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

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Реалізація каналу зв'язку між Raspberry Pi та Android для детектування об'єктів у реальному часі

Спеціальність 123 Комп’ютерна інженерія

Освітня програма «Комп’ютерна інженерія»

Здобувач _____*Ілля БОНДАРЕНКО*
підпис

«__» _____ 202__ р.

Керівник ст. викладач

_____ Катерина ОБУХОВА
підпис
«__» _____ 202__ р.

Миколаїв – 2025

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
«_____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Бондаренко_Ілля Юрійович_____
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Реалізація каналу зв'язку між Raspberry Pi та Android для детектування об'єктів у реальному часі.

Затверджена наказом по ЧНУ ім. Петра Могили від 30.10.2024 № 297.

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

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є реалізування апаратного та програмного забезпечення каналу зв’язку між одноплатним комп’ютером Raspberry Pi та Android-пристроєм.

4. Перелік питань, що підлягають розробці:
- обґрунтування актуальності проблеми розпізнавання перешкод та проведення аналізу теоретичної складової розробки проєкту та існуючих рішень;
- вибір програмної платформи для реалізації процесу фіксування та передачі даних, а також методи та технології для формулювання вимог до АПЗ;
- проєктування системи, включаючи визначення архітектури та взаємодії компонентів;
- розробка програмне забезпечення, реалізація алгоритму для детектування перешкод та інтеграції з апаратною платформою;
- тестування та набуття навичок керування FPV-дроном.

5. Перелік графічних матеріалів

Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Катерина ОБУХОВА

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Ілля БОНДАРЕНКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «29» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: __Реалізація каналу зв'язку між Raspberry Pi та Android для детектування об'єктів у реальному часі

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	01.11.2024	02.11.2024	Виконано
2	Огляд літератури за темою роботи	03.11.2024	10.11.2024	Виконано
3	Складання календарного плану КБР	11.11.2024	12.11.2024	Виконано
4	Аналіз предметної області	06.05.2025	13.05.2025	Виконано
5	Розробка проєктних рішень	06.05.2025	13.05.2025	Виконано
6	Моделювання та конструювання АПЗ	01.05.2025	30.05.2025	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	30.05.2025	12.06.2025	Виконано
8	Відгук керівника КБР	14.06.2025	14.06.2025	Виконано
9	Оформлення КБР та презентації	06.05.2025	04.06.2025	Виконано
10	Попередній захист	21.05.2025	04.06.2025	Виконано
11	Рецензування	14.06.2025	16.06.2025	Виконано
12	Завершення оформлення КБР та презентації	04.06.2025	16.06.2025	Виконано
13	Захист кваліфікаційної бакалаврської роботи	25.06.2025	25.06.2025	Виконано

Керівник роботи _____ __Катерина ОБУХОВА

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ ___Ілля БОНДАРЕНКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Реалізація каналу зв'язку між Raspberry Pi та Android
для детектування об'єктів у реальному часі»
Студент 405 гр.: Бондаренко Ілля Юрійович
Керівник: ст. викладач Обухова Катерина Олександрівна

У зв’язку з розширенням сфер використання безпілотних літальних апаратів (БПЛА), зокрема FPV-дронів, зростає потреба у створенні ефективних систем автономного моніторингу. Важливу роль у цьому відіграє здатність до виявлення об'єктів і перешкод у реальному часі. Існуючі рішення не завжди задовольняють вимоги щодо вартості, енергоефективності та технічних характеристик, що актуалізує розробку оптимізованих апаратно-програмних забезпечень із застосуванням сучасних технологій комп’ютерного зору.

Акцент зроблено на важливості реалізації швидкого, стабільного та енергоефективного каналу зв’язку між елементами системи. Це дає змогу досягти високої точності розпізнавання об’єктів, зменшити затримки передачі даних та забезпечити користувача оперативною інформацією. Реалізація взаємодії через локальну бездротову мережу та застосування протоколів RTSP сприяє підвищенню адаптивності системи до різних технічних умов.

Об’єктом дослідження є процес передачі даних між одноплатним комп’ютером і мобільним пристроєм для подальшої обробки та детектування об’єктів у режимі реального часу.

Предметом дослідження є методи та засоби реалізації ефективного каналу зв’язку між Raspberry Pi та Android-пристроєм для забезпечення передачі даних детектування об’єктів у реальному часі.

Метою роботи є створення ефективного програмно-апаратного забезпечення для виявлення перешкод та координування дій оператора з використанням TensorFlow-фреймворку.

Практична значимість полягає в реалізації стабільного каналу передачі зображень і результатів детектування з одноплатним комп’ютером Raspberry Pi на Android-пристрій із мінімальною затримкою, що забезпечує оперативне інформування оператора. Використання оптимізованого фреймворку TensorFlow Lite дозволяє інтегрувати алгоритми комп’ютерного зору в системи з обмеженими ресурсами.

Пояснювальна записка кваліфікаційної бакалаврської роботи (КБР) складається зі вступу, трьох розділів, висновків, переліку джерел посилання та 3 додатків.

У вступі обґрунтовано актуальність проблематики автоматизованого моніторингу, визначено об’єкт і предмет дослідження, сформульовано мету та завдання роботи, зазначено апробацію матеріалів КБР.

У першому розділі проведено аналітичний огляд предметної сфери та аналіз існуючих систем детектування, обґрунтовано вибір апаратного та програмного забезпечення, з урахуванням обмежень платформи та цінової доступності, а також ергономічних та вагових параметрів.

У другому розділі розроблено алгоритм функціонування системи, який включає захоплення фото- або відеопотоку, детектування об’єктів та маркування рівня небезпеки. Проведено технічний аналіз впливу компонентів на характеристики дрона, що дозволило визначити ключові обмеження та рішення для ефективної роботи системи в польових умовах.

У третьому розділі реалізовано програмне забезпечення із використанням фреймворку TensorFlow Lite та бібліотеки OpenCV, забезпечено підключення камери та розроблено Android-застосунок для обробки та виводу результатів, а також реалізовано архітектуру апаратного забезпечення, розташування компонентів відповідно до ценрту тяжіння РТС.

У висновках підсумовано результати виконаної роботи, проаналізовано ефективність розробленої системи та сформульовано основні висновки на основі проведених досліджень.

В цілому КБР містить 68 сторінок (без додатків), 35 рисунків, 3 таблиці, 23 джерела посилання, 3 додати.

Ключові слова: роботизована технічна система, FPV-дрон, Raspberry Pi 4, фреймворк TensorFlow Lite, детектування об’єктів та перешкод.

ABSTRACT

of the Bachelor's Thesis
"Implementation of a communication channel between Raspberry Pi and Android for real-time object detection"
Student: Bondarenko Illia
Supervisor: Senior Lecturer Obukhova Kateryna

Due to the expanding range of applications of unmanned aerial vehicles (UAVs), particularly FPV drones, the demand for efficient autonomous monitoring systems is increasing. A key aspect of such systems is the ability to detect objects and obstacles in real time. Existing solutions often fail to meet requirements for cost, energy efficiency, and technical performance, highlighting the need for optimized hardware-software complexes employing modern computer vision technologies.

The focus is placed on implementing a fast, stable, and energy-efficient communication channel between system components. This enables high-accuracy object recognition, reduces data transmission latency, and provides real-time information to the user. The use of a local wireless network and implementation of RTSP protocol improves system adaptability to various technical conditions.

The object of the study is the process of data transmission between a microcomputer and a mobile device for subsequent processing and object detection in real time.

The subject of the study is the methods and means of implementing an efficient communication channel between a Raspberry Pi and an Android device to ensure real-time object detection data transfer.

The goal of the study is to develop an effective hardware-software solution for obstacle detection and operator coordination using the TensorFlow framework.

The practical significance lies in implementing a stable transmission channel for images and detection results from the Raspberry Pi microcontroller to an Android device with minimal latency, thereby ensuring timely operator awareness. The use of the optimized TensorFlow Lite framework enables the integration of computer vision algorithms into resource-constrained systems.

This bachelor's thesis explanatory note consists of an introduction, three chapters, conclusions, a list of references, and 3 appendices.

The introduction justifies the relevance of automated monitoring systems, defines the object and subject of the study, formulates the objectives and tasks of the work, and outlines the practical implementation of the research findings.

Chapter one presents an analytical review of the subject area and existing detection systems. It also substantiates the selection of hardware and software components, taking into account platform limitations, cost-effectiveness, ergonomic and weight considerations.

Chapter two outlines the system's functional algorithm, which includes image or video capture, object detection, and threat-level marking. A technical analysis of component influence on UAV performance was conducted, allowing the identification of key limitations and solutions for effective field deployment.

Chapter three describes the software implementation using the TensorFlow Lite framework and OpenCV library. It includes camera integration and the development of an Android application for result processing and display, as well as implemented hardware architecture, component location according to the center of gravity of the RTS.

The conclusions summarize the results of the thesis, evaluate the performance of the developed system, and formulate the key findings based on the research.

The paper consists of 68 pages (excluding appendices), 35 figures, 3 tables, 23 references, and 3 appendices.

Keywords: robotic technical system, FPV drone, Raspberry Pi 4, TensorFlow Lite framework, object and obstacle detection.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПЗ – апаратно-програмне забезпечення

БПЛА – безпілотній літальний апарат

КБР – кваліфікаційна бакалаврська робота

ОС – операційна система

РТС – робототехнічна система

FPV – First Person View

IP – Internet Protocol

SoC – Security Operations Center

ВСТУП

Сучасний світ потребує швидкого та ефективного опрацювання великої кількості даних у реальному часі. У багатьох сферах, зокрема безпеці, моніторингу та управлінні об'єктами, завдання виявлення й обробки інформації поступово передається від людини до автоматизованих систем. Вони в свою чергу надають змогу удосконалити цей процес, а також зменшити час на виконання такого алгоритму, порівняно з працею, яку виконує лише людина.

Детектуванню об’єктів завжди надавали певну цінність, оскільки це дозволяло оцінити становище досліджуваного сектору та, в залежності від результату, прийняти рішення щодо покращення або радикальної зміни ситуації. З появою безпілотних літальних апаратів, вдалось удосконалити цей процес, адже тепер інформація може передаватися безпосередньо до користувача, без потреби в додаткових проміжних етапах. Оператор залучається лише дотично, керуючи процесом зйомки, тоді як її обробка та детектування лягає на технологічний аспект.

Актуальність теми зумовлена зростаючим попитом з боку потенційних користувачів на доступні та стабільні системи моніторингу зовнішнього середовища. Такі системи здатні здійснювати виявлення (детектування) об’єктів або перешкод і візуально позначати їх, зокрема за рівнем небезпеки. Такий підхід дозволяє оперативно реагувати на зміну умов або загрози. Проте, існуючі рішення нерідко ігнорують баланс між якістю, функціональністю та вартістю, а саме вони не враховують обмеження щодо вартості впровадження, технічних характеристик платформи та енергоефективності, що ускладнює їхнє використання в умовах обмежених ресурсів. Це створює потребу в розробці більш оптимізованих, економічно доцільних рішень.

Прикладне та практичне значення обраної теми полягає у створенні автономної платформи виявлення об'єктів на базі смартфону в режимі реального часу за рахунок включення сучасних інформаційних технологій в галузі комп'ютерного зору, спеціалізованих мобільних обчислювальних систем та бездротових каналів зв'язку на базі РТС. У запропонованому рішенні також

використовується зв'язок між платою Raspberry Pi 4 та Android-пристроєм для реалізації обміну даними на високій швидкості, мінімізуючи при цьому затримку передачі результатів обробки зображень та полегшуючи роботу оператору БПЛА. Підсумок дослідження може бути застосований при вдосконалені систем розвідувального моніторингу, охорони приватного або державного сектору, пошуково-рятувальних роботах, а також у сільському господарстві. Передусім, результат відіграє важливу роль у впровадженні радіокерованих систем в життя людей та є підгрунтям для подальшого покращення робототехніки в цілому.

Об’єктом дослідження є процес передачі даних між одноплатним комп’ютером і мобільним пристроєм для подальшої обробки та детектування об’єктів у режимі реального часу.

Предметом дослідження є методи та засоби реалізації ефективного каналу зв’язку між Raspberry Pi та Android-пристроєм для забезпечення передачі даних детектування об’єктів у реальному часі.

Метою роботи є створення ефективного програмно-апаратного забезпечення для виявлення перешкод та координування дій оператора з використанням TensorFlow-фреймворку.

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

– обґрунтувати актуальність проблем розпізнавання перешкод;

– провести аналіз теоретичної складової розробки проєкту та існуючих рішень;

– обрати програмну платформу для реалізації процесу фіксування та передачі даних, а також методи та технології для формулювання вимог до АПЗ;

– здійснити проєктування та визначити архітектуру системи;

– розробити програмне забезпечення, реалізувати алгоритми та інтегрувати з апаратною платформою;

– виконати тестування, набути навичок керування FPV-дроном через віртуальну симуляцію для оцінки ефективності запропонованого рішення та верифікації результатів в реальних умовах.

Дослідження та реалізація каналу зв’язку саме таким чином, має практичне застосування, адже дозволяє зберегти цілісність цінного обладнання, зменшити людський фактор та слугує помічником в сфері прийняття рішень оператором FPV-дрона.

Таким чином, обране рішення демонструє значимість систем автономного моніторингу при ведені віддаленого керування. Мета роботи була визначена, завдання – вказані, а об’єкт та предмет занотовані.

Апробація: Бондаренко І. Ю. (бакалаврант), Обухова К. О. (ст. викладач каф. комп’ютерної інженерії, Чорноморський нац. ун-т ім. Петра Могили, м. Миколаїв, Україна). Реалізація каналу зв'язку між Raspberry Pi та Android для детектування об'єктів у реальному часі. «Ольвійський форум – 2025: стратегії країн Причорноморського регіону в геополітичному просторі» [1].

1 АНАЛІЗ І КЛАСИФІКАЦІЯ ІСНУЮЧИХ ПІДХОДІВ

1.1 Актуальність методики детектування в реальному часі

З розвитком цифрових технологій, передусім у сфері відеопередачі, та завдяки мініатюризації апаратних компонентів, стало можливим створювати легкі, мобільні комплекси з підтримкою високоякісного відеозв’язку, значною дальністю польоту та підвищеною стабільністю каналу. Це сприяло активному поширенню FPV-дронів у цивільному секторі – зокрема для інспекцій промислових об’єктів, участі в FPV-перегонах, екологічного моніторингу, а також в енергетичній сфері. Важливо зазначити, що кожен аспект повинен відповідати стандартам проєктування та виготовлення. Згідно з дослідженням, присвяченим гоночним FPV-дронам, для радіокерованих систем ключовими характеристиками є маневреність та мала маса, тоді як в окремих випадках тривалістю роботи від акумулятора можна знехтувати [22]. Водночас, важливим параметром таких систем залишається дальність дії відеозв’язку, що визначає ефективність взаємодії оператора з РТС у реальному часі.

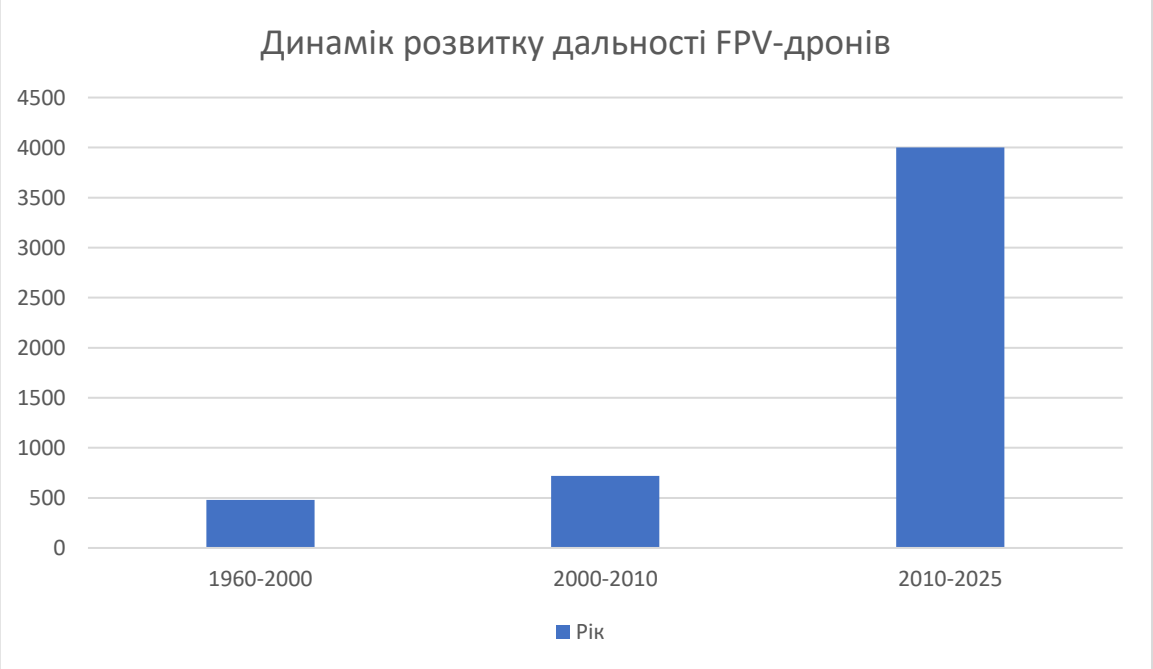


Рисунок 1.1 – Динаміка розвитку дальності сигналу FPV-дронів

На рис. 1.1 представлено діаграму порівняння максимальної дальності передачі відеосигналу для різних типів FPV-систем (аналогові, цифрові, гібридні), яка ілюструє динаміку розвитку технології та зростання її застосовності у сфері автономного моніторингу та виявлення об'єктів [17].

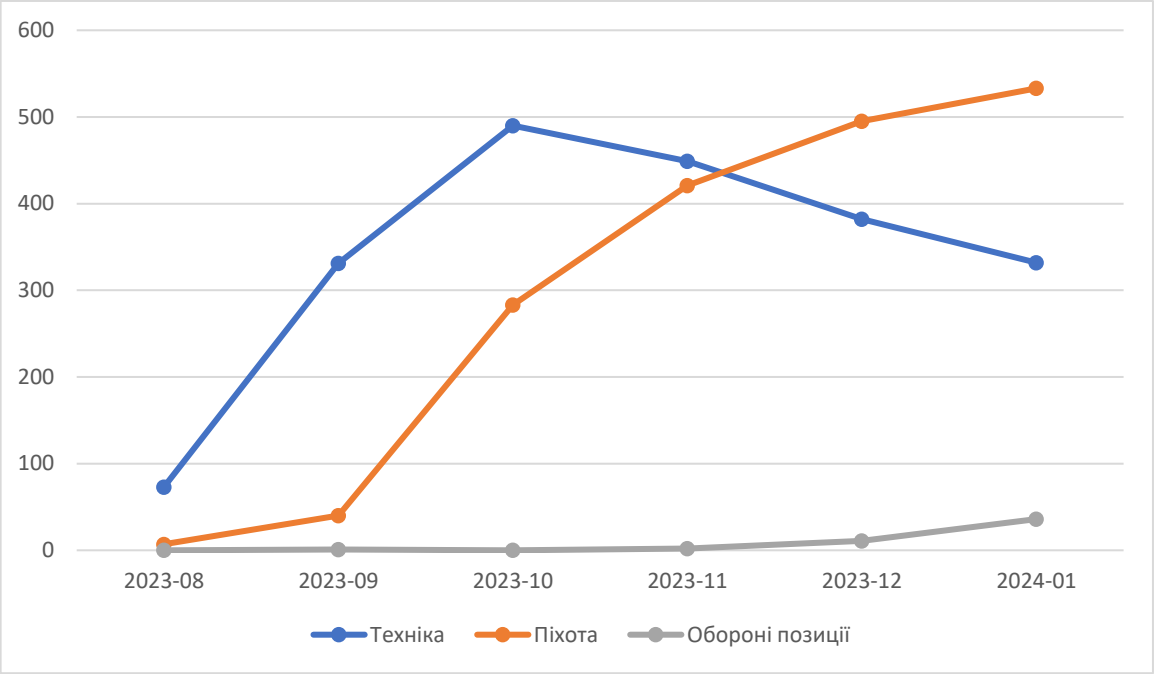


Рисунок 1.2 – Вибірка цілей для FPV-дрону

Дослідження демонструються зростання використання FPV-дронів на період 2023–2024 рр. проти піхоти та зменшення використання проти техніки [18]. Таку різку зміну тенденції можна пов’язати не тільки з недостачею зброї, а й з перерозподілом ресурсів [9].

В той же час, дрони просто втрачаються через збиття їх ворожими силами, глушіння сигналу РЕБ-системами та через порушену координацію пристрою. Тільки 20–40 % безпілотних апаратів, за різними підрахунками, досягають своєї цілі або завершують розвідувальне завдання [10].

Український уряд планує закупити 4,5 млн FPV-дронів на 2025 рік, що втричі більше порівняно з попередніми роками [11].

Попит використання саме FPV-дронів, як зброї обумовлений низкою факторів, серед яких, як залучення молоді до військової справи, так і факт відсутності оператора безпосередньо на лінії зіткнення [5].

Інформаційна обізнаність в таких випадках здатна збільшити відсоткову вірогідність успіху поставлених завдань, оскільки зменшується вплив людського фактору. Всю роботу з детектування покладено на плату з камерою, які при правильному налагодженні повинні розпізнавати рівні небезпеки за трьома критеріями:

– зелений колір, вказує на відсутність небезпеки та можливість безперешкодного перетину сектору;

– жовтий колір, вказує на можливість пошкодження та/або втрати БПЛА;

– червоний колір повідомляє про необхідність негайно покинути територію.

На даний момент, активно практикується впровадження саме таких застосунків до структури керування польотом. Однак, все частіше спосіб передачі інформації ґрунтується на використанні оптоволоконного кабелю, як більш надійного та завадостійкого варіанту. Попри всі переваги, доцільніше використовувати передавач частот, що попередньо налагоджений на потрібну частоту, оскільки таке рішення є дешевшим в порівнянні з оптоволоконним кабелем.

Оператор FPV-дрона, який працює самостійно, виконує досить складне завдання, адже не завжди можливо вчасно виявити замасковану або нерухому перешкоду. Це особливо актуально в умовах складної місцевості або обмеженої видимості, коли звичайного візуального спостереження недостатньо.

Використання одноплатного комп’ютеру на базі Raspberry PI 4 доцільне саме завдяки його характеристикам, що безпосередньо підходять умові ціна-якість, а Android-система є однією з найрозповсюдженіших на території України, що робить її вибір зрозумілим.

1.2 Огляд існуючих рішень

Станом на 2025 рік з ростом попиту на фіксування об’єктів та розпізнавання об’єктів операторами, зросла кількість саморобних систем на базі одноплатних комп’ютерів та фреймворків. Проте, до сих пір, можна примітити

використання комерційних проєктів в громадських цілях, як наприклад, захист окремо-виділеної території, її патрулювання та пошук окремих осіб в пошуково-рятувальних операцій. Для подібних завдань компанія DJI розробила мультикоптер Matrice 300 RTK [7]. Сенсорний екран в поєднанні з керованим дроном та високоякісною камерою, здатні передавати зображення на пульт керування (рис. 1.3).

Рисунок 1.3 – Фіксування цілі в режимі реального часу
DJI Matrice 300 RTK [3]

Дрон використовується для зйомок просторів, а тому має можливість фіксації об’єкта. На базі штучного інтелекту було розроблено алгоритми детектування та розпізнавання об’єктів. Процем збору інформації можливий вночі через вбудовану тепловізійну камеру [3]. Також, мультикоптер підтримує інтеграцію інших сенсорних пристроїв, а отже являє собою один з кращих варіантів для детектування та переслідування об’єкту.

Parrot Anafi USA – спеціально-розроблений дрон для рятувальних та спостережних місій, що застосовується, як для приватних операцій так і державних [12]. Включає в себе вбудовані алгоритми для детектування людей та машин, а також функцію «повернення додому», що повертає дрон або до власника, або до попередньо-заданої точки. Для детектування використовує

термокамеру високої роздільної якості. Можлива передача зображення на телефон (рис. 1.4).

Рисунок 1.4 – Терморежим Parrot Anafi USA [12]

FLIR Systems – FLIR Duo Pro R – камера, що дозволяє детектувати теплові аномалії. Сфера застосування повторює попередні випадки [23]. З особливостей саме цієї моделі, умови освітлення не відіграють значимої ролі, оскільки камера розроблювалась з урахуванням несприятливих погодних умов. Можлива передача даних у реальному часі та інтеграція з іншими моделями дронів.

Рисунок 1.5 – Чорно-білий режим зображення [23]

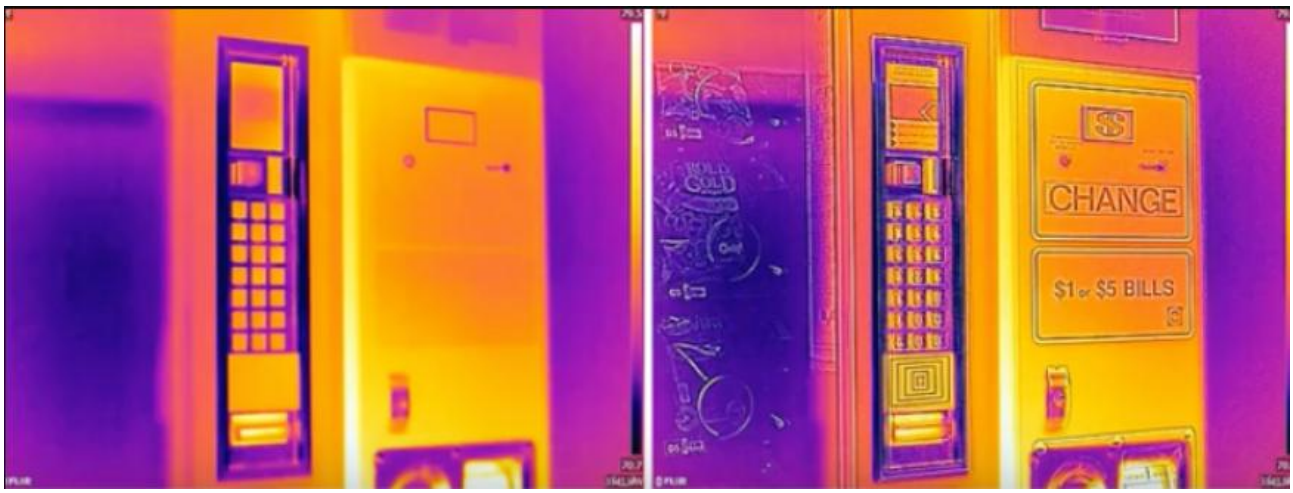


Рисунок 1.6 – Порівняння технологій теплового фіксування (FLIR Duo Pro R справа) [8]

Закордоні аналоги кращі зі сторони якості, проте ціноутворення унеможливлює використання передових технологій у масовому сегменті. До того ж, аналоги хоч і показують перешкоду, проте не застерігають про рівень її небезпеки.

З іншого боку, ведеться розробка FPV-дронів з машинним зором, здатними розпізнавати небезпеку та автоматично наводитись на неї або ж уникати. Для застосування на практиці необхідно активувати окремий важіль на пульті управління. Поки що, розробка на етапі детектування людей, проте удосконалення обіцяють додавання пріоритетності скупченням людей та техніки (рис. 1.7).

Порівняно з іноземними рішеннями, саме це додасть до собівартості БПЛА лише 50 доларів [21]. Тим не менш, основною проблемою для розробників стала розробка системи координат. Рій дронів має розуміти своє місцезнаходження та активуватись одним рухом.

Рисунок 1.7 – Технологія захоплення об’єкту VYRIY [21]

Порівнюючи всі три системи відеоспостереження (табл. 1.1), слід зазначити, що вони не є оптимальними для реалізації виявлення об’єктів та перешкод. Основним негативним критерієм є переплата за здебільшого непотрібні особливості, без яких можна було б обійтись або замінити на дещо гірший, проте значно дешевший варіант. Стосовно сфери використання, РТС орієнтовані на професійне та військове використання, однак, враховуючи той факт, що здебільшого дрони – це витратний матеріал на цей момент часу, переплата не є ключовим об’єктом оцінки. Відсутність відкритої екосистеми також негативно впливає на можливість удосконалення та заміни компонентів вже на сам перед БПЛА.

Отже, для детектування об’єктів доцільне використання відкритих рішень, які легко налаштувати, підлаштовувати, а в деяких випадках, навіть, вдосконалювати.

Таблиця 1.1 – Порівняння основних характеристик камер

Назва дрона	**Тип детектування**	**Роздільна здатність, Мп**	**Вартість**	**Плюси**	**Мінуси**
DJI Zenmuse X7 (DJI Matrice 300 RTK)	Обведення об’єкта	24	Висока	Висока якість зображення та дальність польоту. Можливість використання якісного штучного інтелекту	Висока вартість товару, відсутність детектування окремих об’єктів в режимі реального часу, фіксація тільки на одному
FLIR Boson 320 (Parrot Anafi USA)	Терморежим	21	Середня–висока	Висока якість зображення та дальність польоту. Підходить для ведення довготривалих операцій	Відсутність детектування деяких об’єктів
FLIR Duo Pro R	Терморежим та чорно-білий режим	12	Середня–висока	Якісний терморежим з можливістю розгляду деталей об’єкту. Наявність додаткового режиму	Найменша роздільна здатність

Реальний світ та ситуація вносять свої корективи. Дороге рішення не завжди якісне та оптимальне для виконання певних задач. Інколи масовість та посередня якість краще, ніж однорідність та функціональність.

1.3 Вимоги до системи

У цьому розділі розглядаються вимоги до системи виявлення об'єктів для FPV-дрона, яка використовує одноплатний комп’ютер та мобільний пристрій Android. Система повинна працювати в реальному часі, мінімізувати затримки та своєчасно передаючи попередження оператору, попередньо перефарбовуючи їх в відповідні кольори. Сферою застосування виступає FPV-дрон, як пристрій переміщення фіксуючої апаратної частини.

Слід виокремити потенційних користувачів, що потребують послуги з детектування об'єктів. До цього списку входять рятувальники, адже іноді виникає необхідність у виявленні об'єктів, недосяжних для людини. Така система здатна проникати в важкодоступні місця та виявляти загрозу або жертву катаклізму. Чергове застосування, пошук людей або об'єктів в великому діапазоні. PTC допомагає швидко переміщуватись по сектору, а IP-камера якісно передавати зображення. Окрім цього, наразі FPV-дрони використовують і в військових цілях.

Апаратне забезпечення. Основним компонентом є Raspberry Pi 4 Model B [20], який бере на себе обробку даних. Його потужності буде достатньо для реалізації TensorFlow Lite на низькому рівні FPS. Крім того, він підтримує підключення до камер, периферії, а також бездротовий зв'язок через Wi-Fi, що дозволяє передавати дані на Android без зайвих затримок. При дослідженні буде використовуватись саме такий метод передачі інформації. Проте, на рівні з ним можливе використання третьої та нульової моделі, проте ефективність таких рішень ставиться під сумнів через навантаження на центральний процесор.

Вибір саме 4 версії комп’ютера обумовлена кореляцією ефективність-стабільність, що випереджає попередні версії контролера, більшою кількістю FPS, чого буде достатньо для розпізнавання об’єкту та зроблені фотографій. Кількість FPS можливо доповнити, шляхом додавання акселератору Coral USB.

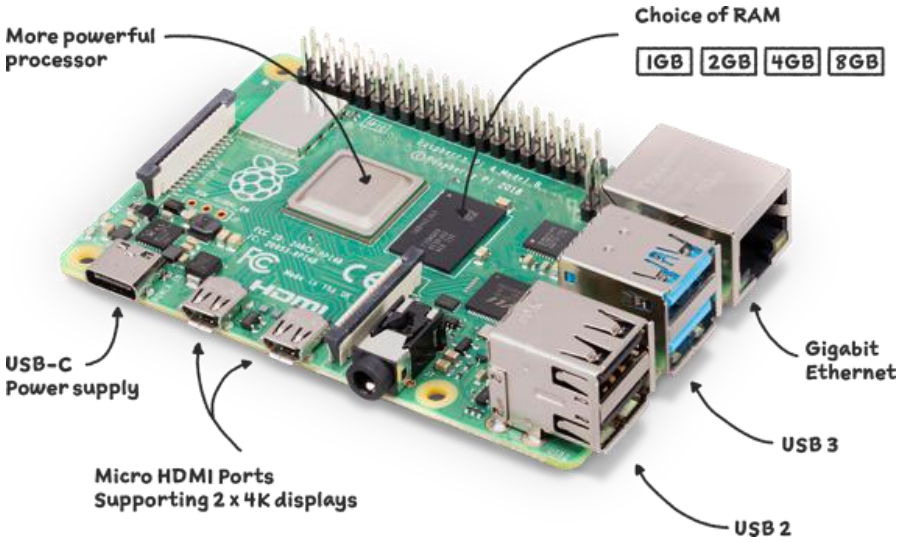


Рисунок 1.8 – Raspberry Pi 4 Model B та його складові [20]

Основні характеристики Raspberry Pi 4 Model B:

– процесор: Broadcom BCM2711, ARM Cortex-A72 (64-біт), 4 ядра, 1,5 ГГц;

– графічний процесор (GPU): Broadcom VideoCore VI, підтримка OpenGL ES 3.0 і OpenCL;

– оперативна пам'ять (RAM): LPDDR4-3200 SDRAM: 8 Гбайт;

– Wi-Fi: 802.11ac, Ethernet: Gigabit;

– порти вводу/виводу (I/O): USB 3.0 (2 порти), USB 2.0 (2 порти), HDMI (2 порти micro-HDMI), CSI (камера);

– MicroSD карта (підтримка до 1 Тбайт);

– потужність: 5V через USB-C (3A).

Дрон виступає сферою застосування, а отже займає вагоме місце в процесі підбору компонентів для майбутньої роботи. Можливе використання дронів DJI, так як умови проведення дослідження дозволяють його використання. 30 хв вільного польоту буде достатньо для фіксування даних та отримання результатів. Швидкість апаратного забезпечення не відіграє важливої ролі, тому нею можна знехтувати та дотримуватись 30 км/год, що є середньою для цього БПЛА. Дальність передачі становить 20 км, за допомогою O4 Transmission.

Головною умовою для підбору компонентів стало їх розміщення та вага, котра не повинна була перевищувати 300 г. На рис. 1.9 наведено схематичне планування РТС, показано ергономіку та правильне розташування компонентів. Враховується аеродинаміка та центр тяжіння для комфортного та ефективного керування.

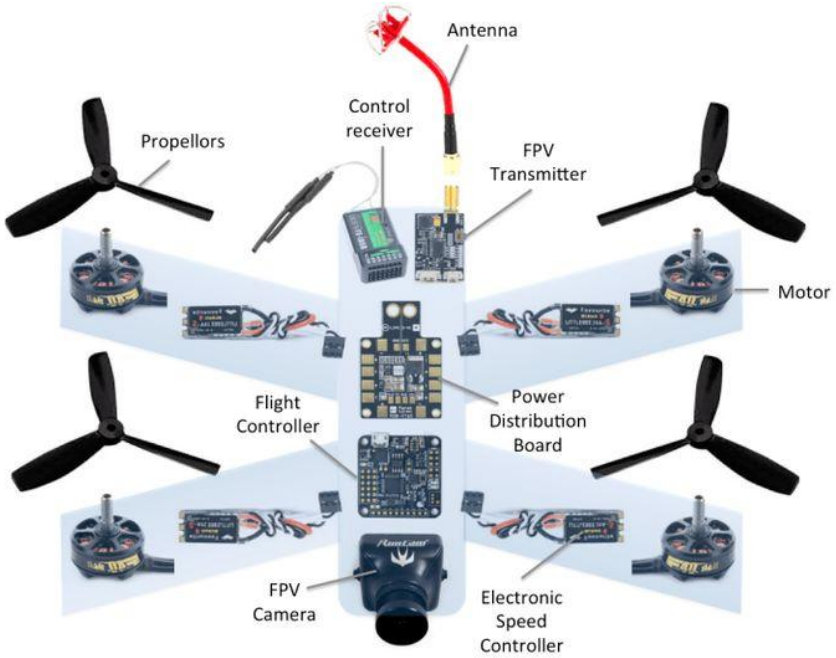


Рисунок 1.9 – Ергономіка РТС [2]

Додатково, плата не застрахована від перегріву, адже використання плати з РТС потребує значних обчислювальних ресурсів для запуску фреймворку та високого коефіцієнту надійності, щоб у разі перегріву система не застосовувала троттлінг або ж зовсім не викнулась. Рішення повинно охолоджувати центральний процесор та забезпечувати тривалу роботу плати.

Такий варіант кейсу є оптимальним, оскільки здатний захистити плату від випадкової деформації зі сторони зовнішнього середовища та підтримувати сталу температуру процесора. Він не екранує Wi-Fi та Bluetooth модулі, а кулери майже не видають звуку, порівняно з іншими моделями. Проте, можливе застосування і простіших рішень, як наприклад, кейс роздрукований на 3D-принтері з використанням полімерних матеріалів. Його характеристик буде достатньо для захисту від пилу та дефектів зовнішнього середовища, однак

модель повинна мати місце для охолоджувальної системи, як то кулер чи радіатор.

Реалізація проєкту залучає використання IP камери, для якої у Raspberry є окремий роз’єм. Загалом, можливе встановлення майже будь-якої камери, проте попри специфікацію та сферу застосування саме цього дослідження, необхідно забезпечити стабілізацію камери на дроні та чіткість зображення. Ідеальним рішенням може стати камера з режимами теплодетектування та нічного бачення. Проте, така камера за собівартістю буде перевищувати всі складові в цілому, а тому доцільне використання простіших версій.

Функціональні характеристики IP-камери:

– роздільна здатність: мінімум 720p (HD), бажано 1080p;
– інтерфейс підключення: Wi-Fi (802.11 b/g/n) або Ethernet;
– кут огляду: від 100° до 120° в залежності від задачі;
– клас захисту не нижче IP65 для використання в польових умовах.

Raspberry Pi не має вбудованого жорсткого диска або іншого внутрішнього накопичувача, тому для зберігання операційної системи, програмного забезпечення та даних потрібно використовувати саме microSD карту. Всі необхідні файли, від операційної системи до інструментів для роботи з комп'ютерним зором, будуть зберігатися на карті пам'яті.

Необхідні характеристики для microSD карти:

– тип: UHS-I, UHS-II або A1/A2;
– 32 Гбайт, але залежно від потреб проєкту;
– швидкість читання: 100 Мбіт/с і більше для ефективної обробки даних;
– швидкість запису: мінімум 10 Мбіт/с для базових задач, рекомендовано 30–90 Мбіт/с для відео чи великих файлів.

Програмне забезпечення:

Архітектура програмної системи є багаторівневою. Вона складається з блоку локальної обробки даних на Raspberry Pi 4, блоку взаємодії з камерою, та мобільного додатку на Android.

Вибір фреймворку TensorFlow Lite обумовлений використанням одноплатного комп’ютера, адже його ресурси обмежені та їх буде недостатньо для використання в комплексі зі звичайною TensorFlow версією. Окремо слід зазначити можливість інтеграції бібліотеки TensorFlow Lite в Android-застосунок.

TensorFlow Lite діє спільно з спеціалізованими моделями, такими як MobileNet, YOLO або SSD [13]. Саме вони дозволяють розпізнати та детектувати об’єкти в реальному часі й оптимізувати роботу фреймворку на платах малої потужності.

Рисунок 1.10 – Raspberry Pi 4 Model B в комплексі з TensorFlow Lite [15]

Результати надсилаються на смартфон через локальну або WiFi-мережу.

Окрім TensorFlow Lite, обов’язковим є підключення бібліотеки OpenCV. Її функція захоплення відео з камери та обробка зображень. При встановленні бібліотек, можуть виникати конфлікти. Virtualenv – інструмент, що попереджує їх виникнення та створює ізольоване середовище Python.

Операційна система – Raspberry Pi OS забезпечує повну сумісність із Python, OpenCV та TensorFlow Lite. Операційна система Android не менша за 9 версію, адже дозволяє використання протоколів HTTP та WebSocket. Пофарбовані зображення передаються як JSON-повідомлення.

Основна мова програмування власне Python. Саме вона відповідає за написання скриптів обробки зображень та фарбування їх в різні кольори. Плюсом застосування саме цих технологій є їхня відкритість, що передбачає їх

модифікування та корегування до існуючого проєкту. Інтерфейс користувача передбачає реалізацію графічного відображення процесу виявлення об'єктів у мобільному застосунку на базі Android. Особливу увагу приділено відображенню результатів обробки відео потоку в реальному часі, що дозволяє оператору миттєво реагувати на зміну обстановки. Інтерфейс повинен бути інтуїтивно зрозумілим, адаптованим до мобільних пристроїв, а також підтримувати взаємодію з системою через прості дії – запуск, зупинка, перехід між режимами.

Апаратний інтерфейс забезпечується портами Raspberry Pi 4 Model B, включаючи GPIO для підключення сенсорів або виконавчих пристроїв, CSI-порт для підключення камери, а також USB-порти, які використовуються для зовнішніх модулів, таких як WiFi-адаптер або накопичувач. Зважаючи на енергоспоживання, підключення блоку живлення повинно відповідати вимогам пристрою – не менш ніж 5В 3А для стабільної роботи всіх компонентів.

Програмний інтерфейс забезпечує взаємодію між програмними модулями обробки даних на Raspberry Pi та клієнтським застосунком Android. Для реалізації зв’язку доцільно використовувати REST API або WebSocket, що забезпечує передачу результатів детекції, команд управління та діагностичних повідомлень.

У проєкті передбачає передачу відеопотоку з борту дрона на мобільний пристрій у реальному часі, а також двосторонній обмін даними. Оптимальним є використання Wi-Fi з підтримкою RTSP або MJPEG-протоколів, що дозволяє зменшити затримки при передачі та забезпечити стабільність під час польоту.

Отже, реалізація проєкту вимагає поєднання апаратної та програмної складової та їх симбіоз. В першу чергу ставиться питання відповідності програмного забезпечення до вимог та завдань, поставлених на початку роботи. Потім, визначається апаратна структура, що дозволяє підібрати оптимальні компоненти.

Висновки до розділу 1

У процесі написання першого розділу були визначені необхідні тези для початку написання та реалізації кваліфікаційної бакалаврської роботи. Було порівняно та протипоставлено пріоритетні сфери застосування методики детектування об’єктів та перешкод в режимі реального часу. Вагома частина потенційних споживачів так чи інакше відносяться до воєнної тематики, оскільки головні завдання з охорони території, спостереження за секторами та виявлення можливої загрози, підходять і для ведення війни. Основний аргумент – низька ціна БПЛА порівняно з іншою зброєю віддаленого ураження. Проте, в мирних умовах, така система може застосовуватись в побутових, охоронних або сільськогосподарських цілях.

Було порівняно системи фіксування об’єктів іноземних БПЛА та вітчизняних аналогів. Висока вартість робить використання перших недоцільним, оскільки їх значний потенціал перевищує потреби споживачів, а отже потенційний клієнт переплачує за якість та непотрібні функції. Натомість, вітчизняний аналог розроблюється на основі конкретно поставлених завдань, та детектує саме небезпечні та важкопомітні об’єкти. Проте, слід зазначити, що функціонал та зручність використання на стороні іноземних варіантів.

Апаратні вимоги були зазначені та виконані, зважаючи на доступні умови. Було обрано Raspberry Pi 4 Module B, адже його потужності достатньо для реалізація прийому, обробки та передачі інформації на Android-пристрій. Це простий, а головне не дорогий спосіб отримання необхідної інформації не вдаючись до включання в проєкт ще пультів з виводом зображення. Від перегріву плату захищає алюмінієвий корпус та два кулери розташовані над центральним процесором. Вага та ергономіка компонентів була підібрана зважаючи на вантажопідйомність дрону DJI Mini 4 Pro таким чином, щоб вони не заважали здійсненню маневрів, коригуючи центр тяжіння. Вибором акселератора прийшлось знехтувати через його ціноутворення, проте на якість зображення це ніяк не впливатиме, а лише зменшить кількість FPS до мінімальних значень.

TensorFlow Lite показав себе з кращого боку при роботі з одноплатного комп’ютера типу Raspberry, ніж звичайна версія фреймворку. Обумовлено це недостатньою потужністю та можливістю перегріву центрального процесора. Було реалізовано підключення бібліотек для обробки інформації з IP-камери та доставку її до користувача, шляхом створення ізольованого середовища для Python. Саме за допомогою неї налагоджено зміна зображення на відповідний колір, в залежності від рівня небезпеки.

Було обрано Android платформу зважаючи на її доступність та сумісність з фреймворком та бібліотеками, а також легким маніпулюванням процесом отримання інформації.

2 ПРОЄКТУВАННЯ СИСТЕМИ КАНАЛУ ЗВ’ЯЗКУ

2.1 Алгоритм роботи метода детектування об’єкту

Проблема реалізації каналу зв’язку для детектування об’єкту потребує не лише апаратного чи програмного рішення. Необхідно чітко розуміти можливості та специфіку реалізації алгоритму роботи всієї системи, її особливості, сильні та слабкі сторони. Логіка розробки схеми пов’язана з потребою в чіткому та швидкому координуванні в процесах розгортання системи. В цьому розділі розглядаються можливі рішення для реалізації каналу зв’язку та детектування об’єктів в цілому.

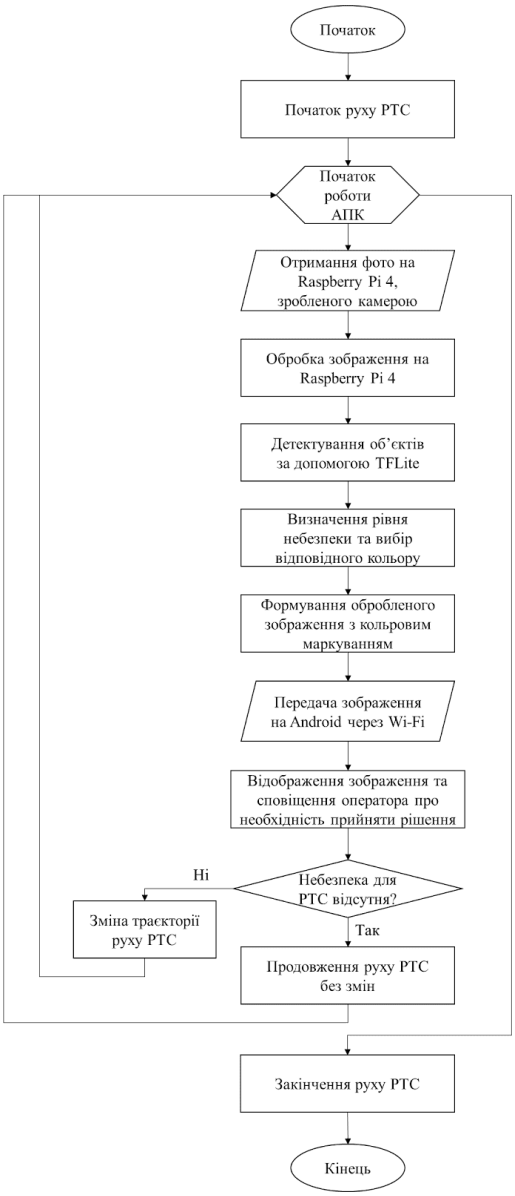


Рисунок 2.1 – Блок-схема реалізації апаратно-програмного забезпечення

Блок-схема спирається на можливі рішення з детектування об’єктів в режимі реального часу та опирається на дії оператора РТС. Кінцевий результат залежить від умов, в яких апаратне середовище знаходиться. Головна ціль – передати зображення в роздільній якості користувачу на мобільний пристрій та не допустити збоїв в роботі алгоритму.

Початок роботи системи ініціалізує готовність роботи всіх компонентів системи. Перевіряється поведінка модуля захоплення відео, одноплатний комп’ютер Raspberry Pi 4 Model B та попереднє з’єднання з Android пристроєм. Модуль захоплення відео включає в себе перевірку коректної працездатності IP-камери та фреймворку TensorFlow Lite. Плата попередньо запускає встановлену на неї операційну систему. Відбувається запуск та взліт БПЛА на висоту, що завчасно передбачена умовами експерименту. У разі виникнення помилок відповідні повідомлення передаються користувачу або записуються напряму в лог-файл для подальшої діагностики. Етап вважається значущим, оскільки подальша робота з детектування об’єктів неможлива без попередньої підготовки та тестування компонентів. У випадку коректної ініціалізації всіх складових система переходить до режиму очікування обраного заздалегідь інтервалу часу. Загалом навантаження на систему класифікуються за часом виконання циклу фіксування зображення.

Таблиця 2.1 – Порівняння ефективності циклів фіксування

Вид циклу	**Час, с**	**Навантаження на ЦП, %**	**Особливості**
Цикл високої частоти	0,5	65–80	Температура може сягати 60–70 °C. Потребує активне охолодження. Можлива затримка у процесі передачі даних, залежить від фіксуючого пристрою.

Вид циклу	**Час, с**	**Навантаження на ЦП, %**	**Особливості**
Цикл середньої частоти	1,0	40–55	Низький рівень перегріву, стабільність результату.
Цикл низької частоти	2,0–3,0	20–30	Зменшення точності виявлення в динамічному середовищі. Повільний сценарій – моніторинг.

Рекомендується дотримуватись циклу середньої або низької частоти. Доцільне використання першого при наявності активного охолодження та останнього при впровадженні інших ресурсозатратних рішень.

IP-камера здатна вести як постійний відеопотік, що притаманно для ресурсоємних систем, так і з певною частотою, в іншому випадку. Отриманий кадр переводиться в матрицю пікселів (NumPy-масив) та одразу передається в модуль попередньої обробки. Вважається за потрібне розгляд бібліотеки OpenCV та її методів, адже бібліотека підготовлює скріншот для подальшої взаємодії з TensorFlow Lite.

OpenCV ініціалізує підключення до камери методом *cv2.VideoCapture()*. В залежності від типу камери значення може варіюватись від 0, якщо камера підключена безпосередньо до USB, або якщо це IP-камера – вказується адреса потокового відео (RTSP, HTTP, або MJPEG).

```
ret, frame = cap.read()
```

OpenCV отримує кадр з відеопотоку, де frame – кадр, у вигляді масиву. Особливість роботи з фреймворком TensorFlow Lite вимагає конвертації зображення до заданого розміру. Для прикладу було взято розмір 300 × 300 пікселів.

```
resized_frame = cv2.resize(frame, (300, 300))
```

OpenCV за замовчуванням зчитує кадри у форматі BGR, тоді як більшість моделей машинного навчання працюють з RGB:

```
rgb_frame = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2RGB)
```

За необхідності функціонал бібліотеки дає можливість працювати на форматуванням кадру для чіткішого детектування об’єктів TensorFlow. До цих функцій входить використання фільтрів згладжування та контрасту. В умовах темного або занадто світлого навколишнього середовища такі методи обробки зображення покращать точність, а головне – надійність програмної складової системи. OpenCV зменшує навантаження на ЦП при детектуванні зображення, адже повноцінна реалізація обробки відео в реальному часі проблематична в умовах Raspberry Pi.

Також, бібліотека допомагає не тільки з вводом, але і з виводом інформації, організовуючи обробку кольору зображення (зелений, жовтий, червоний), а при необхідності – рамки та підписи детектованих зображень.

Алгоритм роботи TensorFlow Lite представлений у вигляді блок-схеми (рис. 2.2), яка демонструє послідовність дій від моменту захоплення зображення до прийняття рішень на основі класифікації об’єктів. Уцьому процесі ключову роль відіграє бібліотека OpenCV, що відповідає за попередню обробку та форматування зображення. Зокрема, вона забезпечує базову калібровку камери для усунення внутрішніх оптичних похибок, виправлення геометричних спотворень, а також налаштування параметрів зображення, таких як баланс білого, контрастність і яскравість. Ці попередні кроки істотно підвищують якість вхідних даних, що у свою чергу підвищує точність та стабільність роботи моделі штучного інтелекту.

З технічної точки зору, описаний алгоритм реалізує типовий цикл системи комп'ютерного зору з функціоналом динамічного оновлення категорій, що дає змогу адаптувати модель до нових або змінених об'єктів у середовищі. Це особливо актуально в умовах реального часу, де ситуаційні зміни відбуваються швидко, наприклад при зміні освітлення або появі нових об'єктів у полі зору камери. Такий підхід забезпечує гнучкість і масштабованість системи,

відкриваючи можливість для розширення функціоналу – наприклад, інтеграції з модулем трекінгу рухомих об'єктів або зі збереженням історії кадрів.

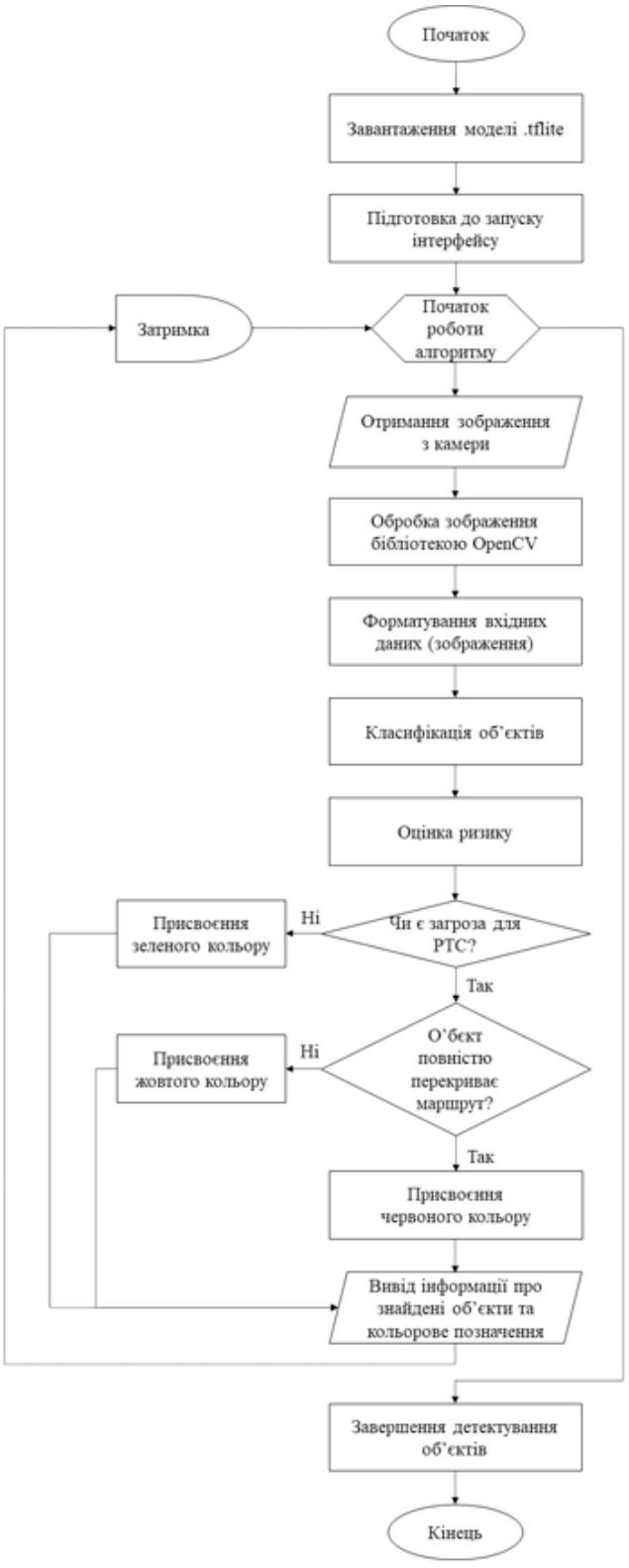


Рисунок 2.2 – Блок-схема алгоритму роботи TensorFlow Lite

При налагодженні детектування об’єктів доцільно обрати модель виявлення об’єктів. Google надає зразок квантованої моделі виявлення об'єктів SSDLite-MobileNet-v2, яка навчена на наборі даних MSCOCO та перетворена для роботи на TensorFlow Lite. Вона може виявляти та ідентифікувати 80 різних поширених об'єктів, таких як люди, автомобілі, чашки тощо. Квантова модель в цьому випадку означає, що вона використовує 8-бітні цілочисельні значення, а не 32-бітні. Це дозволяє їй працювати швидше, шляхом зменшення затримки пам’яті та використовуючи процеси оптимізації в середині процесора. Така модель прискорює виявлення об’єкта при мінімальному коефіцієнту падіння точності.

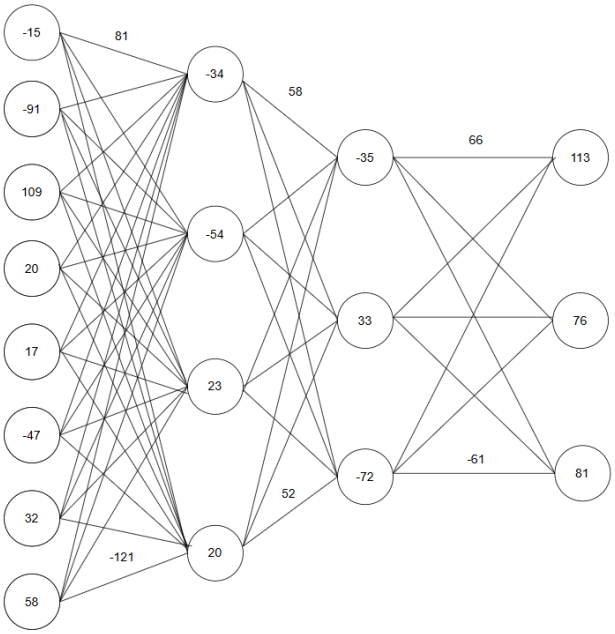


Рисунок 2.3 – Принцип роботи квантової моделі

Існує варіант розробки власної моделі розпізнавання об’єктів на випадок необхідності розпізнавання нестандартних об’єктів [14]. Обидві моделі призначені для своїх цілей. Якщо опиратись на цивільне-аматорське застосування, то SSDLite-MobileNet-v2 рішення є достатнім, проте, якщо необхідне професіональне виявлення нетипових об’єктів, доцільно використовувати власноруч розроблену модель. Незважаючи не це, в рамках проєкту доцільніше використовувати простішу Google модель, адже її моделей

достатньо в рамках детектування загроз. Проте, можлива інтеграція кастомних моделей за умови кращого відгуку з боку програмного забезпечення та заміна рішення на кастомну модель.

Після обробки зображення модель повертає перелік знайдених об’єктів в форматі класу об’єкта, ймовірності виявлення та координат прямокутника навколо об’єкта. Рівень загрози залежить від попередньо-визначених об’єктів та їх кількості. До потенційно небезпечних об’єктів належать, скупчення людей, дерев або електричних ліній тощо. Знайдені об’єкти активують оцінку ризику.

Зелений колір, з програмної точки зору, присвоюється коли:

– не виявлено жодного об’єкта зі списку загроз;
– ймовірність виявлення < 0,5;
– об’єкти не перекривають центральну частину зображення;
– сектор вважається безпечним для продовження польоту.

Жовтий колір присвоюється у випадку коли:

– виявлено окремі об’єкти загрози з середньою впевненістю (0,50–0,75):
– їх кількість або положення вказує на можливу небезпеку;
– є часткове перекриття маршруту.
– сектор потребує уваги.

Червоний колір присвоюється коли:

– виявлено щільну концентрацію загроз (напр., густий ліс, стовпи, мости);
– об’єкти повністю перекривають маршрут або займають велику площу;
– ймовірність виявлення > 0,8;
– сектор вважається критично небезпечним, потрібно змінити маршрут або припинити рух.

На рис. 2.4 відображено бажаний результат обробки зображення. Приклад ілюструє значення кількості дерев у виборі кольору зображення.

Рисунок 2.4 – Результат обробки зображення

Результати класифікації (разом із кольором, координатами об’єктів та їх класами) структуруються в JSON. Оператор отримує детальну інформацію про стан сектору на свій Android-пристрій. У разі критичної ситуації, оператор змінює маршрут або уникає перешкод,

Наприкінці, система повертається до очікування наступного циклу. Таким чином, формується безперервний потік моніторингу заданого сектору.

2.2 Підключення TensorFlow Lite до Raspberry Pi 4 Model B

Модель TensorFlow Lite в поєднанні з Raspberry Pi 4 Model B працюють набагато швидше ніж звичайні TensorFlow. При тому, що звичайна версія здатна видавати вдвічі меншу частоту кадрів, аніж Lite, цей процес можливо зробити продуктивнішим, шляхом додавання Edge TPU [16]. Проте, таке рішення вимагає додаткових витрат. Для дослідження, методом створення скріншотів, достатньо першочергової апаратної моделі.

Перш ніж перейти до встановлення додаткових компонентів та реалізації прикладного рішення, необхідно провести первинну підготовку пристрою Raspberry Pi. Цей етап включає оновлення операційної системи до актуального стану, а також увімкнення необхідних апаратних інтерфейсів, зокрема модуля камери, що є критично важливим для реалізації функціоналу проєкту.

Оновлення здійснювалося через інтерфейс командного рядка, використовуючи пакетний менеджер APT (Advanced Packaging Tool), який є стандартним інструментом управління пакетами у дистрибутивах на базі Debian.

Для ініціалізації оновлення необхідні наступні команди:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Перша команда ініціює оновлення локального списку доступних пакетів з офіційних репозиторіїв для отримання інформації про версії програмного забезпечення, доступного для встановлення. Друга команда виконує оновлення встановлених пакетів, враховуючи залежності між ними, оновлює або замінює пакети.

Залежно від дати останнього оновлення системи, тривалість цього процесу може варіюватися від кількох хвилин до години. Оскільки реалізація проєкту передбачає роботу з візуальними даними, необхідно активувати апаратний інтерфейс камери Raspberry Pi. Ця дія виконується через графічне меню конфігурації Raspberry Pi OS. Зміни в конфігурації апаратних інтерфейсів набувають чинності лише після перезавантаження системи.

У межах підготовки постала необхідність в наявності прикладних скриптів. Отримання скриптів можливе завдяки клонування репозиторію, з відкритими безкоштовними рішеннями. Окрім цього, додатковою вимогою для спрощення процесу встановлення залежностей та налаштувань є скрипт-оболонка.

З метою уникнення потенційних конфліктів між версіями бібліотек і залежностей, що можуть бути присутні на системі Raspberry Pi, було прийнято рішення використовувати ізольоване віртуальне середовище Python. Це особливо актуально, коли на пристрої вже встановлені інші версії TensorFlow чи суміжних бібліотек, які не повинні змінюватися.

Для створення віртуального середовища було попередньо встановлено пакет virtualenv, який є стандартним інструментом для організації ізольованих Python-середовищ:

```
sudo pip3 install virtualenv
```

Після інсталяції virtualenv, безпосередньо в каталозі tflite1 було створено нове віртуальне середовище з іменем tflite1-env за допомогою наступної команди:

```
python3 -m venv tflite1-env
```

У результаті виконання цієї команди в поточному каталозі з’являється нова директорія tflite1-env, яка містить всі необхідні каталоги та файли для ізольованого запуску Python-проєктів, включаючи директорії bin, lib, а також локальну копію інтерпретатора Python.

Для активації використовується команда:

```
source tflite1-env/bin/activate
```

Після цього у запрошенні командного рядка з’являється префікс з його назвою, що візуально підтверджує правильність активації. У цьому режимі всі подальші інсталяції бібліотек за допомогою pip відбуватимуться лише в межах цього середовища, не зачіпаючи глобальну систему.

Цей процес є тимчасовим, оскільки після завершення сесії терміналу або його закриття, середовище деактивується. Тому щоразу при новому запуску терміналу користувач має вручну активувати його.

Наступним кроком є інсталяція ключових бібліотек, необхідних для реалізації системи розпізнавання об’єктів. У цьому проєкті основну роль відіграють дві програмні бібліотеки: TensorFlow Lite – оптимізований для пристроїв з обмеженими ресурсами фреймворк машинного навчання, та OpenCV – бібліотека для комп’ютерного зору, яка забезпечує засоби обробки зображень і відео в реальному часі.

Хоча TensorFlow Lite самостійно не вимагає OpenCV для виконання процесів інференсу, OpenCV активно використовується в скриптах цього проєкту для захоплення відео з камери, обробки зображень та візуалізації результатів роботи моделей розпізнавання об’єктів.

З метою полегшення процесу встановлення та уникнення помилок при ручному додаванні великої кількості залежностей, використано спеціально підготовлений bash-скрипт get_pi_requirements.sh. Цей скрипт завантажує та встановлює усі необхідні пакети, включно з TensorFlow, OpenCV, NumPy, Pillow,

та іншими бібліотеками, що підтримують роботу з машинним навчанням, зображеннями та відео.

Запуск скрипта здійснюється у вже активованому середовищі командою:

```
bash get_pi_requirements.sh
```

Скрипт автоматично виконує завантаження та інсталяцію приблизно 400 Мбайт інсталяційних файлів, що може зайняти певний проміжок часу залежно від якості інтернет-з’єднання та продуктивності комп’ютера. У разі нестабільного з’єднання або помилки при завантаженні (наприклад, через перевищення часу очікування), рекомендується повторити виконання команди кілька разів.

Файл get_pi_requirements.sh викликає інший скрипт – get_pi_dependencies.sh, який містить повний список усіх інстальованих компонентів. Це дозволяє за потреби переглянути зміст інсталяції або змінити його вручну.

Скрипт автоматично встановлює останні доступні версії TensorFlow та пов’язаних бібліотек. Проте, у деяких випадках (наприклад, для забезпечення сумісності з попередніми проєктами або з певними моделями машинного навчання) може виникнути потреба у встановленні конкретної версії TensorFlow. У такому разі після завершення роботи скрипта слід виконати команду встановлення вручну, вказавши потрібну версію:

```
pip3 install tensorflow==X.XX,
```

де X.XX слід замінити на номер версії (наприклад, 2.10 або 2.5.0).

З моделлю виявлення пов’язано два файли: файл сама модель, та файл-карта міток моделей. Попередньо було вказано про можливість застосування двох видів моделей. Від Google та самостійного розроблення.

Після завершення етапів налаштування середовища та встановлення необхідних бібліотек, наступним кроком є безпосередній запуск моделі виявлення об’єктів у режимі реального часу із використанням TensorFlow Lite.

Перш ніж ініціювати виконання скрипта, доцільно забезпечити максимальну ефективність використання апаратних ресурсів. Для цього рекомендується:

– закрити всі фонові процеси та додатки, які не є необхідними для експерименту;

– переконатися в коректному підключенні USB-камери або камери Raspberry Pi (Picamera);

– активувати попередньо створене віртуальне середовище tflite1-env за допомогою команди: `source tflite1-env/bin/activate.`

Підтвердженням активного середовища є наявність префікса (tflite1-env) перед запрошенням у командному рядку.

Запуск детектора об’єктів у режимі реального часу виконується за допомогою Python-скрипта, який реалізує наступні етапи:

– захоплення відеопотоку з підключеної камери

– попередню обробку зображення згідно з вимогами моделі TensorFlow Lite;

– подачу зображення на вхід моделі для здійснення інференсу (інтерпретації);

– виведення результатів у вигляді зображення з нанесеними обмежувальними рамками та мітками класів виявлених об’єктів.

Для запуску скрипта з типовим каталогом моделі використовується команда:

```
python3 TFLite_detection_webcam.py --modeldir=Sample_TFLite_model
```

Якщо користувач працює з власною моделлю, необхідно вказати відповідну назву директорії з файлами моделі, наприклад:

```
python3 TFLite_detection_webcam.py --modeldir=MyCustom_TFLite_Model
```

Після успішної ініціалізації, яка може зайняти кілька хвилин (враховуючи обмежені обчислювальні ресурси Raspberry Pi), з’являється графічне вікно з відеопотоком. При успішному виявленні об’єктів на зображення будуть

накладені прямокутники з відповідними назвами класів. Виявлення відбувається у режимі майже реального часу з частотою кадрів, що залежить від розміру моделі, роздільної здатності відео та конфігурації пристрою [15].

2.3 Вибір необхідних технологій та компонентів

Для реалізації практичних аспектів кваліфікаційної роботи відібрано апаратні складові, які відповідають вимогам системи виявлення об’єктів у реальному часі та передачі оброблених результатів на мобільний пристрій. Вибір обладнання здійснюється за критеріями якості, сумісності, продуктивності та вартості, що важливо в дослідницьких та освітніх проектах на кшталт цього. Усі компоненти мають відповідний рівень технічної надійності, придатний для побудови стабільної роботизованої системи з фіксації та редагування кадрів.

Першим на черзі DJI Phantom 4 – БПЛА, квадрокоптерного типу [4]. На рис. 2.5 зображена його розібрана версія, без IP-камери та пропелерів. Такий стан РТС тимчасовий, оскільки квадрокоптер використовується як літальна платформа для написання дослідницьких робіт інших типів.

Рисунок 2.5 – DJI Phantom 4 в розібраному стані

Як основа для подальшої платформи, DJI представляє собою апарат, котрий задовольняє потреби визначення типових перешкод та небезпек. Він обладнаний поліпшеною системою навігації, що включає GPS/GLONASS, а також підтримує режим стабілізації польоту. Таким чином, ці фактори є ключовим для точного захоплення відео. Його вантажопідйомність дозволяє інтегрувати додаткові компоненти.

Окремо слід зазначити плату Raspberry Pi 4 Model B 8 GB (рис. 2.6). Її характеристики було згадано розділом вище, проте доцільно згадати її габарити. Фізичні габарити становлять 85,6 мм × 56,5 мм, при цьому товщина плати (без урахування радіаторів або корпусу) становить близько 17 мм, що є ідеальним для розташування плати нижче корпусу DJI або інтеграції в нього. Надмірне навантаження та порушення аеродинаміки виключене. Стандартне розташування портів забезпечує підключення зовнішніх інтерфейсів та модулів.

Рисунок 2.6 – Raspberry Pi 4 Module B, залучена до дослідницької роботи

Отже, важливою складовою коректної реалізації є дотримування низки конструкційно-технічних вимог до кріплення [6]. По-перше, кронштейн повинен бути ударостійким, тому що БПЛА під час польоту генерують сильну вібрацію, яка може вплинути на роботу плату та спричинити нестабільну обробку сигналу.

Для цього рекомендується використовувати демпфери з гуми, силікону або антивібраційну прокладку.

По-друге, кріпильні елементи повинні бути термостійкими. Враховуючи, що друкована плата може нагріватися через високе навантаження під час обробки в реальному часі, слід забезпечити вільну циркуляцію повітря або додаткове охолодження. Неправильне розміщення або надмірне затягування в закритому корпусі може призвести до перегріву та зниження продуктивності.

По-третє, під час роботи зовнішні інтерфейси Raspberry Pi, такі як GPIO, HDMI, USB і слот microSD, повинні бути доступні. Тому кріплення має забезпечувати легку установку без блокування портів. Крім того, враховуючи розмір плати, необхідно переконатись в наявності стандартних монтажних отворах на корпусі (4 отвори в кожному куті, діаметром 2,75 мм). На рис. 2.7 зображено готове охолоджуюче рішення для плати, з можливістю фіксації безпосередньо в середовищі PTC.

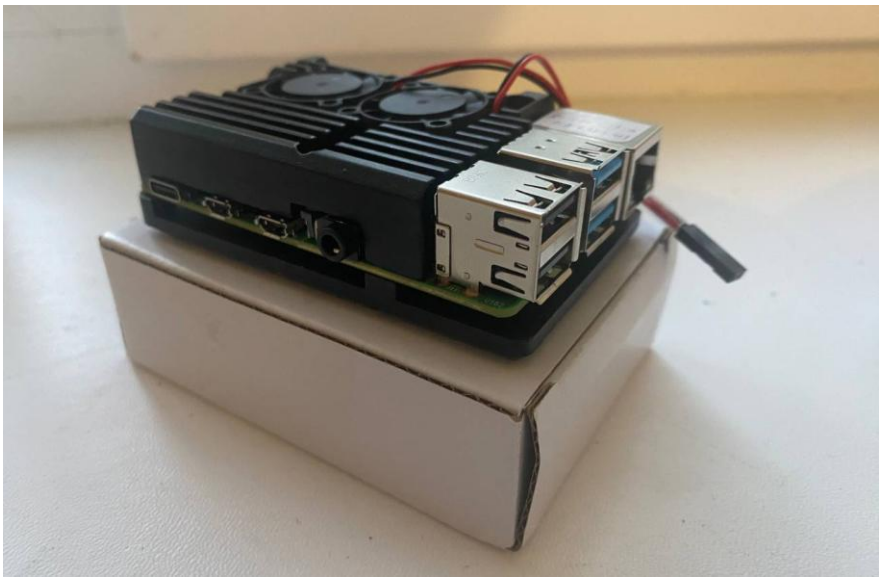


Рисунок 2.7 – Алюмінієвий корпус з радіатором та двома кулерами

Таке рішення обумовлене наявністю специфічної програмної складової, використання якої неможливе на одноплатних комп'ютерах попереднього покоління [19]. До того ж, навіть Raspberry Pi 4 Module B, не здатна видавати достатню кількість FPS при частому циклі, без перегріву компонентів (табл. 2.1).

А отже використання корпусу є цілком обумовленим апаратно-програмними особливостями при детектуванні об'єктів віддалено та з мінімальним вкладом в цей процес оператору FPV-дрона. Подальші складові роботи розглядаються в умовах постійного охолодження системи.

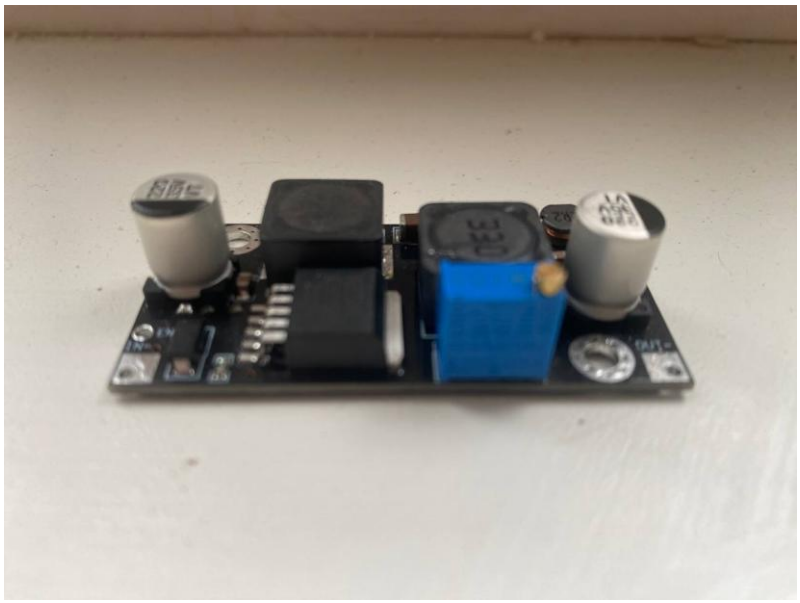


Рисунок 2.8 – Модуль XL6019 регульований підвищувально-понижувальний перетворювач Uin 3–32V, Uout 5–35V

Модуль XL6019 є регульованим імпульсним стабілізатором напруги типу «підвищувально-понижувальний» (англ. boost-buck), здатним працювати як у режимі підвищення, так і зниження вхідної напруги (рис. 2.8). Модуль можливо використовувати для передачі напруги з літій-іонних батарей до контактів плати без ризику перегріти або пошкодити компоненти плати. XL6019 зручний у ситуаціях, коли джерело живлення має широкий діапазон напруги – напр., при живленні від акумулятора, який поступово розряджається (напр., Li-ion 18650, 3,0–4,2 В). У той час як Raspberry Pi потребує стабільних 5,0–5,2 В, XL6019 автоматично регулює вихід – навіть якщо вхідна напруга падає нижче або піднімається вище бажаної вихідної. В умовах обробки відео з камери, аналізу моделей TensorFlow Lite або роботи з Edge TPU, просідання напруги навіть на 0,3 В може призвести до помилок або вимкнення системи. Модуль виступає як буфер та регулятор, що забезпечує безперервну подачу енергії. До того ж, він має

вбудований гвинтовий потенціометр, що дозволяє налаштувати вихідну напругу. Проте все одно потребує мультиметра.

Рисунок 2.9 – Дві батареї 18650MJI1

Доречно використання двох послідовно з’єднаних літій-іонних елементів 18650, які забезпечують номінальну напругу близько 7,4 В (повна зарядка – до 8,4 В, повне розрядження – до 6,0 В) (рис. 2.9). Така конфігурація дозволяє забезпечити значну автономність роботи системи, а також створює оптимальні умови для стабільної роботи XL6019, що має достатній запас напруги на вході для ефективної стабілізації.

Їх використання є доцільним у мобільних системах на базі Raspberry Pi через їх високу енергетичну щільність, низьке саморозрядження, доступність, а також підтримку повторної зарядки.

Виробник офіційно не вказує максимальне рекомендоване корисне навантаження для Phantom 4, оскільки дрон не призначений для транспортування додаткових речей. Проте фактичні випробування показали, що безпілотник може стабільно переміщатись з додатковим навантаженням до 200–300 г, а час польоту та маневреність істотно не зменшаються. Регулюючи кронштейни і противаги, іноді вдається піднімати вантажі до 400 г, але це на

межі безпеки. Шляхом аналізу існуючих компонентів практичним шляхом було визначено вагу радіатора з двома кулерами та одноплатного комп’ютера без інтегрованої SD-карти.

Рисунок 2.10 – Вага головної частини компонентів

Хоч, було зважено найважчі елементи системи (рис. 2.10), проте варто звернути увагу на додаткові, ті що зостаються поза кадром, такі як батарея живлення плати та кріплення та провідники. Необхідно ретельно збалансувати центр ваги. Дуже важливо зменшити кількість непотрібних елементів та рекомендуються легкі батареї та камери.

Таблиця 2.2 – Компоненти проєкту та їх вага

Компонент	**Вага, г**	**Примітки**
Raspberry Pi 4 Model B (8GB)	47	–
Верхня частина кейса	57	Можливе розділення на окремі кулери

Компонент	**Вага, г**	**Примітки**
Нижня часитна кейса	60	Можливе зменшення ваги, шляхом укорочення платформи
Raspberry Cam. Rev. 1.3	3	Можливе застосування стандартної IP-камери
Карта пам'яті microSD 32 Гбайт	< 1	Мінімальне значення для Raspberry OS.
Літій іонні акумулятори	111	Необхідне постійне спостереження за напругою. Існує ризик виведення системи зі строю
Понижуючий перетворювач	10	–
Кріплення	35	Полімерні матеріали, розподіл навантаження
Всього	324	–

Згідно з розрахунками, маса системи майже досягає граничного значення її несучої здатності. Висновок: ця система технічно можлива з DJI Phantom 4, але масу окремих компонентів потрібно було б мінімізувати, особливо використовуючи більш компактне джерело живлення або легку камеру. Перевищення вантажопідйомності може призвести до скорочення часу польоту та збільшення ризику втрати дрона у разі аварії. Можливим шляхом вирішення проблеми є укорочення корпусу алюмінієвого кейсу та заміна кріплення на полімерні аналоги.

Висновки до розділу 2

В межах описаного дослідження особливу увагу було приділено аналізу підходів до побудови РТС. Розглянуто алгоритм роботи всієї системи, його відповідність до поставлених завдань та аспекти можливої реалізації. Передача відеоінформації та виконання алгоритмів комп'ютерного зору є основою для роботизованої системи.

Розроблена блок-схема роботи РТС передбачає ініціалізацію апаратних компонентів в комплексі з програмною складовою. З огляду на особливості реалізації в обмежених умовах було визначено опитимальний режим захоплення кадрів та інтерпретації зображення. Для підвищення точності виявлення об'єктів та обробки зображення було обрано бібліотеку OpenCV, що дозволяє реалізувати потребу в маркуванні вихідних даних та сповіщення оператора БПЛА про рівень небезпеки середовища. Продемонстровано особливості роботи з ліцензійною квантовою моделлю від Google – SSDLite-MobileNet-v2 та принцип її роботи. Також акцентовано увагу на альтернативі у вигляді розробки власної моделі виявлення об’єктів. Таке рішення може бути обумовлене необхідністю обробки специфічних зображення, здебільшого в нестандартних умовах.

Також було представлено потенціальний результат дослідження у вигляді оброблених скріншотів. Описано фактори, що впливають на пріоритетність надання статусу зображення, а також взаємовідношення між ймовірністю виявлення та кольором.

Актуальне обладнання було відібрано, зважаючи на ціноутворення в сумі з ефективністю та продуктивністю. В якості пересувного БПЛА було відібрано DJI Phantom 4, опираючись на його вантажопідйомні характеристики. До того ж, при ньому виділено місце для кріпильних елементів, що з одного боку дає простір для реалізації розташування основних елементів детектування, а з іншого сама ідея втілення обмежує місце допустимого знаходження об’єкта, через розташування основної камери. Тим не менш, було розраховано ваговий та геометричний аспект системи. Як результат – необхідність регулювання центра маси роботизованої системи через перевантаження з боку плати та охолоджувального кейсу. Можливе рішення включає залучення полімерних аналогів компонентів або зменшення ваги, шляхом ручного урізання стандартних рішень, наприклад, корпусу алюмінієвого кейсу. Слід зазначити про доцільність уникнення перевищення порогу в 300 г. Хоча, БПЛА здатний підіймати в повітря вагу до 400 г, аеродинаміка та керування погіршується навіть з вирівняним центром мас. Неправильне рішення з цього боку здатне не лише

зробити неможливим подальше пересування, а й звести на ні попередньо-поставлені завдання, тим самим не допомагаючи оператору, а тільки заважаючи виконувати роботу.

3 РЕАЛІЗАЦІЯ СИСТЕМИ ДЕТЕКТУВАННЯ ОБ’ЄКТІВ

У цьому розділі показано процес реалізації віддаленого детектування об’єктів на базі апаратної платформи Raspberry Pi, Android та програмного забезпечення TensorFlow Lite. Система побудована з використанням одноплатного комп’ютера Raspberry Pi 4 Model B, до якого підключено камеру, модуль живлення, модуль бездротового зв’язку та інші необхідні компоненти. Raspberry Pi виконує функцію обробки зображень у реальному часі за допомогою попередньо навченої моделі машинного навчання, скомпільованої для TensorFlow Lite. Результати розпізнавання передаються на Android-пристрій для візуалізації та взаємодії з користувачем.

Рис. 3.1 ілюструє схему підключення основних елементів систем.

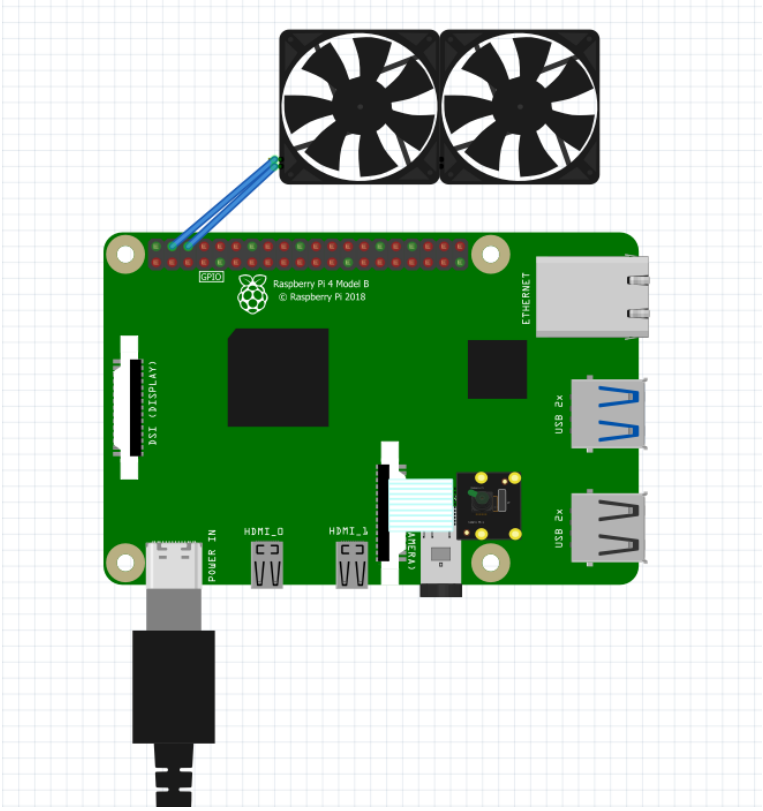


Рисунок 3.1 – Схема підключення Raspberry Pi 4 Module B та компонентів (джерело живлення – блок)

Блок виступає джерелом живлення на моменті реалізації. Подальше тестування замінне джерело живлення на автономне.

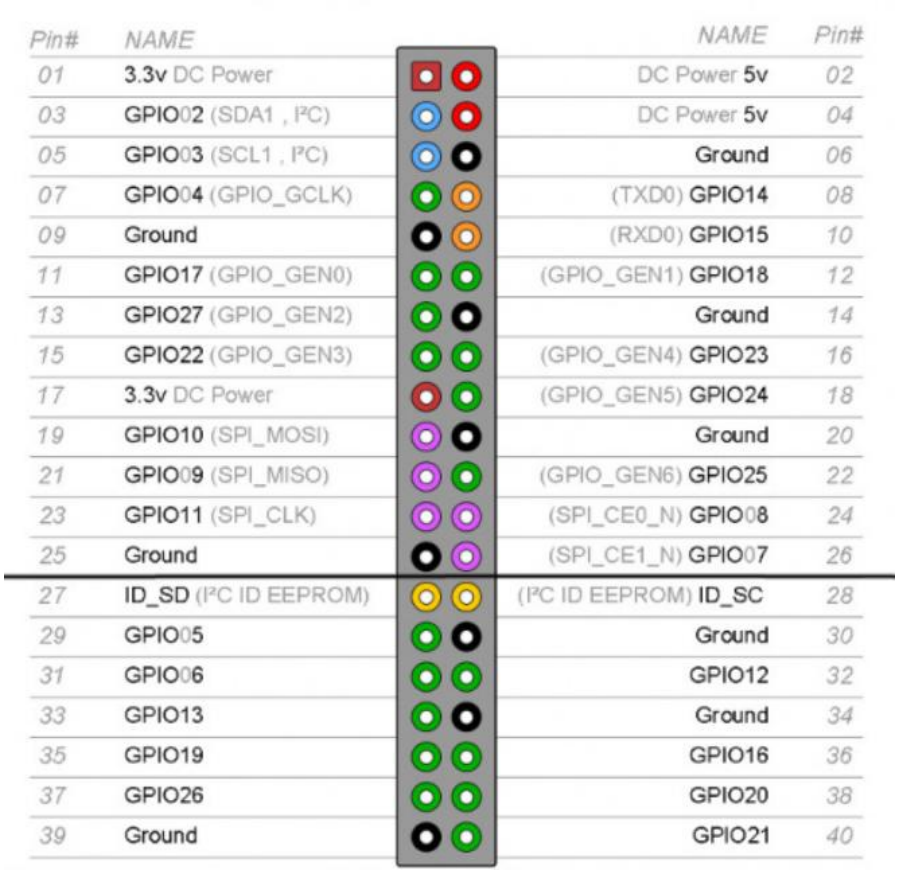
3.1 Реалізація апаратного забезпечення

Для забезпечення надійного захисту апаратної частини Raspberry Pi 4 Model B у даному проєкті було використано металевий корпус, який виконує одразу кілька важливих функцій. Основною з них є покращення тепловідведення від основних елементів плати, зокрема від центрального процесора (англ. Security Operations Center, SoC), оперативної пам’яті та чіпа USB-контролера. Металеві корпуси ефективно працюють як пасивний радіатор, особливо в умовах підвищеного тепловиділення. Зважаючи на необхідність додержання умовного обмеження зі сторони вантажопідйомності РТС, було прийнято рішення про закріплення лише верхньої частини кейсу, тоді як нижня частина плати має кріпитись на полімерний матеріал. Таким чином, вся конструкція була полегшена на 15 %, що є критично важливим на момент проведення тестового польоту, адже полегшує роботу оператора БПЛА.

Raspberry Pi 4 Model B оснащено 40-піновим GPIO-роз’ємом, що дозволяє підключати зовнішні пристрої, зокрема вентилятори. Вони мають номінальну напругу в 5 В та були підключені до GPIO для живлення: пін 4 і пін 6 (земля). Нехтування інструкціями з підключення може призвести до ситуації, коли процесор використовує не всю свою потужність або, в гіршому випадку, перегрів наближає плату до внутрішніх лімітів безпеки. Тоді, при довготривалому використанні, неминуче пришвидшений знос обладнання. У випадку детектування та обробки скріншотів, без активного та пасивного охолодження не минути тротлінга, особливо в замкнутому середовищі, де показник температури швидко перевищить значення в 80 °C.

Окрім цього, схема підключення GPIO-роз’єму, представлена на рис. 3.2, демонструє розташування основних пінів живлення, заземлення та даних, що можуть бути використані для підключення не лише охолодження, а й різноманітних сенсорів, індикаторів або модулів (наприклад, I2C, SPI чи UART), що розширює функціональні можливості системи в цілому. Саме гнучкість і модульність є ключовими перевагами при створенні систем на базі Raspberry Pi

в умовах обмеженого простору й енергоспоживання, таких як автономні безпілотні платформи.



Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Рисунок 3.2 – Розташування пінів для Raspberry Pi 4 Model B

В свою чергу, основною зазначеною функцією є автономність системи детектування, тому замість блоку живлення, було розглянуто альтернативу у вигляді двох літій-іонних акумуляторів. Слід зазначити про неможливість використання моделей акумуляторів 16500MJ1 напряму з одноплатним комп’ютером. Одним з мінусів останнього є відсутність автоматичного захисту від перенапруги, глибокого розряду та перезарядження. Тому для живлення одноплатного комп’ютера від літій-іонних елементів необхідне використання проміжних електронних модулів.

У типовій конфігурації застосовується наступна послідовність компонентів:

– літій-іонний акумулятор підключається до модуля зарядки TP4056, оснащеного захистом;

– вихід TP4056 (3,7 В) подається на підвищувальний перетворювач (Boost Converter), який стабілізує напругу на рівні 5 В;

– вихід Boost Converter з’єднується з піном живлення Raspberry Pi (GPIO 5V, пін 2 або 4), або з роз’ємом USB-C через відповідний адаптер.

За потреби тривалого автономного живлення доцільним є використання декількох акумуляторних елементів з балансувальним модулем (BMS), який забезпечує рівномірний заряд та розряд усіх елементів у пакеті. При проєктуванні систем живлення необхідно дотримуватись вимог техніки безпеки та передбачити захист від короткого замикання.

Полімерні матеріали широко застосовувались при проєктуванні апаратної складової дослідження. Було надрукована кейс для камери Raspberry version 1.3 та каркас конструкції. З огляду на ергономіку DJI Phantom 4 від розташування компонентів прямо на БПЛА вирішено відмовитись, адже це унеможливлювало стабільне переміщення компонентів детектування та зменшувало захист плати в умовах жовтої та червоної зон. Водночас, підтвердило ефективність рішення розташувати компоненти прямо під камерою, так як нижня конструкція DJI, що призначена для посадки БПЛА, підходить для кріплення кронштейнів. Проте, єдина проблема – це стандартне посереднє розташування камери. Воно ускладнило процес розміщення компонентів по поверхні кронштейну, адже висота камери майже повністю дорівнювала висоті «ніг» БПЛА. Було прийнято рішення розташування плати позаду FPV-дрона, разом з його охолодженням, тоді як елементи живлення перенести наперед, щоб нівелювати різницю балансів. Камера детектування об'єктів була закріплена збоку основної камери. Крипіжним елементом виступають стяжки, як більш дешевий та легкий аналог, порівняно зі спеціальними вбудованими в кріплення кейсами.

Елемент керування БПЛА в цьому дослідженні передбачає пульт DJI Phantom 4 з вбудованим крипіжним елементом під телефон. При тестуванні використовується телефон марки Iphone для відеопередачі напряму з БПЛА до оператора, тоді як Android – як середовище, де показано вже детектуванні загрози.

а)

б)

Рисунок 3.3 – Готове апаратне рішення: а) фото всієї задіяної системи; б) детальне розташування компонентів

Використання полімерних матеріалів, а також стяжок – як елемента кріплення знизило вагу вантажу. Вага всієї системи становить 264 г, а вага разом з РТС – 1 кг 158 г. Результат дає можливість замислитись про інтегрування інших складових, таких як полімерні кейси для акумуляторів, планка для камери тощо.

3.2 Реалізація програмного забезпечення

Розробка програмного забезпечення є важливим кроком для проєкту, оскільки він забезпечує функціональну взаємодію апаратної частини з користувачем та зовнішнім середовищем. У даній системі програмне забезпечення було реалізовано на базі операційної системи Raspberry Pi OS x32, яка є оптимізованим дистрибутивом Linux, спеціально адаптованим для апаратної платформи Raspberry Pi. У контексті обмеженого середовища виникає потреба використовувати дисплей ноутбука як вивідний пристрій. Оскільки HDMI-порт ноутбука зазвичай реалізований як вихідний інтерфейс, пряме підключення не є можливим, що зумовлює використання альтернативних підходів.

3.2.1 Процес налагодження Raspberry Pi

HDMI-порти у більшості ноутбуків функціонують лише в режимі передачі сигналу, а не його прийому, що унеможливлює пряме підключення Raspberry Pi через HDMI. Виходом із ситуації є застосування протоколів віддаленого доступу, таких як VNC (Virtual Network Computing) або RDP (Remote Desktop Protocol), які дозволяють здійснювати графічну взаємодію через мережу.

Raspberry Pi 4 потребує джерела живлення з номінальною напругою 5 В та струмом не менше 3 А, що обумовлено значно вищим енергоспоживанням у порівнянні з попередніми моделями. Нестача живлення або використання нестабільного джерела може призводити до різноманітних збоїв у роботі, зокрема до зниження продуктивності, перезапусків системи, втрати даних або некоректної роботи підключених модулів, таких як камери, накопичувачі або інші USB-пристрої.

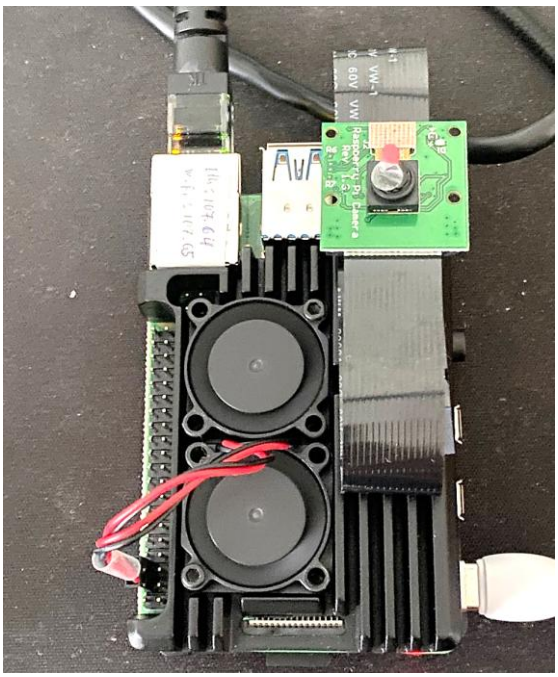


Рисунок 3.4 – Підключення Rapsberry Pi 4 Model B
для подальшого налаштування

Для забезпечення надійного електроживлення використовуються сертифіковані адаптери живлення з інтерфейсом USB Type-C, які відповідають технічним вимогам Raspberry Pi Foundation. В умовах мобільного або автономного використання можливе підключення літій-іонних акумуляторів через спеціалізовані модулі живлення, що забезпечують стабілізацію напруги та захист від перенапруги й глибокого розряду. Такі модулі дозволяють організувати безперебійне живлення системи, що є важливим для проектів, які працюють у середовищі з нестабільним доступом до мережі електропостачання. Грамотний підбір джерела живлення є необхідною передумовою для тривалої та безпечної експлуатації Raspberry Pi 4 Model B у складі складних електронних систем. Однак, при дослідженнях в лабораторних умовах, вирішено відмовитись на деякий час від використання літій-іонних аналогів через небезпеку пошкодження плати та модулів. Було обрано підключення блоком живлення.

На завершальному етапі взаємодія з Raspberry Pi здійснюється через SSH-клієнт (наприклад, ssh у терміналі Linux/macOS або через PuTTY у середовищі Windows). У результаті встановлюється повноцінне термінальне підключення,

яке дозволяє керувати пристроєм, виконувати команди, розгортати програмне забезпечення тощо. В даному випадку було використано під’єднання через PuTTY середовище. Попередньо було перенесено ssh файл до директорії системи. Інтеграція PuTTY та VNC Viever дає змогу запустити середовище операційної системи (ОС) на ноутбуці.

Таким чином, процес підключення Raspberry Pi до ноутбука реалізується як комплекс послідовних процедур апаратного та програмного характеру, що у своїй сукупності забезпечують ефективну експлуатацію пристрою у контексті навчання, досліджень або розробки вбудованих систем.

Після підключення оновлюється система Raspberry Pi. В залежності від WiFi-підключення, процес може бути вдалим або не вдалим. Ранні версії одноплатного комп’ютера дозволяли вмикати камеру командою:

```
sudo raspi-config
```

Після встановлення останньої версії ПЗ потрібно впевнитись, що встановлено libcamera та що вона встановлена правильно:

```
sudo apt update
sudo apt install libcamera-apps
libcamera-jpeg -o test.jpg
```

Остання команда робить знімок екрана та зберігає його на SD-накопичувач (рис. 3.5).

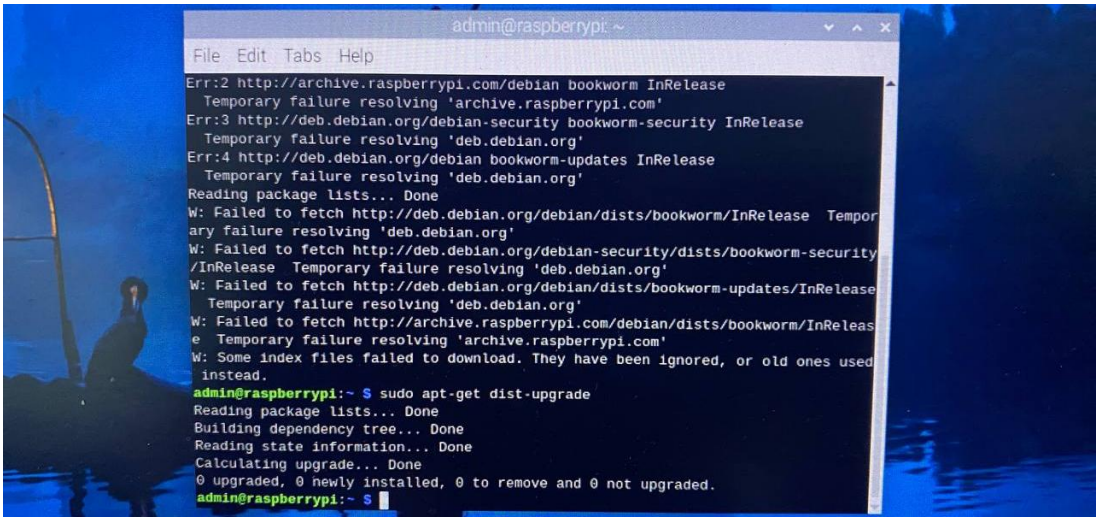


Рисунок 3.5 – Приклад встановлення оновлення

Приклад дозволяє перевірити не лише функціональність самої камери, а й коректність конфігурації драйверів, доступ до файлової системи та базові можливості комп’ютерного зору. На прикладі показано можливість детектування таким чином проблем з підключенням до інтернет мережі.

3.2.2 Встановлення проблемних бібліотек

Проблема реалізації постала при спробі встановити програмне забезпечення для TensorFlow Lite. Остання версія Raspberry Pi OS Bookworm зазнала змін внутрішньої архітектури системи. Через оновлену версію значна кількість бібліотек не підтримувалась, а отже не могла функціонувати інтегровано. З цієї причини, було прийнято рішення змінити ОС на Raspberry Pi os legacy. Всіх недоліків з бібліотеками вона не вирішує, однак надає необхідні інструменти для роботи існуючими та все ж дає змогу встановити окремі з них.

Окремо слід виділити системні бібліотеки, що встановлювались через apt для сумісності. Серед них основною є *libcamera*. Паралельно, бібліотека являє собою масштабне оновлення нової ОС та є аналогом старших бібліотек. Більшість гайдів та мануалів не враховують специфіку роботи саме з нею, а тому є застарілими. *libjpeg-dev, libpng-dev, libtiff5* – бібліотеки, що застосовувались для роботи з графічними форматами, а *v4l-utils, libv4l-dev* – при взаємодії з відеозахопленням.

Команди, що використовувались для підключення системних бібліотек:

- `sudo apt install -y \`
- `python3-picamera2 \`
- `libatlas-base-dev \`
- `libjpeg-dev \`
- `libtiff5 \`
- `libjasper-dev \`
- `libpng-dev \`
- `libavcodec-dev \`
- `libavformat-dev \`
- `libswscale-dev \`

- `libv4l-dev \`
- `v4l-utils \`
- `libgtk2.0-dev \`
- `libcanberra-gtk* \`

Деякі з них відсутні в нових версіях Raspberry PI, деякі видалили зовсім.

Переходячи до основних бібліотек, слід зазначити, що здебільшого нові версії викликають конфлікти та заважають коректному запуску моделі детектування об’єктів. *NumPy* потребував даунгрейд до стабільної версії програмного забезпечення. Більшість бібліотек на основі OpenCV та tflite були зібрані з *numpy 1.x*, і несумісні з *NumPy 2.x*. Також OpenCV вимагала встановлення нижчої версії через несумісність нових із бібліотекою *picamera2*.

Основні команди, що використовувались для підключення бібліотек:

- `sudo apt install python3-picamera2`
- `pip install opencv-python`
- `pip install "numpy<2.0"`
- `pip install tflite-support`

Під час підготовки до запуску системи детектування об’єктів було розроблене віртуальне середовище *venv-picam2*. Також проведено ряд дій, що безпосередньо стосуються взаємодії з ядром операційної системи. Метою цих дій стало забезпечення валідної роботи відеопідсистеми та сумісність з зовнішніми бібліотеками, що знаходяться на нижчому рівні доступу до пристроїв. Відбулась спроба компіляції та встановлення модуля ядра *v4l2loopback*. Для цього використовувався інструмент DKMS, що автоматизує процес компіляції та інтеграції модулів ядра при змінні або оновленні ядра. У зв’язку з припиненням підтримки бібліотеки *picamera* для нових моделей Raspberry Pi (з ядром 64-біт), було впроваджено *libcamera*, що надає доступ до камер на основі стандартів V4L2 через інтерфейси ядра.

У межах реалізації рішення відбулось налаштування системного файлу *config.txt*, що забезпечило стабільне підключення параметрів для роботи з камерою. На рис. 3.6 було замінено значення всіх білих рядків. *Arm_64bit=0* – для 32-бітної версії завжди має зоставатись 0, інакше потрібні бібліотеки можуть

не встановитись, а запуск файлу *object_detection.py*, що відповідає за виявлення об’єктів, робить неможливим. Попередньо, було виконано налаштування розширення екрану через окремі аспекти встановлення зв’язку між одноплатним комп’ютером та ноутбуком.

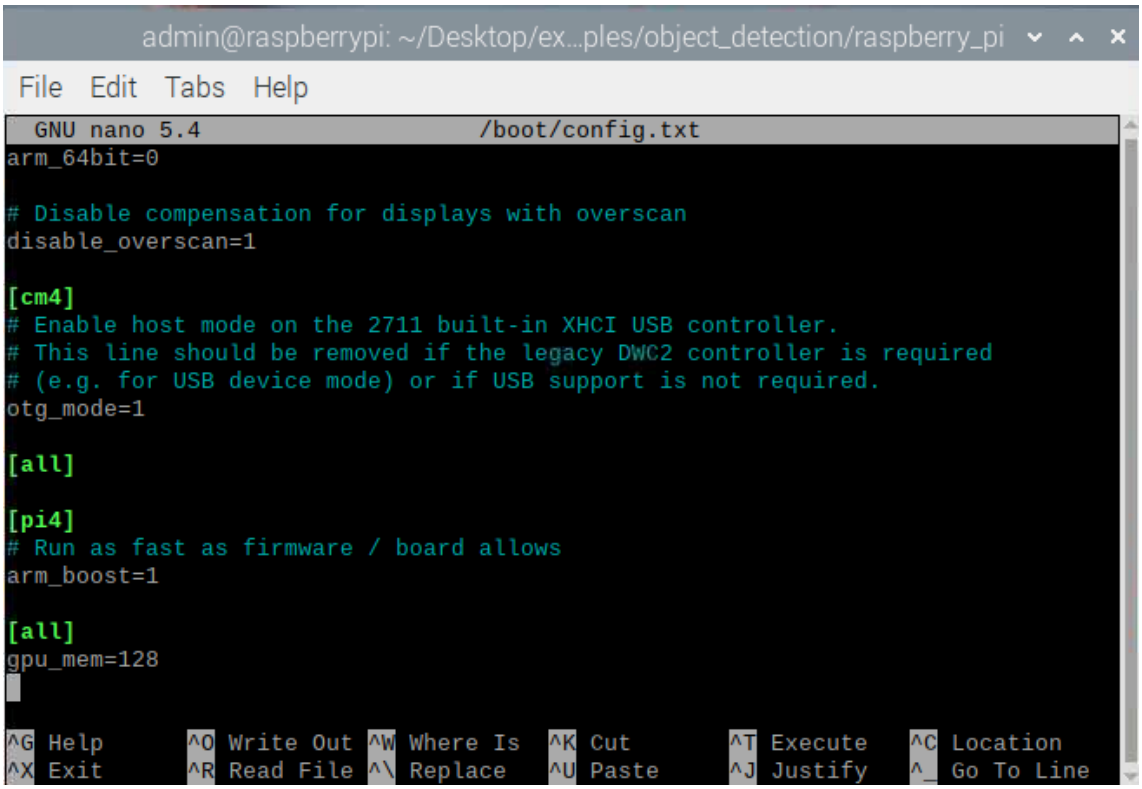


Рисунок 3.6 – Вміст config.txt

Після всіх вищеописаних дій, виконується команда:

```
python3 detect.py --model efficientdet_lite0.tflite,
```

що запускає інтерпретатор мови програмування Python версії 3 для виконання скрипта *detect.py*, реалізує процедуру виявлення об’єктів у реальному часі на базі моделі глибокого навчання *efficientdet_lite0.tflite*.

3.2.3 Архітектура програмного забезпечення

Спочатку відбувається ініціалізація камери. В даному випадку, вона конфігурується на задану роздільність (рис. 3.7).

```
# Init PiCamera2
picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(main={"size": (width, height)}))
picam2.start()
```

Рисунок 3.7 – Ініціалізація камери

Далі відбувається ініціалізація моделі об'єктів (рис. 3.8). Формуються параметри екземпляра детектора об'єктів, а саме файл моделі, кількість CPU потоків, максимальна кількість результатів на кадр та поріг впевненості. Опціонально, код включає використання апаратного прискорювача Edge TPU.

```
# Initialize the object detection model
base_options = core.BaseOptions(
    file_name=model, use_coral=enable_edgetpu, num_threads=num_threads)
detection_options = processor.DetectionOptions(
    max_results=3, score_threshold=0.3)
options = vision.ObjectDetectorOptions(
    base_options=base_options, detection_options=detection_options)
detector = vision.ObjectDetector.create_from_options(options)
```

Рисунок 3.8 – Ініціалізація моделі детектування об'єктів
на основі TensorFlow Lite

Розпочинається включення в роботу основного циклу (рис. 3.9). У ньому відбувається все, що пов’язано з кадром, а якщо точніше то саме захоплення, перевірка коректності, конвертація зображення у RGB формат, створення тензора та отримання результатів детекції. Для спрощення взаємодії оператора РТС з вихідним зображенням було додано дзеркальне відображення кадру та накладання результатів у вигляді рамки та назви детектованого об'єкту на вихідне зображення.

```
while True:
    # Capture frame from Pi Camera
    image = picam2.capture_array()
    if image is None:
        sys.exit("ERROR: Failed to capture image from Raspberry Pi Camera.")

    counter += 1
    image = cv2.flip(image, 1)
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    input_tensor = vision.TensorImage.create_from_array(rgb_image)
    detection_result = detector.detect(input_tensor)

    image = utils.visualize(image, detection_result)
```

Рисунок 3.9 – Основний цикл обробки відео

В подальшому в код було додано відображення кількості кадрів в секунду. Логіка виконання полягає в фіксуванні середньої частоти за останні 10 кадрів та відображення результату на відео. Проте, не менш важливим є детектування рівня загроз. В цьому випадку, як було вказано раніше було обрано цикл низької частоти з можливістю кореляції вихідного зображення за часом. Так як, робота задіює відеопотік разом з автоматичним скрінуванням, зрозуміло, що на центральний процесор плати покладене навантаження, що здатне призвести до зависань та перегріву. Тому, в якості тестування ефективності роботи саме цього сегменту було обрано 10 секунд, як час створення фотографії. На рис. 3.10 зображено вже робочий варіант коду.

```
current_time = time.time()
if current_time - last_photo_time >= 2.0:
    photo_filename = f"photo_{photo_counter}.jpg"
    cv2.imwrite(photo_filename, image)
    print(f"Saved photo: {photo_filename}")
    photo_counter += 1
    last_photo_time = current_time
```

Рисунок 3.10 – Цикл виконання скріншоту

Цикл перевіряє, чи пройшло 2 секунди з моменту останньої фотографії. Якщо так, зображення зберігається в файл з ім’ям типу *photo_x.jpg*. Попередньо до цього виконується аналіз об'єктів на рівень безпеки. В залежності від нього, надається червоний або жовтий колір.

Якщо небезпеку не виявлено – зелений. Об'єкти, що потенційно перекривають маршрут з високою впевненістю або великі за розміром, класифікуються як червона загроза, а об'єкти з середнім рівнем впевненості класифікуються як жовта небезпека. Крім того, кореляція коефіцієнта небезпеки можливо регулювати в залежності від необхідності та обставин, за яких виконується детектування (рис. 3.11).

```
red_count = 0
yellow_count = 0

for det in detection_result.detections:
    score = det.categories[0].score
    bbox = det.bounding_box
    x_min, y_min = bbox.origin_x, bbox.origin_y
    x_max, y_max = x_min + bbox.width, y_min + bbox.height

    obj_in_center = (
        center_x - center_margin < (x_min + x_max) / 2 < center_x + center_margin and
        center_y - center_margin < (y_min + y_max) / 2 < center_y + center_margin
    )

    if score > 0.8 or (obj_in_center and bbox.width * bbox.height > 0.2 * frame_w * frame
        red_count += 1
    elif 0.5 <= score <= 0.75:
        yellow_count += 1
```

Рисунок 3.11 – Аналіз зображення та надання йому статусу

Важливо зазначити, що завдяки візуальному кодуванню ступеня небезпеки за допомогою напівпрозорого кольорового фону, оператор може не лише виявити наявність загрози, а й швидко зорієнтуватися в її масштабах та положенні. При тому, що зображення паралельно оброблюється та зберігається на пристрій, а не тільки передається на Android-пристрій. Таким чином, можливий подальший аналіз та форматування коду ідентифікації, виявлення помилок та їх усунення. Проте, потрібно враховувати складність збереження великого потоку інформації в обмеженому апаратному середовищі. Як рішення – змінити частоту збереження фотографій.

З практичного боку, варто враховувати необхідність оптимізації ресурсів Raspberry Pi: зменшення роздільної здатності зображень, використання форматів із компресією, а також впровадження буферизації чи пакетного збереження даних для уникнення перевантаження системи.

Проєкт демонструє потенціал масштабування та адаптації до різних сфер – від безпілотних літальних апаратів до систем відеоспостереження або охорони критичної інфраструктури.

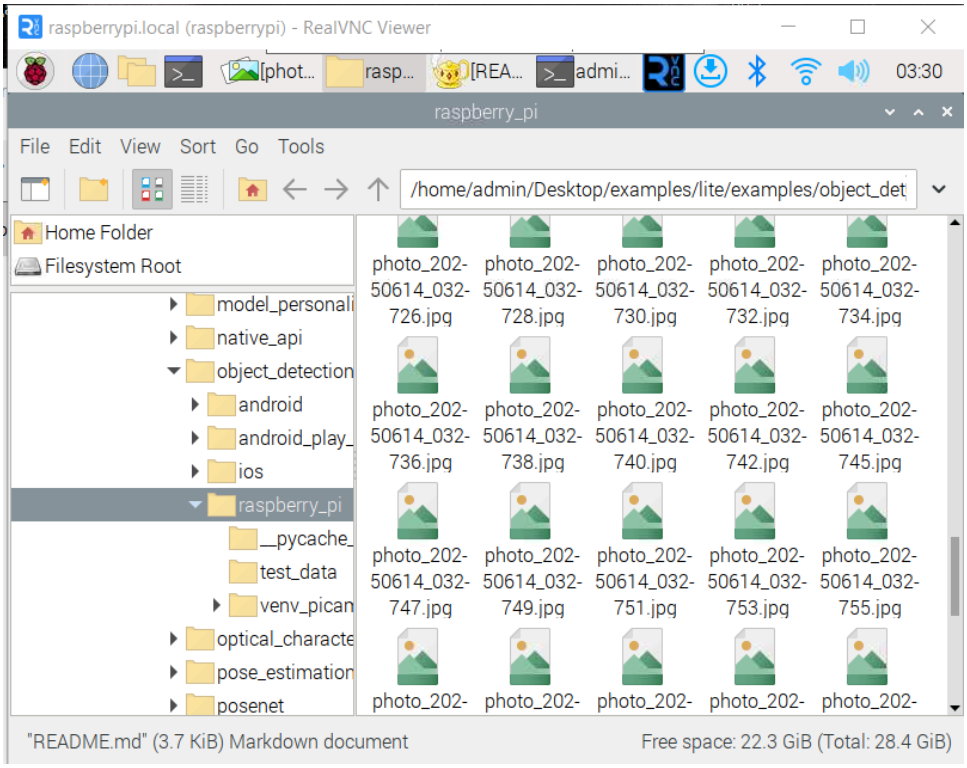


Рисунок 3.12 – Збереження файлів з фотографіями

Слід враховувати, що модель детектування об’єктів не є ідеальною через складність реалізації оптимального навчання. Проте, незважаючи на це, вона є доволі точною, оскільки спроможна детектувати об’єкти на великій відстані від себе. Для прикладу, було проведене тестування в рамках якого було детектовано птаха, дерево та людину. 3 об’єкти різні за фізичними характеристиками були детектовані, а отже це робить модель ефективною в умовах не тільки лабораторних, а й навколишнього середовища.

На рис. 3.13 зображений процес надання статусу, кольору об’єкту. Також показано, що саме за об’єкт детектовано. В цьому випадку, на надання статусу вплинуло розташування навушників та перекриття ними камери, що імітує можливе наближення FPV-дрону до потенціальної перешкоди. При віддалені камери, статус змінюється на зелений, а при додаванні інших перешкод – на червоний.

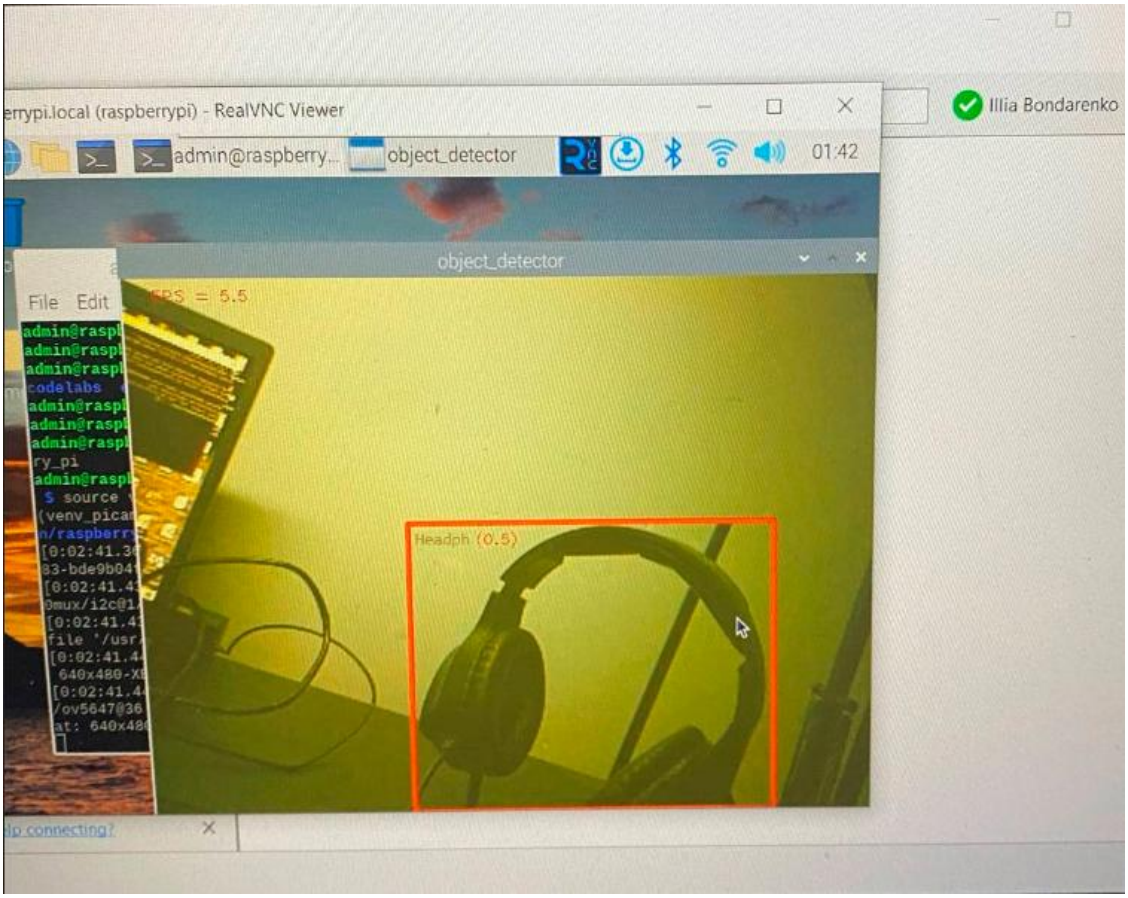


Рисунок 3.13 – Детектування навушників та присвоювання статусу

Модель додатково показує границі об'єкта та статус його можливого виявлення та ідентифікування в цифрах біля назви. Таким чином, це дає додаткову інформацію оператору про об'єкт, дозволяючи завагатись у випадку, коли модель неправильно проводить процес ідентифікації.

3.2.4 Реалізація передачі даних між Raspberry Pi та Android-пристроєм

Передача даних напряму на Андроїд пристрій можлива завдяки встановлення бібліотеки *flask* та редагування *object_detection.py* для виводу вже обробленого зображення на пристрій. У процесі роботи Flask приймає HTTP-запити від клієнта, аналізує адресу запиту та метод, а далі передає управління визначеній у коді функції-обробнику, що прив’язана до цієї адреси. Результатом роботи обробника є повернення відповіді клієнту у вигляді HTML-сторінки, JSON-даних або інших форматів.

Фреймворк використовує шаблонізатор Jinja2 для генерації HTML-контенту, що дозволяє відділити логіку застосунку від його візуального представлення. Завдяки цьому можна вставляти змінні, керувати відображенням за допомогою умов і циклів безпосередньо в HTML-шаблонах.

У Flask також реалізована підтримка сесій, cookies та обробки форм, що дає змогу зберігати стан між запитами, ідентифікувати користувача, керувати доступом до частин застосунку. Робота сервера в Flask заснована на інтерфейсі WSGI, що дозволяє інтегрувати його з більш складними вебсерверами.

Для підключення моделі необхідне віртуальне середовище. Було використано попередньо-створене середовище та введено команду:

```
pip install flask
```

Створюється файл з вебсервером, зі стандартним для нього 5000 хостом. В якості адреси використовується IP-адреса Raspberry. Опціонально, було додано автоматичний запуск Flask-сервера при запускі плати, що є особливо важливим в умовах підключення та застосування з РТС.

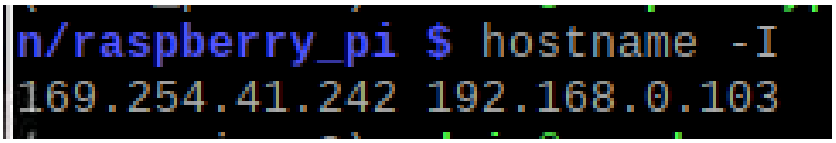


Рисунок 3.14 – Вивід команди hostname -I

При введені команди *hostname*, вивід – 2 адреси:

1) 169.254.41.242 – це автоматична (link-local) IP-адреса, яка з’являється, коли пристрій не може отримати адресу від DHCP-сервера (тобто маршрутизатора). Вона не використовується для нормального з'єднання в мережі;

2) 192.168.0.103 – це локальна IP-адреса у вашій мережі, яку видав маршрутизатор. Це саме та адреса, через яку можна підключатися до Raspberry Pi по SSH, HTTP, VNC.

Доцільне налаштування статичної IP-адреси, адже таким чином можливо позбутись постійного введення команд для детектування поточної адреси.

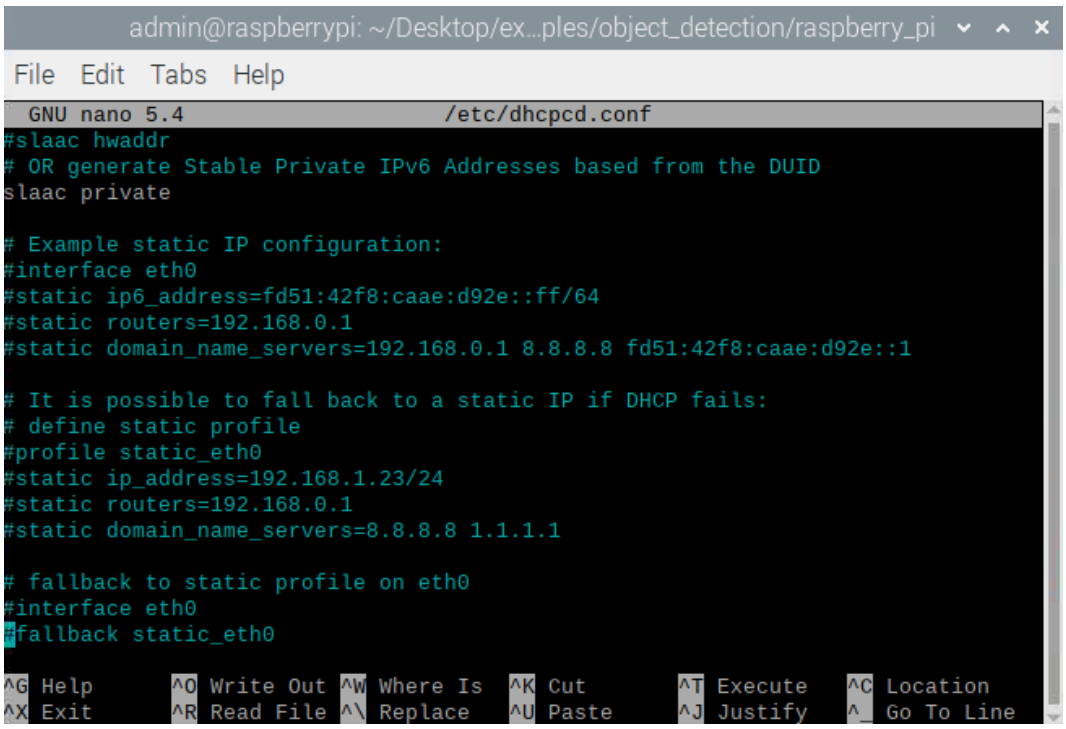


Рисунок 3.15– Вміст dhcpcd.conf

У Raspberry Pi мережеві інтерфейси за замовчуванням обслуговуються демоном dhcpcd, що динамічно отримує IP-адреси. Проте, якщо редагувати файл /etc/dhcpcd.conf, користувач має змогу перевизначити цю поведінку і встановити фіксовану адресу для конкретного інтерфейсу.

Таким чином, IP-адреса була налагоджена та в подальшому, вона використовувалась для підключення Android-пристроя. Для цього в будь-якому браузері вводиться IP-адреса та порт. Відбувається підключення та вивід переданих даних на екран. Такий підхід дозволяє не зосереджуватись лише на Android-застосунку, а навпаки дозволити без зайвих програмних середовищ передавати дані на пристрій, що підключений до спільної мережі Wi-Fi.

3.3 Напрямки потенційної модернізації та масштабування

Серед основних недоліків можна зазначити недостатню вантажопідйомність РТС, що негативно впливає на якість та стабільність системи цілому. Фактор унеможливлює залучення передового обладнання, таких як IP-камера, металевий каркас чи покращений акумулятор із захистом від дії зовнішніх факторів. Серед основних недоліків поточної реалізації апаратно-

програмного забезпечення варто виділити недостатню вантажопідйомність і енергетичну спроможність обчислювальної платформи Raspberry Pi 4 Model B, що суттєво обмежує можливості системи як у функціональному, так і у фізичному аспектах. Обмеження потужності живлення, тепловідведення та підтримки зовнішніх пристроїв призводять до нестабільної роботи всієї системи, особливо при спробах підключення високопродуктивних або енергоємних компонентів.

Зокрема, підключення IP-камери, яка вимагає стабільного живлення та високої пропускної здатності мережі, може спричинити перевантаження інтерфейсів або призвести до падіння системи внаслідок недостатнього струму живлення або перегріву процесора. Такі умови несумісні з побудовою надійної системи відеоспостереження або машинного зору, де необхідна постійна передача великих обсягів даних та їх обробка в реальному часі.

Додатково, використання металевого корпусу – хоч і покликаного покращити захист та охолодження – створює нові виклики для теплообміну, особливо в разі обмеженої вентиляції. Такий корпус має більшу масу і може збільшити навантаження на монтажну конструкцію, що важливо при створенні мобільних або портативних систем. Через обмежені можливості плати щодо підключення та живлення вентиляторів, реалізація активного охолодження всередині металевого корпусу стає складним завданням без зовнішніх джерел живлення або додаткової електроніки.

Аналогічні обмеження спостерігаються і у випадку з використанням вдосконалених акумуляторних систем. Літій-іонні батареї з вбудованими модулями захисту та контролю заряду мають більші габарити, вагу і вимоги до живлення, ніж може забезпечити стандартний інтерфейс живлення Raspberry Pi. Це унеможливлює інтеграцію систем автономного енергоживлення без використання проміжних компонентів, таких як понижуючі або підвищуючи перетворювачі, що вимагає додаткової технічної реалізації, впливає на загальну вартість і складність проєкту.

Шлях модернізації програмної складової включає в себе насамперед модернізацію існуючої моделі детектування об’єктів, її навчання та удосконалення точності. В такому випадку моделі необхідно надати потрібну кількість об’єкта потрібного класу.

Висновок до розділу 3

В ході проведеного дослідження була розроблена структура апаратно-програмного забезпечення. Виділено місце для системи детектування об'єктів з погляду на стабільність, енергоефективність та захищеність системи вцілому. Кут огляду камери дозволяє виявляти об'єкти на тій же відстані, що й основна камера БПЛА, що з боку зручності, не заважає пілоту орієнтуватись в двох вимірах. Паралельно до цього, було налагоджено канал зв'язку між Android-пристроєм та одноплатним комп’ютером Raspberry Pi. В лабораторних умовах, передача кольорового зображення здійснювалась через WiFi-з'єднання, проте подальше удосконалення здатне замінити таке рішення на інший бездротовий аналог.

Було засвоєно принцип роботи з фреймворком TensorFlow Lite в комплексі з одноплатним комп’ютером Raspberry Pi 4 Model B. Були задіяні бібліотеки обробки та масштабування зображень. Як результат – детектування та позначення потенційно небезпечних ділянок, надання звіту та попередження оператора РТС. Дві камери дозволяють уникнути затримки з боку виведення зображення, адже зображення виводяться на два пристрої, що дозволяє керувати РТС паралельно ведучи моніторинг місцевості. Проте, слід відзначити те, що шлях встановлення програмного забезпечення, таким чином з часом може застаріти, оскільки Raspberry OS оновлюється, старі бібліотеки замінюються новими, а нові доповнюються.

Також слід виокремити вибагливість одноплатного комп’ютера щодо живлення. Необхідне встановлення лише оригінального блоку живлення для стабільного налагодження роботи системи. Окрім цього, відсутній захист від пошкодження плати через застосування GPIO портів напряму в якості

провідника енергії. В таких випадках, краще переконатись в можливості сприймати електроенергію контролером та додати посередника між платою та акумулятором. З іншого боку, таке рішення збільшить навантаження на систему. Для таких випадків БПЛА здатний підняти в повітря ще 40 г попередньо залишеного місця. Все це, в комплексі з навичками оператора надає останньому недорогий та ефективний засіб запобігання утрати, а в випадку все ж таки утрати, детектування останнього місцезнаходження FPV-дрона.

Додатково варто зазначити перспективи розвитку запропонованої системи. Зокрема, можлива інтеграція з модулями штучного інтелекту, що працюють у реальному часі, включно з модифікованими моделями класифікації загроз на основі контексту (наприклад, виявлення руху в зоні потенційної небезпеки або прогнозування траєкторій перешкод). У разі впровадження таких функцій зростає актуальність питань кібербезпеки. Забезпечення безпечного каналу зв’язку, автентифікація на рівні пристроїв та шифрування переданих даних стають ключовими факторами під час розгортання системи в реальних умовах, особливо в зонах з обмеженим доступом або підвищеним ризиком втрати зв’язку.

ВИСНОВКИ

В результаті виконання кваліфікаційної бакалаврської роботи було розроблено апаратно-програмне забезпечення для виявлення перешкод та координування дій оператора з використанням TensorFlow-фреймворку.

Досягнута поставлена мета – реалізовано ефективний каналу зв’язку між Raspberry Pi та Android-пристроєм для забезпечення передачі даних детектування об’єктів у реальному часі. Зазначену мету досягнуто завдяки розв’язанню наступних завдань:

– обґрунтовано актуальність проблем розпізнавання перешкод;

– проведено аналіз теоретичної складової розробки проєкту та існуючих рішень;

– обрано програмну платформу для реалізації процесу фіксування та передачі даних, а також методи та технології для формулювання вимог до АПЗ;

– здійснено проєктування та визначено архітектуру системи;

– розроблено програмне забезпечення, реалізовано алгоритми та інтегровано їх з апаратною платформою;

– виконано тестування, набуто навички керування дроном через віртуальну симуляцію для оцінки ефективності запропонованого рішення та верифікації результатів в реальних умовах.

На основі теми КБР проведено власне дослідження з питання ефективності РТС в цивільній та військовій сфері. Зазначено необхідність залучення БПЛА до процесів оборони приватної власності та пошукових місій, а також їх важливість для науково-технологічного прогресу в цілому.

У процесі підготовки здобуто практичні навички роботи з РТС, здійснювалося їх налаштування та оптимізація для конкретних завдань. Також опрацьовано спеціалізовану термінологію, яка використовується у відповідному середовищі, та проаналізовано низку технічних джерел. Наведено результати наукових досліджень, представлено діаграми та схеми для уточнення сказаного. При використанні зовнішніх матеріалів особлива увага приділялася дотриманню

академічної доброчесності, правильному оформленню цитувань і уникненню плагіату.

Висвітлено логіку роботи бібліотеки TensorFlow Lite, розроблено функціональну структуру взаємодії між дроновою системою та Android-платформою, а також розглянуто питання навантаження на одноплатний комп’ютер та вплив частоти циклів детектування на загальну ефективність системи. Опрацьовано можливі рішення та пропозиції з покращення роботи проєкту та визначено необхідний функціонал системи.

На основі зібраних даних визначено актуальні напрями вдосконалення РТС з урахуванням новітніх підходів до розпізнавання об'єктів у реальному часі. Також розглянуто можливості інтеграції відкритого програмного забезпечення для створення гнучких та адаптивних систем, що відповідають сучасним вимогам безпеки та ефективності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондаренко І., Обухова К. Реалізація каналу зв'язку між Raspberry PI та Android для детектування об'єктів у реальному часі. *Ольвійський форум – 2025: стратегії країн Причорноморського регіону в геополітичному просторі* : тези доп. XXII Міжнар. наук. конф., Миколаїв, 16–21 черв. 2025 р. Миколаїв : ЧНУ ім. Петра Могили, 2025 *(подано до друку)*.

2. Ергономіка PTC. URL: https://aerobotlabs.com/unveiling-the-inner-workings-exploring-the-components-of-uavs/ (дата звернення: 12.05.2025).

3. Зображення пульту управління DJI Matrice 300 RTK URL: https://ecodrive.in.ua/kvadrokopter-dron-dji-matrice-350-rtk-enterprise-cp-en-00000468-01-dji-zenmuse-h20t-cp-zm-00000121-01/ (дата звернення: 12.05.2025).

4. Квадрокоптер DJI Phantom 4. URL: https://www.dronarium.com.ua/shop/drony/kvadrokopter-dji-phantom-4/ (дата звернення: 20.05.2025).

5. Мультикоптер DJI Matrice 300 RTK. URL: https://store.quadro.ua/multikopter-dji-matrice-300-rtk/?srsltid=AfmBOorB-BBsv1vF3Eux32VO-PVla1x10kzYfHUpevWDj7BC_omJOERS (дата звернення: 08.05.2025).

6. Огляд FPV дронів: Розбираємо компоненти та їхню роль у відтворенні захоплюючого польотного досвіду. URL: https://drone-azimyth.com.ua/index.php?route=blog/article&article_id=126&srsltid=AfmBOoojW8PMCN56_taK2oFH0GHyC9uUA5RD2cwB9HY_RLoUDaOnSnfR (дата звернення: 20.05.2025).

7. Підвіс та камера DJI Zenmuse X7 (без об'єктива). URL: https://dronestore.com.ua/shop/dji-zenmuse-x7/?srsltid=AfmBOopUVVqOGboQNmm7ZIbZ5hWswHf_3pz6VtXQc9xTRjzPy2rrt6bs (дата звернення: 08.05.2025).

8. Рій дронів та божевільні покупці. Як інженери перетворюють виробництво FPV на велику індустрію. URL:

https://epravda.com.ua/publications/2024/07/25/717109/ (дата звернення: 08.05.2025).

9. Статистика використання FPV-дронів на полі бою. URL: https://defence-ua.com/minds_and_ideas/zibrano_detalnu_statistiku_vikoristannja_fpv_droniv_na_poli_boju_jaki_mozhna_zrobiti_visnovki-14298.html (дата звернення: 08.05.2025).

10. Статистика ефективності безпілотних літальних апаратів. URL: https://www.ukrinform.ua/rubric-ato/3972253-droniv-na-optovolokni-u-sil-oboroni-stane-bilse-do-kinca-lita-komandir-polku-ahilles.html (дата звернення: 08.05.2025).

11. Україна закупить 4,5 мільйони FPV-дронів у 2025 році. URL: https://militarnyi.com/uk/news/ukrayina-zakupyt-4-5-miljony-fpv-droniv-u-2025-rotsi/?utm_source=chatgpt.com (дата звернення: 08.05.2025).

12. ANAFI USA model. URL: https://www.parrot.com/en/drones/anafi-usa/buy (дата звернення: 08.05.2025).

13. Bursa S., Incel O. Building Lightweight Deep learning Models with TensorFlow Lite for Human Activity Recognition on Mobile Devices. *Annals of Telecommunications*. 2023. Vol. 78, Is. 11. P. 687–702. DOI: 10.1007/s12243-023-00962-x/.

14. Demosthenous G., Vassiliades V. Continual Learning on the Edge with TensorFlow Lite. May, 2021 (Preprint). DOI: 10.48550/arXiv.2105.01946.

15. How to Run TensorFlow Lite Object Detection Models on the Raspberry Pi (with Optional Coral USB Accelerator). URL: https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/deploy_guides/Raspberry_Pi_Guide.md (Last accessed: 12.05.2025).

16. Kharve A. Leveraging TensorFlow Lite and Camera2 API for Efficient Real-Time Object Detection in Android Apps Using Kotlin. *International Scientific Journal of Engineering and Management*. 2025. Vol. 4, Is. 6. P. 1–9. DOI: 10.55041/ISJEM04033.

17. 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.

18. Oleksenko O., Huk O., Snitsarenko V., Ikaiev D., Rahulin V. Regarding the features of the combat application of FPV drones in the war between russia and Ukraine. *Наукові праці Державного науково-дослідного інституту випробувань і сертифікації озброєння та військової техніки*. 2024. Vol. 22. P. 99–105. DOI: 10.37701/dndivsovt.22.2024.12.

19. Raspberry Pi 3 vs Raspberry Pi 4 Performance with TensorFlow, TF Lite, & Coral USB Accelerator. URL: https://www.youtube.com/watch?v=TiOKvOrYNII (дата звернення: 20.05.2025).

20. Raspberry Pi 4 Model B. URL: https://evo.net.ua/korpus-dlya-raspberry-pi-4-model-b-c-aktivnym-okhlazhdeniem/?srsltid=AfmBOooRIyQR6S7K9o3TxqXAQowBDfW7tAf1LNDRBKGdYPTgM665OMXv (дата звернення: 08.05.2025).

21. Raspberry Pi 4 Model B. URL: https://rozetka.com.ua/388063668/p388063668/ (дата звернення: 12.05.2025).

22. Siddiqui W. Design and Fabrication of FPV Racing Drone. *Journal of Mechanical and Construction Engineering (JMCE)*. 2024. Vol. 4. P. 1–8. DOI: 10.54060/a2zjournals.jmce.38.

23. The FLIR Duo Pro R: A Powerful, Professional Tool for Drones Built for the Future. URL: https://dronelife.com/2017/11/02/thermal-camera-right-flir-duopro/ (Last accessed: 08.05.2025).

ДОДАТОК А

Блок-схеми алгоритмів

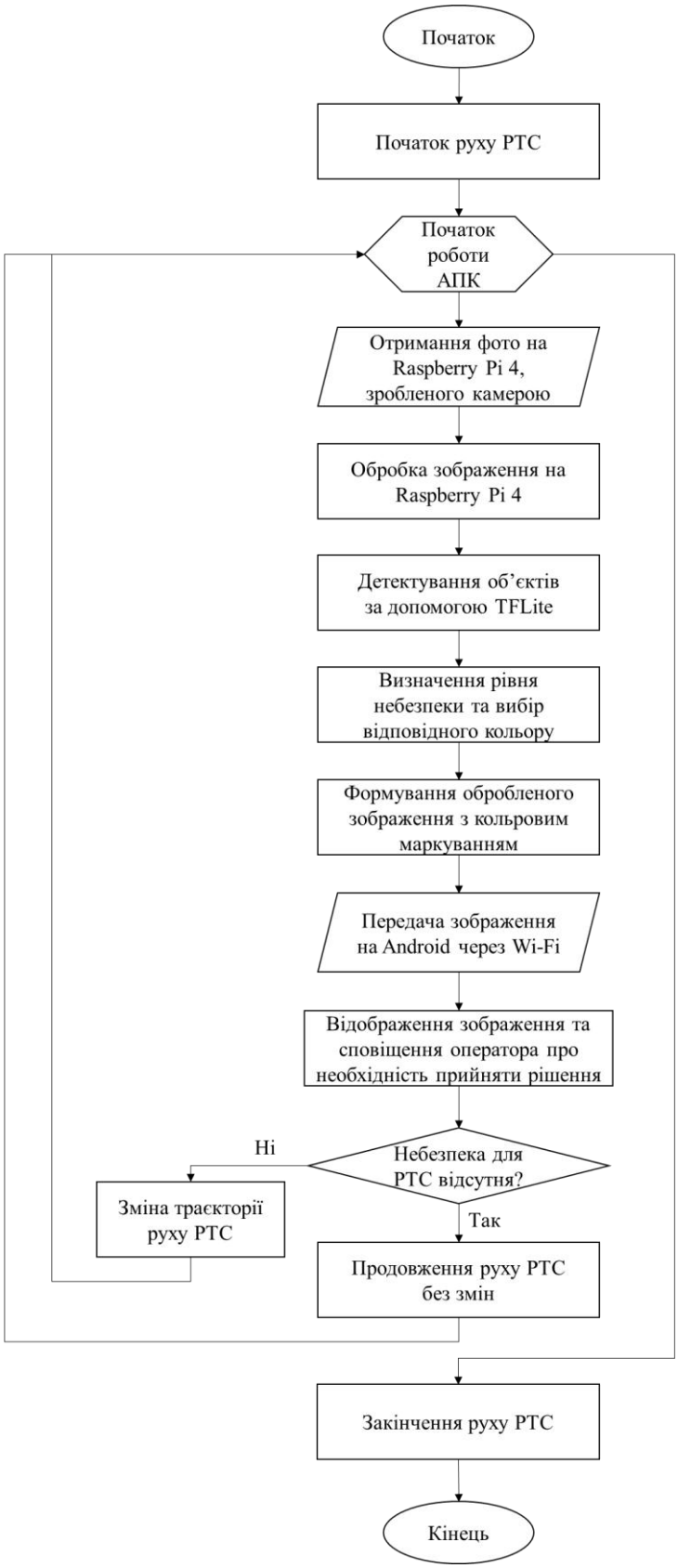


Рисунок А.1 – Блок-схема алгоритму роботи РТС

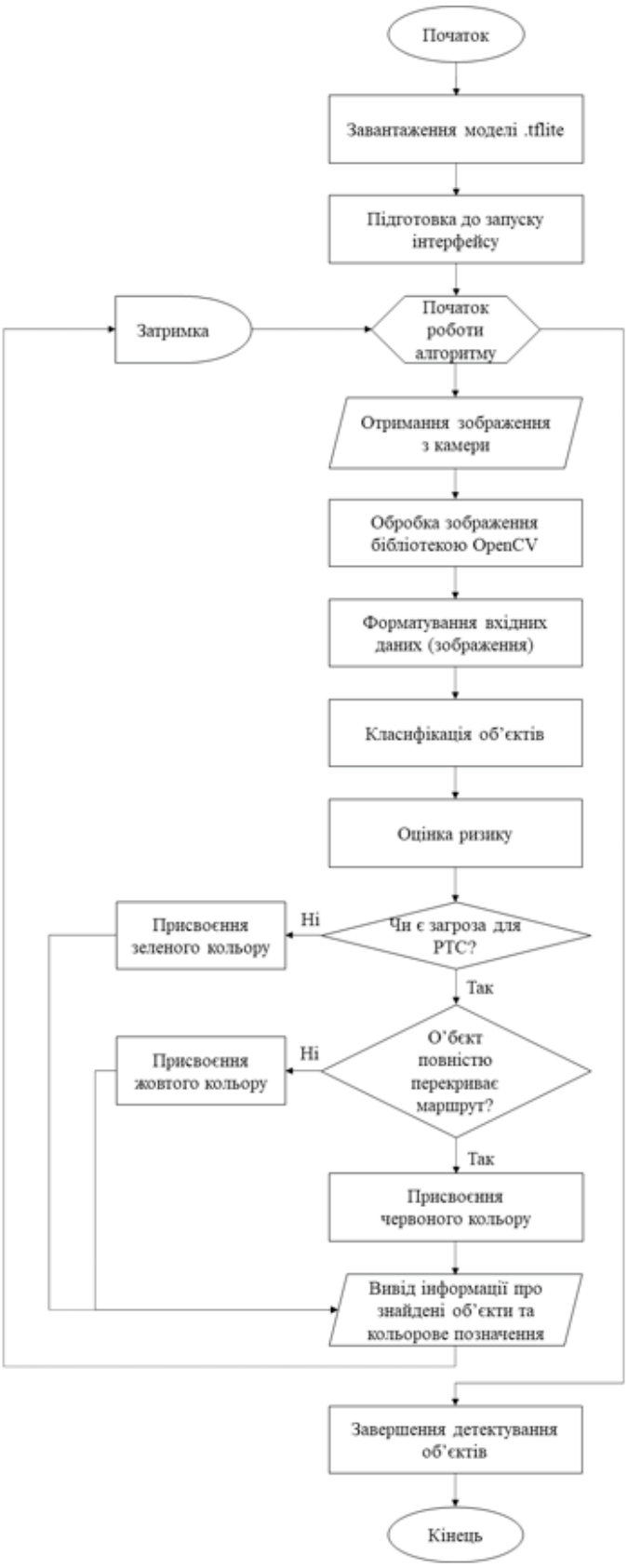


Рисунок А.2 – Блок-схема алгоритму визначення рівня небезпеки на основі TensorFlow Lite

ДОДАТОК Б

Код програми детектування

Лістінг коду object_detection.py

```
import argparse
import sys
import time
import cv2
import numpy as np
from picamera2 import Picamera2
from tflite_support.task import core
from tflite_support.task import processor
from tflite_support.task import vision
import utils
from datetime import datetime
def run(model: str, width: int, height: int, num_threads: int, enable_edgetpu: bool) ->
None:
    counter, fps = 0, 0
    start_time = time.time()
    last_photo_time = time.time()
    picam2 = Picamera2()
    picam2.configure(picam2.create_preview_configuration(main={"size": (width,
height)}))
    picam2.start()
    row_size = 20
    left_margin = 24
    text_color = (0, 0, 255)
    font_size = 1
    font_thickness = 1
    fps_avg_frame_count = 10
    base_options = core.BaseOptions(file_name=model, use_coral=enable_edgetpu,
num_threads=num_threads)
    detection_options = processor.DetectionOptions(max_results=3, score_threshold=0.3)
    options = vision.ObjectDetectorOptions(base_options=base_options,
detection_options=detection_options)
    detector = vision.ObjectDetector.create_from_options(options)
    frame_w, frame_h = width, height
    center_x, center_y = frame_w // 2, frame_h // 2
    center_margin = 100
    while True:
        image = picam2.capture_array()
        if image is None:
            sys.exit("ERROR: Failed to capture image from Raspberry Pi Camera.")
        counter += 1
        image = cv2.flip(image, 1)
        rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        input_tensor = vision.TensorImage.create_from_array(rgb_image)
        detection_result = detector.detect(input_tensor)
        # === Аналіз об'єктів ===
        red_count = 0
        yellow_count = 0
        for det in detection_result.detections:
            score = det.categories[0].score
            bbox = det.bounding_box
            x_min, y_min = bbox.origin_x, bbox.origin_y
            x_max, y_max = x_min + bbox.width, y_min + bbox.height
            obj_in_center = (
                center_x - center_margin < (x_min + x_max) / 2 < center_x +
center_margin and
```

```
                center_y - center_margin < (y_min + y_max) / 2 < center_y +
center_margin
            )
            if score > 0.8 or (obj_in_center and bbox.width * bbox.height > 0.2 *
frame_w * frame_h):
                red_count += 1
            elif 0.5 <= score <= 0.75:
                yellow_count += 1
        # === Вибір кольору ===
        if red_count >= 2:
            overlay_color = (0, 0, 255)  # RED
        elif yellow_count >= 1:
            overlay_color = (0, 255, 255)  # YELLOW
        else:
            overlay_color = (0, 255, 0)  # GREEN
        # === Накладання напівпрозорого кольору ===
        overlay = image.copy()
        cv2.rectangle(overlay, (0, 0), (frame_w, frame_h), overlay_color, -1)
        alpha = 0.3
        image = cv2.addWeighted(overlay, alpha, image, 1 - alpha, 0)
        # === Візуалізація ===
        image = utils.visualize(image, detection_result)
        # === FPS ===
        if counter % fps_avg_frame_count == 0:
            end_time = time.time()
            fps = fps_avg_frame_count / (end_time - start_time)
            start_time = time.time()
        fps_text = 'FPS = {:.1f}'.format(fps)
        text_location = (left_margin, row_size)
        cv2.putText(image, fps_text, text_location, cv2.FONT_HERSHEY_PLAIN,
                    font_size, text_color, font_thickness)

        # === Збереження фото кожні 2 секунди ===
        current_time = time.time()
        if current_time - last_photo_time >= 2.0:
            filename = datetime.now().strftime("photo_%Y%m%d_%H%M%S.jpg")
            cv2.imwrite(filename, image)
            last_photo_time = current_time
        # === Показ зображення ===
        cv2.imshow('object_detector', image)
        if cv2.waitKey(1) == 27:
            break
    cv2.destroyAllWindows()
    picam2.stop()
def main():
    parser =
argparse.ArgumentParser(formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('--model', default='efficientdet_lite0.tflite',
                        help='Path of the object detection model.')
    parser.add_argument('--frameWidth', default=640, type=int,
                        help='Width of frame to capture from camera.')
    parser.add_argument('--frameHeight', default=480, type=int,
                        help='Height of frame to capture from camera.')
    parser.add_argument('--numThreads', default=4, type=int,
                        help='Number of CPU threads to run the model.')
    parser.add_argument('--enableEdgeTPU', action='store_true',
                        help='Whether to run the model on EdgeTPU.')
    args = parser.parse_args()
    run(args.model, args.frameWidth, args.frameHeight,
        args.numThreads, args.enableEdgeTPU)
if __name__ == '__main__':
    main()
```

ДОДАТОК В

Матеріали апробації роботи

DOI

Ілля Бондаренко,
Катерина Обухова

РЕАЛІЗАЦІЯ КАНАЛУ ЗВ'ЯЗКУ МІЖ RASPBERRY PI ТА ANDROID ДЛЯ ДЕТЕКТУВАННЯ ОБ'ЄКТІВ У РЕАЛЬНОМУ ЧАСІ

Тези присвячено аналізу інтеграції мобільних і вбудованих платформ для створення апаратно-програмного комплексу реального часу детектування об'єктів і потенційних загроз. Основна увага зосереджена на забезпеченні стабільного та енергоефективного каналу зв'язку між компонентами системи.

Ключові слова: *роботизована технічна система, FPV-дрон, Raspberry Pi 4, фреймворк TensorFlow Lite, детектування об'єктів та перешкод.*

З розвитком цифрових технологій, передусім у сфері відеопередачі, а також завдяки мініатюризації апаратних компонентів стало можливим створювати легкі мобільні комплекси з підтримкою високоякісного відеозв'язку, значною дальністю польоту та підвищеною стабільністю каналу. Це сприяло активному поширенню FPV-дронів у цивільному секторі – зокрема для інспекцій промислових об'єктів та інфраструктури, сільськогосподарського й екологічного моніторингу, рятувальних операцій, а також у воєнній сфері. Дедалі частіше роботизовані технічні системи (РТС) використовуються в розважальних цілях (напр., кінозйомки та телевізійного виробництва), проте навіть в такому випадку потребують спеціального налаштування [1]. Все більше стає помітним втручання РТС в сфери, де людська діяльність може бути неефективною або надто тривалою. Актуальність детектування перешкод спричинена необхідністю зменшення людського фактору під час процесів ідентефікації об'єктів та унеможнення можливих загроз апаратної складової. Основним завданням є забезпечення безперебійного зв'язку між мікроконтролером та оператором, і як результат – фотозвіт з вказаним рівнем безпеки навколишнього середовища.

Необхідність у РТС для детектування пояснюється тим, що під час польоту FPV-дрон може зіштовхнутися з непередбачуваними об'єктами – як природними (дерева, птахи), так і техногенними (лінії електропередач, будівельні конструкції), які важко заздалегідь виявити візуально. У таких умовах автоматичне фіксування та аналіз перешкод дозволяє своєчасно змінити маршрут польоту або попередити оператора про потенційну небезпеку. Це особливо важливо в сценаріях автономного або напівавтономного управління, де зволікання може спричинити втрату цінного обладнання або створити небезпеку для людей.

При виборі платформи враховувалась потреба в моніторингу масштабних територій та швидкому реагуванні на об'єкт. Сферою застовування виступає FPV-дрон як окрема рухома багатоядерна комп'ютерна система з можливістю розміщення на ній елементів детектування та обробки інформації з подальшим передаванням на мобільний пристрій. DJI Phantom 4 є оптимальним вибором, адже забезпечує стабільність польоту та дальність польоту (до 5 км), що є достатнім для моніторину територій середнього масштабу [2]. Батарея розміщується під корпусом, що покращує аеродинамічні властивості дрона. Орієнтована вантажопідйомність становить 200–300 гр., чого достатньо для перевезення апаратних компонентів (рис. 1). Саме кріплення має бути збалансованим, щоб не зміщувати центр мас, а габарити компонентів дозволять їх розташування відносно центру.

Рис. 1. Підбір компонентів з урахуванням їх ваги

Основним апаратним компонентом є мікроконтролер Raspberry Pi 4 Model B, який виконує обробку даних. Його потужності достатньо для реалізації детектування об’єктів за допомогою фреймворку TensorFlow Lite (TFLite) по фото або відео з низькою частотою кадрів (FPS). Окрім цього, він забезпечує підтримку камер, периферійних пристроїв, а також оснащений WiFi-модулем, що дозволяє передавати дані на Android без додаткових затримок. Центральний процесор обробляє вхідні дані (фото- або відеопотік) та виконує алгоритми машинного навчання з використанням відповідного фреймворку. Значний обсяг оперативної пам’яті гарантує підтримку стабільної роботи операційної системи, службових процесів та дає змогу обробки проміжних результатів у межах моделі глибинного навчання.

Графічний процесор прискорює обробку графічних даних, зокрема у декодуванні відеопотоку, який передається до основного обчислювального модуля. Wi-Fi та Ethernet дозволяють здійснювати двосторонню комунікацію з зовнішніми пристроями, зокрема Android-системами, для передачі інформації про виявлені об’єкти чи отримання керуючих сигналів.

Оскільки запуск фреймворку потребує значних обчислювальних ресурсів, плата не застрахована від перегріву. Отже потрібно охолоджувати центральний процесор для забезпечення тривалої роботи мікроконтролера. Обрано алюмінієвий корпус зі вбудованими двома кулерами для запобігання перегріву та ушкодженням від можливих перешкод на шляху РТС.

Архітектура програмної частини комплексу є багаторівневою. Вона складається з блоку локальної обробки даних на Raspberry Pi 4, блоку взаємодії з камерою та мобільного застосунку на Android.

Вибір фреймворку TensorFlow Lite обумовлений використанням мікроконтролера, ресурси якого обмежені. Їх недостатньо для використання в комплексі з повнофункціональною версією TensorFlow. TFLite є оптимізованою версією TensorFlow, розробленою спеціально для запуску попередньо навчених моделей на мобільних та вбудованих пристроях із обмеженими ресурсами, таких як Raspberry Pi або мобільні платформи. Окремо слід зазначити можливість інтеграції бібліотеки TFLite в Android-застосунок, що забезпечує швидке виконання алгоритмів машинного навчання з низькою затримкою та енергоефективністю. TFLite працює з спеціалізованими моделями MobileNet, YOLO або SSD та дозволяє розпізнавати та детектувати об’єкти в реальному часі. Обов’язковим є підключення бібліотеки OpenCV [3], для використання функцій захоплення відео з камери та обробки зображень. Можливі конфлікти під час встановлення бібліотек

вирішуються за допомогою virtualenv – інструменту для створення ізольованого Python-середовища. Операційна система (ОС) – Raspberry Pi OS забезпечує повну сумісність із Python, OpenCV та TensorFlow Lite. ОС Android має бути не нижче 9-ї версії, адже вона дозволяє використання протоколів HTTP та WebSocket. Оброблені зображення передаються у форматі JSON-повідомлень.

На рис. 2 зображено процес детектування перешкод. Приклад демонструє ефективність фреймворку TensorFlow для фіксування та розпізнавання об'єктів.

Рис. 2. Приклад детектування перешкод (дерев) [4]

На етапі обробки кожен кадр отримує пріоритет залежно від кольору:

– зелений колір вказує на відсутність небезпеки та можливість безперешкодного перетину сектору;

– жовтий колір вказує на наявність ймовірної небезпеки та можливість пошкодження дрону;

– червоний колір повідомляє про небезпечну ділянку та необхідність негайно покинути територію через загрозу втрати РТС.

На рис. 3 наведено результат детектування перешкод на прикладі густини лісу: чим щільніше розташовані дерева, тим важче продовжувати політ.

Рис. 3. Приклад результату обробки

Таким чином, оператор отримує оброблене зображення та на основі нього має можливість координувати траєкторію FPV-дрону відповідно до прийнятого рішення. Цей підхід дозволяє прискорити процес реагування на перешкоди, а в разі помилки програмного забезпечення керування переходить напряму до оператора. Зважаючи на це, доцільно підсумувати загальний процес передачі інформації та детектування небезпеки (рис. 4).

Рис. 4. Алгоритм роботи апаратно-програмного комплексу

Запропонований підхід ґрунтується на поєднанні мобільних обчислювальних платформ та сучасних алгоритмів машинного навчання. Це рішення забезпечує високу швидкість обробки даних (фото- та відеопотоку) та стабільний зв’язку між FPV-дроном і оператором. Використання Raspberry Pi 4 у поєднанні з Android-пристроями через бездротовий канал дозволяє створити компактний, гнучкий й енергоефективний комплекс, здатний адаптуватися до різних умов. Додаткові компоненти, такі як кулери та алюмінієвий корпус, підвищують функціональність мікроконтролера, забезпечуючи його надійну роботу при будь-яких умовах навколишнього середовища. Застосування фреймворку TensorFlow Lite дозволяє ефективно виконувати завдання детектування без перевантаження апаратних ресурсів.

Реалізація такого рішення не лише відповідає сучасним вимогам до безпеки та оперативності, а й відкриває нові перспективи для розвитку автономних систем контролю з використанням роботизованих технічних систем у цивільному та військовому секторах.

Список використаних джерел

1. Siddiqui W. Design and Fabrication of FPV Racing Drone. *Journal of Mechanical and Construction Engineering (JMCE)*. 2024. Vol. 4, No. 1. P. 1–8. DOI: 10.54060/a2zjournals.jmce.38.

2. Support for Phantom 4. URL: https://www.dji.com/support/product/phantom-4 (Last accessed: 17.05.2025).

3. How to Run TensorFlow Lite Object Detection Models on the Raspberry Pi (with Optional Coral USB Accelerator). URL: https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/deploy_guides/Raspberry_Pi_Guide.md (Last accessed: 12.05.2025).

4. How to detect a single tree from a drone imagery of a dense forest? Nothing is simpler than custom object detection using TensorFlow. URL: https://blog.skygate.io/how-to-detect-a-single-tree-from-a-drone-imagery-of-a-dense-forest-de190f64cdb7 (Last accessed: 17.05.2025)

Illia Bondarenko,
Kateryna Obukhova

IMPLEMENTATION OF A COMMUNICATION CHANNEL BETWEEN RASPBERRY PI AND ANDROID FOR REAL-TIME OBJECT DETECTION

The paper are devoted to the integration of mobile and embedded platforms for the development of a hardware-software complex for objects and threats detection in real-time. Special attention is given to ensuring a stable and energy-efficient communication channel between system components.

Keywords: *robotic technical system, FPV drone, Raspberry Pi 4, TensorFlow Lite framework, object and obstacle detection.*

Відомості про автора / Information about the Author

Ілля Бондаренко, бакалаврант кафедри комп'ютерної інженерії, Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: ilyabondarenko66@gmail.com, orcid: https://orcid.org/0009-0000-9345-0286.

Illia Bondarenko, Bachelor's student of the Computer Engineering Department at Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. Email: ilyabondarenko66@gmail.com, orcid: https://orcid.org/0009-0000-9345-0286.

Катерина Обухова, ст. викладач кафедри комп'ютерної інженерії, Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: kateryna.obukhova@chmnu.edu.ua, orcid: https://orcid.org/0000-0001-8793-7055.

Kateryna Obukhova, senior lecturer of the Computer Engineering Department at Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. E-mail: kateryna.obukhova@chmnu.edu.ua, orcid: https://orcid.org/0000-0001-8793-7055.