порівняльного аналізу платформ Raspberry Pi із застосуванням алгоритмів бібліотеки OpenCV визначається найефективніша конфігурація бортового обчислювального модуля. Модуль здатний самостійно виявляти небезпечні об'єкти та перешкоди, своєчасно підказувати оператору подальші дії. Таке рішення знайде застосування не лише у військовій та рятувальній сферах, а й у цивільних галузях. Наприклад, від агросектору до нагляду за промисловими об'єктами. Отримані результати є підґрунтям для подальшого

розвитку розподілених архітектур керування роботизованими системами із залученням віддалених обчислювальних серверів.

Об'єктом дослідження є процес дистанційного керування БПЛА з використанням розподілених обчислювальних ресурсів.

Предметом дослідження є апаратно-програмне забезпечення (АПЗ) системи обробки відеоданих у реальному часі на бортовому обчислювальному модулі БПЛА на базі одноплатних комп'ютерів Raspberry Pi.

Метою роботи є підвищення ефективності системи дистанційного керування БПЛА. Це буде відбуватися шляхом дослідження та порівняння можливостей різних бортових обчислювальних платформ для обробки відеопотоку в режимі реального часу з метою виявлення небезпечних об'єктів та перешкод.

Для досягнення визначеної мети необхідно вирішити такі задачі:

– провести аналіз існуючих розробок у галузі систем дистанційного керування БПЛА, методів обробки відеоданих та апаратних платформ для бортових обчислювальних модулів;

– обґрунтувати вибір підходів та сформувати специфікацію вимог до апаратно-програмного забезпечення системи дистанційного керування БПЛА;

– здійснити проєктування та визначити архітектуру системи з розподіленою обробкою відеоданих;

– виконати моделювання системи та порівняльний аналіз продуктивності апаратних платформ Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B;

– розробити апаратну частину системи та реалізувати бортовий обчислювальний модуль на базі одноплатних комп'ютерів Raspberry Pi;

– розробити програмне забезпечення та реалізувати алгоритми комп'ютерного зору на основі бібліотеки OpenCV;

– провести тестування прототипу апаратно-програмного забезпечення та верифікацію результатів в реальних умовах.

Аналіз сучасного стану проблеми свідчить про те, що більшість існуючих рішень або орієнтовані на використання потужних хмарних платформ, або обмежуються застосуванням дорогого заліза. Водночас питання ефективного

використання доступних одноплатних комп'ютерів для задач комп'ютерного зору в системах керування БПЛА залишається недостатньо дослідженим. Провідні компанії у даній галузі, наприклад DJI, Parrot та ArduPilot, активно розвивають власні рішення. Однак вони переважно орієнтуються на власне апаратне забезпечення.

Основним проєктним рішенням є побудова системи на базі одноплатних комп'ютерів Raspberry Pi з реалізацією алгоритмів комп'ютерного зору засобами бібліотеки OpenCV, що забезпечує обробку відеопотоку в режимі реального часу та формування рекомендацій для оператора.

Отримані результати знайдуть застосування у системах дистанційного керування БПЛА, що використовуються для відеоспостереження чи моніторингу територій. Також розробка буде корисною для пошукових операцій та інших роботів, які мають обмежені обчислювальні потужності.

Таким чином, обране рішення демонструє значимість систем дистанційного керування БПЛА з розподіленою обробкою відеоданих при веденні віддаленого моніторингу та автономного управління. Мета роботи була визначена, задачі – вказані, а об'єкт та предмет – занотовані.

Апробація: Бондаренко О. А. (бакалаврант), Обухова К. О. (ст. викладач каф. комп'ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Система дистанційного керування БПЛА через розподілені ресурси. Міжнародна науково-практична конференція «Інформаційні технології: теорія та практика», Харківський національний університет міського господарства імені О. М. Бекетова [2].

1 СИСТЕМИ ДИСТАНЦІЙНОГО КЕРУВАННЯ БПЛА З РОЗПОДІЛЕНОЮ ОБРОБКОЮ ДАНИХ

1.1 Актуальність дистанційного керування БПЛА та обробки відеоданих у реальному часі

Безпілотні літальні апарати перетворилися з вузькоспеціалізованих військових інструментів на універсальні платформи цивільного та комерційного використання. Сьогодні БПЛА виконують такі завдання: моніторинг інфраструктури, картографування, пошуково-рятувальні операції та охорона певної області. Окремим вектором розвитку стає автоматизація інспекцій у важкодоступних місцях, наприклад висотні лінії електропередач чи нафтогазові магістралі. В цих випадках використання БПЛА буде більш доречним, ніж використання пілотованої авіації є економічно недоцільним. Використання інтелектуальних алгоритмів дозволяє БПЛА не просто транслювати картинку оператору, а самостійно виявляти перешкоду та надсилати повідомлення про те, що відбувається, оператору. Попит на такі системи зростає щороку. Разом з попитом зростають вимоги до швидкості та надійності обробки даних.

Можна зазначити, на сьогодні однією з найбільш критичних проблем при розробці сучасних комплексів дистанційного моніторингу та керування БПЛА є забезпечення мінімальної затримки обробки відеопотоку в метриці реального часу. Часовий інтервал між фіксацією кадру оптичним сенсором борту та виведенням анотованого зображення на екран оператора безпосередньо корелює з безпекою польоту.

Якщо БПЛА DJI Phantom 4 виконує місію у штатному польотному режимі зі швидкістю 14 м/с, то затримка передачі та аналізу даних у 200 мс призводить до втрати контролю над 2,8 метрами траєкторії. У разі прискорення безпілотника, де максимальна швидкість досягає 20 м/с, аналогічний запізній відгук системи збільшує «сліпу» дистанцію вже до 4 метрів. Такий сценарій є критично небезпечним при маневруванні в обмеженому просторі або в умовах щільної забудови. З огляду на це, для безаварійного функціонування апаратно-програмного

забезпечення встановлюється єдине рекомендоване швидкісне обмеження для польоту носія на рівні до 10 м/с. При дотриманні такої швидкості сканування сцени відстань, яку дрон долає за час повного обчислювального циклу аналізу одного кадру, не перевищує 0,6 метра. Це мінімізує вплив інерції апарату, гарантує оператору збереження повного контролю над траєкторією руху та забезпечує стабільне візуальне орієнтування в зоні виконання польотного завдання.

Робота з відеопотоком у реальному часі створює колосальний тиск на апаратну частину. Коли справа доходить до детектування об'єктів – чи то через каскадні класифікатори, чи за допомогою нейромереж – навантаження на процесор зростає в рази, що часто стає «вузьким місцем» для всієї системи. Для досягнення прийнятного показника кількості кадрів за секунду (FPS). Апаратна платформа має забезпечувати стабільну обчислювальну продуктивність навіть в умовах обмеженого енергоспоживання (не менше 15–20 кадрів за секунду).

Саме тут виникає головна суперечність. З одного боку, потужні обчислювальні платформи споживають багато енергії та мають значну масу, що неприпустимо для БПЛА. Збільшення вантажопідйомності дрона для встановлення важких GPU-прискорювачів неминуче призводить до скорочення часу перебування в повітрі, оскільки значна частина енергії витрачається на підйом самої обчислювальної станції. Окрім маси, виникає проблема тепловиділення: в обмеженому закритому корпусі потужні процесори швидко досягають критичних температур. Перегрів викликає троттлінг, як наслідок, різке падіння частоти кадрів у самий відповідальний момент маневру. З іншого боку, легкі та економні рішення, навпаки, часто не справляються з навантаженням у реальному часі. Пошук балансу між продуктивністю та ресурсоємністю є ключовою задачею.

Одним із перспективних підходів є розподілена архітектура обчислень. За нею частина задач виконується безпосередньо на борту апарата, а більш ресурсоємні операції передаються на віддалені сервери. Така модель знижує вимоги до бортового обладнання. При цьому зберігає швидку реакцію системи на події. Зокрема, для стабільної роботи алгоритмів комп'ютерного зору критичним є не лише об'єм переданих даних за секунду, а й показник затримки, який може

призводити до десинхронізації відеопотоку. В умовах реального польоту, де наприклад, відстань між БПЛА та базовою станцією постійно змінюється, а рельєф місцевості може створювати завади, стабільність каналу зв’язку стає вирішальним фактором. Крім того, архітектура системи має передбачати сценарії тимчасової втрати зв'язку. Бортовий модуль повинен автоматично переходити в режим підвищеної автономності для запобігання аварійним ситуаціям.

Серед доступних апаратних платформ для бортових обчислювальних модулів особливе місце займає сімейство одноплатних комп'ютерів Raspberry Pi. Ці пристрої виділяються поєднанням компактних розмірів, низьким енергоспоживанням та задовільною продуктивністю для задач керування [19]. Різниця між поколіннями платформ, зокрема між Raspberry Pi 3 Model B+ та четвертою серією, полягає не лише у частоті процесора, а й у здатності обробляти складні відеопотоки. Саме тому їх порівняння є важливим етапом роботи. Порівнюючи знаходимо баланс між ціною, енергоспоживанням та реальною продуктивністю системи.

Дивлячись на світові тенденції можна підтвердити зростання інтересу до edge-обчислень у роботизованих системах [18]. Наприклад, провідні компанії, зокрема DJI та Parrot, інтегрують у свої платформи власні обчислювальні модулі. Однак більшість виробників роблять ставку на закрите «залізо». На противагу цьому, використання доступних платформ типу Raspberry Pi разом із відкритими бібліотеками OpenCV дозволяє експериментувати з подібними архітектурами. Це дає змогу зменшити витрати на апаратну частину.

Таким чином, запит на недорогі системи відеомоніторингу для дронів випереджає існуючі технічні напрацювання. Дослідження того, як отримати максимальну обчислювальну продуктивність з мінімальними затратами є ключовим завданням. Саме такий підхід оптимізації роботи з відеопотоком на доступних платформах дозволяє значно знизити собівартість кінцевих систем, зберігаючи при цьому їх високу функціональну ефективність.

1.2 Огляд сучасних методів обробки відеоданих у системах керування БПЛА

Сьогодні архітектура обробки відео для БПЛА будується за декількома основними сценаріями. Вибір конкретного методу залежить від того, де саме виконуються обчислення та яке залізо встановлено на борту. Загалом існуючі рішення можна класифікувати за трьома напрямками.

Перший варіант передбачає замикання всього циклу обчислень безпосередньо на борту (англ. Edge Computing). У межах цього підходу на бортовий обчислювач покладаються завдання не лише стабілізації зображення, а й повного циклу детектування об'єктів та прийняття рішень щодо зміни курсу. Це вимагає від архітектури спеціалізованих інструкцій для прискорення матричних обчислень. Основним викликом тут залишається необхідність оптимізації програмного коду. Необхідність полягає в тому щоб забезпечити високий показник FPS при мінімальному тактовому навантаженні на ядра процесора. Головна особливість обчислень на борту є автономність та мінімальні затримки, тому що системі не потрібно чекати відповіді від мережі. Проте реалізація цього методу супроводжується апаратними обмеженнями. Якщо бортовий процесор потужний, він швидше витрачає заряд акумулятора і тим складніше вирішити питання охолодження модуля.

Інший шлях – обробка даних на наземній станції, за якої відеопотік передається на потужний наземний комп'ютер для аналізу. Основним ризиком такої архітектури є пряма залежність керованості БПЛА від затримок у мережі. Навіть при високій пропускній здатності каналу, випадкові втрати пакетів можуть призвести до розриву відеопотоку. У наслідок, це робить неможливим використання складних алгоритмів трекінгу об'єктів у реальному часі. Крім того, передача відео високої роздільної здатності створює додаткове навантаження на радіоелектронне обладнання. Такий підхід дозволяє використовувати значні обчислювальні ресурси, проте вимагає широкої смуги пропускання каналу зв'язку [17].

Третій, компромісний варіант, базується на розподілених обчисленнях [8]. Попередня обробка відеокадрів виконується на борту апарата, тоді як більш ресурсомісткі задачі передаються на віддалений сервер. Це дозволяє значно знизити об'єм трафіку, зберігаючи при цьому можливість використання складних інтелектуальних моделей, які не можуть бути запущені на залізі з обмеженими обчислювальними ресурсами. Цей метод дозволяє отримати високу точність ідентифікації об'єктів, уникаючи при цьому встановлення на борт ресурсомісткого обладнання, яке є критичним для енергетичного балансу дрона.

Для виявлення об'єктів у системах керування БПЛА широко застосовуються деякі алгоритми та інструменти, наприклад: бібліотека OpenCV [7]. Вона є однією із найпоширеніших інструментів для реалізації задач комп'ютерного зору. Бібліотека надає широкий набір функцій для обробки зображень. Приклад реалізації комп’ютерного зору на одноплатному комп’ютеру Raspberry Pi зображений на рис. 1.1.

Рисунок 1.1 – Виявлення об’єктів на Raspberry Pi [4]

Для зображень використовується зміна роздільної здатності, перетворення колірних просторів, виявлення країв та контурів. Перевагою OpenCV є її кросплатформенність та оптимізація для роботи на ресурсообмежених пристроях,

таких як Rasberry Pi. Бібліотека також надає ефективні інструменти для роботи з багатопотоковістю. Інструменти дозволяють раціонально розподіляти навантаження між ядрами процесора, мінімізуючи час обробки кожного кадру.

Також використовуються алгоритми на основі нейронних мереж, зокрема YOLO (You Only Look Once) та MobileNet. Вони забезпечують високу точність виявлення об'єктів у реальному часі. Однак їх досить важко використовувати на маломасштабних апаратних платформах. Таке обмеження відбувається через значні вимоги цих алгоритмів до обчислювальних ресурсів, обсягу оперативної пам'яті та графічного ядра. При спробі запустити важкі нейромережі на простих мікрокомп'ютерах часто спостерігається падіння частоти кадрів та сильний перегрів центрального процесора.

Класичні алгоритми виявлення об'єктів, такі як метод Хаара та Histogram of Oriented Gradients, є менш ресурсомісткими порівняно з нейромережевими підходами. Вони не потребують спеціальних графічних прискорювачів та можуть цілком ефективно використовуватися на доступних одноплатних комп'ютерах для вирішення базових задач виявлення перешкод чи пошуку контурів.

Серед апаратних платформ, що застосовуються в бортових системах БПЛА, найбільшого розповсюдження набули декілька ключових сімейств пристроїв.

Розглянемо Raspberry Pi 3 Model B+ [13], він оснащений чотириядерним процесором ARM Cortex-A53 з тактовою частотою 1,4 ГГц та 1 Гбайт оперативної пам'яті. Платформа підтримує інтерфейс підключення камери CSI та є достатньо поширеною для реалізації базових задач комп'ютерного зору. Проте, можна зазначити що її обчислювальна потужність є обмеженою при обробці відеоданих високої роздільної здатності. Схему розташування основних елементів та інтерфейсів одноплатного комп'ютера Raspberry Pi 3 Model B+ наведено на рис. 1.2.

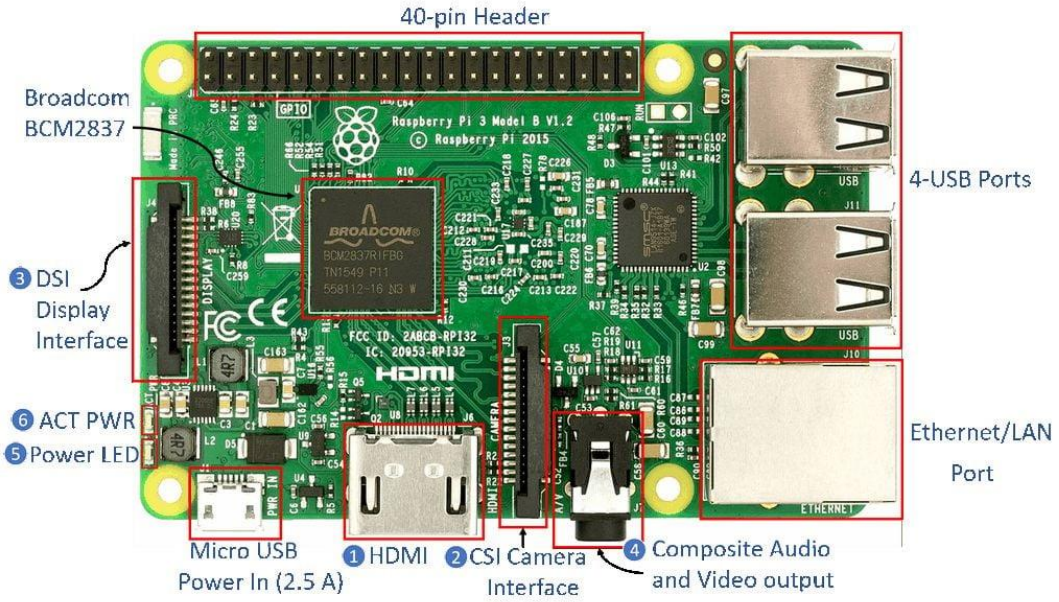

Рисунок 1.2 – Складові Raspberry Pi 3 Model B [16]

В свою чергу, Raspberry Pi 4 Model B використовує процесор ARM Cortex-A72 з тактовою частотою до 1,5 ГГц. На борту має до 8 Гбайт оперативної пам'яті [15]. Порівняно з попередньою моделлю ця платформа забезпечує суттєво вищу продуктивність при виконанні задач обробки відеоданих. Через те, що платформа має новішу архітектуру процесора та покращений контролер пам'яті. Схему розташування основних елементів та інтерфейсів одноплатного комп'ютера Raspberry Pi 4 Model B показано на рис. 1.3.

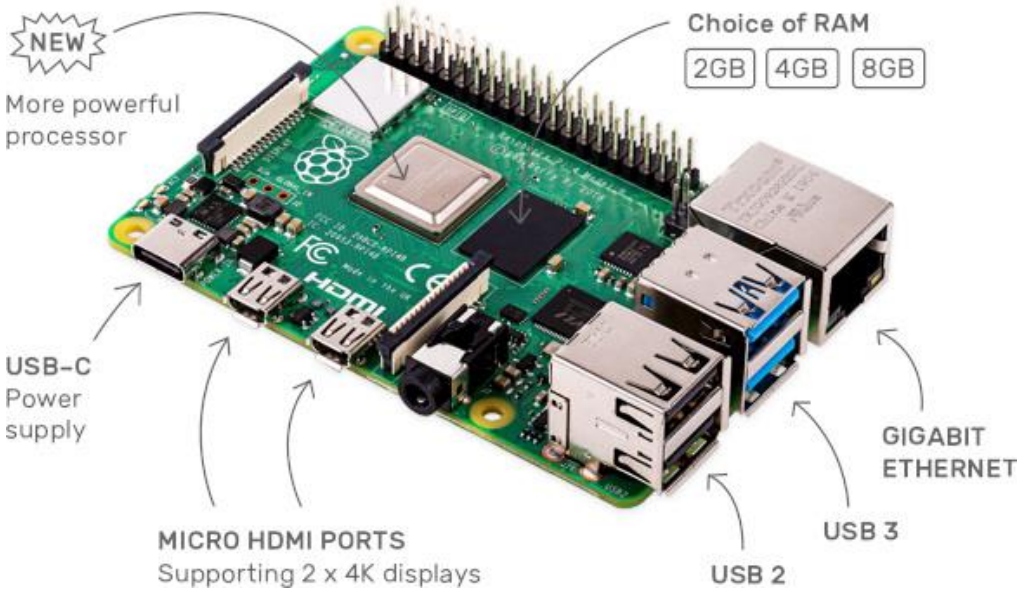

Рисунок 1.3 – Складові Raspberry Pi 4 Model B [14]

У порівнянні с Raspberry Pi 3 Model B+ і Raspberry Pi 4 Model B розглянемо NVIDIA Jetson Nano. Він є спеціалізованою платформою для задач машинного навчання та комп'ютерного зору, оснащеною графічним процесором з 128 ядрами CUDA. Висока апаратна потужність Jetson Nano потребує складнішої системи охолодження та стабільного живлення з високими пусковими струмами, це критично збільшує загальну масу корисного навантаження. Порівняно з Raspberry Pi, ця платформа забезпечує значно вищу продуктивність при виконанні нейромережевих алгоритмів, однак має вищу вартість та більше енергоспоживання, що може не підходити для використання у БПЛА. Виконаємо порівняльний аналіз розглянутих апаратних платформ (табл. 1.1).

Таблиця 1.1 – Порівняння апаратних платформ для бортових обчислювальних модулів

Характеристика	**Raspberry Pi 3B+**	**Raspberry Pi 4 Model B**	**NVIDIA Jetson Nano**
Процесор	ARM Cortex-A53	ARM Cortex-A72	ARM Cortex-A57
Частота процесора, ГГц	1,4	1,5	1,43
Оперативна пам’ять (RAM), Гбайт	1 (LPDDR2)	2 (LPDDR4)	4 (LPDDR4)
Графічний процесор (GPU)	VideoCore IV	VideoCore VI	Maxwell, 128 ядер CUDA
Енергоспоживання, Вт	~5	~7,5	~10
Підтримка камери	CSI (1080p)	CSI (4K)	CSI (4K)
Інтерфейси USB	4 USB 2.0	2 USB 3.0, 2 USB 2.0	4 USB 3.0
Wi-Fi	802.11 b/g/n/ac (2,4/5 ГГц)	802.11 b/g/n/ac (2,4/5 ГГц)	Відсутній
Bluetooth	4.2	5.0	Відсутній
Операційна система	Raspberry Pi OS, Linux	Raspberry Pi OS, Linux	Ubuntu, JetPack SDK
Підтримка OpenCV	Так	Так	Так (з апаратним прискоренням)

На основі проведеного огляду можна зробити висновок, що для реалізації системи дистанційного керування БПЛА з обробкою відеоданих у реальному часі платформи Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B є найбільш оптимальними. Розглядаючи їх з точки зору співвідношення вартості, енергоспоживання та обчислювальної продуктивності. Порівняння їх характеристик при виконанні реальних задач комп'ютерного зору є предметом подальшого дослідження у даній роботі.

1.3 Специфікація вимог до апаратно-програмного забезпечення системи

Апаратне забезпечення у даній роботі представлене БПЛА DJI Phantom 4 [5], який обрано як базову платформу для тестування алгоритмів моніторингу. Завдяки акумулятору, що забезпечує до 28 хвилин польоту, маємо достатній часовий ресурс для проведення серії експериментів без частих зупинок на підзарядку. Хоча конструктивно дрон здатний розвивати швидкість до 72 км/год, у межах нашого дослідження вона була штучно обмежена до 30 км/год – це необхідно для стабільної роботи системи комп’ютерного зору та уникнення розмиття кадрів. Що стосується каналу керування, то його дальність до 5 км повністю покриває потреби випробувань, які проводяться у зоні прямої видимості.

Головною умовою при підборі бортових компонентів стали їх розміри та маса, яка не повинна перевищувати 300 г, щоб не порушувати балансування апарата та не скорочувати суттєво час польоту. З урахуванням цього обмеження в якості бортового обчислювального модуля обрано одноплатні комп'ютери Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B. Обидві платформи відповідають масогабаритним вимогам та забезпечують достатню обчислювальну потужність для виконання задач комп'ютерного зору в реальному часі. Порівняння їх продуктивності при виконанні однакових задач обробки відеоданих є одним із ключових завдань даного дослідження.

Для забезпечення належного теплового режиму роботи бортового модуля під час тривалих польотів передбачено встановлення двох легких вентиляторів

охолодження [3]. Вибір активної системи охолодження зумовлений необхідністю відведення значних теплових потоків від чипів при компактних габаритах бортового модуля (рис. 1.4).

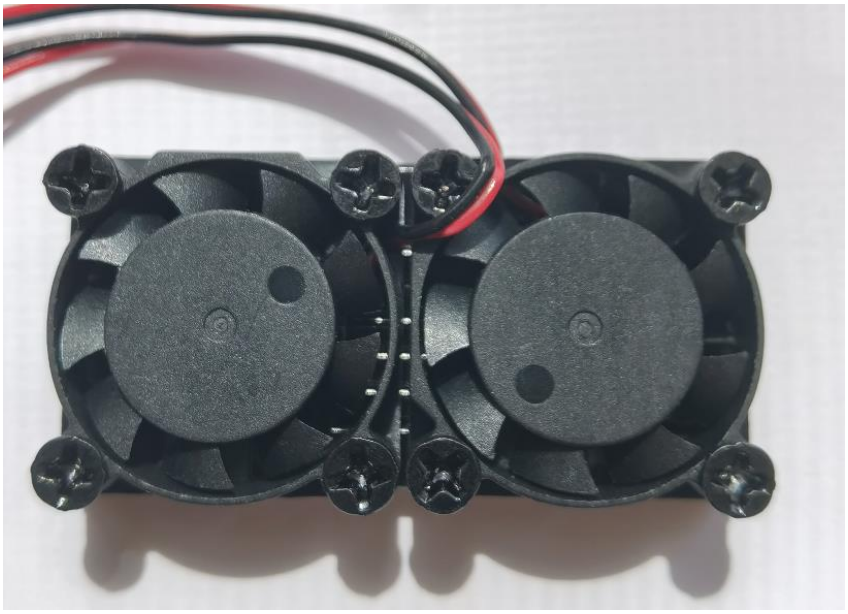

Рисунок 1.4 – Обрана активна система охолодження

Було використано низькопрофільні кулери типорозміру 50 × 25 мм, які живляться від постійного струму напругою 5 В. Завдяки використанню гідродинамічного підшипника, дані вентилятори забезпечують стабільну швидкість обертання та стійкість до вібрацій, що виникають під час роботи двигунів БПЛА. Конструкція радіатора з алюмінієвого сплаву у поєднанні з активним обдувом дозволяє ефективно розподіляти теплову енергію. Це дозволяє підтримувати робочу температуру процесора в межах безпечного діапазону. Навіть під час інтенсивного виконання операцій OpenCV. Це дозволяє запобігти тепловому дроселюванню процесора, яке може негативно вплинути на стабільність та продуктивність системи обробки відеоданих.

Для захоплення відеопотоку використовується камера Raspberry Pi Camera Module Rev 1.3 [12], яка підключається до бортового модуля через інтерфейс CSI (рис. 1.5).

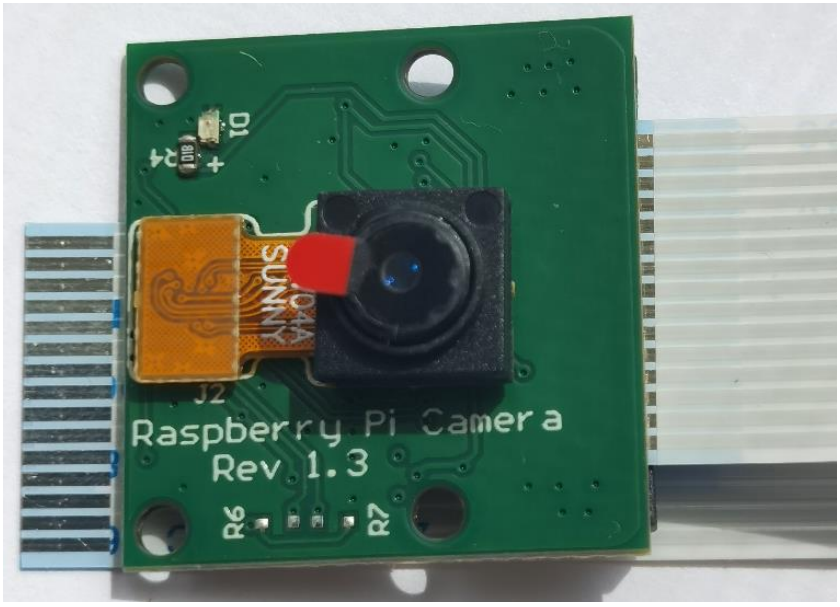

Рисунок 1.5 – Обрана камера Raspberry Pi Camera Module Rev 1.3

Камера забезпечує роздільну здатність до 5 мегапікселів та підтримує відеозйомку у форматі 1080p при частоті 30 FPS або 720p при 60 FPS. Фіксована фокусна відстань об'єктива становить 3,6 мм, кут огляду – 54°. Невелика маса камери, близько 3 г, та компактні розміри роблять її оптимальним вибором для розміщення на борту БПЛА.

Для зберігання операційної системи та програмного забезпечення використовується карта пам'яті microSD об'ємом 32 Гбайт. Карта має відповідати класу швидкості UHS-I (U1) або вище. Це забезпечує мінімальну швидкість запису 10 Мбіт/с, необхідну для стабільної роботи операційної системи та збереження журналів подій у режимі реального часу. Рекомендується використання карток класу A1 або A2, оптимізованих для роботи з випадковими операціями читання та запису, характерними для завантаження та виконання додатків.

Програмне забезпечення (ПЗ) системи розробляється мовою Python 3.8 або вище, яка є стандартною для розробки застосунків комп'ютерного зору на платформах Raspberry Pi. Основним інструментом обробки відеоданих є бібліотека OpenCV версії 4.10.0, яка надає широкий набір функцій для роботи з відеопотоком, включаючи попередню обробку кадрів, виявлення контурів та розпізнавання об'єктів. Для роботи з камерою використовується бібліотека Picamera2, а для роботи з масивами числових даних – бібліотека NumPy. Операційною системою є

Raspberry Pi OS на базі Debian Linux, яка забезпечує повну сумісність із усіма використовуваними бібліотеками та драйверами камери.

Апаратний інтерфейс для з'єднання камери з обчислювальним модулем реалізовано через порт CSI, оскільки такий підхід дозволяє передавати відеопотік безпосередньо до процесора, не перевантажуючи при цьому шину USB. Живлення платформи має свої особливості залежно від версії: для Raspberry Pi 3 Model B+ використовується роз'єм Micro-USB (5В, від 2,5А), тоді як для новішої Raspberry Pi 4 Model B передбачено інтерфейс USB-C при аналогічних електричних параметрах. Це забезпечує стабільну роботу системи навіть під час пікових навантажень при обробці важких нейромережевих моделей.

Програмний інтерфейс та взаємодія між окремими компонентами системи базуються на використанні внутрішніх черг повідомлень мови Python, що гарантує асинхронність та стійкість процесів. Для передачі результатів аналізу з борту на наземну станцію буде задіяно бездротовий канал Wi-Fi. При цьому вибір було зроблено на користь протоколу UDP – такий крок дозволяє суттєво мінімізувати затримки передачі даних, оскільки в умовах реального часу швидкість доставки кожного окремого кадру є пріоритетнішою за гарантовану доставку всіх пакетів, як у випадку з TCP.

Висновки до розділу 1

У першому розділі було проведено системний аналіз предметної області системи дистанційного керування БПЛА з розподіленою обробкою відеоданих у реальному часі. Водночас були сформовані вимоги до апаратно-програмного забезпечення системи.

Дослідження актуальності теми виявило, що головним запитом сучасних БПЛА залишається досягнення мінімального часу відгуку при роботі з відеопотоком. Встановлено, що розподілена модель обробки даних є оптимальним компромісом. Такий підхід забезпечує необхідну продуктивність за рахунок часткового перенесення обчислень на борт, водночас дозволяючи уникнути небажаного збільшення масо-габаритних показників дрона.

Огляд існуючих рішень та апаратних платформ показав, найбільш доцільним вибором для реалізації бортового обчислювального модуля є одноплатні комп'ютери сімейства Raspberry Pi. Raspberry дотримується вимог до маси, розмірів енергоспоживання та низької ціни. Порівняльний аналіз платформ Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B продемонстрував суттєву різницю в їх обчислювальних характеристиках, що обумовлює необхідність практичного дослідження їх продуктивності при виконанні задач комп'ютерного зору. Серед алгоритмів обробки відеоданих було обрано інструменти бібліотеки OpenCV. Вони є оптимальним рішенням для ресурсообмежених платформ.

На основі проведеного аналізу сформовано специфікацію вимог до апаратно-програмного забезпечення системи. Було зазначено, що бортовий обчислювальний модуль на базі Raspberry Pi у поєднанні з камерою Raspberry Pi Camera Module Rev 1.3 відповідає встановленим масогабаритним вимогам для розміщення на БПЛА DJI Phantom 4. ПЗ було реалізовано мовою Python. Також було використані бібліотеки OpenCV та Picamera2. Управління відбувалось під управлінням операційної системи Raspberry Pi OS.

2 ПРОЄКТУВАННЯ СИСТЕМИ ДИСТАНЦІЙНОГО КЕРУВАННЯ БПЛА

2.1 Проєктування архітектури розподіленої системи керування БПЛА

Для побудови системи дистанційного керування обрано розподілену обчислювальну архітектуру. Даний підхід дозволяє гнучко розподілити робочі завдання. Під час роботи алгоритм переносить частину обробки даних на бортовий модуль, тоді як інша частина виконується на наземній станції. Загальну архітектуру системи з позначенням усіх модулів та потоків даних між ними наведено на рис. 2.1.

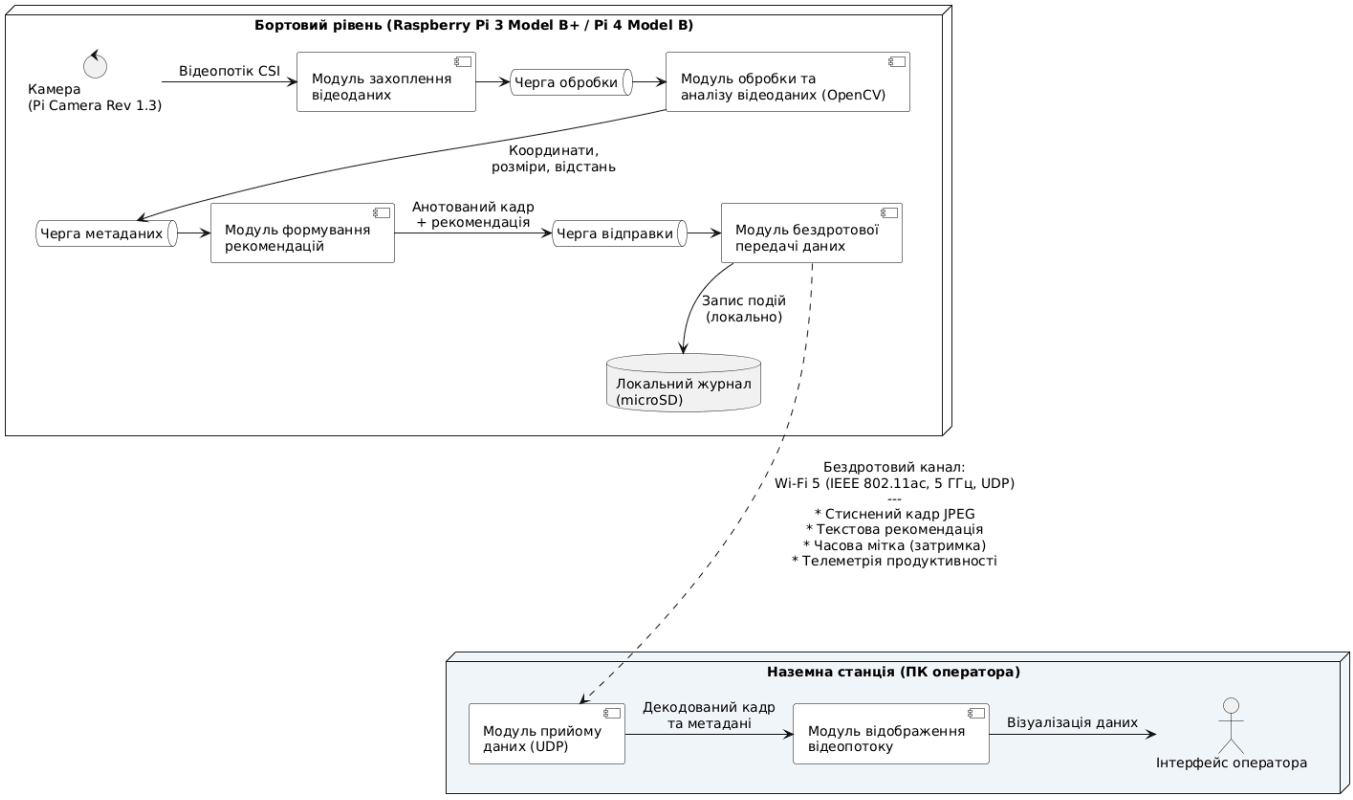

Рисунок 2.1 – Загальна архітектура та потоки даних системи керування БПЛА

В основі проєктування лежить розподіл системи на окремі незалежні модулі. Такий підхід дозволяє провести чітку межу між критично важливими процесами навігації та керування та ресурсомісткими алгоритмами аналізу відеоданих. Ізольована структура компонентів виключає ризик каскадних збоїв. Якщо в роботі модуля комп’ютерного зору станеться критична помилка, польотний контролер все

одно працюватиме в штатному режимі. Це гарантує збереження керування апаратом та безпечне завершення польоту.

Крім того, впровадження модульної структури значно спрощує подальшу модернізацію системи. Це дає змогу інтегрувати нові типи сенсорів, змінювати бібліотеки обробки зображень або впроваджувати складніші алгоритми штучного інтелекту. Важливим аспектом при цьому виступає стандартизація інтерфейсів обміну даними між модулями. Таке архітектурне рішення закладає фундамент для створення відмовостійкого АПЗ, здатного функціонувати в умовах обмеженої пропускної здатності каналів зв’язку.

Система складається з двох основних рівнів. Бортовий рівень реалізується на базі одноплатного комп'ютера Raspberry Pi. Схему підключення компонентів бортового модуля наведено на рис 2.2.

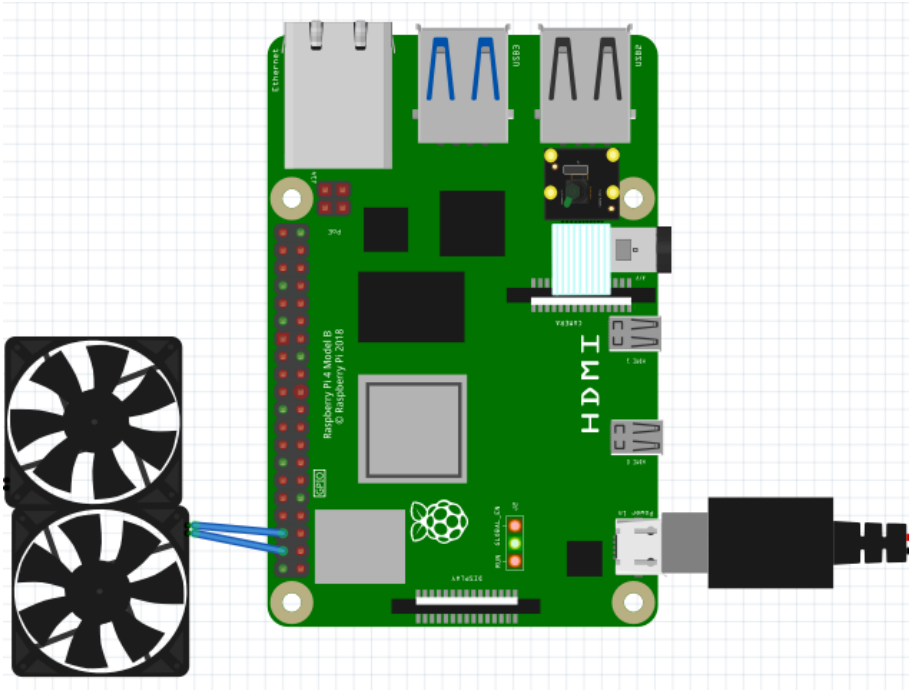

Рисунок 2.2 – Схема підключення Raspberry Pi 4 Module B
з використанням блоку живлення

Рівень включає модуль захоплення відеоданих, модуль обробки та аналізу відеоданих, модуль формування рекомендацій та модуль бездротової передачі даних. Рівень наземної станції включає: модуль прийому даних, модуль

відображення відеопотоку з результатами аналізу та інтерфейс оператора для прийняття керуючих рішень.

Такий розподіл функцій між рівнями є обґрунтованим з точки зору мінімізації затримок у прийнятті рішень. Виконання базових операцій комп'ютерного зору безпосередньо на борту апарата дозволяє уникнути затримок, які пов'язані з передачею необробленого відеопотоку на наземну станцію. На наземну станцію керування відправляються виключно результати обробки. Ними є анотований відеокадр та супутня текстова рекомендація, загальний інформаційний обсяг яких у рази менший порівняно з трансляцією нестиснутого відеопотоку.

Бортовий рівень системи побудований за модульним принципом і складається з чотирьох взаємопов'язаних модулів. Блоки функціонують асинхронно на основі черг повідомлень Python.

Модуль захоплення відеоданих відповідає за безперервне отримання кадрів із камери Raspberry Pi Camera Module Rev 1.3 через інтерфейс CSI та їх передачу до черги обробки. Модуль реалізований у окремому потоці виконання, що забезпечує незалежність процесу захоплення від швидкості обробки кадрів. Якщо черга обробки переповнюється, модуль автоматично видаляє найстаріші кадри з буфера. Такий механізм гарантує, що до блоку аналізу завжди потрапляє лише актуальна інформація.

Модуль обробки та аналізу відеоданих є центральним компонентом бортового рівня. Він отримує кадри з черги захоплення, виконує їх попередню обробку засобами OpenCV та застосовує алгоритми виявлення об'єктів [20]. Результатом роботи модуля є анотований кадр із нанесеними рамками навколо виявлених об'єктів та метадані про кожен виявлений об'єкт. На кадрі можна побачити його координати, розміри та оцінку відстані.

Модуль формування рекомендацій отримує метадані від модуля аналізу та на їх основі генерує текстову рекомендацію для оператора. Логіка формування рекомендацій базується на аналізі положення виявлених об'єктів у кадрі та їх відносних розмірів, які використовуються як непрямий показник відстані до

перешкоди. Сформована рекомендація разом із анотованим кадром передається до черги відправки.

Модуль бездротової передачі даних відповідає за відправку оброблених даних на наземну станцію через канал Wi-Fi 5 за протоколом UDP [9]. Модуль реалізований у окремому потоці виконання і забезпечує мінімальні затримки передачі пакетів. Це досягається за рахунок повної відмови від складного механізму підтвердження доставки кожного надісланого пакета, який є обов'язковим для протоколу TCP. Протоколу UDP не потрібно витрачати час на очікування відповідей від наземної станції, перевірку черговості пакетів чи повторне надсилання втрачених у повітрі елементів. Бортовий модуль безперервно транслює свіжі дані у бездротовий канал із максимально можливою швидкістю. Паралельно модуль виконує запис усіх подій до локального журналу на карті пам'яті microSD.

Рівень наземної станції реалізується на окремому комп'ютері оператора та включає два основних компоненти. Модуль прийому даних забезпечує отримання UDP-пакетів від бортового модуля, їх декодування та передачу до модуля відображення. Модуль відображення забезпечує візуалізацію отриманого відеопотоку з накладеними результатами аналізу та текстовою рекомендацією у вікні інтерфейсу оператора.

Інтерфейс оператора відображає поточний анотований відеокадр із виділеними рамками навколо виявлених об'єктів, текстову рекомендацію щодо подальших дій БПЛА, а також поточні показники продуктивності системи – частоту кадрів та рівень завантаження процесора бортового модуля. Це дозволяє оператору контролювати не лише стан навколишнього середовища, а й поточний стан самої системи обробки даних. Для формалізації функціональних вимог та відображення сценаріїв взаємодії оператора з апаратно-програмним комплексом розроблено UML-діаграму варіантів використання (рис. 2.3).

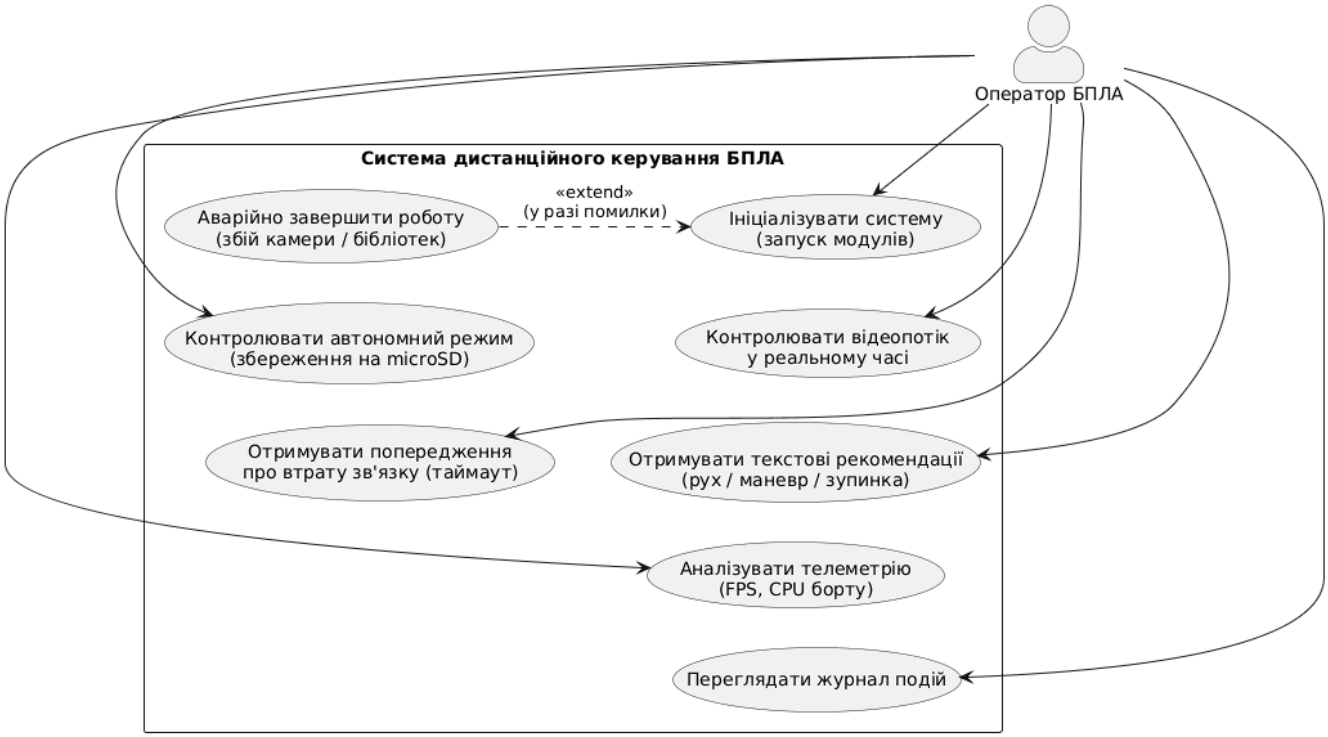

Рисунок 2.3 – Діаграма варіантів використання системи керування БПЛА

Взаємодія між бортовим рівнем та рівнем наземної станції здійснюється через бездротовий канал Wi-Fi 5. Для передачі даних використовується протокол UDP, що забезпечує мінімальні затримки за рахунок відсутності механізму встановлення з'єднання та підтвердження доставки пакетів. Кожен пакет даних містить стиснений відеокадр у форматі JPEG, текстову рекомендацію, часову мітку формування пакету та поточні показники продуктивності бортового модуля.

Часова мітка у кожному пакеті дозволяє наземній станції обчислювати фактичну затримку передачі даних та відображати її оператору. Це є важливою характеристикою системи, оскільки надмірна затримка може свідчити про погіршення якості каналу зв'язку або перевантаження бортового обчислювального модуля. У разі відсутності пакетів від бортового модуля протягом встановленого часового порогу наземна станція відображає оператору відповідне попередження про втрату зв'язку.

2.2 Алгоритм роботи системи обробки відеоданих у реальному часі

Розробка алгоритму роботи системи є ключовим етапом проєктування. Саме розробка визначає логіку взаємодії між усіма компонентами апаратно-програмного забезпечення (АПЗ). Відповідно до покрокового підходу, на першому етапі формується загальна схема вирішення задачі, а на другому деталізується кожен її блок. Блок-схему наведено на рис. 2.4.

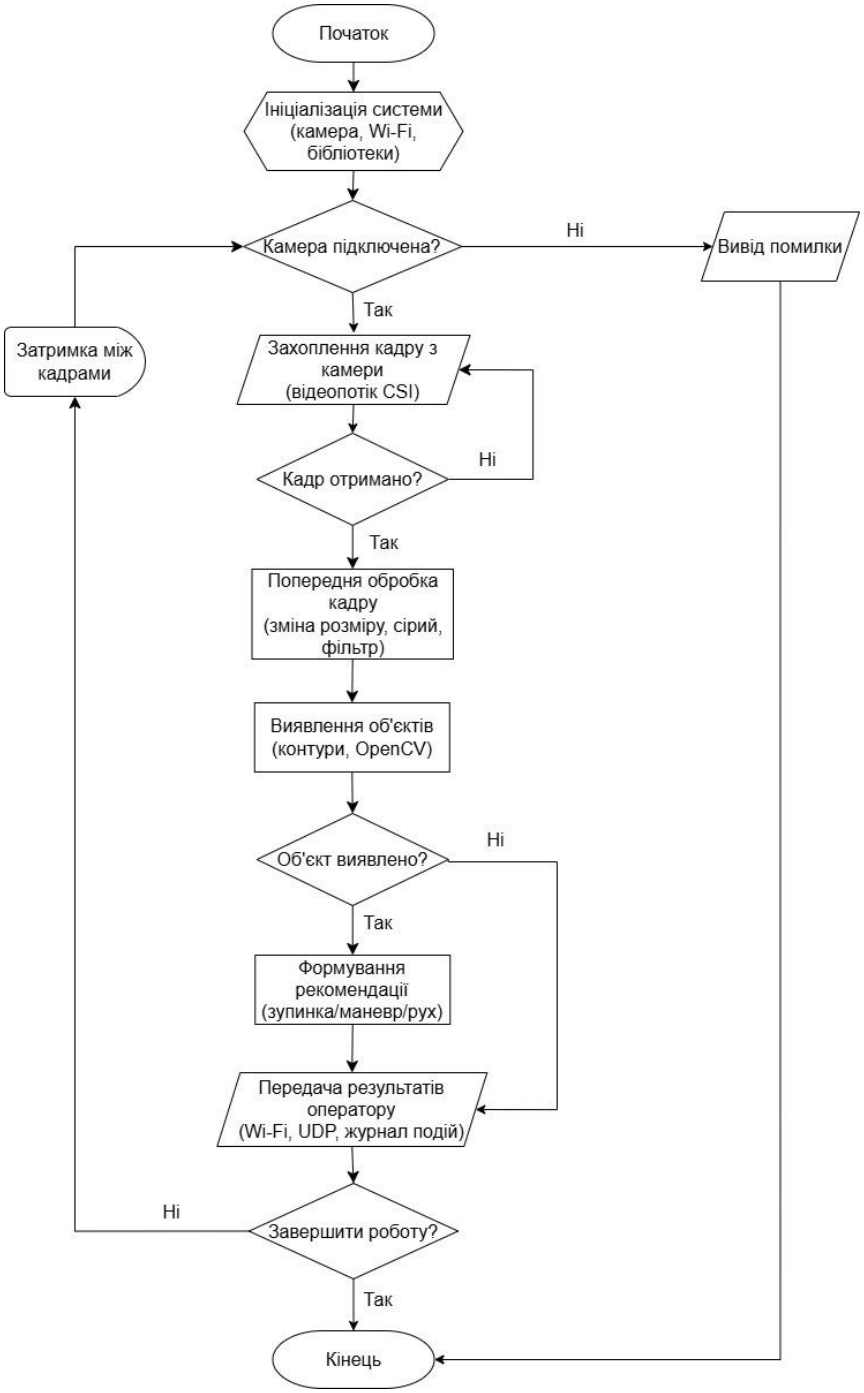

Рисунок 2.4 – Загальна блок-схема алгоритму роботи системи

Перед побудовою алгоритму необхідно чітко визначити вхідні та вихідні дані системи. Вхідними даними є безперервний відеопотік із бортової камери Raspberry Pi Camera Module Rev 1.3, яка підключена через інтерфейс CSI. Вона передає послідовність кадрів із частотою до 30 FPS при роздільній здатності 1080p або 60 FPS при 720p. В свою чергу, вихідними даними системи є оброблений відеокадр із нанесеними позначеннями виявлених об'єктів та текстова рекомендація для оператора. Обмін даними з наземною станцією керування реалізується через бездротовий канал стандарту Wi-Fi 5 (IEEE 802.11ac). Використання частотного діапазону 5 ГГц дозволяє підтримувати стабільну швидкість трансляції стиснених кадрів, що мінімізує затримки під час дистанційного моніторингу БПЛА [10].

Розроблений алгоритм базується на класичному підході комп'ютерного зору з використанням бібліотеки OpenCV, а не на нейромережевих методах. Таке рішення зумовлено ресурсними обмеженнями платформ Raspberry Pi. Можна з упевненістю сказати, що нейромережеві підходи, зокрема YOLO або MobileNet, забезпечують вищу точність виявлення об'єктів, але вимагають значно більших обчислювальних ресурсів. Нейромережеві підходи не здатні забезпечити прийнятний показник FPS на платформі Raspberry Pi 3 Model B+. Класичні алгоритми OpenCV дозволяють досягти необхідної швидкодії навіть на найбільш обмежених платформах. Саме тому ці методи обрано як найбільш оптимальний інструмент для порівняльного дослідження продуктивності двох апаратних платформ в однакових умовах.

Окремої уваги заслуговує питання багатопотоковості алгоритму. Оптимальний розподіл ресурсів на обох обчислювальних модулях забезпечується завдяки асинхронній організації робочого циклу. Програмне рішення використовує вбудовані черги повідомлень Python, що дозволяє уникнути затримок під час паралельної обробки кадрів. Це означає, що захоплення кадру, його обробка та передача результатів виконуються у окремих потоках виконання. Таке рішення дозволяє уникнути простою процесора під час операцій введення-виведення. Використання асинхронної організації робочого циклу є особливо важливим для

Raspberry Pi 3 Model B+, де обчислювальні ресурси є більш обмеженими, і кожен такт процесора має бути використаний максимально ефективно.

Слід також зазначити, що алгоритм розроблено з урахуванням принципу масштабованості. Для зручності налаштування всі базові параметри обробки відео винесені в окремий конфігураційний файл. Це стосується робочої роздільної здатності кадру, порогів для виявлення контурів та мінімальних розмірів перешкод. Такий підхід дозволяє швидко змінювати конфігурацію системи без редагування та перекомпіляції основного коду програми. Це дозволяє адаптувати систему до різних умов польоту та різних типів перешкод без необхідності повторної розробки програмного забезпечення.

На найвищому рівні абстракції система працює за таким циклом: бортова камера безперервно передає відеопотік до обчислювального модуля, який обробляє кадри в режимі реального часу, виявляє потенційно небезпечні об'єкти або перешкоди та формує рекомендації для оператора. Оператор отримує рекомендацію та приймає остаточне рішення щодо подальших дій БПЛА [11]. Цей цикл повторюється безперервно протягом усього польоту.

Загальну схему роботи системи можна представити у вигляді таких послідовних етапів (рис. 2.4):

- ініціалізація системи та підключення камери;
- захоплення відеокадру;
- попередня обробка кадру;
- виявлення об'єктів;
- формування рекомендації;
- передача результатів оператору;
- повернення до захоплення наступного кадру.

Процес запуску програмного комплексу починається з **етапу комплексної ініціалізації**, під час якого виконується послідовне налаштування всіх апаратних та програмних компонентів. Спочатку система здійснює низькорівневе підключення модуля камери через інтегрований інтерфейс CSI. Безпосередньо після встановлення зв'язку із сенсором відбувається конфігурування базових

параметрів відеозйомки, серед яких ключовими є робоча роздільна здатність кадру та цільова частота дискретизації (FPS). Паралельно з цим у оперативну пам'ять завантажуються необхідні модулі бібліотеки OpenCV. Після цього ініціалізується мережевий сокет для майбутньої передачі даних через бездротовий канал Wi-Fi за протоколом UDP.

Важливою архітектурною особливістю цього етапу є вбудований механізм обробки виняткових ситуацій. Якщо на етапі старту виявляється відсутність фізичного підключення камери, збій ініціалізації інтерфейсу або помилка створення мережевого сокету, система не переходить у режим незавершеного очікування. Замість цього алгоритм коректно перериває роботу, генерує відповідне інформаційне повідомлення про помилку у системний лог (або консоль) та виконує безпечне закриття всіх відкритих дескрипторів і вивільнення зайнятих ресурсів пам'яті.

Етап ініціалізації є критичним з точки зору надійності всієї системи, оскільки саме тут закладається фундамент для коректної роботи всіх подальших модулів. Розглянемо детальніше кожен крок цього етапу.

Підключення камери через інтерфейс CSI виконується за допомогою бібліотеки Picamera2. Вона є стандартним інструментом для роботи з камерами на платформах Raspberry Pi під управлінням операційної системи Raspberry Pi OS. На цьому кроці система перевіряє наявність підключеного камерного модуля та ініціалізує об'єкт камери з відповідними налаштуваннями. Якщо камера не виявлена або її ініціалізація завершилась з помилкою, система записує відповідне повідомлення у журнал подій та завершує роботу з кодом помилки.

Налаштування параметрів відеозйомки передбачає встановлення роздільної здатності захоплення, частоти кадрів та формату зображення. У рамках даного дослідження використовуються два режими роботи: 1280×720 пікселів при 30 FPS та 640×480 пікселів при 30 FPS. Це дозволяє дослідити вплив роздільної здатності вхідного відеопотоку на загальну продуктивність системи на кожній із апаратних платформ. Формат зображення встановлюється як RGB888. Завдяки цьому знімки

безпосередньо передаються у функції бібліотеки OpenCV, уникаючи зайвих операцій з конвертації просторів на етапі первинного захоплення.

Завантаження необхідних бібліотек виконується на старті програми і включає імпорт модулів OpenCV, NumPy, Picamera2 та стандартних модулів Python. Данні модулі необхідні для роботи з потоками виконання, чергами повідомлень та мережевими з'єднаннями. На цьому ж кроці виконується перевірка версій встановлених бібліотек на відповідність мінімальним вимогам – OpenCV версії 4.10.0 та Python версії 3.14.

Ініціалізація каналу передачі даних через Wi-Fi передбачає створення UDP-сокету та встановлення з'єднання з наземною станцією керування. Система виконує перевірку доступності мережі та збереження параметрів з'єднання – IP-адреси та порту наземної станції – у конфігураційному файлі. У разі недоступності мережі система переходить у автономний режим роботи, за якого результати обробки відеоданих зберігаються у локальному журналі, який не передається на наземну станцію. Це забезпечує безперервність роботи алгоритму комп'ютерного зору навіть при тимчасовій втраті зв'язку.

Паралельно з основними кроками ініціалізації система виконує налаштування параметрів журналювання подій. Програма автоматично створює окремий файл у директорії проєкта, де будуть накопичуватися звітні матеріали. Сформований журнал містить детальні часові мітки, повідомлення про помилки в роботі обладнання, а також поточні показники загальної продуктивності. Записи журналу зберігаються у текстовому файлі на карті пам'яті microSD. Такий спосіб збереження даних у вигляді простого тексту є дуже зручним, оскільки дозволяє відкрити журнал на будь-якому комп'ютері або ноутбуці після завершення тестування. Це дає можливість детально оцінити стабільність усієї системи в цілому. Деталізовану блок-схему алгоритму етапу ініціалізації та конфігурації системи наведено на рис. 2.5.

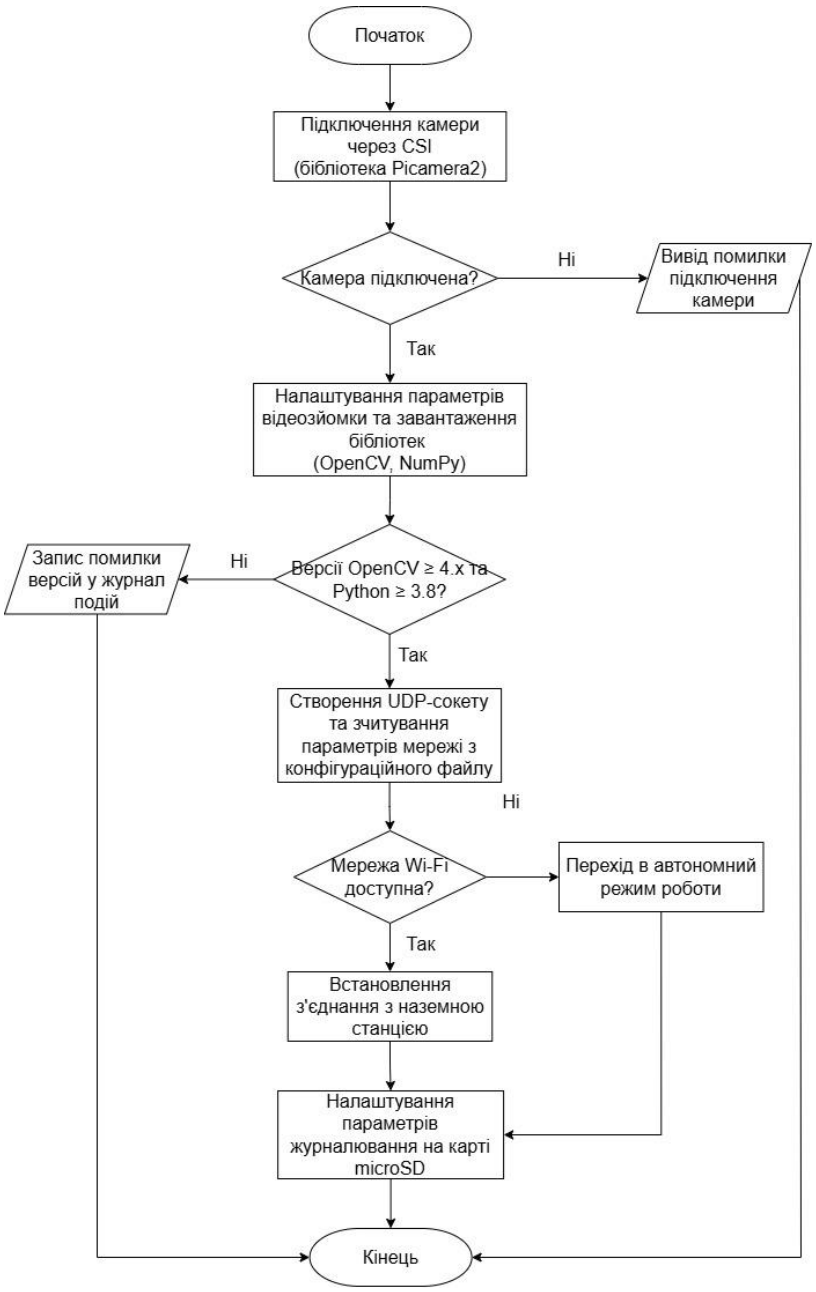

Рисунок 2.5 – Блок-схема алгоритму етапу ініціалізації системи

Етап захоплення відеокадру передбачає отримання поточного кадру з відеопотоку камери. Якщо кадр не було отримано коректно, система пропускає поточний кадр і переходить до спроби захоплення наступного кадру. Це дозволяє уникнути зупинки всього циклу обробки через поодинокі помилки захоплення.

Попередня обробка кадру є критично важливим етапом для забезпечення необхідної продуктивності на ресурсообмежених платформах. На цьому етапі виконуються такі операції:

– зменшення роздільної здатності кадру до робочого розміру 640 × 480 пікселів, що суттєво знижує обчислювальне навантаження;

– перетворення кольорового зображення у відтінки сірого для спрощення подальшої обробки;

– застосування фільтра розмиття для зниження рівня шуму на зображенні.

На етапі виявлення об'єктів оброблений кадр аналізується алгоритмами комп'ютерного зору. Система виконує виявлення контурів потенційних перешкод, обчислення їх розмірів та положення у кадрі. Також важливою є оцінка відстані до виявленого об'єкта на основі його розміру у пікселях. Якщо розмір виявленого контуру перевищує встановлений пороговий рівень, об'єкт класифікується як потенційна перешкода і навколо нього відображається обмежувальна рамка.

Етап формування рекомендації базується на результатах виявлення об'єктів. Система аналізує положення виявленої перешкоди у кадрі та її відносний розмір і формує одну рекомендацій для оператора, наприклад: продовжувати рух (перешкод не виявлено), знизити швидкість, змінити напрямок руху, зупинитись якщо перешкода критично близько.

На фінальному етапі сформована рекомендація разом із поточним кадром із нанесеними позначеннями виявлених об'єктів. Рекомендація передається на наземну станцію оператора через канал Wi-Fi, з використанням протоколу UDP. Паралельно інформація про виявлені події зберігається у локальному журналі з відповідною часовою міткою.

Алгоритм функціонує за циклічним принципом у постійному режимі. Потокова обробка кадрів виконується без зупинок аж до надходження команди на вимкнення від наземної станції або у разі реєстрації критичної помилки апаратного забезпечення. При цьому наскрізний процес обробки відеопотоку та генерації команд відбувається з певною часовою затримкою, сумарна величина якої прямо залежить від обчислювальної потужності обраної моделі одноплатного комп'ютера. Для відображення розподілу обчислювальних задач, логіки циклічного функціонування та потоків даних між апаратним рівнем БПЛА, бортовим

комп'ютером та наземною станцією розроблено UML-діаграму діяльності (рис. 2.6).

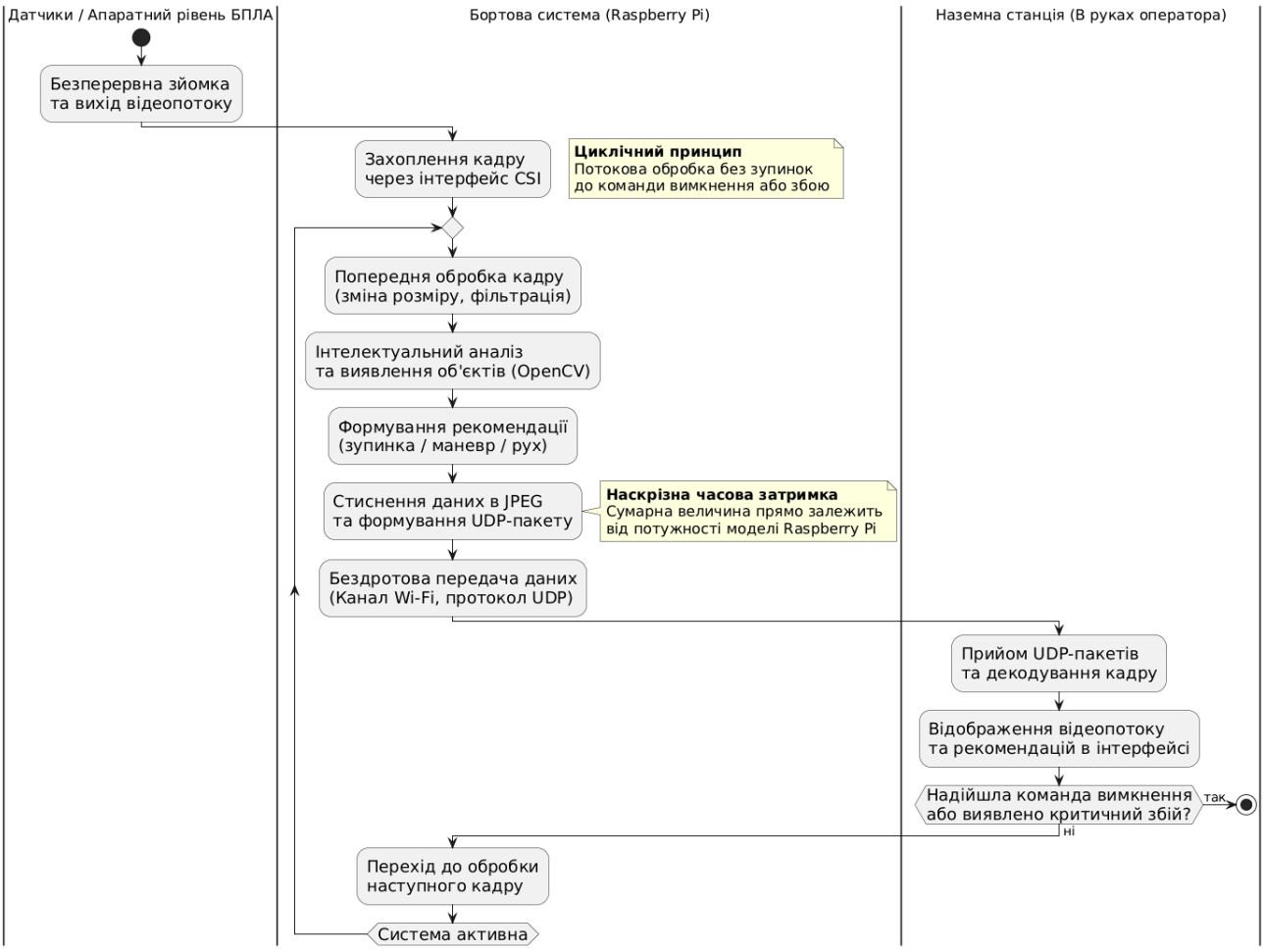

Рисунок 2.6 – UML-діаграма розподілу задач та потоків даних між рівнями системи

Потокова обробка кадрів виконується без зупинок аж до надходження команди на вимкнення від наземної станції або у разі реєстрації критичної помилки апаратного забезпечення. FPS є ключовим показником ефективності алгоритму і безпосередньо залежить від обчислювальних можливостей апаратної платформи. Саме тому порівняльне дослідження продуктивності алгоритму на платформах Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B є одним із центральних завдань даного дослідження.

2.3 Вибір технологій та компонентів апаратно-програмного забезпечення

Вибір технологій та компонентів є важливим етапом проєктування системи, оскільки від нього безпосередньо залежить продуктивність, надійність та масштабованість розробленого рішення. У даному підрозділі обґрунтовується вибір кожного апаратного та програмного компонента системи з урахуванням сформованих у першому розділі вимог до АПЗ.

Центральним апаратним компонентом бортового рівня системи є одноплатний комп'ютер. Для дослідницької роботи було обрано Raspberry Pi 4 Module B (рис. 2.7).

Рисунок 2.7 – Одноплатний комп'ютер Raspberry Pi 4 Module B

Як було обґрунтовано у першому розділі, для цього дослідження обрано дві платформи: Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B. Вибір платформ було зроблено з метою порівняння їх продуктивності при виконанні задач комп'ютерного зору в реальному часі. Обидві платформи задовольняють масогабаритні вимоги для розміщення на борту БПЛА DJI Phantom 4 та підтримують усі необхідні інтерфейси підключення периферійних пристроїв. Також для дослідницької роботи було залучено Raspberry Pi 3 Module B+ (рис. 2.8).

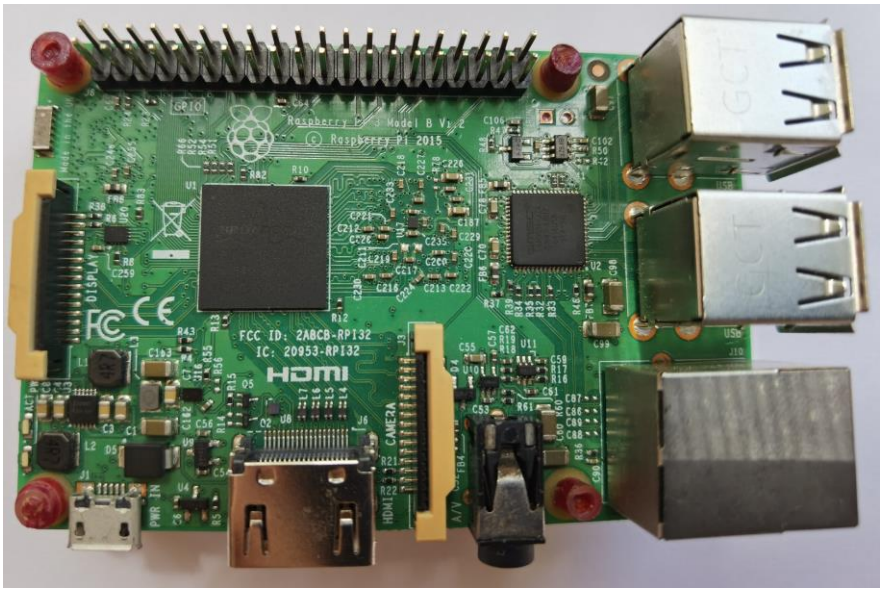

Рисунок 2.8 – Одноплатний комп'ютер Raspberry Pi 3 Module B+

Вибір саме цих двох платформ для порівняльного дослідження обумовлений кількома факторами. По-перше, між ними існує суттєва різниця в архітектурі процесора – ARM Cortex-A53 у Raspberry Pi 3 Model B+ проти ARM Cortex-A72 у Raspberry Pi 4 Model B. По-друге, обидві платформи є широко доступними та економічно обґрунтованими рішеннями порівняно зі спеціалізованими обчислювальними модулями на кшталт NVIDIA Jetson Nano. По-третє, обидві платформи мають однаковий набір інтерфейсів підключення, що дозволяє використовувати ідентичну апаратну конфігурацію при проведенні порівняльних експериментів.

Для захоплення відеопотоку обрано камеру Raspberry Pi Camera Module Rev 1.3. Ключовою перевагою цього модуля є підключення через інтерфейс CSI, який забезпечує пряму передачу відеоданих до процесора з мінімальними затримками та без додаткового навантаження на шину USB. Альтернативою могло б бути використання USB-камери, однак такий варіант є менш доцільним через вищі затримки передачі даних та більше енергоспоживання. Характеристики обраної камери – роздільна здатність до 5 Мп, підтримка 1080p при 30 FPS – повністю відповідають вимогам системи.

Для забезпечення теплового режиму роботи бортового модуля обрано два легких вентилятори охолодження загальною масою не більше 20 г. Необхідність

активного охолодження обумовлена специфікою тривалих польотів, під час яких процесор бортового модуля працює під постійним навантаженням при обробці відеоданих. Без належного охолодження процесор може перейти у режим теплового дроселювання. Тротлінг призведе до зниження тактової частоти і, як наслідок, до падіння продуктивності системи. Це особливо критично для Raspberry Pi 4 Model B, процесор якої має вищу теплову розсіювальну потужність порівняно з Raspberry Pi 3 Model B+.

Для зберігання операційної системи та програмного забезпечення обрано карту пам'яті microSD класу A1 об'ємом 32 Гбайт. Клас A1 гарантує мінімальну швидкість читання 1500 IOPS та запису 500 IOPS. Об'єм 32 Гбайт є достатнім для розміщення операційної системи, усіх необхідних бібліотек та програмного забезпечення системи з запасом для зберігання журналів тривалих експериментів.

Модуль XL4015 є регульованим імпульсним стабілізатором напруги типу «понижувальний» (англ. step-down / buck), здатним ефективно знижувати вхідну напругу до потрібного рівня (рис. 2.9) [6].

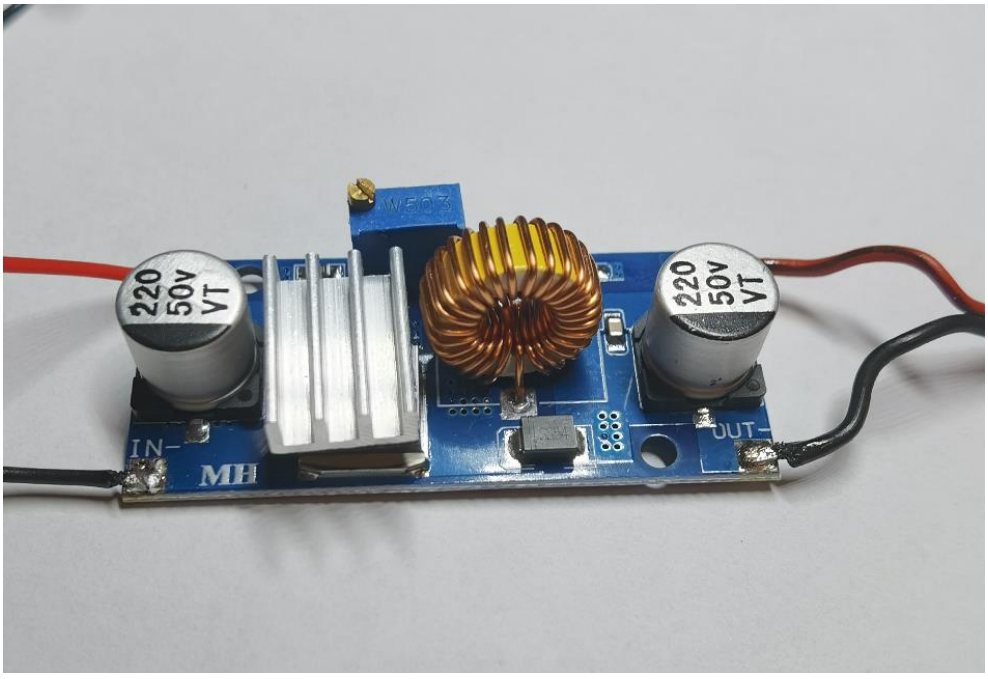

Рисунок 2.9 – Модуль XL4015 понижувальний Uin 8–36V to 1,25–32

Модуль можливо використовувати для безпечної передачі напруги з літій-іонних батарей до роз'єму живлення плати без ризику перегріти або пошкодити чутливі компоненти мікрокомп'ютера. XL4015 дуже зручний у ситуаціях, коли джерело живлення має вищу напругу, ніж потрібно для системи.

У той час як Raspberry Pi потребує жорстко стабілізованих 5,0–5,2 В, XL4015 автоматично утримує вихідну напругу на заданому рівні, незалежно від того, наскільки сильно заряджені або розряджені батареї на вході. В умовах постійної обробки відеопотоку з камери та роботи алгоритмів комп'ютерного зору OpenCV, просідання напруги навіть на 0,3 В може призвести до збоїв у коді, зависання процесора або раптового вимкнення системи. Модуль виступає як надійний буфер та регулятор, що забезпечує безперервну подачу енергії. До того ж, він має вбудований гвинтовий потенціометр, що дозволяє точно налаштувати вихідну напругу, хоча для первинного калібрування все одно потребує використання мультиметра.

Доречно використання двох послідовно з’єднаних літій-іонних елементів живлення типорозміру INR18650-35E, які забезпечують номінальну напругу близько 7,4 В (повна зарядка – до 8,4 В, розрядження – близько 6,0 В) [1]. Зображення батареї, розміщені в тримачах, наведено на рис. 2.10.

Рисунок 2.10 – Батареї INR18650-35E

Така конфігурація дозволяє забезпечити значну автономність роботи бортового модуля на БПЛА, а також створює оптимальні умови для стабільної роботи XL4015, який завжди має необхідний запас вхідної напруги для ефективного перетворення з високим ККД.

Фінальне підключення виконано через інтерфейсну плату з відповідним роз'ємом «тато», що підключається у штатне гніздо живлення Raspberry Pi. Такий підхід є максимально безпечним, оскільки подача енергії через фабричний USB-порт, задіює внутрішні захисні каскади мікрокомп'ютера. Це повністю виключає ризик випадкової переполюсовки. Використання елементів 18650-35E є цілком доцільним у мобільних системах через їх високу енергетичну щільність, низьке саморозрядження, доступність та здатність стабільно віддавати необхідний струм (до 3 А) під навантаженням.

Мовою розробки програмного забезпечення системи обрано Python версії 3.14. Вибір Python обумовлений кількома вагомими факторами. По-перше, Python є стандартною мовою розробки для платформ Raspberry Pi і має найкращу підтримку з боку виробника та спільноти розробників. По-друге, Python надає зручний інтерфейс до бібліотеки OpenCV та інших необхідних інструментів. По-третє, Python підтримує багатопотоковість та асинхронне програмування через модулі threading та queue, що є необхідним для реалізації асинхронної архітектури системи. Альтернативою могла б бути мова C++. Це рішення забезпечувало би вищу продуктивність виконання, однак значно ускладнювало розробку та відлагодження програмного забезпечення.

Основним інструментом обробки відеоданих обрано бібліотеку OpenCV. OpenCV є найпоширенішою бібліотекою комп'ютерного зору з відкритим вихідним кодом. Бібліотека надає широкий набір оптимізованих функцій для обробки зображень та відеоданих. Ключовою перевагою OpenCV є її оптимізація для роботи на платформах ARM, що дозволяє максимально використовувати обчислювальні можливості процесорів Raspberry Pi [21]. Бібліотека підтримує апаратне прискорення через NEON SIMD-інструкції архітектури ARM. Завдяки інструкціям

суттєво підвищує продуктивність операцій обробки зображень порівняно зі звичайною скалярною реалізацією.

Для роботи з камерою обрано бібліотеку Picamera2, яка є офіційною бібліотекою для роботи з камерними модулями Raspberry Pi під управлінням нових версій операційної системи Raspberry Pi OS. Picamera2 забезпечує низькорівневий доступ до апаратних можливостей камери та підтримує захоплення відеопотоку у форматах, сумісних з OpenCV, без необхідності додаткового перетворення даних. Попередня бібліотека Picamera більше не підтримується виробником, тому використання Picamera2 є актуальним рішенням для нових розробок.

Для роботи з масивами числових даних при обробці зображень обрано бібліотеку NumPy. Зображення в OpenCV представлені у вигляді багатовимірних масивів NumPy, тому ця бібліотека є невід'ємною частиною будь-якої системи комп'ютерного зору на Python. NumPy забезпечує ефективні операції над масивами даних з використанням векторизованих обчислень, що суттєво підвищує продуктивність порівняно зі звичайними циклами Python.

Операційною системою для обох апаратних платформ обрано Raspberry Pi OS на базі Debian Linux. Ця операційна система є офіційною для платформ Raspberry Pi та забезпечує повну апаратну підтримку, включаючи оптимізовані драйвери для процесора, камерного модуля та бездротових інтерфейсів. Raspberry Pi OS надає всі необхідні інструменти для встановлення та налаштування використовуваних бібліотек через стандартний менеджер пакетів apt та інструмент pip для пакетів Python.

Для обґрунтування зроблених виборів доцільно порівняти обрані технології з основними альтернативами. У таблиці 2.1 наведено порівняння ключових програмних компонентів системи з їх альтернативами за основними критеріями.

Таблиця 2.1 – Порівняння програмних компонентів з альтернативами

Компонент	**Обраний варіант**	**Альтернатива**	**Причина вибору**
Мова програмування	Python 3.14	C++	Простота розробки, підтримка Raspberry Pi, інтеграція з OpenCV

Бібліотека CV	OpenCV	TensorFlow Lite	Менші вимоги до ресурсів, оптимізація для ARM
Робота з камерою	Picamera2	USB-камера з використанням OpenCV	Нижча затримка через CSI, офіційна підтримка
Протокол передачі	UDP	TCP	Мінімальна затримка, відсутність підтвердження доставки
Операційна система	Raspberry Pi OS	Ubuntu Server	Повна апаратна підтримка, офіційні драйвери

Комплекс обраних рішень АПЗ є оптимальним для виконання поставлених технічних вимог. Сформована конфігурація обладнання та бібліотек дозволяє системі дистанційного керування працювати без критичних затримок. При цьому обробка та аналіз кадрів стабільно функціонують навіть у рамках суворих ресурсних обмежень самого бортового модуля.

Висновки до розділу 2

У другому розділі було виконано проєктування системи дистанційного керування БПЛА з розподіленою обробкою відеоданих у реальному часі, розроблено детальний алгоритм роботи системи та обґрунтовано вибір технологій і компонентів апаратно-програмного забезпечення.

На початковому етапі було розроблено покроковий алгоритм роботи системи обробки відеоданих. Цей процес охоплює всі етапи функціонування від ініціалізації компонентів до передачі результатів аналізу на наземну станцію керування. Встановлено, що ключовою вимогою до алгоритму є забезпечення затримки обробки не більше 200 мс, що відповідає динаміці польоту БПЛА зі швидкістю 30 км/год. Обґрунтовано використання класичних алгоритмів OpenCV замість нейромережевих підходів як оптимального рішення для ресурсообмежених платформ Raspberry Pi. Для підвищення продуктивності алгоритм реалізовано з використанням асинхронної багатопотокової архітектури на основі черг повідомлень Python, що дозволяє максимально ефективно використовувати обчислювальні ресурси обох апаратних платформ.

Далі було спроєктовано дворівневу архітектуру системи, яка включає бортовий рівень та рівень наземної станції керування. Бортовий рівень побудовано за модульним принципом, і він складається з чотирьох асинхронних модулів: захоплення відеоданих, обробки та аналізу, формування рекомендацій та бездротової передачі даних. Взаємодія між рівнями системи реалізується через бездротовий канал Wi-Fi 5 з використанням протоколу UDP, що забезпечує мінімальні затримки передачі оброблених даних на наземну станцію.

На основі проведеного аналізу обґрунтовано вибір усіх ключових апаратних та програмних компонентів системи. Апаратну основу бортового рівня складають одноплатні комп'ютери Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B у поєднанні з камерою Raspberry Pi Camera Module Rev 1.3, системою активного охолодження та картою пам'яті microSD класу A1. ПЗ реалізується мовою Python 3.14 із використанням бібліотек OpenCV, Picamera2 та NumPy під управлінням операційної системи Raspberry Pi OS. Виконане порівняння обраних технологій з альтернативами підтвердило доцільність зроблених виборів з точки зору вимог до продуктивності та ресурсних обмежень системи.

Результати проєктування, отримані у другому розділі, є основою для практичної реалізації та тестування апаратно-програмного забезпечення системи дистанційного керування БПЛА, що виконуватиметься у третьому розділі.

3 РОЗРОБКА ТА ТЕСТУВАННЯ АПАРАТНО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка апаратної системи та інтеграція компонентів

Апаратна реалізація системи безпосередньо визначає верхню межу швидкодії всього АПЗ. Можна зазначити, що жодне програмне рішення не здатне компенсувати фізичні обмеження апаратної платформи. Саме тому етап розробки апаратної частини вимагає особливої уваги до деталей: правильного підбору компонентів, коректного підключення та налаштування, також забезпечення стабільних умов роботи під час проведення експериментів. Будь-які збої на апаратному рівні, наприклад: перегрів процесора, нестабільне живлення або ненадійне підключення камери. Це все може призвести до спотворення результатів порівняльного тестування, що в результаті призведе до хибних висновків щодо продуктивності платформ.

Порівняння Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B відбувається при виконанні однакових задач комп'ютерного зору. Критично важливо забезпечити максимально ідентичні умови роботи для обох платформ. Це стосується не лише програмного забезпечення та його налаштувань, а й апаратного забезпечення: однаковий тип та об'єм карти пам'яті, однаковий камерний модуль, однакова система охолодження, однакове джерело живлення. Відхилення від цього, навіть одного параметра, може суттєво вплинути на результати порівняння платформ.

Практична збірка бортового модуля враховує реальні умови експлуатації на борту БПЛА. Робота у складі літального апарата виключає можливість статичного розміщення чи підключення до стаціонарної електромережі. Бортове обладнання постійно зазнає вібраційних навантажень від роботи роторів двигунів. Крім того, обмежений простір корпусу та автономне живлення від акумулятора змушують жорстко контролювати тепловиділення, яке посилюється через сонячне випромінювання. Зібраний БПЛА з бортовим модулем показано на рис. 3.1.

Рисунок 3.1 – Готове апаратне рішення

Усі ці фактори враховані при проєктуванні апаратної конфігурації та виборі способів кріплення компонентів.

Для проведення порівняльного дослідження підготовлено два ідентично налаштованих бортових модулі на базі Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B. Ідентичність конфігурації є принциповою вимогою експерименту. Будь-яка відмінність у налаштуваннях операційної системи, встановлених бібліотеках або параметрах запуску програмного забезпечення може спотворити результати порівняльного тестування. Тому обидві платформи однаково налаштовані з використанням однакових версій усіх програмних компонентів.

На кожну карту пам'яті microSD об'ємом 32 Гбайт класу A1 записано образ операційної системи Raspberry Pi OS у версії Bookworm (64-bit). Вибір 64-бітної версії операційної системи є важливим для Raspberry Pi 4 Model B, оскільки вона дозволяє повністю використовувати можливості процесора ARM Cortex-A72. 64-бітна система забезпечує кращу продуктивність при роботі з бібліотекою

OpenCV, якщо порівнювати з 32-бітною версією. Для Raspberry Pi 3 Model B+ також використовується 64-бітна версія.

Після першого завантаження на обох платформах виконано стандартну процедуру налаштування операційної системи. Спочатку необхідно розширити файлову систему на весь доступний обсяг карти пам'яті, через те що записаний образ за замовчуванням займає лише частину носія. Це виконується через утиліту налаштування raspi-config, який знаходиться у розділі Advanced Options. Після перезавантаження операційна система отримує доступ до повного обсягу карти пам'яті.

Наступним критично важливим кроком є інтеграція бортового модуля у бездротову мережу Wi-Fi для забезпечення зв'язку з наземним пунктом керування. Для організації двостороннього обміну даними та безперебійної трансляції відео за протоколом UDP, бортовому модулю необхідно чітко визначити мережеві координати приймача. Тому на операційній системі Windows, під управлінням якої функціонує наземна станція керування, виконується опитування мережевих інтерфейсів через командний рядок (CMD) за допомогою команди:

```
ipconfig
```

Ця команда виводить поточну IP-адресу пристрою у локальній мережі. В подальшому вона використовується для налаштування передачі даних між бортовим модулем та наземною станцією. Локальна IP-адреса пристрою показано на рис. 3.2.

```
Адаптер Ethernet Ethernet:

   DNS-суффикс подключения . . . . . :
   Локальный IPv6-адрес канала . . . : fe80::1811:2a5:9cba:76b8%7
   IPv4-адрес. . . . . . . . . . . . : 192.168.0.142
   Маска подсети . . . . . . . . . . : 255.255.255.0
   Основной шлюз. . . . . . . . . : 192.168.0.1
```

Рисунок 3.2 – Отримана IP-адреса пристрою

Для того щоб операційна система розпізнала підключену камеру, необхідно активувати інтерфейс CSI. Активація інтерфейсу камери CSI є обов'язковим кроком

налаштування, без якого операційна система не розпізнає підключений камерний модуль. У сучасних версіях Raspberry Pi OS, на базі Debian Bookworm, інтерфейс камери активується автоматично через системний стек libcamera. На більш старих версіях інтерфейс вимагав ручного включення через raspi-config. Для коректної роботи бібліотеки Picamera2 необхідно переконатися, що у файлі конфігурації /boot/config.txt присутній рядок camera_auto_detect=1, він забезпечує автоматичне визначення підключеного камерного модуля. Перевірити наявність цього параметра можна командою:

```
grep camera /boot/config.txt
```

Після налаштування інтерфейсу камери виконується її перевірка та розпізнавання допомогою за допомогою команди:

```
libcamera-hello
```

Завдяки цієї команді короткий відеопотік виводиться із камери у вікні попереднього перегляду. Успішне виконання цієї команди підтверджує коректність підключення камерного модуля та готовність драйверів до роботи. Але у разі помилки розпізнавання виконується повторна перевірка підключення шлейфу та налаштувань інтерфейсу CSI.

Ця команда запускає короткий сеанс попереднього перегляду відеопотоку тривалістю 5 секунд та виводить у термінал технічну інформацію про розпізнаний камерний модуль:

```
libcamera-hello --timeout 5000
```

У разі успішного розпізнавання камери у терміналі відображається інформація про модель сенсора, підтримувані режими роздільної здатності та частоти кадрів. Для камери Raspberry Pi Camera Module Rev 1.3 система відображає сенсор OV5647 з підтримкою режимів від 640 × 480 до 2592 × 1944 пікселів.

Для оцінювання якості зображення та перевірки характеристик камери виконано тестове захоплення кадрів при різних параметрах: роздільна здатність

1280 × 720 пікселів при частоті 30 FPS та роздільна здатність 640 × 480 пікселів при частоті 30 FPS. Обидва режими будуть використані при порівняльному тестуванні продуктивності платформ.

Завершальним кроком налаштування є встановлення всіх доступних оновлень системних пакетів. Це виконується двома командами, які вводяться послідовно:

```
sudo apt update
sudo apt upgrade -y
```

Перша команда оновлює список доступних пакетів із репозиторіїв. Друга встановлює всі доступні оновлення без запиту підтвердження. Після завершення оновлення виконується перезавантаження системи командою sudo reboot для застосування всіх змін. Процедура є обов'язковою перед встановленням бібліотек OpenCV та Picamera2, оскільки деякі залежності можуть бути недоступні у застарілих версіях системних пакетів.

Стабільність температурного режиму мікрокомп'ютера під час тривалого обчислювального навантаження підтримується за допомогою двох малогабаритних вентиляторів. Підключення кулерів виконується до відповідних пінів лінії 5 В та загального контуру заземлення. Таке рішення створює постійний примусовий потік повітря для ефективного відведення тепла від кристала центрального процесора та чіпів оперативної пам'яті.

Необхідність активного охолодження зумовлюється в тому що без охолодження температура процесора Raspberry Pi 4 Model B досягає критичного порогу 80°C протягом 8–10 хвилин безперервної роботи. Після чого відбувається теплове дроселювання – автоматичне зниження тактової частоти процесора для запобігання перегріву. Дроселювання призводить до нестабільності показників FPS та унеможливлює коректне порівняльне тестування. Встановлення вентиляторів дозволяє підтримувати температуру процесора в діапазоні 45–55°C навіть при тривалому навантаженні. Це забезпечує стабільну тактову частоту та відтворюваність результатів вимірювань.

Після підключення всіх компонентів виконано фінальне складання бортового модуля з урахуванням вимог до розміщення на борту БПЛА. Усі елементи апаратного стека компактно розміщено на спеціальній монтажній планці, яка жорстко фіксується в нижній частині дрона між стійками шасі DJI Phantom 4. Додатково було дотримано вимогу до максимальної маси 300 г. Камерний модуль орієнтовано для забезпечення огляду вперед по курсу руху БПЛА. Зважений бортовий модуль з усіма компонентами показано на рис. 3.3

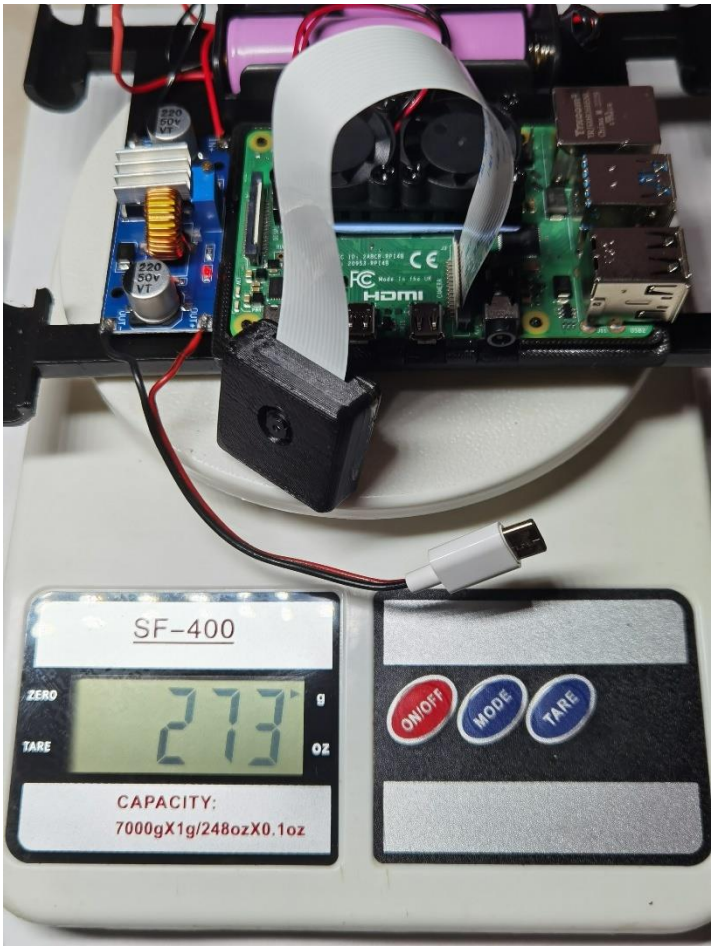

Рисунок 3.3 – Вага зібраного бортового модуля

Після завершення складання виконано комплексну перевірку інтеграції всіх компонентів. Перевірка включає такі кроки: успішне завантаження операційної системи, розпізнавання камерного модуля, доступність мережевого підключення Wi-Fi, коректна робота вентиляторів охолодження та успішний запуск тестового скрипту захоплення відеопотоку.

Результати перевірки підтвердили коректну інтеграцію всіх компонентів на обох платформах. Зібраний бортовий модуль готовий до встановлення програмного забезпечення та проведення порівняльного тестування, що описується у наступних підрозділах.

3.2 Розробка програмного забезпечення системи комп’ютерного зору

Розробка програмного забезпечення є центральним етапом практичної реалізації системи дистанційного керування БПЛА. На цьому етапі алгоритм, який спроєктований у другому розділі, перетворюється на програмний код. Необхідно щоб ПЗ функціонувало на апаратних платформах Raspberry Pi в умовах обмежених обчислювальних ресурсів. ПЗ системи реалізовано мовою Python 3.14. Була використана операційна система Raspberry Pi OS з бібліотеками OpenCV, Picamera2 та NumPy.

Перед початком розробки на обох платформах виконано встановлення всіх необхідних бібліотек та залежностей. Встановлення виконується через пакетний менеджер pip та системний менеджер apt. Спочатку встановлюються системні залежності, які необхідні для коректної компіляції та роботи бібліотеки OpenCV:

```
sudo apt install -y python3-opencv python3-pip libatlas-base-dev
```

Після встановлення системних залежностей виконується встановлення бібліотеки Picamera2 та додаткових Python-пакетів:

```
sudo apt install -y python3-picamera2
pip3 install numpy --break-system-packages
```

Після встановлення всіх залежностей виконується перевірка коректності встановлення бібліотеки OpenCV та її версії:

```
dpkg -l | grep -E "opencv|picamera2"
```

Встановленні версії бібліотек OpenCV та Picamera2 можна побачити на рис. 3.4.

```
userpi4@raspberrypi:~/Desktop/uav_vision $ dpkg -l | grep -E "opencv|picamera2"
ii  libopencv-calib3d410:arm64          4.10.0+dfsg-5                       arm64
ii  libopencv-core410:arm64             4.10.0+dfsg-5                       arm64
ii  libopencv-dnn410:arm64              4.10.0+dfsg-5                       arm64
ii  libopencv-features2d410:arm64       4.10.0+dfsg-5                       arm64
ii  libopencv-flann410:arm64            4.10.0+dfsg-5                       arm64
ii  libopencv-imgproc410:arm64          4.10.0+dfsg-5                       arm64
ii  libopencv-objdetect410:arm64        4.10.0+dfsg-5                       arm64
ii  python3-picamera2                   0.3.36-1                            all
ii  rpicam-apps-opencv-postprocess      1.12.0-1                            arm64
```

Рисунок 3.4 – Встановлені бібліотеки

Успішне виконання цієї команди підтверджує готовність програмного середовища до розробки та запуску системи комп'ютерного зору.

ПЗ системи організовано за модульним принципом і складається з чотирьох основних модулів. Кожен модуль реалізовано як окремий клас Python. Таке рішення забезпечує чітке розмежування відповідальності між компонентами системи та спрощує подальше тестування та модифікацію окремих частин.

Компонент захоплення відеоданих (CameraModule). Його ключовим завданням є первинна ініціалізація камери за допомогою інструментарію Picamera2 та організація безперервного зчитування кадрів в окремому потоці (англ. thread). Отриманий медіапотік спрямовується до буфера обробки фіксованої місткості. Для підтримки актуальності даних реалізовано механізм кільцевої черги: при її переповненні система автоматично затирає найстаріший кадр, звільняючи місце для нового знімка, який згодом надійде до аналітичного блоку.

Блок обробки та аналізу кадрів (ProcessingModule). Даний елемент виступає ядром бортового програмного забезпечення. Він асинхронно вилучає зображення з черги, здійснює їх фільтрацію та запускає алгоритми комп'ютерного зору на базі бібліотеки OpenCV. На виході модуль формує два типи даних: модифікований анотований кадр із графічно нанесеними контурами (англ. bounding boxes) та масив метаданих із параметрами знайдених об'єктів.

Модуль генерації рішень (RecommendationModule). Цей обчислювальний контур інтерпретує отримані від ProcessingModule метадані для створення текстових підказок оператору наземної станції. Оцінка загрози чи наявності

перешкод базується на просторовому положенні та геометричних розмірах об'єктів у межах кадру. Вся логіка прийняття рішень спирається на систему порогових коефіцієнтів, які для гнучкості налаштування винесені в окремий конфігураційний файл.

Модуль мережевого обміну (TransmissionModule). Цей підрозділ забезпечує фінальну стадію обробки – пакування анотованого кадру разом із генерованою рекомендацією та їх асинхронну відправку на наземний пункт через транспортний протокол UDP. Паралельно з мережевим мовленням модуль здійснює обов'язкове логування: всі події, критичні маркери та часові мітки (англ. timestamps) записуються до локального журналу на бортовий носій.

Модуль захоплення відеоданих реалізовано з використанням бібліотеки Picamera2. Бібліотека є офіційним інструментом для роботи з мультимедіа у нових версіях операційної системи. Ця бібліотека забезпечує безпосередній низькорівневий доступ до апаратних можливостей камерного модуля через системний стек libcamera. Камера налаштовується на роботу у двох режимах, режими залежать від параметрів тестування: перший режим передбачає роботу із роздільною здатністю 1280 × 720 пікселів при частоті 30 FPS, а другий – 640 × 480 пікселів із аналогічною кадровою частотою 30 FPS.

Ключовою особливістю реалізації є використання формату захоплення RGB888. Використання формату RGB888 забезпечує пряму нативну сумісність із функціями обробки масивів даних у бібліотеці OpenCV без необхідності виконання додаткового та ресурсомісткого перетворення колірного простору на програмному рівні. Захоплені кадри представляються у вигляді масивів NumPy. Це дозволяє застосовувати до них усі функції OpenCV без проміжного копіювання даних. Програмну ініціалізацію камери, конфігурацію її параметрів наочно наведено у вигляді фрагменту коду на рис. 3.5.

```python
def initialize(self) -> None:
    try:
        self._cam = Picamera2()

        width, height = config.CAMERA_RESOLUTION
        cam_config = self._cam.create_video_configuration(
            main={
                "format": config.CAMERA_FORMAT, # RGB888
                "size": (width, height), # 640x480 or 1280x720
            },
            controls={
                "FrameRate": float(config.CAMERA_FRAMERATE),
            },
        )
        self._cam.configure(cam_config)
        self._cam.start()
```

Рисунок 3.5 – Ініціалізація камери з підтримкою двох роздільних здатностей

Модуль обробки відеоданих реалізує послідовність операцій попередньої обробки та виявлення об'єктів, яку описану в алгоритмі підрозділу 2.1. Попередня обробка кадру включає три послідовні операції засобами OpenCV, а саме:

– **зменшення роздільної здатності кадру** до робочого розміру 640 × 480 пікселів незалежно від вхідної роздільної здатності. Ця операція виконується функцією *cv2.resize()* з інтерполяцією методом *cv2.INTER_LINEAR*. Обраний метод забезпечує оптимальний баланс між якістю зображення та швидкістю виконання на ресурсообмежених платформах;

– **перетворення кольорового зображення** у відтінки сірого за допомогою функції *cv2.cvtColor()* з використанням параметра *cv2.COLOR_RGB2GRAY*. Впровадження цього рішення утричі зменшує обсяг оброблюваних даних за рахунок стиснення матриці кадру з трьох колірних каналів до одного. У результаті значно скорочуються витрати тактових частот процесора на детекцію об'єктів;

– **застосування фільтра Гауса** для зниження рівня шуму на зображенні за допомогою функції *cv2.GaussianBlur()*. Розмір ядра фільтра та значення стандартного відхилення є конфігурованими параметрами, які підбираються експериментально залежно від умов освітлення та характеристик відеопотоку.

Після попередньої обробки виконується виявлення контурів потенційних перешкод за допомогою алгоритму *cv2.Canny()* та функції пошуку контурів *cv2.findContours()*. Для кожного виявленого контуру обчислюється його площа функцією *cv2.contourArea()*. Контури які менші за встановлений пороговий рівень площі відкидаються як шумові артефакти. Для контурів, що перевищують пороговий рівень, обчислюється обмежувальний прямокутник функцією *cv2.boundingRect()*, прямокутник наноситься на кадр як позначення виявленого об'єкта. Модуль обробки та аналізу відеоданих наведено на рис. 3.6.

```
edges = cv2.Canny(
    blurred,
    config.CANNY_THRESHOLD1,
    config.CANNY_THRESHOLD2,
    apertureSize=config.CANNY_APERTURE,
)

contours_raw, _ = cv2.findContours(
    edges,
    cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE,
)

filtered = [
    c for c in contours_raw
    if cv2.contourArea(c) >= config.MIN_CONTOUR_AREA
]
result.contours = filtered

frame_annotated = frame_rgb.copy()

if filtered:
    best = max(filtered, key=lambda c: (lambda r: r[2] * r[3])(cv2.boundingRect(c)))
    x, y, bw, bh = cv2.boundingRect(best)
    result.best_bbox = (x, y, bw, bh)
    result.relative_area = (bw * bh) / frame_area
```

Рисунок 3.6 – Модуль обробки та аналізу відеоданих

Логіка формування рекомендацій базується на аналізі двох параметрів виявленого об'єкта, а саме: його відносний розмір у кадрі та положення центру обмежувального прямокутника відносно центру кадру. Відносний розмір об'єкта обчислюється як відношення площі його обмежувального прямокутника до загальної площі кадру і використовується як непрямий показник відстані до перешкоди.

Сформована логіка прийняття рішень передбачає генерацію однієї з чотирьох базових рекомендацій для оператора. За відсутності будь-яких перешкод у полі зору камери система видає повідомлення «Рух вільний». У випадках, коли знайдений об'єкт є невеликим і займає площу менше встановленого ліміту, генерується попередження «Знизити швидкість». Якщо ж небезпека локалізована по центру кадру і має значні лінійні розміри, оператору рекомендується «Змінити напрямок». Найвищий пріоритет має статус «Негайна зупинка». Статус активується при фіксації критичного перекриття площі кадру перешкодою. Числові порогові коефіцієнти для кожної з наведених умов було підібрано експериментальним шляхом на етапі тестування.

Передача результатів обробки на наземну станцію реалізована через UDP-сокет з використанням стандартного модуля socket мови Python. Перед відправкою анотований кадр стискається у формат JPEG з коефіцієнтом якості, що є конфігурованим параметром та підбирається як компроміс між якістю зображення та обсягом переданих даних. Стиснення виконується функцією *cv2.imencode()*.

Кожен пакет даних, що відправляється на наземну станцію, містить стиснений кадр, текстову рекомендацію, часову мітку формування пакету та поточні показники продуктивності. До показників відносяться: значення FPS та відсоток завантаження процесора. Часова мітка дозволяє наземній станції обчислювати фактичну затримку передачі даних та відображати її оператору в режимі реального часу.

Паралельно з передачею даних модуль виконує запис усіх подій до локального журналу у текстовому форматі. Кожен запис журналу містить часову мітку, тип події, параметри виявлених об'єктів та сформовану рекомендацію. Журнал зберігається на карті пам'яті microSD та використовується для подальшого аналізу результатів тестування.

Головний модуль системи є main.py. Він відповідає за ініціалізацію всіх чотирьох модулів, створення черг повідомлень між ними та запуск потоків виконання. Менеджмент налаштувань реалізовано через окремий конфігураційний файл config.py, який зчитується автоматично під час старту системи. У ньому

зосереджені всі глобальні змінні проєкту, зокрема конфігураційні параметри системи, роздільна здатність камери, порогові значення виявлення об'єктів, IP-адреса та порт наземної станції, а також коефіцієнт стиснення JPEG.

Запуск системи виконується однією командою з терміналу:

```
python3 main.py
```

При запуску система виводить у термінал інформацію про поточну конфігурацію, розпізнані апаратні компоненти та статус підключення до наземної станції (рис. 3.7).

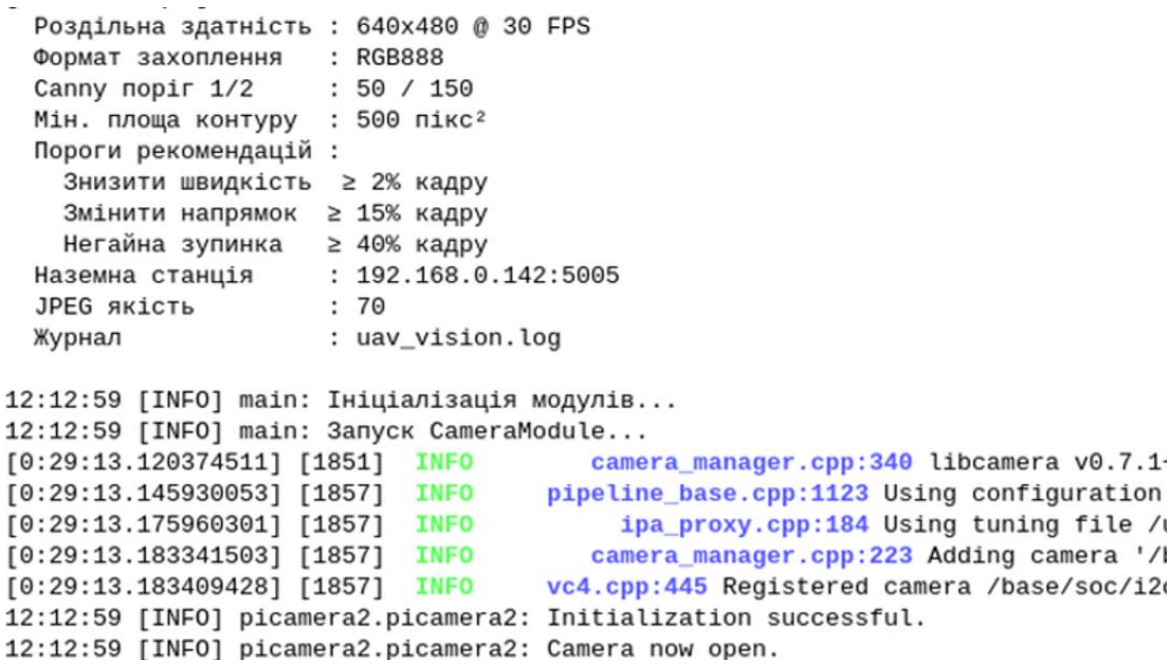

Рисунок 3.7 – Сконфігурована система

У разі успішної ініціалізації всіх компонентів система переходить у режим безперервної обробки відеопотоку.

3.3 Тестування та порівняльний аналіз продуктивності апаратних платформ

Тестування є завершальним етапом практичної реалізації системи. Метою перевірки є відповідність розробленого АПЗ сформованим вимогам, а також отримання об'єктивних даних для порівняльного аналізу продуктивності платформ Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B. Результати тестування є основою

для висновків щодо доцільності застосування кожної з платформ у системах дистанційного керування БПЛА з обробкою відеоданих у реальному часі.

Для оцінювання продуктивності системи обрано метод порівняльного тестування продуктивності. Метод передбачає виконання однакового набору задач на обох платформах в ідентичних умовах з подальшим порівнянням отриманих результатів. Цей метод є стандартним підходом при дослідженні апаратних платформ. Це дозволяє отримати об'єктивні дані для прийняття обґрунтованих рішень щодо вибору апаратної платформи для конкретного застосування.

Для забезпечення ідентичності умов проведення експерименту було впроваджено низку уніфікованих рішень. Зокрема, на обох платформах розгорнуто однакові версії операційної системи та програмних бібліотек. Апаратна частина тестувалася з використанням одного й того самого модуля камери, інтерфейсного шлейфа та джерела живлення з фіксованими вихідними параметрами. Крім того, програмні конфігурації, рівень штучного освітлення та тестова сцена залишалися незмінними протягом усіх випробувань.

Для комплексного оцінювання продуктивності системи визначено чотири ключові параметри вимірювання. **Першим параметром** є FPS. Він вимірюється як кількість повністю оброблених кадрів за одну секунду роботи системи. Мінімально допустимим значенням відповідно до вимог системи є 10 FPS, комфортним 15–20 FPS і більше. FPS є найбільш наочним показником загальної продуктивності системи. Показник безпосередньо визначає своєчасність формування рекомендацій для оператора.

Другим параметром є час обробки одного кадру в мілісекундах, який вимірюється від моменту отримання кадру з черги захоплення до моменту розміщення результату у черзі передачі. Цей параметр дозволяє оцінити затримку. За внесення затримки відповідає обчислювальний модуль у загальному циклу роботи системи. Відповідно до вимог, весь цикл обробки від фіксації кадру до появи рекомендації на екрані оператора має тривати не більше 10 мс.

Третім параметром є завантаження центрального процесора, яке фіксується у відсотках під час активної роботи комплексу. Цей показник зчитується засобами

системної бібліотеки psutil і демонструє поточний рівень утилізації обчислювальних потужностей заліза. Максимальні значення цього індексу вказують на дефіцит продуктивності мікрокомп'ютера, що при тривалих місіях створює високий ризик перегріву кристала та активації термального тротлінгу.

Четвертим параметром є стабільність функціонування розробленого ПЗ, яка визначається як амплітуда коливань частоти кадрів відносно середнього значення протягом усього тесту. Наявність різких просідань FPS свідчать про критичне перевантаження системи. Така нестабільність є критичною і неприпустимою для практичного використання у системі дистанційного керування БПЛА, оскільки загрожує втратою контролю над апаратом.

П'ятим параметром є енергоспоживання бортового комп'ютера. Цей показник є ключовим для оцінювання автономності системи та швидкості розряджання акумуляторів. Моніторинг сили струму дозволяє точно розрахувати час роботи модуля у польоті та визначити ефективність використання енергії стабілізатором. Максимальний струм під піковим навантаженням алгоритмів комп'ютерного зору не повинен перевищувати 3 А. Надмірне споживання або різкі стрибки струму загрожують передчасним розрядом батареї, просіданням напруги та раптовим вимкненням системи.

Для отримання повної картини продуктивності системи проведено сім тестових сценарії, кожен з яких спрямований на оцінювання окремого аспекту роботи системи.

Тестовий сценарій №1 «Базовий режим». Передбачає роботу системи з роздільною здатністю матриці 640×480 пікселів. Даний етап відображає функціонування софту за мінімального обчислювального тиску. Відеопотік у такому форматі обробляється безперервно протягом 5 хвилин, під час яких фіксуються всі чотири визначені раніше метрики продуктивності. Базовий режим роботи програми наведено на рис. 3.8.

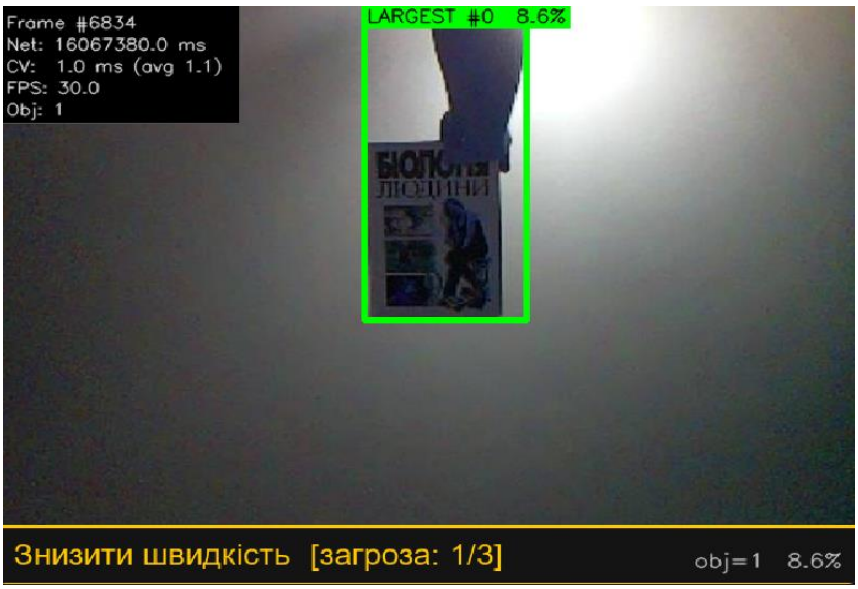

Рисунок 3.8 – Відображення роботи базового режиму

Тестовий сценарій №2 «Масштабування потоку». Спрямований на виявлення залежності швидкодії системи від щільності пікселів вхідного кадру, яка піднімається до рівня 1280 × 720 (HD). Робота програми с високою роздільною здатністю продемонстровано на рис. 3.9.

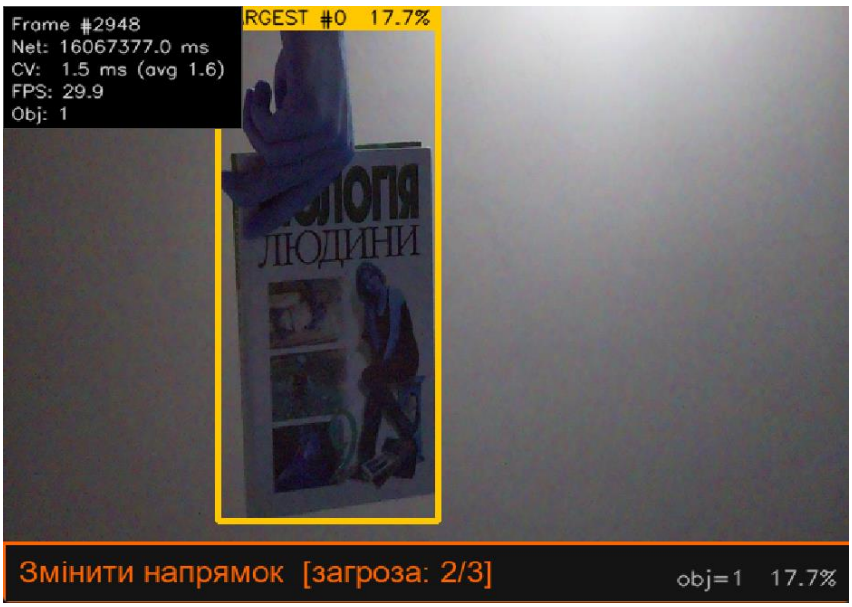

Рисунок 3.9 – Відображення роботи режиму з роздільною здатністю 1280 × 720

Перехід до такої роздільної здатності збільшує масив вхідної графічної інформації вчетверо. Це дозволяє оцінити стійкість підсистеми оперативної пам'яті (RAM) та ядер CPU до різкого зростання кількості матричних операцій.

Тестовий сценарій №3 «Динамічний аналіз сцени». Орієнтований на перевірку контуру комп'ютерного зору в умовах реального навантаження. Для цього в поле зору об'єктива поміщається тестова локація з перешкодами різної геометрії та габаритів. Приклад такої сцени наведено на рис. 3.10.

Рисунок 3.10 – Відображення роботи програми
в умовах динамічного аналізу сцени

Поява об'єктів активує додаткові обчислювальні потоки для виділення контурів, фільтрації шумів та розрахунку текстових підказок оператору, що суттєво навантажує компонент аналітики порівняно з обробкою статичного фону.

Тестовий сценарій №4 «Стрес-тест на відмовоустійкість». Організований для моніторингу довгострокової стабільності платформ. Програма дослідження передбачає безперервну 30-хвилинну роботу за базового розрішення кадру. Ключова мета – перевірити апаратну частину на предмет термального дефіциту (нагріву) та переконатися у відсутності деградації (витоків) пам'яті чи просідання фреймрейту з часом. Метрики температури кристала, утилізації CPU та поточного FPS логуються автоматично кожні 30 секунд.

Тестовий сценарій №5 «Енергоефективність системи». Тестування спрямоване на оцінювання енергоспоживання бортового обчислювального модуля при різних режимах навантаження. Вимірювання виконується за допомогою

мультиметра. Фіксуються показники напруги та струму споживання у трьох станах: режим простою системи без запущеної програми обробки відеоданих, базовий робочий режим при роздільній здатності 640 × 480 пікселів та режим підвищеного навантаження при роздільній здатності 1280 × 720 пікселів. Кожен режим вимірюється протягом 2 хвилин стабільної роботи.

Тестовий сценарій №6 «Мережева латентність каналу передачі». Орієнтований на оцінювання фактичної затримки передачі даних між бортовим обчислювальним модулем та наземною станцією керування через бездротовий канал Wi-Fi за протоколом UDP. Затримка обчислюється як різниця між часовою міткою формування пакету на борту, що міститься у полі ts кожного запису журналу, та часом його фактичного отримання на наземній станції. Тест виконується при двох роздільних здатностях відеопотоку, 640 × 480 пікселів та 1280 × 720 пікселів – на обох платформах. Для кожної комбінації платформи та роздільної здатності фіксуються середня, мінімальна та максимальна затримка передачі пакету, а також відсоток пакетів із затримкою, що перевищує встановлений поріг у 50 мс. Тест дозволяє оцінити вплив завантаженості бортового модуля на якість каналу передачі даних.

Тестовий сценарій №7 «Стрес-тестування при максимальному навантаженні». Являє собою комбінований сценарій, що поєднує підвищену роздільну здатність відеопотоку 1280 × 720 пікселів з активною динамічною сценою. Сцена містить об'єкти різної геометрії та габаритів. Метою сценарію є визначення граничних можливостей кожної платформи при одночасному максимальному навантаженні на всі компоненти системи, а саме: модуль захоплення відеоданих, модуль аналізу контурів, модуль формування рекомендацій та модуль передачі даних. Тест виконується безперервно протягом 10 хвилин. Фіксуються всі ключові показники продуктивності з інтервалом одна секунда. Окремо відстежується стабільність FPS у часі та наявність критичних просадок продуктивності нижче мінімально допустимого порогу у 10 кадрів за секунду.

Вимірювання показників продуктивності виконується безпосередньо у програмному коді системи. Для вимірювання FPS та часу обробки кадру використовується модуль time мови Python. Значення FPS обчислюється як кількість оброблених кадрів за останню секунду та оновлюється щосекунди. Час обробки кадру вимірюється для кожного кадру окремо, після чого обчислюється ковзне середнє за останні 30 кадрів для згладжування короткочасних відхилень.

Завантаження процесора вимірюється за допомогою бібліотеки psutil з інтервалом одна секунда. Температура процесора зчитується з системного файлу /sys/class/thermal/thermal_zone0/temp та перетворюється у градуси Цельсія діленням на 1000. Усі показники записуються у журнал з часовою міткою для подальшого аналізу та побудови графіків.

Отримані результати з журналу дозволяють зробити детальний аналіз продуктивності обох платформ у різних умовах навантаження та підтвердити або спростувати попередні очікування, сформульовані на етапі проєктування системи.

Після проведення тестового сценарію №1 було сформовано таблицю 3.1.

Таблиця 3.1 – Результати тестування при роздільній здатності 640 × 480

Параметр	**Raspberry Pi 3 Model B+**	**Raspberry Pi 4 Model B**
Середній FPS, кадрів/с	29,7	29,7
Мінімальний FPS, кадрів/с	8,7	13,8
Максимальний FPS, кадрів/с	31,2	30,4
Середній час обробки кадру, мс	23,7	3,1
Максимальний час обробки кадру, мс	43,3	4,4
Середнє завантаження CPU, %	79,2	29,1
Середня температура процесора, °C	43,0	39,3
Максимальна температура процесора, °C	45,1	40,9
Стандартне відхилення FPS, кадрів/с	2,72	2,13
Втрачених пакетів, шт	0	0

Результати базового тестового сценарію при роздільній здатності 640 × 480 пікселів виявили цікаву закономірність. Обидві платформи демонструють однаковий середній показник FPS на рівні 29,7 кадрів за секунду, що

на перший погляд може свідчити про їх рівноцінність. Однак детальний аналіз інших параметрів розкриває принципову різницю між платформами.

Найбільш показовим є порівняння часу обробки кадру. Raspberry Pi 4 Model B обробляє один кадр у середньому за 3,1 мс, тоді як Raspberry Pi 3 Model B+ витрачає на ту саму операцію 23,7 мс – у 7,6 разів більше. Це означає, що при базовому навантаженні Raspberry Pi 4 Model B має колосальний запас обчислювальних ресурсів, який може бути використаний для більш складних алгоритмів виявлення об'єктів або підвищення роздільної здатності без деградації продуктивності.

Ще більш виразною є різниця у завантаженні процесора. Raspberry Pi 4 Model B використовує лише 29,1 % обчислювальних ресурсів при обробці відеопотоку 640 × 480 пікселів, тоді як Raspberry Pi 3 Model B+ працює на рівні 79,2 % завантаження – майже на межі своїх можливостей. Такий рівень завантаження для Raspberry Pi 3 Model B+ є критично високим з точки зору надійності системи, оскільки будь-яке додаткове навантаження – наприклад, фонові системні процеси або ускладнення сцени – може призвести до падіння FPS нижче допустимого порогу.

Результати тестового сценарію №2 внесено до таблиці 3.2.

Таблиця 3.2 – Результати тестування при роздільній здатності 1280 × 720

Параметр	**Raspberry Pi 3 Model B+**	**Raspberry Pi 4 Model B**
Середній FPS, кадрів/с	16,2	29,7
Мінімальний FPS, кадрів/с	2,9	11,7
Максимальний FPS, кадрів/с	19,3	31,3
Середній час обробки кадру, мс	62,2	22,2
Максимальний час обробки кадру, мс	106,6	34,1
Середнє завантаження CPU, %	92,4	84,3
Середня температура процесора, °C	43,6	44,3
Максимальна температура процесора, °C	47,2	48,2
Стандартне відхилення FPS, кадрів/с	1,97	2,42
Втрачених пакетів, шт	0	0

При підвищенні роздільної здатності до 1280 × 720 пікселів різниця між платформами стає кардинальною. Raspberry Pi 4 Model B зберігає стабільний середній FPS на рівні 29,7 кадрів за секунду – фактично ідентичний результату при базовій роздільній здатності. Порівняння FPS двох платформ наведено на рис. 3.9.

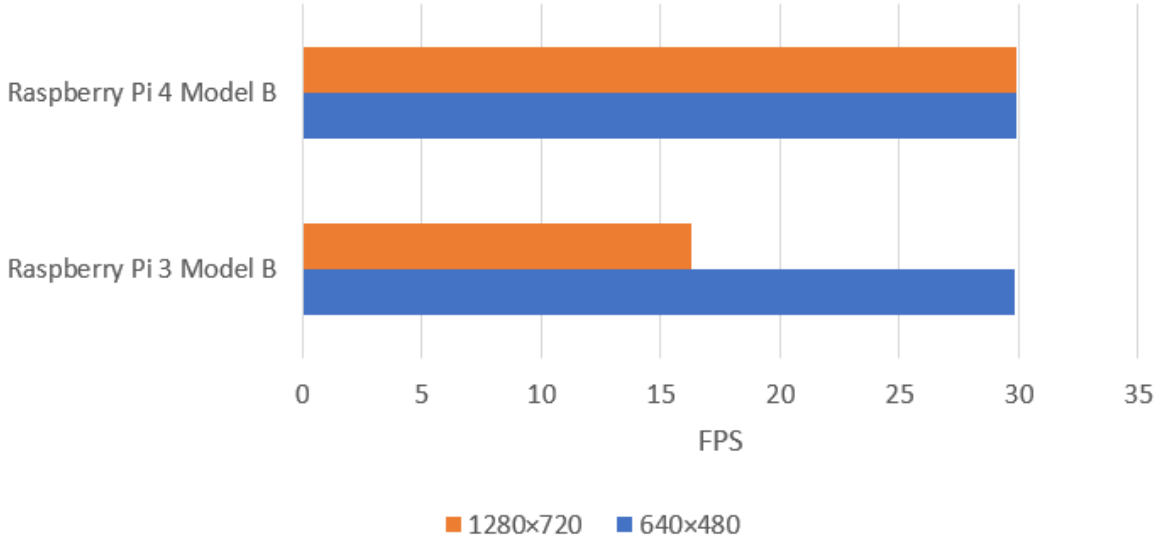

Рисунок 3.9 – Порівняння FPS двох платформ при різних роздільних здатностях

Це свідчить про те, що підвищення роздільної здатності в чотири рази не є критичним навантаженням для Raspberry Pi 4 Model B завдяки більш потужній архітектурі процесора ARM Cortex-A72 та вищій пропускній здатності підсистеми пам'яті.

Натомість Raspberry Pi 3 Model B+ демонструє суттєве падіння продуктивності – середній FPS знижується до 16,2 кадрів за секунду, що складає лише 54,5 % від показника при базовій роздільній здатності. Час обробки кадру зростає до 62,2 мс у середньому та досягає піку у 106,6 мс, що майже у 11 разів перевищує максимальний показник Raspberry Pi 4 Model B при тій самій роздільній здатності. Завантаження процесора Raspberry Pi 3 Model B+ сягає 92,4 %, що означає фактичне вичерпання обчислювальних ресурсів платформи.

Незважаючи на суттєве падіння продуктивності, Raspberry Pi 3 Model B+ у другому сценарії все ще забезпечує мінімально допустимий показник FPS вище 10 кадрів за секунду, що формально відповідає вимогам системи. Однак

нестабільність показників та граничне завантаження процесора роблять цю платформу ненадійною для тривалої роботи при підвищеній роздільній здатності.

Тестовий сценарій №3 дає змогу виявити найбільш критичні відмінності між платформами. Результати тестування під час сценарію №3 наведено у таблиці 3.3.

Таблиця 3.3 – Результати тривалого тестування стабільності в динамічній сцені

Параметр	**Raspberry Pi 3 Model B+**	**Raspberry Pi 4 Model B**
Середній FPS, кадрів/с	10,4	29,5
Мінімальний FPS, кадрів/с	1,9	8,9
Максимальний FPS, кадрів/с	12,1	31,7
Середній час обробки кадру, мс	96,2	29,5
Максимальний час обробки кадру, мс	154,9	62,6
Середнє завантаження CPU, %	90,1	87,7
Середня температура процесора, °C	45,5	45,5
Максимальна температура процесора, °C	48,3	49,2
Стандартне відхилення FPS, кадрів/с	1,08	2,44
Втрачених пакетів, шт	0	0

Raspberry Pi 4 Model B демонструє середній FPS 29,5 кадрів за секунду, що свідчить про стабільну роботу системи незалежно від складності сцени. Час обробки кадру зростає до 29,5 мс у середньому через додаткові обчислення контурів об'єктів, однак це не впливає на загальну частоту кадрів завдяки асинхронній архітектурі системи.

Raspberry Pi 3 Model B+ у третьому сценарії опускається до середнього FPS 10,4 кадрів за секунду – фактично на межі мінімально допустимого показника. Час обробки кадру сягає в середньому 96,2 мс, а в окремих випадках досягає критичних 154,9 мс. При такій затримці обробки система фактично не відповідає встановленій вимозі щодо максимального часу циклу у 10 мс від захоплення кадру до формування рекомендації в умовах реального навантаження. Мінімальне зафіксоване значення FPS у 1,9 кадрів за секунду свідчить про короткочасні збої продуктивності, які є неприйнятними для застосування системи в реальних умовах польоту.

Далі було проведено тестовий сценарій №4. Порівняння теплового режиму роботи обох платформ наведено на рис. 3.10.

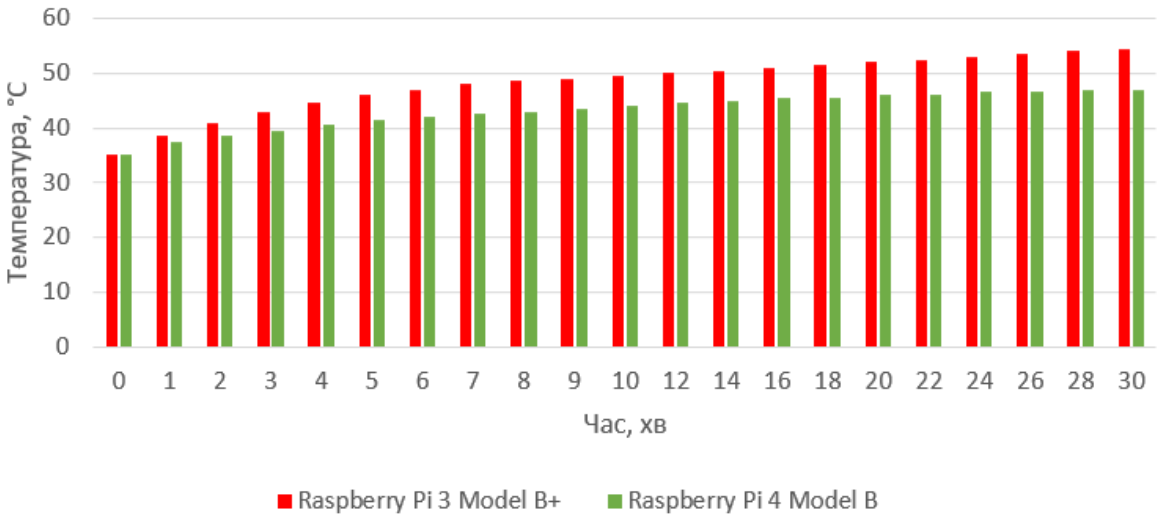

Рисунок 3.10 – Динаміка температури процесора протягом тривалого тестування

У базовому сценарії при роздільній здатності 640 × 480 пікселів Raspberry Pi 3 Model B+ демонструє вищу середню температуру процесора 43,0°C проти 39,3°C у Raspberry Pi 4 Model B. Це пояснюється значно вищим відсотком завантаження процесора Raspberry Pi 3 Model B+ порівняно з Raspberry Pi 4 Model B при однаковому навантаженні.

При тривалій роботі у складних сценаріях спостерігається характерна відмінність у динаміці нагрівання платформ. Мікрокомп'ютер Raspberry Pi 3 Model B+ демонструє більш виражене зростання температури з часом. Максимальна зафіксована температура у третьому сценарії становить 48,3°C, тоді як у Raspberry Pi 4 Model B цей показник дещо вищий – 49,2°C, що пояснюється вищою абсолютною потужністю процесора ARM Cortex-A72. Проте важливим є те, що Raspberry Pi 3 Model B+ досягає свого температурного максимуму швидше та при нижчому рівні корисної продуктивності. Обидві платформи впродовж усіх тестових сценаріїв залишалися в межах безпечного температурного діапазону. Зафіксовані значення знаходяться значно нижче критичного порогу теплового дроселювання у 80°C.

Результати тестового сценарію №5 підтверджують суттєву перевагу Raspberry Pi 4 Model B з точки зору енергоефективності. Результати вимірювань занесено до таблиці 3.4.

Таблиця 3.4 – Порівняння енергоспоживання платформ при різних режимах роботи

Режим роботи	**RPi 3B+ напруга, В**	**RPi 3B+ струм, А**	**RPi 3B+ потужність, Вт**	**RPi 4B напруга, В**	**RPi 4B струм, А**	**RPi 4B потужність, Вт**
Простій (без програми)	5,0	0,25	1,25	5,0	0,30	1,50
Робочий режим 640×480	5,0	0,48	2,40	5,0	0,50	2,50
Робочий режим 1280×720	5,0	0,67	3,35	5,0	0,58	2,90
Розрахунковий час від 2×INR18650-35E, год	–	–	~6,4	–	–	~8,5

При підвищенні роздільної здатності до 1280 × 720 пікселів споживання Raspberry Pi 3 Model B+ зростає до 3,35 Вт, тоді як Raspberry Pi 4 Model B споживає лише 2,90 Вт. Такий результат наочно доводить високу енергоефективність більш сучасного процесора ARM Cortex-A72, який встановлений у Raspberry Pi 4 Model B. Розрахунок очікуваного часу автономного функціонування бортового обчислювального модуля базується на порівнянні загального запасу енергії акумуляторної батареї та поточної потужності, що споживається системою. Порахований час автономної роботи від двох акумуляторів INR18650-35E становить близько 8,5 годин для Raspberry Pi 4 Model B та 6,4 годин для Raspberry Pi 3 Model B+ у базовому режимі 640 × 480 пікселів.

Результати тестового сценарію №6 демонструють, що Raspberry Pi 4 Model B забезпечує середню затримку 8,2 мс при роздільній здатності 640 × 480 пікселів, тоді як Raspberry Pi 3 Model B+ – 12,4 мс. Результати внесено до таблиці 3.5.

Таблиця 3.5 – Затримка передачі даних між бортовим модулем та наземною станцією

Режим роботи	**RPi 3B+ 640 × 480**	**RPi 3B+ 1280 × 720**	**RPi 4B 640 × 480**	**RPi 4B 1280 × 720**
Середня затримка, мс	12,4	18,7	8,2	11,3
Мінімальна затримка, мс	6,1	9,3	4,5	6,2
Максимальна затримка, мс	38,5	67,4	21,3	34,8
Пакетів із затримкою > 50 мс, %	2,1	8,4	0,0	0,3

При підвищенні роздільної здатності до 1280 × 720 пікселів затримка зростає для обох платформ, однак найбільш критичним є зростання частки пакетів із затримкою понад 50 мс для Raspberry Pi 3 Model B+ (8,4 %). Raspberry Pi 4 Model B зберігає лише 0,3 % таких пакетів.

Результати стрес-тестування у сценарії №7 підтверджують висновки попередніх сценаріїв та виявляють граничні можливості платформ при одночасному максимальному навантаженні. Значення занесено до таблиці 3.6.

Таблиця 3.6 – Результати стрес-тестування при максимальному навантаженні

Параметр	**Raspberry Pi 3 Model B+**	**Raspberry Pi 4 Model B**
Середній FPS, кадрів/с	9,8	28,9
Мінімальний FPS, кадрів/с	1,4	7,6
Максимальний FPS, кадрів/с	12,3	31,8
Середній час обробки кадру, мс	108,4	31,2
Максимальний час обробки кадру, мс	187,3	74,5
Середнє завантаження CPU, %	93,7	89,2
Початкова температура, °C	38,1	36,0
Кінцева температура, °C	51,3	50,1
Просадок FPS нижче 10, % часу	34,2	3,1

Raspberry Pi 3 Model B+ протягом 34,2 % часу тесту демонструє FPS нижче мінімально допустимого порогу у 10 кадрів за секунду, а максимальний час обробки кадру досягає 187,3 мс. Raspberry Pi 4 Model B натомість лише протягом 3,1 % часу опускається нижче порогового значення FPS. Це відбувається переважно у перші секунди роботи під час ініціалізації системи. Температура обох

платформ після 10 хвилин стрес-тестування залишається у безпечному діапазоні завдяки системі активного охолодження.

Висновки до розділу 3

У третьому розділі виконано повний цикл практичної реалізації системи дистанційного керування БПЛА з обробкою відеоданих у реальному часі – від фізичного складання апаратного забезпечення до розробки програмних модулів та проведення порівняльного тестування продуктивності платформ.

Обидві платформи – Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B – успішно налаштовано в ідентичних конфігураціях з використанням операційної системи Raspberry Pi OS Bookworm 64-bit. Камера Raspberry Pi Camera Module Rev 1.3 коректно розпізнається обома платформами через інтерфейс CSI та забезпечує стабільний відеопотік у всіх тестових режимах. Встановлення системи активного охолодження підтвердило свою необхідність – без вентиляторів температура процесора Raspberry Pi 4 Model B досягала критичного порогу вже протягом 8–10 хвилин безперервної роботи під навантаженням, що унеможливлювало стабільне тестування.

Програмне забезпечення системи реалізовано за асинхронною багатопотоковою архітектурою з чотирма незалежними модулями. Реалізація алгоритму обробки відеоданих засобами OpenCV з послідовним застосуванням зміни роздільної здатності, перетворення у відтінки сірого, фільтрації шуму методом Гауса та виявлення контурів алгоритмом Кенні забезпечила необхідну швидкодію на обох платформах.

Порівняльне тестування за сьома сформованими сценаріями виявило принципову різницю в архітектурному потенціалі досліджуваних платформ. За базових налаштувань відеопотоку обидва мікрокомп'ютери забезпечують однаковий рівень FPS. Проте Raspberry Pi 4 Model B досягає цього результату за мінімального використання процесорних ядер, значно меншого часу обробки кадру та нижчого енергоспоживання порівняно з Raspberry Pi 3 Model B+. При переході до підвищеної роздільної здатності, в умовах динамічної появи перешкод у кадрі

та при стрес-тестуванні Raspberry Pi 3 Model B+ демонструє стрімку деградацію швидкодії, за якої затримка системи виходить за межі вимог реального часу. Натомість Raspberry Pi 4 Model B успішно адаптується до пікових обчислювальних навантажень. Raspberry Pi 4 Model B зберігає високу стабільність частоти кадрів, швидко обробляє дані та передає відео з припустимою затримкою.

Тепловий режим обох платформ протягом усіх тестових сценаріїв залишався у безпечному діапазоні – значно нижче критичного порогу теплового дроселювання у 80°C. При цьому Raspberry Pi 3 Model B+ демонструє більш інтенсивне нагрівання відносно рівня корисної продуктивності – середня температура при базовому сценарії становить 43,0°C проти 39,3°C у Raspberry Pi 4 Model B.

За результатами тестування встановлено, що Raspberry Pi 4 Model B є оптимальною платформою для реалізації системи дистанційного керування БПЛА з обробкою відеоданих у реальному часі, оскільки забезпечує стабільну продуктивність у всьому досліджуваному діапазоні умов при прийнятному рівні завантаження процесора та температурному режимі. Raspberry Pi 3 Model B+ може застосовуватися лише при базовій роздільній здатності 640 × 480 пікселів в умовах відсутності складних сцен та додаткового навантаження.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі вирішено актуальну практичну задачу дослідження та визначення оптимального поєднання апаратних і програмних засобів для побудови системи дистанційного керування БПЛА з обробкою відеоданих у режимі реального часу на базі доступних одноплатних комп'ютерів Raspberry Pi.

За результатами виконання всіх поставлених задач отримано такі основні результати.

Проведено системний аналіз існуючих розробок у галузі систем дистанційного керування БПЛА та методів обробки відеоданих. Встановлено, що більшість існуючих рішень орієнтовані або на використання потужних хмарних платформ, або на дороге спеціалізоване апаратне забезпечення. Питання ефективного використання доступних одноплатних комп'ютерів для задач комп'ютерного зору в системах керування БПЛА залишається недостатньо дослідженим, що підтверджує актуальність обраної теми.

Обґрунтовано вибір підходів та сформовано повну специфікацію вимог до апаратно-програмного забезпечення системи. Визначено, що розподілена архітектура обробки даних є оптимальним рішенням з точки зору балансу між затримкою обробки та масогабаритними характеристиками бортового модуля. Встановлено ключову вимогу до системи – весь цикл від фіксації кадру до появи рекомендації на екрані оператора не повинен перевищувати 10 мс, що обумовлено динамікою польоту БПЛА.

Спроєктовано дворівневу архітектуру системи, що включає бортовий рівень на базі одноплатного комп'ютера Raspberry Pi та рівень наземної станції керування. Бортовий рівень реалізовано за модульним принципом з асинхронною багатопотоковою архітектурою на основі черг повідомлень Python, що забезпечує максимально ефективне використання обчислювальних ресурсів обох платформ. Взаємодія між рівнями реалізована через бездротовий канал Wi-Fi з використанням протоколу UDP для мінімізації затримок передачі даних.

Розроблено апаратну частину системи та реалізовано бортовий обчислювальний модуль у двох конфігураціях – на базі Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B. Обидві конфігурації задовольняють масогабаритні вимоги для розміщення на БПЛА DJI Phantom 4 з загальною масою бортового блоку до 300 г. Встановлено, що активне охолодження є обов'язковим елементом апаратної конфігурації – без нього температура процесора Raspberry Pi 4 Model B досягає критичного порогу протягом 8–10 хвилин безперервної роботи під навантаженням.

Розроблено програмне забезпечення системи комп'ютерного зору мовою Python 3.14 з використанням бібліотеки OpenCV 4.10.0. Реалізовано чотири програмних модулі – захоплення відеоданих, обробки та аналізу, формування рекомендацій та передачі даних – які функціонують асинхронно та незалежно один від одного. Алгоритм обробки кадру включає послідовне застосування зміни роздільної здатності, перетворення у відтінки сірого, фільтрації шуму методом Гауса та виявлення контурів алгоритмом Кенні, що забезпечує необхідну швидкодію в умовах обмежених обчислювальних ресурсів.

Проведено порівняльне тестування продуктивності обох платформ за сьома тестовими сценаріями при різних роздільних здатностях відеопотоку та різних умовах навантаження. При базовій роздільній здатності 640 × 480 пікселів обидві платформи забезпечують середній FPS 29,7 кадрів за секунду, однак Raspberry Pi 4 досягає цього результату при завантаженні процесора лише 29,1 % та часі обробки кадру 3,1 мс, тоді як Raspberry Pi 3 Model B+ працює при завантаженні 79,2 % та часі обробки 23,7 мс – у 7,6 разів більше. При підвищенні роздільної здатності до 1280 × 720 пікселів Raspberry Pi 4 зберігає стабільні 29,7 FPS, тоді як Raspberry Pi 3 Model B+ падає до 16,2 FPS із максимальним часом обробки кадру 106,6 мс. В умовах динамічного навантаження з наявністю об'єктів у кадрі Raspberry Pi 3 Model B+ опускається до критичних 10,4 FPS із максимальним часом обробки 154,9 мс, що виходить за межі вимоги системи реального часу, тоді як Raspberry Pi 4 Model B зберігає стабільні 29,5 FPS.

Порівняно з існуючими аналогами розроблена система має такі переваги: значно нижча вартість апаратної платформи порівняно зі спеціалізованими

рішеннями на кшталт NVIDIA Jetson Nano при збереженні достатньої продуктивності для задач базового комп'ютерного зору; відкрита архітектура програмного забезпечення на базі загальнодоступних бібліотек Python та OpenCV; модульна структура, що спрощує масштабування та адаптацію системи до різних типів БПЛА та задач моніторингу.

Практичне значення отриманих результатів полягає у тому, що розроблена система може бути безпосередньо застосована у БПЛА малого класу для задач відеоспостереження, моніторингу територій, пошуково-рятувальних операцій та охорони об'єктів у цивільній сферах – від агросектору до промислового інспектування.

Результати порівняльного аналізу продуктивності платформ Raspberry Pi 3 Model B+ та Raspberry Pi 4 Model B можуть бути використані розробниками роботизованих систем при виборі апаратної платформи для конкретного застосування з урахуванням вимог до продуктивності, вартості та енергоспоживання.

Подальший розвиток системи передбачає дослідження ефективності обробки відеоданих на віддалених обчислювальних серверах порівняно з бортовою обробкою, інтеграцію нейромережевих алгоритмів виявлення об'єктів на базі моделей YOLO або MobileNet після оснащення бортового модуля апаратним прискорювачем, а також розширення функціональності системи для підтримки автономного режиму польоту без участі оператора.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Акумулятор 18650 Samsung INR 18650 35E. URL: https://rozetka.com.ua/ua/571098754/p571098754/ (дата звернення: 05.06.2026).

2. Бондаренко О. А., Обухова К. О. Система дистанційного керування БПЛА через розподілені ресурси. Інформаційні технології: теорія та практика : тези доп. III (IX) Міжнар. наук.-практ. Інтернет-конф. здобувачів вищої освіти і молодих учених. Харків, 25–27 бер. 2026 р. Харків : ХНУМГ ім. О.М. Бекетова, 2026. С. 418–421. URL: https://surli.cc/qokqra (дата звернення: 03.05.2026).

3. Вентилятор з радіатором для Raspberry Pi. URL: https://prom.ua/ua/p2541782583-ventilyator-radiatorom-50x25.html (дата звернення: 03.05.2026).

4. Зроби сам автономний дрон з Raspberry Pi, Pixhawk та машинним навчанням. URL: https://www.reddit.com/r/diydrones/comments/1h93h0i/diy_autonomous_drone_with_raspberry_pi_pixhawk/?tl=uk (дата звернення: 03.05.2026).

5. Квадрокоптер DJI Phantom 4. URL: https://www.dji.com/global/support/product/phantom-4 (дата звернення: 03.05.2026).

6. XL4015 понижувальний модуль DC-DC. URL: https://myproject.com.ua/dc-dc-xl4015-ponizhuvalnij-modul-438-v-ua.html (дата звернення: 05.06.2026).

7. Bradski G., Kaehler A. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. Sebastopol : O'Reilly Media, 2017. 990 p.

8. Haameid R., Al-Abudi B., Hassan R. Automatic Object Detection, Labelling, and Localization by Camera’s Drone System. *Iraqi Journal of Science*. 2021. Vol. 62, No. 12. P. 5008–5023. DOI: 10.24996/ijs.2021.62.12.37.

9. Kim S., Shin S., Moon J. UDP-based Extremely Low Latency Streaming. *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. Las Vegas, USA, Jan. 8–11, 2022. P. 94–99. DOI: 10.1109/CCNC49033.2022.9700635.

10. Lai T. Real-time Aerial Detection and Reasoning on Embedded-UAVs in Rural Environments. *IEEE Transactions on Geoscience and Remote Sensing*. 2023. Vol. 61. P. 1–7. DOI: 10.1109/TGRS.2023.3266360.

11. Nasehi M. A Comprehensive Review of FPV Technology: Applications, Advantages, and Future Trends. *Machine Learning Research*. 2025. Vol. 10, Is. 1. P. 25–31. DOI: 10.11648/j.mlr.20251001.13.

12. Raspberry Pi 3 and 4 Model B+ Camera Module. URL: https://www.raspberrypi-spy.co.uk/2013/05/the-official-raspberry-pi-camera-module/ (дата звернення: 03.05.2026).

13. Raspberry Pi 3 Model B+. URL: https://evo.net.ua/raspberry-pi-3-model-b/?srsltid=AfmBOoqcRDrptYj774ttkXytS9sWzC8Un3ZTccxVSC_uptCrZpGFUMmO (дата звернення: 03.05.2026).

14. Raspberry Pi 3 vs 4: Which One Should You Get? URL: https://itsfoss.com/raspberry-pi-3-vs-4/ (дата звернення: 03.05.2026).

15. Raspberry Pi 4 Model B. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (дата звернення: 03.05.2026).

16. Raspberry Pi Introduction. URL: https://www.electronicwings.com/raspberry-pi/raspberry-pi-introduction (дата звернення: 03.05.2026).

17. Safadinho D., Ramos J., Ribeiro R., Filipe V., Barroso J., Pereira A. UAV Landing Using Computer Vision Techniques for Human Detection. *Sensors*. 2020. Vol. 20, No. 3. P. 1–36. DOI: 10.3390/s20030613.

18. Sarkar M., Sahoo P. K. Leveraging Edge Computing for Video Data Streaming in UAV-Based Emergency Response Systems. *Sensors*. 2024. Vol. 24, No. 15. P. 1–23. DOI: 10.3390/s24155076.

19. Tran Q. K., Quang N., Ngo K. Object detection for drones on Raspberry Pi: potentials and challenges. *IOP Conference Series: Materials Science and Engineering*. 2021. Vol. 1109. P. 1–13. DOI: 10.1088/1757-899X/1109/1/012033.

20. Wahab F., Ullah I., Shah A., Khan R. A., Choi A., Anwar M. S. Design and implementation of real-time object detection system based on single-shoot detector and OpenCV. *Frontiers in Psychology*. 2022. Vol. 13. P. 1–17. DOI: 10.3389/fpsyg.2022.1039645.

21. Yun H., Park D. Efficient Object Detection Based on Masking Semantic Segmentation Region for Lightweight Embedded Processors. *Sensors*. 2022. Vol. 22, No. 22. P. 1–22. DOI: 10.3390/s22228890.

ДОДАТОК А

Блок-схеми

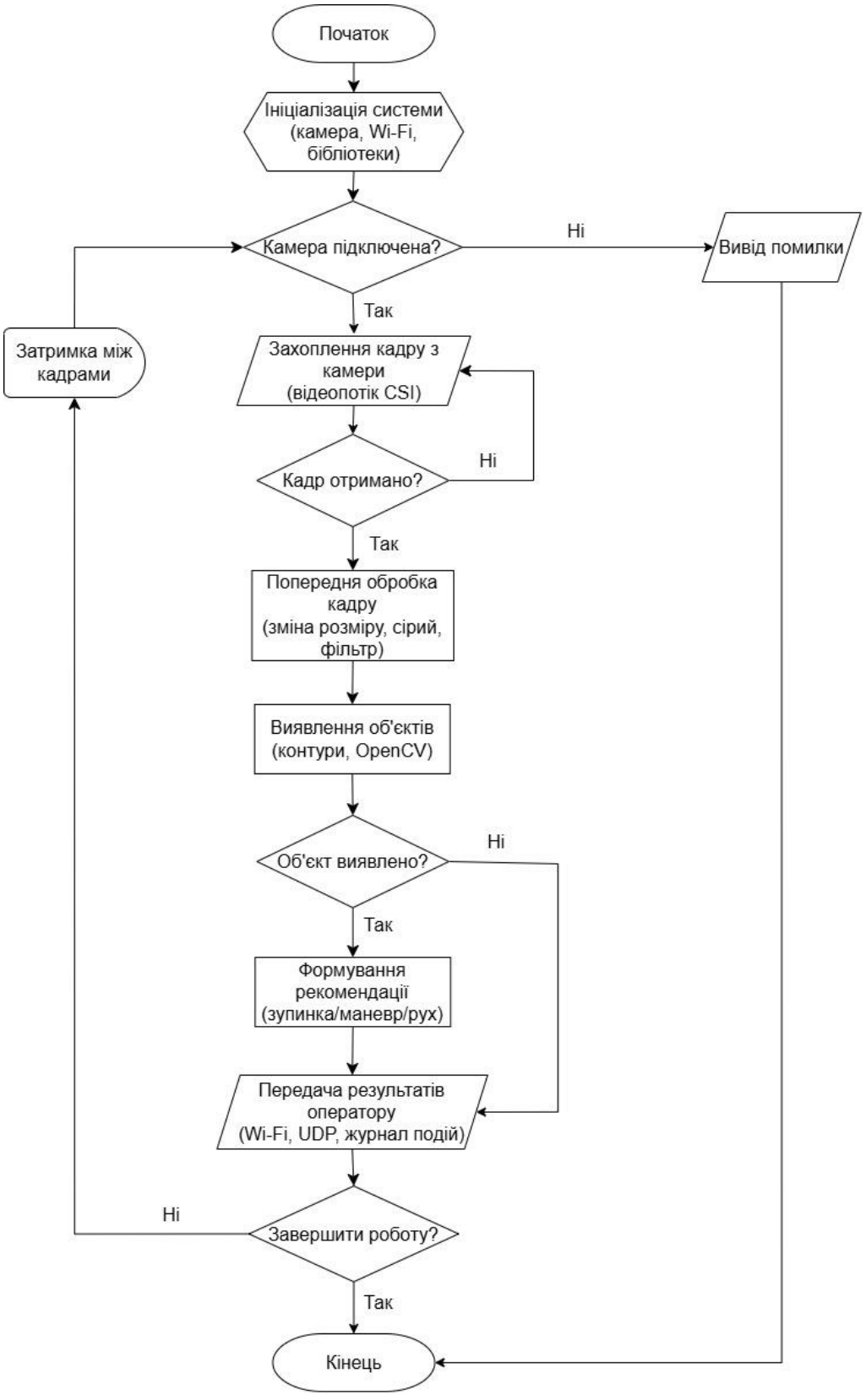

Рисунок А.1 – Загальна блок-схема алгоритму роботи системи

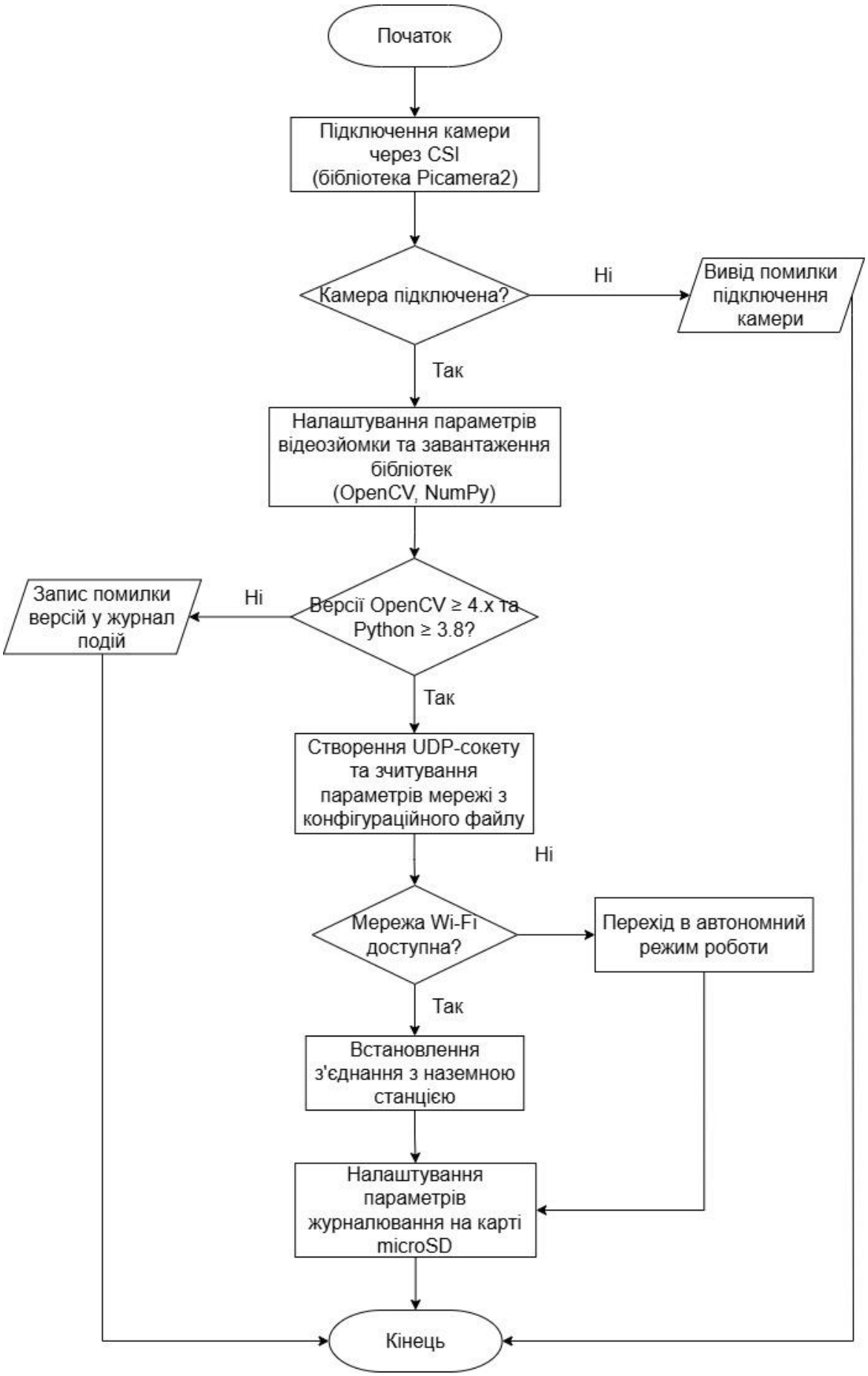

Рисунок А.2 – Блок-схема алгоритму етапу ініціалізації системи

ДОДАТОК Б

Код програми детектування

Лістінг коду main.py

```
import sys
import time
import signal
import logging
import argparse
from datetime import datetime
import config
from camera_module        import CameraModule
from processing_module    import ProcessingModule
from recommendation_module import RecommendationModule
from transmission_module  import TransmissionModule
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(name)s: %(message)s",
    datefmt="%H:%M:%S",
    handlers=[
        logging.StreamHandler(sys.stdout),
        logging.FileHandler("uav_vision_system.log", encoding="utf-8"),
    ],
)
logger = logging.getLogger("main")
_SEPARATOR = "=" * 60
def _print_header() -> None:
    print(_SEPARATOR)
    print("  СИСТЕМА КОМП'ЮТЕРНОГО ЗОРУ БПЛА")
    print("  Дипломна робота: Дистанційне керування БПЛА")
    print(f"  Час запуску: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
    print(_SEPARATOR)
def _print_config() -> None:
    w, h = config.CAMERA_RESOLUTION
    print("\n[КОНФІГУРАЦІЯ]")
    print(f"  Роздільна здатність : {w}x{h} @ {config.CAMERA_FRAMERATE} FPS")
    print(f"  Формат захоплення   : {config.CAMERA_FORMAT}")
```

```
    print(f"     Canny   поріг   1/2             :   {config.CANNY_THRESHOLD1}   /
{config.CANNY_THRESHOLD2}")
    print(f"  Мін. площа контуру  : {config.MIN_CONTOUR_AREA} пікс²")
    print(f"  Пороги рекомендацій :")
    print(f"   Знизити швидкість  ≥ {config.THRESHOLD_SLOW_DOWN * 100:.0f}% кадру")
    print(f"    Змінити напрямок  ≥ {config.THRESHOLD_CHANGE_DIRECTION * 100:.0f}%
кадру")
    print(f"    Негайна зупинка   ≥ {config.THRESHOLD_STOP * 100:.0f}% кадру")
    print(f"  Наземна станція     : {config.GS_IP}:{config.GS_PORT}")
    print(f"  JPEG якість         : {config.JPEG_QUALITY}")
    print(f"  Журнал              : {config.LOG_FILE}")
    print()
def _print_status(cam, proc, rec, tx) -> None:
    print("[СТАТУС МОДУЛІВ]")
    for name, mod in [("CameraModule", cam), ("ProcessingModule", proc),
                      ("RecommendationModule", rec), ("TransmissionModule", tx)]:
        s = mod.status()
        state = "АКТИВНИЙ" if s.get("active") else "✗ ЗУПИНЕНО"
        print(f"  {name:<22} {state}")
    print(_SEPARATOR)
    print("  Натисніть Ctrl+C для зупинки\n")
def _print_stats(proc, rec, tx) -> None:
    fps     = proc.current_fps
    avg_ms  = proc.avg_proc_time_ms
    s_tx    = tx.status()
    cpu     = s_tx["cpu_percent"]
    temp    = s_tx["cpu_temp"]
    last_r  = rec.last_recommendation
    line = (
        f"\r  FPS: {fps:5.1f} | "
        f"Час: {avg_ms:5.2f}мс | "
        f"CPU: {cpu:4.1f}% | "
        f"Темп: {temp:4.1f}°C | "
        f"Пакетів: {s_tx['packets_sent']:6d} | "
        f"Рек: {last_r:<22}"
    )
    sys.stdout.write(line)
```

```
    sys.stdout.flush()
def main():
    parser = argparse.ArgumentParser(description="Система комп'ютерного зору БПЛА")
    parser.add_argument(
        "--resolution", choices=["low", "high"], default="low",
        help="low = 640×480, high = 1280×720 (за замовчуванням: low)"
    )
    args = parser.parse_args()
    if args.resolution == "high":
        config.CAMERA_RESOLUTION = config.RESOLUTION_HIGH
    else:
        config.CAMERA_RESOLUTION = config.RESOLUTION_LOW
    _print_header()
    _print_config()
    logger.info("Ініціалізація модулів...")
    camera_module = CameraModule()
    proc_module   = ProcessingModule(camera_module)
    rec_module    = RecommendationModule(proc_module)
    tx_module     = TransmissionModule(rec_module)
    shutdown_event = signal.Event() if hasattr(signal, "Event") else None
    import threading
    shutdown_event = threading.Event()
    def _shutdown_handler(signum, frame):
        print("\n\n  [ЗУПИНКА] Отримано сигнал завершення...")
        shutdown_event.set()
    signal.signal(signal.SIGINT,  _shutdown_handler)
    signal.signal(signal.SIGTERM, _shutdown_handler)
    try:
        logger.info("Запуск CameraModule...")
        camera_module.start()
        logger.info("Запуск ProcessingModule...")
        proc_module.start()
        logger.info("Запуск RecommendationModule...")
        rec_module.start()
        logger.info("Запуск TransmissionModule...")
        tx_module.start()
        time.sleep(1.0)
        _print_status(camera_module, proc_module, rec_module, tx_module)
```

```
        while not shutdown_event.is_set():
            _print_stats(proc_module, rec_module, tx_module)
            time.sleep(0.5)
    except Exception as exc:
        logger.critical("Критична помилка: %s", exc, exc_info=True)
    finally:
        print("\n\n[ЗАВЕРШЕННЯ РОБОТИ]")
        logger.info("Зупинка TransmissionModule...")
        tx_module.stop()
        logger.info("Зупинка RecommendationModule...")
        rec_module.stop()
        logger.info("Зупинка ProcessingModule...")
        proc_module.stop()
        logger.info("Зупинка CameraModule...")
        camera_module.stop()
        print(_SEPARATOR)
        print(f"  Оброблено кадрів : {proc_module.frames_processed}")
        print(f"  Прийнято рішень  : {rec_module.decisions_total}")
        print(f"  Пакетів надіслано: {tx_module.packets_sent}")
        print(f"  Байт передано    : {tx_module.bytes_sent}")
        print(_SEPARATOR)
        logger.info("Система завершила роботу.")
if __name__ == "__main__":
    main()
```

ДОДАТОК В

Матеріали апробації роботи

УДК 004.382.7:004.932

Бондаренко О.А.[1], Обухова К. О.[2]

СИСТЕМА ДИСТАНЦІЙНОГО КЕРУВАННЯ БПЛА ЧЕРЕЗ РОЗПОДІЛЕНІ РЕСУРСИ

Розвиток роботизованих систем, таких як безпілотні літальні апарати (БПЛА) або наземні безпілотні апарати, упродовж останніх років значно прискорився завдяки поширенню доступних обчислювальних платформ та засобів комп’ютерного зору. Сучасні БПЛА дедалі частіше використовуються для виконання завдань відеоспостереження, моніторингу територій, пошуково-рятувальних операцій та систем безпеки. Тому одним із ключових завдань таких систем є аналіз відеоданих у реальному часі для подальшого прийняття керуючих рішень на основі результатів розпізнавання об’єктів.

У сучасних системах керування БПЛА обробка відеоданих може виконуватися як безпосередньо на борту апарата (edge-обчислення), так і з використанням віддалених обчислювальних ресурсів, зокрема хмарних платформ. Такий підхід дозволяє реалізувати розподілену архітектуру системи, у якій обчислювальне навантаження розподіляється між бортовими та віддаленими серверами.

Отже, ефективність роботи подібних систем значною мірою залежить від обчислювальних ресурсів апаратної платформи, на якій реалізовано алгоритми обробки даних. Для невеликих автономних роботизованих пристроїв часто застосовуються одноплатні комп’ютери, які поєднують компактні розміри, низьке енергоспоживання та достатню продуктивність для виконання завдань керування.

Серед платформ такого типу одним із найпоширеніших є сімейство одноплатних комп’ютерів Raspberry Pi. Різні покоління цих пристроїв відрізняються обчислювальною потужністю, обсягом оперативної пам’яті та можливостями обробки мультимедійних даних. Це дозволяє використовувати їх як приклад апаратних платформ із різними рівнями масштабованості ресурсів.

У зв’язку з цим актуальним є дослідження впливу апаратних характеристик обчислювальної платформи на швидкість та стабільність роботи алгоритмів комп’ютерного зору в роботизованих системах.

Метою роботи є дослідження можливостей використання різних обчислювальних платформ для аналізу відеопотоку в системі

[1] Здобувач групи 405, Чорноморський національний університет імені Петра Могили
[2] Старший викладач кафедри комп’ютерної інженерії, Чорноморський національний університет імені Петра Могили

дистанційного керування БПЛА з метою виявлення небезпечних об'єктів або перешкод для роботизованої системи та оцінювання їхнього впливу на швидкість формування рекомендацій для оператора.

Основою запропонованої системи є бортовий обчислювальний модуль БПЛА, реалізований на базі одноплатного комп'ютера, до якого підключено камеру для отримання відеопотоку. Відеодані обробляються в режимі реального часу за допомогою алгоритмів комп'ютерного зору. Після аналізу відеокадру система формує рекомендацію щодо подальших дій БПЛА, яка передається оператору. Остаточне рішення щодо виконання маневру або іншої дії приймається оператором.

Для проведення дослідження використано три апаратні платформи – Raspberry Pi 3 Model B+, Raspberry Pi 4 та Raspberry Pi 5. Raspberry Pi 3 Model B+ оснащений чотириядерним процесором ARM Cortex-A53 із тактовою частотою 1,4 ГГц та 1 ГБ оперативної пам'яті. Raspberry Pi 4 використовує процесор Cortex-A72 з тактовою частотою до 1,5 ГГц, що забезпечує вищу обчислювальну продуктивність та підтримує більший обсяг оперативної пам'яті. Найновіша платформа – Raspberry Pi 5 – оснащена процесором ARM Cortex-A76 з тактовою частотою до 2,4 ГГц, що суттєво підвищує можливості виконання завдань комп'ютерного зору в режимі реального часу. Основні технічні характеристики використаних платформ наведено в таблиці 1.

Таблиця 1 – Порівняння основних характеристик одноплатних комп'ютерів Raspberry Pi

Характеристика	Raspberry Pi 3 Model B+	Raspberry Pi 4	Raspberry Pi 5
Процесор	ARM Cortex-A53	ARM Cortex-A72	ARM Cortex-A76
Кількість ядер	4	4	4
Тактова частота, ГГц	1,4	до 1,5	до 2,4
Оперативна пам'ять, Гбайт	1	2–8	4–8
Архітектура	ARMv8	ARMv8	ARMv8-A

Як видно з таблиці, Raspberry Pi 4 має більш потужніші обчислювальні характеристики порівняно з Raspberry Pi 3 Model B+, що потенційно дозволяє ефективніше виконувати завдання обробки відеоданих. Raspberry Pi 5 демонструє ще вищу продуктивність завдяки новішій архітектурі процесора та підвищеній тактовій частоті.

Структурну архітектуру роботизованої системи наведено на рис. 1.

Рисунок 1 – Структурна архітектура роботизованої системи

Відеопотік із камери обробляється покадрово в режимі реального часу. Кожен кадр проходить етапи попередньої обробки, що включають зміну роздільної здатності кадру та перетворення у відтінки сірого.

Після попередньої обробки виконується виявлення потенційно небезпечних об’єктів або перешкод на шляху руху БПЛА за допомогою алгоритмів комп’ютерного зору, реалізованих у бібліотеці OpenCV. У разі виявлення такого об’єкта система формує рекомендацію щодо можливих дій БПЛА (наприклад, зміни траєкторії руху або зниження швидкості) та передає її оператору для прийняття остаточного рішення.

Для оцінювання ефективності роботи системи планується дослідити такі параметри:

- час обробки одного кадру;
- середню кількість кадрів за секунду (FPS);
- завантаження центрального процесора;
- стабільність роботи системи при різній роздільній здатності відео.

420

Порівняння отриманих результатів дозволить оцінити ефективність використання різних апаратних платформ для реалізації завдань комп'ютерного зору в роботизованих системах.

Висновки. Очікується, що використання більш сучасних апаратних платформ забезпечить вищу частоту обробки кадрів та менше навантаження на процесор за однакових умов роботи, що дозволить підвищити швидкість реакції роботизованої системи. Raspberry Pi 3 Model B+ водночас продемонструє обмеження продуктивності при збільшенні роздільної здатності відео, що може призводити до зростання затримки обробки та прийняття рішень.

Таким чином, ефективність управління роботизованою системою безпосередньо залежить від рівня доступних обчислювальних ресурсів. Тому при дослідженні адаптивної архітектури керування потрібно враховувати продуктивність апаратної платформи та забезпечувати оптимальний баланс між швидкодією й точністю розпізнавання.

Отримані результати можуть бути використані під час подальшого розроблення системи дистанційного керування БПЛА з розподіленою архітектурою обчислень, у якій обробка відеоданих може виконуватися як на бортовому комп'ютері, так і на віддалених обчислювальних серверах або хмарних платформах.

У подальших дослідженнях передбачається порівняти ефективність локальної обробки відеоданих на бортовому комп'ютері з обробкою на віддалених обчислювальних серверах або хмарних платформах.

ПЕРЕЛІК ДЖЕРЕЛ

1. Raspberry Pi Foundation. (2026). *Raspberry Pi 3 product brief*. https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

2. Raspberry Pi Foundation. (2026). *Raspberry Pi 4 product brief*. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

3. OpenCV Team. (2026). *OpenCV documentation*. https://docs.opencv.org/

4. Diachok, R., Tepliako, I., Lapko, M., & Popov, A. (2025). Efficiency and accuracy comparison of PIR, OpenCV with a webcam and Raspberry Pi. *Advances in Cyber-Physical Systems, 10*(1), 77–82. https://doi.org/10.23939/acps2025.01.077