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

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Система стабілізації параметрів дрона-ретранслятора
Назва теми

КвРКІ 2301261.21.01.46 ПЗ
Шифр

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

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

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

Виконала: студентка IV курсу, група КІ2-21-1 _____ Ярослава КОЖЕМЯКО
Підпис Ініціали, прізвище

Керівник _____ Ольга ПАВЛОВА
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис Ініціали, прізвище

«05» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Ярославі КОЖЕМЯКО

Ім'я, прізвище студента

1. Тема проекту (роботи) Система стабілізації параметрів дрона-ретранслятора

Керівник проекту (роботи) Ольга ПАВЛОВА, доктор філософії

Ім'я, прізвище, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

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

Аналіз предметної області стабілізації FPV-дронів-ретрансляторів із використанням технологій комп'ютерного зору

Вибір компонентів для розробки системи стабілізації FPV-дрона-ретранслятора на основі обробки зображення

Опис програмно-апаратної реалізації системи стабілізації FPV-дрона-ретранслятора на основі обробки зображення

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

Архітектура ПЗ проєкту

Алгоритми роботи системи

Апаратне забезпечення проєкту

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – вибір компонентів для розробки системи стабілізації FPV-дрона-ретранслятора на основі обробки зображення	01.04.2025	виконано
5	Робота над розділом 3 – реалізація системи стабілізації FPV-дрона-ретранслятора на основі обробки зображення	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студентка _____ Ярослава КОЖЕМЯКО
Підпис Ініціали, прізвище

Керівник роботи _____ Ольга ПАВЛОВА
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 2301261.21.01.46 ПЗ	Пояснювальна записка	70		
			Графічні матеріали			
2		КвРКІ 2301261.21.01.46 Е8	Архітектура ПЗ проєкту	1		
3		КвРКІ 2301261.21.01.46 Е8	Алгоритми роботи системи	1		
4		КвРКІ 2301261.21.01.46 Е8	Апаратне забезпечення проєкту	1		

КвРКІ 2301261.21.01.46 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробила		Кожемяко		05.06.25
Перевір.		Павлова		05.06.25
Н. контр.		Кисіль		05.06.25
Затв.		Павлова		05.06.25

Відомість проекту

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ2-21-1

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система стабілізації параметрів дрона-ретранслятора».

Автор роботи: Ярослава КОЖЕМЯКО.

Керівник роботи: Ольга ПАВЛОВА.

Пояснювальна записка: 70 с., 13 рис., 2 табл., 4 дод., 52 джерела.

Графічна частина: 3 креслення.

БПЛА, FPV-дрон, дрон-ретранслятор, комп'ютерний зір, система стабілізації, автономне керування.

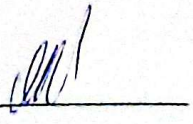
Метою дипломної роботи є розробка системи стабілізації для дрона-ретранслятора з використанням методів комп'ютерного зору та обробки зображень у реальному часі.

Об'єктом дослідження є процес стабілізації FPV-дронів для перенесення ретранслятора.

Предметом дослідження є використання алгоритмів комп'ютерного зору для стабілізації дрона-ретранслятора шляхом аналізу відеопотоку з тепловізійної камери.

У межах дослідження проведено систематичний огляд сучасних наукових і технічних джерел, що дало змогу сформувати концепцію побудови системи стабілізації на основі візуального позиціонування.

Для реалізації програмної частини було застосовано методи структурного проєктування та ітеративної розробки програмного забезпечення, з використанням бібліотек комп'ютерного зору, інтерфейсів телеметрії, а також симуляційного тестування.



Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП

На тлі сучасного воєнного конфлікту в Україні відбувається стрімкий розвиток технологій ведення бойових дій, у якому безпілотні літальні апарати (БПЛА) відіграють ключову роль. Камерні дрони з високоточними системами стабілізації, навігації та автономного керування давно стали невід'ємною частиною оснащення армій світу. Проте їхня висока вартість, обмежена доступність та чутливість до втрат роблять їх уразливими у виснажливій війні, де ресурси швидко вичерпуються.

На противагу їм, FPV-дрони дешевші, швидші та більш масові, активно застосовуються як розвідувальні платформи, ударні системи або засоби ретрансляції сигналу. Їхня низька ціна та простота дозволяють випускати їх у великій кількості, але брак вбудованих систем стабілізації значно обмежує їхню ефективність у завданнях, що вимагають точного позиціонування, зокрема при розгортанні мереж зв'язку в умовах відсутності GPS або в складній місцевості.

Водночас технологічне протистояння на полі бою потребує нових рішень для підвищення автономності дешевих дронів без значного збільшення їхньої вартості. Одним із таких напрямів є впровадження методів комп'ютерного зору та обробки відеосигналу як альтернативи інерційним чи супутниковим системам стабілізації. Зокрема, використання тепловізійних камер дозволяє забезпечити стабільну роботу дрона навіть за умови поганої видимості.

Актуальність дослідження полягає у необхідності розробки ефективних візуальних методів стабілізації для FPV-дронів, здатних функціонувати у складних середовищах без залучення зовнішньої навігаційної інфраструктури. Запропоноване рішення дозволяє забезпечити автономне позиціонування для виконання завдань ретрансляції у реальному часі.

Метою дипломної роботи є проєктування та реалізація системи стабілізації FPV-дрона-ретранслятора на основі аналізу зображення з тепловізійної камери,

здатної забезпечити стійке позиціонування апарата в просторі без використання GPS або інших високовартісних засобів навігації.

Об’єктом дослідження є процес стабілізації FPV-дронів, які використовуються як мобільні платформи для ретрансляції зв’язку в умовах бойових дій.

Предметом дослідження є застосування технологій комп’ютерного зору, зокрема обробки відеосигналу та алгоритмів оптичного потоку, для візуальної стабілізації положення дрона у просторі.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Основні поняття концепції стабілізації FPV-дронів-ретрансляторів

FPV розшифровується як First Person View, тобто пілот дрона бачить саме те, що бачить камера дрона в режимі реального часу за допомогою спеціальних відео-окуляр [1]. Завдяки цьому створюється відчуття, наче пілот і дрон є одним цілим, що забезпечує надзвичайно точне та динамічне керування.

На противагу FPV-дронам, традиційні безпілотні літальні апарати зазвичай керують у зоні прямої видимості або через зображення на екрані смартфона, на якому відображається інформація з камери дрона. Хоча пересічна людина зазвичай може доволі швидко навчитися керувати так званими «камерними дронами», дрони із фокусом на якість відеозйомки, процес оволодіння FPV-дронами є суттєво складнішим і технічно інтенсивнішим. Це зумовлено, зокрема, відсутністю в більшості FPV-дронів автоматичних систем стабілізації.

Однією з ключових відмінностей FPV польотів є ручне керування. На відміну від більшості споживчих дронів, які мають GPS-стабілізацію та систему уникнення перешкод, FPV-дрони зазвичай літають у «акрорежимі» (ACRO mode), де пілот має повний контроль над тангажем, рисканням і креном без автоматичних коригувань. Це дає змогу виконувати сальто, перекиди та інші складні маневри, але також вимагає високого рівня майстерності та координації.

Для досвідченого пілота відсутність системи автоматичної стабілізації під час польоту зазвичай не становить суттєвої проблеми, за умови наявності повного контролю над усіма трьома осями – тангажем, креном і рисканням, та власне тягою. Такий пілот здатен утримувати дрон у стабільному положенні завдяки власним умінням, навіть за відсутності допоміжних алгоритмів стабілізації, характерних для «камерних дронів» [2].

Однак реалізація цього контролю безпосередньо залежить від якості зв’язку між дроном та пультом керування або наземною станцією. Зі збільшенням дистанції або в умовах наявності значних перешкоди в навколишньому середовищі

(наприклад, пересічена місцевість, лісові масиви, густонаселені райони, де сигнал може блокуватись або відбиватись від будівель) якість з’єднання може різко погіршуватися. Це не лише ускладнює управління дроном, а й створює ризики втрати зв’язку або затримки в передачі критичних команд.

У таких випадках доцільним є використання ретранслятора – пристрою або окремої платформи, яка приймає і передає сигнали між оператором і дроном [3]. Ретранслятор забезпечує стабільний радіоканал у важкодоступних зонах, розширює ефективний радіус дії системи керування та дозволяє зберігати повний контроль над дроном у складних умовах, що особливо важливо при FPV-польотах на великі відстані або в умовах обмеженої видимості.

Коли між наземною станцією керування та дроном відсутній прямий радіозв’язок, встановити надійне з’єднання стає складно, іноді зовсім неможливо. Це особливо актуально у випадках, коли сигнал блокується великими фізичними перешкодами, такими як будівлі, пагорби, ліси чи інші особливості рельєфу, які здатні поглинати радіохвилі.

У таких умовах канал зв’язку, незалежно від того, йдеться про телеметрію, відеосигнал чи сигнали керування, може істотно погіршитися, а згодом взагалі зникнути. Це створює загрозу втрати контролю над дроном і унеможливлює виконання завдань.

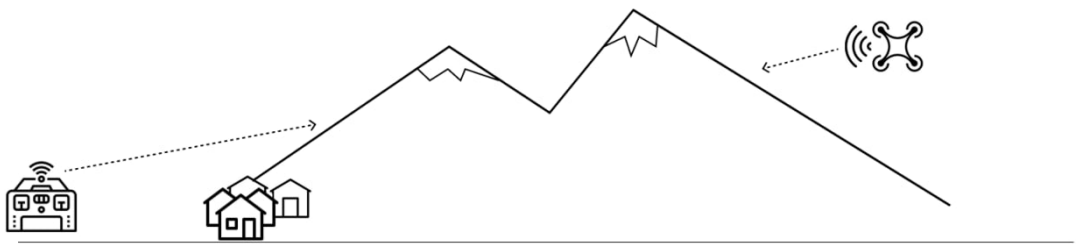


Рисунок 1.1 – Радіозв'язок не може бути встановлений, якщо лінія видимості закрита масивними об'єктами

Щоб уникнути цієї проблеми, доцільно використовувати ретранслятор – проміжний вузол зв’язку, який може бути у вигляді стаціонарної наземної станції, мобільного транспортного засобу або іншого дрона, що зависає на відповідній висоті. Такий ретранслятор передає сигнали між пілотом і дроном, створюючи багатосегментний зв’язок, який дозволяє обійти перешкоди та відновити стабільну лінію радіозв’язку.

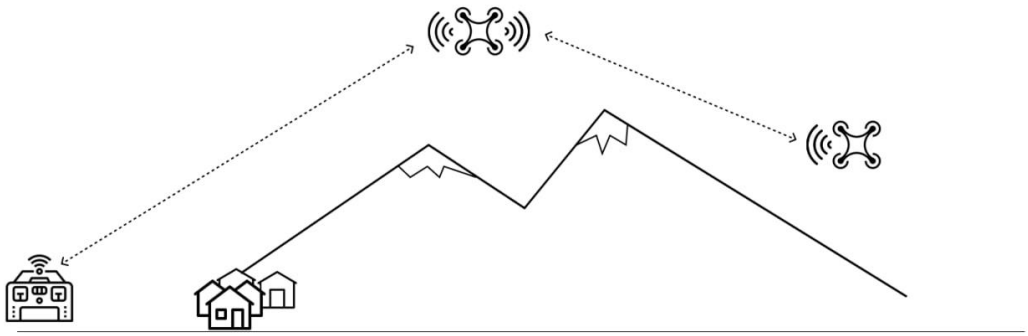


Рисунок 1.2 – Додавання ретранслятора дозволяє встановити радіозв'язок, якщо лінії видимості перешкоджають великі об'єкти

Або ж коли дрон знаходиться занадто далеко, тоді встановити радіозв’язок безпосередньо неможливо. Надмірна відстань між оператором і дроном призводить до втрати сигналу, що унеможливлює стабільне керування, отримання телеметрії й відео в реальному часі.

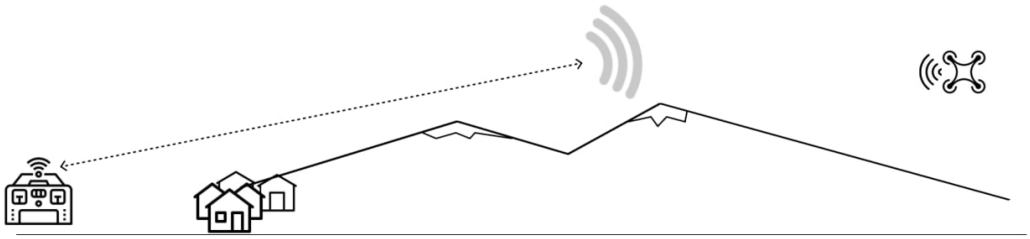


Рисунок 1.3 – Радіозв'язок неможливо встановити, якщо відстань занадто велика

Знову ж таки проблему можна вирішити шляхом використання одного або кількох ретрансляторів. Вони виконують роль проміжних вузлів, що приймають сигнал від наземної станції та передають його далі до дрона (і навпаки). Така схема дозволяє подолати обмеження дальності польоту й відновити зв’язок навіть на великій відстані, коли пряме радіоз’єднання неможливе.

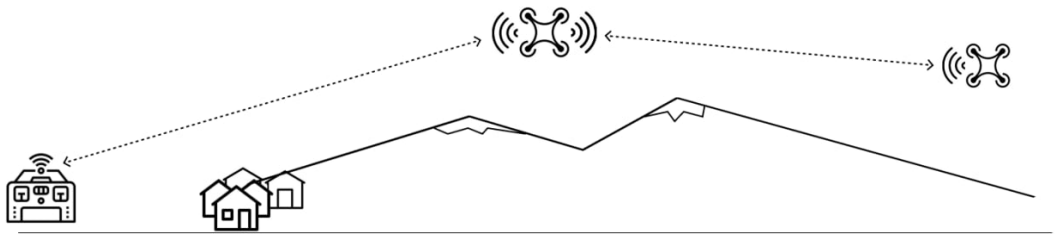


Рисунок 1.4 – Додавання вузла ретрансляції дозволяє встановити радіозв’язок, коли відстань надто велика

Отже, ретрансляція може застосовуватись на різних рівнях комунікації між безпілотним літальним апаратом та оператором, зокрема:

– передача відеосигналу: забезпечення трансляції зображення від FPV-камери дрона до оператора в умовах, коли відсутня пряма видимість або сигнал недостатньо потужний для стабільної передачі даних;

– телеметрія: підтримка постійного обміну критичними даними (GPS-координати, рівень заряду акумулятора, орієнтація, швидкість тощо) між БПЛА та наземною станцією керування;

– сигнали керування: забезпечення безперервного контролю над БПЛА, навіть якщо оператор не може безпосередньо зв’язатися з ним через радіозв’язок.

Оскільки сам ретранслятор може бути як стаціонарним (фіксована точка ретрансляції, розміщена на місцевості, вежі, будівлі чи транспортному засобі), так і мобільним (платформа, здатна змінювати своє положення у просторі). У межах даного проекту предметом є саме FPV-дрон, який переносить відповідне обладнання (ретранслятор) для забезпечення зв’язку.

Як вже зазначалось раніше, однією з основних труднощів при використанні FPV-дронів є відсутність функції автоматичного зависання, що є стандартною для більшості споживчих дронів, зокрема моделей від DJI. У моделях типу DJI, дрон автоматично стабілізується у повітрі й утримує висоту без втручання оператора. Натомість FPV-дрони, особливо в режимі "acro", не мають подібних стабілізаційних систем – пілот самостійно контролює кожен аспект польоту: нахил, висоту, обертання та швидкість. Втрата концентрації або незначна помилка можуть миттєво призвести до втрати контролю та аварії.

Проте зібрати 8-дюймовий FPV-дрон коштує в середньому 10 000 грн, в той час, як середня вартість DJI Matrice 4T становить 276 720 грн [4], що фактично в 20 разів [5] перевищує ціну для FPV-дрона. Для розширення доступності використання дронів-ретрансляторів і покращення ефективності виконання завдань, постала необхідність інтеграції функції зависання для FPV-дронів. Це дозволить підвищити кількість успішно виконаних операцій в умовах складного ландшафту, постійної дії РЕБ або при роботі на дальні відстані.

Отже, розглянемо основні методи для стабілізації таких дронів [6].

Таблиця 1.1 – Методи стабілізації FPV-дрону при зависанні на місці

Метод	**Опис**	**Примітки**
GPS	Використовує супутникові сигнали для визначення позиції дрону.	Потрібна висока точність для позиціонування.
Обробка зображення	Камера аналізує навколишнє середовище для коригування позиції.	Включає методи оптичного потоку, SLAM.
Інерціальні сенсори	Гіроскопи та акселерометри відстежують орієнтацію та зміщення дрону.	Забезпечують точну корекцію нахилу і руху.

Кінець таблиці 1.1

Барометри	Вимірюють атмосферний тиск для визначення висоти дрону.	Корисно для стабілізації на заданій висоті.
Ультразвукові сенсори	Вимірюють відстань до поверхні для стабільності на низьких висотах.	Застосовуються в закритих приміщеннях.
Лідар	Використовує лазерні промені для точного картографування навколишнього середовища.	Підвищує точність локалізації та стабільності в складних умовах.
Мультисенсорні системи	Комбінують дані з різних сенсорів для підвищення точності стабілізації.	Підходить для складних умов, де один сенсор недостатній.

Проаналізувавши різні методи стабілізації FPV-дрону, можна зробити висновок, що кожен із них має свої переваги та обмеження в залежності від умов польоту. Проте для забезпечення максимальної точності і стабільності при зависанні в складних умовах, ми обрали метод обробки зображення, зокрема при використанні термальної камери. Така технологія дозволяє не тільки забезпечити стабільне позиціювання, але й надає додаткові можливості для навігації в умовах обмеженої видимості або при необхідності виявлення теплових об'єктів, що робить її достатньо ефективною для ряду специфічних завдань.

1.2 Аналіз апаратної частини реалізації

Для реалізації стабілізації FPV-дрона на основі аналізу зображення з бортової тепловізійної камери необхідно інтегрувати низку критично важливих апаратних

компонентів. Вони забезпечують ефективне функціонування системи в режимі реального часу, дозволяючи дрону коригувати своє положення в просторі відповідно до візуальних і теплових даних.

До таких компонентів належать:

– FPV-дрон (платформа);
– тепловізійна камера;
– бортовий комп’ютер;
– датчики;
– польтний контролер.

Розглянемо кожну з частин детальніше, нижче наведено аналіз основних компонентів та їх функцій.

Основним апаратним компонентом є сам дрон, який повинен володіти достатньою потужністю, стабільністю та здатністю розміщувати додаткове обладнання, зокрема ретрансятор. Тому при виборі FPV-дрона необхідно враховувати такі технічні характеристики:

– двигуни та електронні регулятори швидкості (ESC), які мають бути достатньо потужними, щоб забезпечити стабільність дрона навіть із додатковим навантаженням;

– рама та конструкція мають бути достатньо міцними та легкими, щоб підтримувати додаткові компоненти без істотного впливу на маневреність чи тривалість польоту.

Тепловізійна камера є ключовим елементом системи стабілізації, оскільки використовується для отримання зображень навколишнього середовища, дозволяє аналізувати зміни температури та виявляти термічні об’єкти. Висока роздільна здатність важлива для точного аналізу та стабільного позиціонування, особливо в складних умовах. Камеру необхідно буде підключити до польотного контролера та бортового комп’ютера, який буде обробляти зображення в режимі реального часу.

У зв’язку з цим необхідний потужний бортовий обчислювальний модуль для обробки зображень з тепловізійної камери за допомогою комп’ютерного зору та, на

основі отриманих даних, передачі нових команд для стабілізації дрону. Високопродуктивний одноплатний комп'ютер, наприклад Raspberry Pi, може бути використаний для підключення до камери, прийому відеопотоку та подальшої його обробки. Цей модуль також взаємодіє з іншими датчиками дрона, такими як інерційні датчики та GPS. Алгоритми комп'ютерного зору, зокрема оптичний потік та SLAM (Simultaneous Localization and Mapping – одночасна локалізація та побудова карти), можуть бути реалізовані безпосередньо на обчислювальному модулі для забезпечення корекції навігаційних даних і підвищення точності позиціонування дрона в умовах обмеженої доступності GPS.

Інерційний вимірювальний пристрій, який включає гіроскопи та акселерометри, відстежує орієнтацію та рух дрона. Точне відстеження відхилень дрона від бажаного положення має важливе значення для подальшої його стабілізації. Гіроскопи вимірюють обертання дрона навколо осі, а акселерометри вимірюють його прискорення, допомагаючи відстежувати зміщення. Барометр, який вимірює зміни атмосферного тиску, використовується для моніторингу та контролю висоти дрона з високою точністю, забезпечуючи стабільність на потрібній висоті. Ці датчики працюють разом, щоб підтримувати точне позиціонування та забезпечувати стабільність дрона.

Польотний контролер – це «мозок» дрона, який отримує дані від усіх датчиків і керує командами двигуна для стабілізації. Він регулює рухи дрона для підтримки стабільного положення на основі опрацьованих даних тепловізійної камери та інерційних датчиків [7].

Отже, основні апаратні компоненти включають: тепловізійну камеру для збору даних, бортовий комп'ютер для обробки зображень, реалізації алгоритмів комп'ютерного зору та взаємодії з іншими сенсорами, інерційні датчики для визначення орієнтації та кутових швидкостей, барометр для вимірювання висоти, платформу дрона для забезпечення фізичної підтримки всіх компонентів та необхідної тяги, польотний контролер для виконання команд і взаємодії з бортовим комп'ютером.

Усі ці компоненти повинні бути інтегровані в єдину систему з узгодженим обміном даними для забезпечення максимальної точності, стабільності та надійності функціонування в умовах реального середовища.

Нижче подано блок-схему, яка показує логіку функціонування алгоритму стабілізації FPV-дрона в процесі зависанн. У цьому методі застосовується тепловізійна камера для здобуття візуальної інформації про навколишнє середовище, а також інерційні датчики та інші сенсори, як-от барометр, для збирання даних про позицію та рух дрона. Обробка цих даних виконується на бортовому комп'ютері в реальному часі, після чого обчислені координати передаються польотному контролеру для коригування руху й підтримки стабільного зависання.

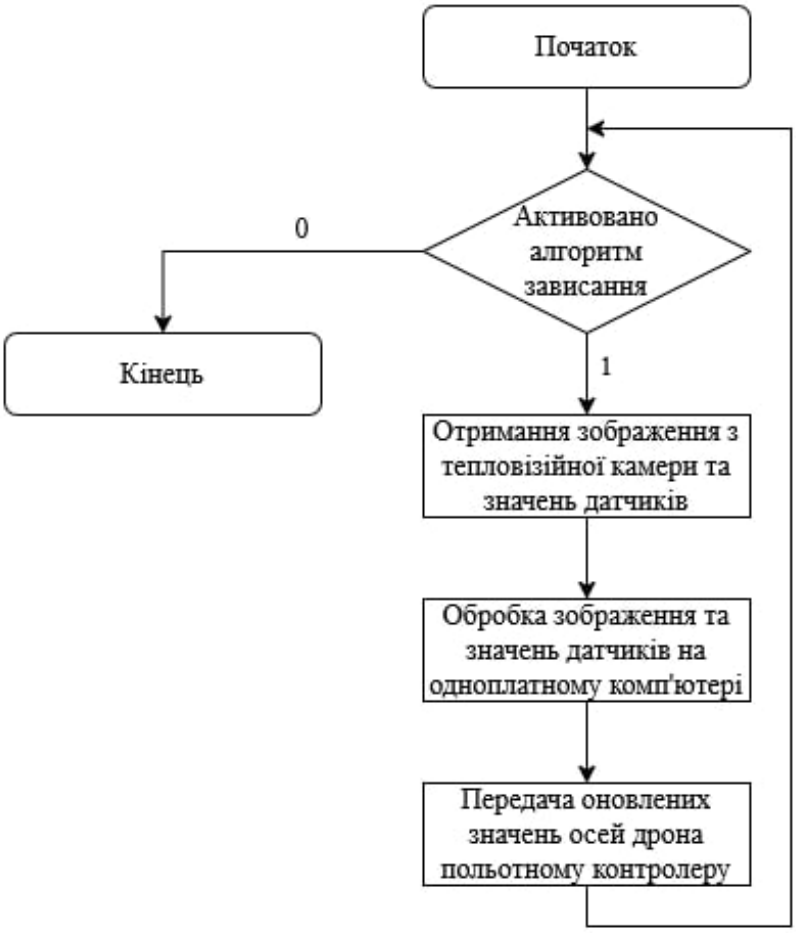


Рисунок 1.5 – Алгоритм роботи апаратної частини

1.3 Програмна частина для реалізації

Для реалізації автономної стабілізації та супроводження цілі під час зависання, FPV-дрон покладається на два основні програмні модулі: протокол зв’язку MAVLink і модуль обробки зображень OpenCV.

Протокол MAVLink забезпечує зв’язок у режимі реального часу між бортовим комп’ютером (наприклад, Raspberry Pi) і польотним контролером дрона, дозволяючи передавати команди управління та отримувати телеметричні дані. Тим часом модуль OpenCV обробляє зображення отримані з тепловізійної камери для виявлення та відстеження об’єктів на яких можна зафіксуватись. Працюючи в тандемі, ці компоненти утворюють замкнуту систему: візуальні дані аналізуються для визначення відносного положення дрона щодо цілі, а відповідні команди управління надсилаються через MAVLink для підтримки стабільності положення, зависання над ціллю.

1.3.1 Протокол зв’язку MAVLink

MAVLink (Micro Air Vehicle Link) – це компактний протокол обміну даними, призначений для забезпечення зв’язку між дроном і зовнішніми пристроями, такими як автопілот, наземна станція або бортовий комп’ютер. Він дозволяє передавати як телеметричну інформацію (наприклад, положення, висоту, стан батареї), так і команди керування (зліт, посадка, навігація) у реальному часі. Завдяки невеликому розміру повідомлень (із накладними витратами 8–14 байтів для службової інформації), MAVLink добре підходить для використання в умовах обмеженої пропускної здатності, зокрема при бездротовому зв’язку. Протокол підтримується такими системами автопілоту, як ArduPilot і PX4, і є стандартом у багатьох сучасних безпілотних платформах [8].

У даному проєкті MAVLink використовується як рівень зв’язку між польотним контролером FPV-дрона та бортовим комп’ютером. Він передає телеметричні дані (наприклад, положення, заряд акумулятора, показання датчиків)

від дрона до вбудованого комп’ютера, а також команди керування (наприклад, зміни режиму польоту, значення тангажу, крену, рискання та власне тяги) від комп’ютера до дрона. Цей двонаправлений зв’язок фактично робить MAVLink «мовою» дрона, забезпечуючи взаємодію між бортовим комп’ютером і корисним навантаженням, тобто обладнанням, яке несе дрон на борту (наприклад, підвісна тепловізійна камера).

MAVLink підтримує як шаблон проєктування публікація-підписка [9], так і взаємодію за принципом «точка-точка» [10], що дозволяє ефективно організовувати передачу даних: періодичні повідомлення (наприклад, про положення чи стан системи) транслюються для всіх «підписаних» компонентів, в той час як команди та параметри можуть передаватися безпосередньо між окремими модулями з підтвердженням отримання. Дана гнучкість забезпечує надійний обмін інформацією між усіма частинами системи.

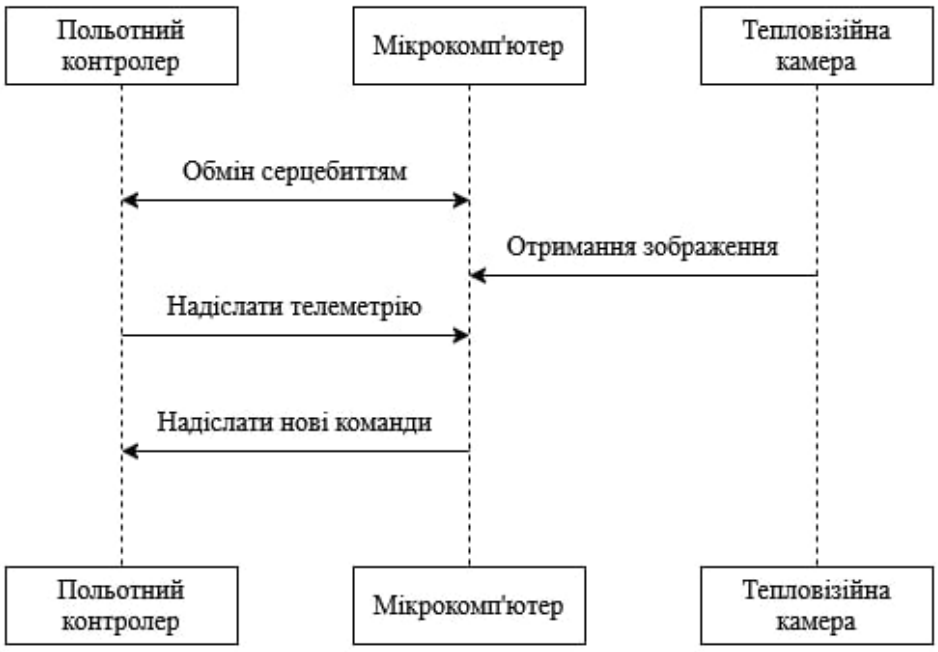


Рисунок 1.6 – Діаграма взаємодії між мікрокомп'ютером, польотним контролером і тепловізійною камерою через протокол MAVLink

Польотний контролер та мікрокомп’ютер обмінюються "серцебиттями" для моніторингу підключення, використовуючи стандартні ідентифікатори повідомлень (визначені в XML) для всіх взаємодій. Основні функції включають: відправку повідомлень про серцебиття/статус, трансляцію даних з датчиків з фіксованими інтервалами, передачу та обробку командних повідомлень (наприклад, PARAM_SET). Оскільки MAVLink розроблений для радіозв’язку з низькою пропускною здатністю, він ефективний і надійний навіть у шумних з'єднаннях.

1.3.2 Бібліотека функцій комп’ютерного зору OpenCV

OpenCV (Open Source Computer Vision Library) – це бібліотека з відкритим кодом, призначена для комп'ютерного зору та обробки зображень. Вона надає широкий набір алгоритмів та інструментів для аналізу зображень і відео, що дозволяє вирішувати різноманітні задачі, зокрема виявлення об'єктів, розпізнавання осіб, аналіз руху, обробка теплових зображень та багато інших [11].

Серед основних можливостей OpenCV:

- фільтрація зображень;
- корекція кольору, зміна контрасту;
- масштабування, обертання;
- відстеження об'єктів і визначення напрямку їхнього руху.

Бібліотека підтримує інтеграцію з різними мовами програмування, такими як C++, Python, Java, і сумісна з численними платформами, що робить її популярним інструментом у галузі комп'ютерного зору та штучного інтелекту. OpenCV також дозволяє здійснювати машинне навчання та створювати інтелектуальні системи на основі отриманих даних.

У контексті даного проєкту бібліотека OpenCV використовується для обробки тепловізійного зображення, отриманого з камери. Зокрема, вона здійснює аналіз зображень, виявляючи та локалізуючи "теплі" об'єкти, що можуть бути

потенційно важливими для подальших операцій. Цей процес включає в себе використання алгоритмів комп'ютерного зору для визначення ділянок з підвищеною температурою, що дозволяє точно ідентифікувати об'єкти, які потребують додаткової уваги або аналізу.

На початковому етапі кожен отриманий тепловізійний кадр піддається обрізанню та приводиться до формату зображення у градаціях сірого або нормалізованої інтенсивності (якщо він ще не відповідає цим вимогам). Після цього виконується згладжування зображення з метою придушення шумів. Зокрема, для зменшення впливу випадкових теплових флуктуацій перед подальшим аналізом застосовується фільтр розмиття Гаусса. Така попередня обробка забезпечує стійкість подальших алгоритмів до незначних коливань значень пікселів і запобігає виникненню хибних контурів.

Потім модуль здійснює вилучення ознак, необхідних для ідентифікації об'єктів на зображенні. Найбільш поширеним підходом на цьому етапі є застосування алгоритму виявлення контурів Кенні, який обчислює градієнти інтенсивності та використовує подвійне порогове значення для побудови бінарної карти контурів.

На практиці попередньо може застосовуватися глобальна або адаптивна порогова обробка, оскільки яскраві області тепловізійного зображення зазвичай відповідають зонам підвищеної температури, тобто потенційним цілям. Після цього на отриманому результаті виконується алгоритм виявлення контурів Кенні. На основі сформованого бінарного зображення здійснюється пошук контурів за допомогою відповідної функції модуля. Контури представляють собою замкнені криві, які окреслюють межі об'єктів, і дозволяють ефективно виокремлювати зв’язані області з підвищеною інтенсивністю як потенційні теплові цілі.

Серед усіх виявлених контурів здійснюється вибір найбільш ймовірної цілі на основі певних критеріїв. Зокрема, це може бути контур із найбільшою площею або контур, що має найвищу середню інтенсивність, що вказує на найгарячішу область зображення. Для вибраного контуру обчислюються його геометричні та

статистичні характеристики, зокрема площа та центроїд – координати центра маси контуру, що використовується як оцінка положення цілі в межах кадру. Це дозволяє визначити цільову координату в піксельному просторі зображення.

Після того, як ціль була ідентифікована та локалізована на тепловізійному зображенні, визначається її зсув відносно центру кадру. Виходячи з цього зсуву, обчислюються необхідні коригування положення дрона. Відповідні повідомлення з оновленими значеннями по координатних осях передаються до польотного контролера через протокол MAVLink. Таким чином, ціль не задається як абсолютна точка у просторі, а формується команда щодо відносного зміщення, що дозволяє FPV-дрону стабілізувати положення, орієнтуватися так, щоб об'єкт залишався в полі зору. Такий підхід реалізує замкнуту систему керування, в якому модуль комп’ютерного зору виявляє відхилення, а MAVLink забезпечує відповідну реакцію автопілота.

1.4 Вибір компонентів та середовища для реалізації

На початковому етапі розробки проєкту важливо визначити оптимальні апаратні компоненти та програмне середовище, які забезпечать стабільну, надійну і енергоефективну роботу дрона.

Одним із ключових аспектів є тип відеосигналу, що передається з тепловізійної камери на бортовий комп’ютер для подальшої обробки. У більшості FPV-дронах використовують аналогову технологію передачі відео, оскільки вона має низку беззаперечних переваг перед цифровими системами. По-перше, аналоговий сигнал споживає значно менше електроенергії, що дозволяє збільшити тривалість польоту дрона. По-друге, аналогові системи відзначаються високою стійкістю до радіоперешкод, що критично важливо в умовах обмеженого або нестабільного зв'язку. Цифрові ж камери можуть втрачати якість зображення, проявляти пікселізацію чи навіть повністю втрачати сигнал. У цьому проєкті виберемо тепловізійну камеру, що має компактні розміри, низьку вагу та сумісність

із більшістю FPV-систем, яка дозволяє передавати зображення в реальному часі та легко інтегрується у корпус дрона без шкоди для його аеродинаміки.

Одним із ключових елементів апаратної архітектури автономного FPV-дрона є обчислювальний блок, який забезпечує обробку вхідних даних із сенсорів, тепловізійної камери, а також реалізацію алгоритмів автономного керування в режимі реального часу. З огляду на специфіку завдань та вимоги до розміру, ваги і енергоспоживання, доцільно використати одноплатний мікрокомп’ютер, зокрема представника лінійки Raspberry Pi.

Такі пристрої ідеально підходять для використання у вбудованих системах через їхню універсальність, компактні габарити, енергоефективність і широкі функціональні можливості. Одноплатний мікрокомп’ютер об'єднує на одній платі центральний процесор, оперативну пам’ять, графічний прискорювач, а також широкий набір інтерфейсів для підключення периферійних пристроїв: USB, GPIO, I2C, UART, SPI та інші. Інтеграція Raspberry Pi з тепловізійною камерою та польотним контролером дозволяє реалізувати гнучку та масштабовану архітектуру, в якій можливо ефективно розподіляти обчислювальні задачі між компонентами. Завдяки високій сумісності з операційною системою Linux і підтримці великої кількості бібліотек Python, мікрокомп’ютер виконує роль центрального вузла в системі автономного керування дроном.

Не менш важливим компонентом системи керування дроном є польотний контролер, який відіграє центральну роль у забезпеченні стабільності, точності та безпеки польоту. Саме цей пристрій приймає сигнали від сенсорів, обробляє їх у режимі реального часу та формує команди на регулятори швидкості, які безпосередньо керують роботою електродвигунів. Сучасні польотні контролери базуються на високопродуктивних мікроконтролерах, що мають значну обчислювальну потужність, енергоефективність та розширені можливості підключення. Це дозволяє їм не лише обробляти сигнали з акселерометрів, гіроскопів та барометрів, а й взаємодіяти з зовнішніми системами, такими як бортовий комп’ютер або наземна станція управління. При виборі польотного

контролера варто ретельно враховувати сумісність із основними компонентами дрона – двигунами, регуляторами швидкості, сенсорами навігації та позиціонування. Крім того, ключову роль відіграє підтримка стандартних протоколів зв'язку, таких як MAVLink, який забезпечує ефективний двосторонній обмін даними між контролером, телеметрією, обчислювальним модулем і зовнішніми інтерфейсами. Додатковими критеріями вибору можуть бути: можливість оновлення прошивки, підтримка автономних режимів польоту, наявність розширених засобів логування та сумісність із програмними симуляторами, зокрема ArduPilot SITL, що дозволяє попередньо тестувати алгоритми управління без ризику для обладнання.

Програмна частина є не менш важливою складовою проєкту, оскільки саме вона забезпечує реалізацію алгоритмів автономного управління, обробку даних із сенсорів і камери, а також взаємодію між усіма компонентами системи. У рамках цього проєкту основою для реалізації програмної логіки обрано операційну систему Linux, яка надає широкий спектр інструментів для налаштування, розгортання та обслуговування фонових процесів, що критично важливо для забезпечення безперебійної роботи дрона в автономному режимі. Завдяки великій кількості підтримуваних драйверів, Linux дозволяє легко інтегрувати нові апаратні компоненти, включно з різними видами сенсорів, камер і модулів зв'язку.

Основною мовою програмування є Python – високорівнева, інтерпретована мова з відкритим кодом, яка характеризується простотою синтаксису, гнучкістю. Python активно використовується в галузях робототехніки, штучного інтелекту та вбудованих систем [12]. У цьому проєкті Python буде застосовується для:

- реалізації комп'ютерного зору (бібліотека OpenCV);
- обробки телеметричних даних (MAVProxy, DroneKit);
- керування мережею та обміном інформацією з іншими модулями;
- взаємодії із сенсорами та протоколами на низькому рівні.

Щоб забезпечити попереднє тестування розроблених алгоритмів без необхідності запуску апаратної частини, використовується ArduPilot SITL

(Software-In-The-Loop) – симуляційне середовище, що дозволяє створювати віртуальні сценарії польоту [13]. Це дає змогу моделювати поведінку дрона в різних умовах, тестувати коректність керування та виявляти помилки на етапі розробки, мінімізуючи ризики у польових випробуваннях.

Загалом, поєднання таких інструментів, як Linux, Python, OpenCV, MAVLink та ArduPilot SITL, у комплексі з апаратними рішеннями (тепловізійна камера Foxeer, мікрокомп'ютер Raspberry Pi, польотний контролер) формує надійну архітектуру для створення автономного FPV-дрона, здатного до самостійної навігації, обробки візуальних даних та адаптації до змін навколишнього середовища.

1.5 Висновки до першого розділу

У першому розділі було розглянуто теоретичні засади, необхідні для реалізації даного проєкту, спрямованого на створення системи стабілізації FPV-дрона з використанням тепловізійної камери та обчислювального модуля. Здійснено комплексний аналіз сучасних методів вирішення проблеми стабілізації безпілотних літальних апаратів, що дозволило виявити переваги та обмеження різних підходів, а також обґрунтувати вибір технологій і засобів, які доцільно застосувати у рамках запропонованого технічного рішення.

На основі проведеного аналізу було сформовано структурну модель апаратної частини проєкту. До її складу включено бортовий комп'ютер, що забезпечує обробку зображень у режимі реального часу та реалізацію алгоритмів комп'ютерного зору; польотний контролер, відповідальний за низькорівневе керування, стабілізацію та обробку сигналів з інерційних датчиків; а також тепловізійну камеру, яка виконує функцію основного сенсора для аналізу навколишнього середовища.

Крім апаратної складової, було визначено програмну частину реалізації, необхідну для реалізації обробки даних. Зокрема, проаналізовано функціональні

можливості бібліотеки OpenCV, яка дозволяє здійснювати попередню обробку зображень, фільтрацію, виявлення об'єктів, а також реалізацію алгоритмів оптичного потоку для визначення положення дрона у просторі.

Отже, результати, отримані в межах першого розділу, створюють концептуальну та технічну основу для практичного етапу реалізації системи. Визначені апаратні та програмні компоненти, а також проаналізовані методи стабілізації, слугуватимуть базою для подальшої розробки, моделювання та тестування інтегрованої системи керування дроном.

ОБҐРУНТУВАННЯ ВИБОРУ КОМПОНЕНТІВ ТА СЕРЕДОВИЩА РЕАЛІЗАЦІЇ

2.1 Апаратне середовище реалізації

Для реалізації автономної стабілізації FPV-дрона з інтегрованою тепловізійною камерою було обрано використання бортового мікрокомп’ютера, який виконує функції обчислювального модуля. Основним його завданням є реалізація алгоритмів зависання та стабілізації дрона у просторі шляхом обробки зображень, що надходять з тепловізійної камери. Завдяки аналізу відеопотоку бортовий мікрокомп’ютер зможе визначати зміщення дрона у просторі та генерувати відповідні коригувальні дії.

Передача команд управління до польотного контролера здійснюватиметься за допомогою протоколу MAVLink – універсального стандарту для обміну даними між автопілотом та іншими системами. Такий підхід забезпечує високий рівень сумісності та дозволяє ефективно реалізовувати інтеграцію між програмною та апаратною частинами проєкту.

2.1.1 Вибір тепловізійної камери

Вибір камери є ключовим етапом у побудові системи автономної стабілізації дрона, оскільки від характеристик сенсора значною мірою залежить якість вхідних даних для подальшої обробки. На цей вибір впливають численні фактори, зокрема роздільна здатність, частота кадрів, тип зображення (кольорове чи температурне), світлочутливість, формат вихідного сигналу, а також енергоспоживання.

У межах даного проєкту пріоритет надається тепловізійним (інфрачервоним) камерам, які забезпечують зображення на основі розподілу температур, а не кольорових характеристик. Такий підхід обґрунтований прагненням зменшити обсяг обчислень, оскільки температурні дані менш варіативні, ніж колірні, і не залежать від умов освітлення. Крім того, тепловізійні камери є надзвичайно

ефективними в умовах недостатньої видимості, зокрема вночі або в умовах диму, туману, що важливо для стабільної навігації в різноманітних сценаріях застосування.

Наступним аспектом, який необхідно врахувати, є тип відеосигналу, що формується камерою. Сучасні тепловізори можуть генерувати як аналоговий сигнал, що часто застосовується завдяки низькій затримці, так і цифровий сигнал, який надає вищу якість зображення та зручність інтеграції з обчислювальними модулями. Отже, порівняємо аналоговий та цифровий сигнал, аби визначити, який сигнал буде кращим для використання у даному проекті.

Таблиця 2.1 – Цифрові та аналогові сигнали [14]

Параметр	**Аналоговий сигнал**	**Цифровий сигнал**
Природа сигналу	Безперервний, значення змінюються по безперервній шкалі.	Дискретний, значення представлені в вигляді 0 і 1.
Якість сигналу	Підлягає спотворенню та шумам, що можуть знижувати якість.	Менше піддається спотворенням, може бути відновлений при помилках.
Носій інформації	Амплітуда та/або частота сигналу.	Бітові послідовності.
Чутливість до шуму	Висока чутливість, може бути спотворено при передачі.	Менше чутливі до шуму, можлива корекція помилок.
Складність обробки	Простіша обробка, але за рахунок шумів якість може погіршуватися.	Складніша обробка, але точність зберігається навіть при помилках.
Пропускна здатність	Менша, оскільки потрібно більше місця для передачі сигналу.	Вища, ефективно використовує пропускну здатність.

Кінець таблиці 2.1

Енергоспоживання	Може споживати менше енергії для передачі простих сигналів.	Часто вищий рівень через складніші процеси обробки та передачі.
Тривалість передачі	Може бути затримка через обробку сигналу або передачу через великі відстані.	Може мати більші затримки через потребу в кодуванні та декодуванні.

Отже, використання аналогових технологій у тепловізійних камерах на FPV-дронах є цілком виправданим, оскільки вони забезпечують низку практичних переваг, що сприяють ефективності їх застосування, а саме:

– менша затримка: аналогові відеосигнали мають мінімальний час обробки, що означає меншу затримку (часову затримку між захопленням і відображенням зображення), вони забезпечують майже миттєву передачу відео, що необхідно для управління дроном в реальному часі;

– зменшена пропускна здатність: зазвичай вимагають меншої пропускної здатності, ніж цифрові сигнали, що дозволяє передавати відео на великі відстані без перевантаження бездротового каналу зв'язку;

– простота та надійність: часто є простішими та надійнішими в нетривіальних умовах, як це буває на FPV-дронах, натомість цифрові сигнали можуть бути більш вразливими до завад, а обробка цифрових сигналів в реальному часі може вимагати більше обчислювальних ресурсів, що може вплинути на продуктивність або час роботи батареї;

– ціна та доступність: зазвичай дешевші за цифрові, що робить їх більш доступними, адже компоненти, що необхідні для аналогових систем, широко представлені на ринку, що сприяє збереженню їхньої вартості на відносно низькому рівні.

З огляду на технічні вимоги та специфіку умов експлуатації, у межах реалізації цього проєкту було прийнято рішення використати тепловізійну камеру Foxeer Cat 3 Mini. Даний вибір зумовлений низкою переваг, які ця модель забезпечує в контексті FPV-систем. Зокрема, основною технічною перевагою є надзвичайно низька затримка передачі зображення, що критично важливо для здійснення безперервного контролю над дроном у реальному часі. Це особливо актуально в умовах високих швидкостей польоту, різких маневрів або в ситуаціях, де оператор змушений реагувати на непередбачувані зміни навколишнього середовища.

Рисунок 2.1 – Тепловізійна камера для FPV Foxeer Mini Cat 3 [15]

Камера Foxeer Cat 3 Mini має роздільну здатність 160×120 пікселів, чого цілком достатньо для розпізнавання теплових об’єктів на середніх дистанціях, а її здатність передавати відео з мінімальними вимогами до пропускної здатності каналу є ключовою для підтримки стабільного відеозв’язку.

Додатковою перевагою є використання аналогового сигналу, що дозволяє суттєво знизити енергоспоживання системи. Це, у свою чергу, позитивно впливає на загальний час автономного польоту безпілотного літального апарата. На відміну від цифрових систем, які схильні до таких проблем, як пікселізація, артефакти або повна втрата зображення при ослабленні сигналу, аналогові рішення, зокрема Foxeer Cat 3 Mini, демонструють високу надійність та стійкість до зовнішніх перешкод.

Окремо варто відзначити компактні розміри і малу вагу камери, що дозволяє інтегрувати її в конструкцію FPV-дрона без суттєвого впливу на його аеродинамічні характеристики чи баланс. Таким чином, обраний пристрій не лише задовольняє технічні критерії проєкту, але й сприяє досягненню високої ефективності й надійності в умовах практичного застосування.

2.1.2 Вибір бортового комп’ютера

При виборі бортового комп’ютера для даного проекту, одноплатні комп’ютери є оптимальним вибором завдяки своїм компактним розмірам, гнучкості та функціональності. Мікрокомп’ютери мають інтегровані компоненти, такі як процесори, пам'ять і порти вводу/виводу, що робить їх ідеальними для виконання різноманітних завдань в межах вбудованих систем.

Одноплатні комп’ютери стали популярними в різних галузях завдяки кільком перевагам. По-перше, вони мають компактні розміри, що є особливо важливим для проектів з обмеженим простором, таких як борт FPV-дрона. По-друге, одноплатні комп’ютери спроектовані з урахуванням енергоефективності, що є критично важливим для застосунків, що працюють від батарей. Крім того, вони є економічними. З точки зору гнучкості, більшість таких комп’ютерів підтримують різноманітні периферійні пристрої та мають піни GPIO (General Purpose Input/Output), що дозволяє взаємодіяти з сенсорами, камерами та іншими компонентами [16].

Одноплатні комп’ютери (Single-Board Computers, SBC) представлені на ринку різними платформами, серед яких найпоширенішими є Arduino, Raspberry Pi та Orange Pi. Кожна з цих платформ має свої унікальні особливості, архітектурні рішення та сфери застосування.

Arduino – це мікроконтролерна платформа, яка орієнтована насамперед на роботу з простими цифровими та аналоговими сигналами. Вона не має операційної системи в традиційному розумінні, натомість виконує прошивку, яка прописується

безпосередньо в пам’ять контролера. Arduino характеризується надзвичайною енергоефективністю, миттєвим запуском і високою надійністю, але водночас має обмежені обчислювальні ресурси. Вона ідеально підходить для задач автоматизації, керування пристроями, зчитування даних із сенсорів, але не може ефективно працювати з потоковим відео або виконувати паралельні обчислення, що необхідні для FPV-дронів [17].

Orange Pi – це платформа, яка за своїми функціональними можливостями близька до Raspberry Pi і часто розглядається як її альтернатива. Orange Pi має конкурентоспроможні характеристики за нижчою ціною, однак ця платформа іноді страждає від обмеженої підтримки драйверів, нестабільної роботи деяких компонентів, а також менш активної спільноти, що ускладнює розробку та налагодження проєкту [18].

Raspberry Pi, натомість, є однією з найпопулярніших платформ у світі SBC. Вона представлена у широкому асортименті моделей від бюджетного та енергоефективного Raspberry Pi Zero до потужних Raspberry Pi 4 і Raspberry Pi 5, що дозволяє обрати оптимальний варіант для різних проєктів. Raspberry Pi вирізняється низькою вартістю, високою продуктивністю багатоядерних процесорів та підтримкою оперативної пам’яті до 8 ГБ, що забезпечує ефективне виконання завдань від базової автоматизації до складних алгоритмів штучного інтелекту. Мала споживана потужність робить цю платформу ідеальною для застосувань з обмеженим енергоспоживанням, наприклад, у проєктах з живленням від акумуляторів або альтернативних джерел енергії. Важливою перевагою є підтримка повноцінних операційних систем, таких як Raspberry Pi OS (на базі Debian), Ubuntu та інших дистрибутивів. Крім того, велика кількість документації, бібліотек і активна спільнота користувачів значно спрощують розробку та впровадження інноваційних рішень. Завдяки поєднанню доступності, продуктивності, енергоефективності та гнучкості у виборі програмного забезпечення, Raspberry Pi є оптимальним вибором для широкого спектра технічних завдань і проєктів [19].

Оскільки одним із найбільш поширених та затребуваних мінікомп’ютерів на сучасному ринку є Raspberry Pi, то для реалізації даного проєкту нами було обрано модель даного виробника (Raspberry Pi Zero 2 W), яка оптимально відповідає вимогам технічного завдання завдяки нижче наведеним характеристикам.

Рисунок 2.2 – Мікрокомп’ютер Raspberry Pi Zero 2 W [20]

Raspberry Pi Zero 2 W не оснащено апаратним аналоговим відеовходом, оскільки дана модель, як і більшість сучасних одноплатних комп’ютерів, розрахована на роботу з цифровими інтерфейсами. Через відсутність спеціалізованих аналогових пінів для відеосигналу підключення пристроїв з аналоговим виходом, зокрема тепловізійної камери, потребує додаткового вузла – зовнішнього аналого-цифрового перетворювача. Це зумовлено тим, що стандартні GPIO-піни Raspberry Pi не підтримують прийом аналогового сигналу в жодному форматі, тому безпосередній ввід аналогових відеоданих неможливий.

Застосування зовнішнього аналого-цифрового перетворювача (АЦП) є технічно доцільним рішенням, що забезпечує конверсію аналогового сигналу у цифрову форму, придатну для обробки модулем Raspberry Pi. Сучасні АЦП характеризуються низьким енергоспоживанням, компактними розмірами та мінімальною інерційністю. Затримка, що виникає під час перетворення, є практично нульовою і не впливає на загальну продуктивність системи.

Ця модель вирізняється компактними розмірами (65 мм × 30 мм) та надзвичайно низькою вагою – лише 13 грамів. Така фізична легкість і мінімальний розмір роблять пристрій ідеальним для вбудованих систем, де мобільність та мінімальна вага мають ключове значення, що в свою чергу підвищує загальну ефективність системи. Крім того, доступність Raspberry Pi Zero 2 W за помірною ціною забезпечує економічну вигоду, що особливо важливо на стадії прототипування або при необхідності виготовлення декількох пристроїв у межах обмеженого бюджету [21].

Щодо продуктивності, то Raspberry Pi Zero 2 W оснащений чотириядерним процесором ARM Cortex-A53, який є одним із найбільш ефективних і широко застосовуваних процесорів у вбудованих системах середнього рівня продуктивності. Архітектура Cortex-A53 базується на 64-бітовій ARMv8-A технології, що забезпечує покращену продуктивність при одночасному збереженні енергоефективності, що є важливим для пристроїв із обмеженими ресурсами живлення. Чотириядерна конфігурація дозволяє розподіляти обчислювальні задачі між ядрами, забезпечуючи ефективну багатозадачність та підвищуючи загальну швидкодію системи. Це особливо важливо для обробки паралельних потоків даних, таких як відео в реальному часі, одночасне зчитування з кількох сенсорів. Архітектура Cortex-A53 підтримує сучасні технології керування енергоспоживанням, що дозволяє оптимізувати баланс між продуктивністю і тривалістю роботи пристрою, особливо в автономних системах. Окрім того, цей процесор сумісний із широким спектром програмного забезпечення на основі Linux, що спрощує розробку і впровадження складних додатків.

Для організації комунікації пристрій обладнаний інтегрованими модулями Wi-Fi та Bluetooth, що надає можливість бездротового підключення до інших пристроїв або мережі Інтернет. Ця функціональність сприяє реалізації віддаленого управління системою, а також ефективному збору, передачі та обробці даних у режимі реального часу.

Наявність UART-порту у Raspberry Pi Zero 2 W є важливою характеристикою з огляду на специфіку застосування протоколу MAVLink у вбудованих та автономних системах. MAVLink здатен функціонувати через різні фізичні та транспортні рівні зв'язку, серед яких можна виділити UART, USB, TCP/IP, UDP, CAN та інші. Серед цих варіантів UART залишається одним із найбільш поширених і практичних способів передачі MAVLink-повідомлень. Це зумовлено кількома ключовими факторами. По-перше, UART забезпечує простий і відносно недорогий апаратний інтерфейс, який легко інтегрується в більшість мікроконтролерів та вбудованих платформ. По-друге, він характеризується низьким енергоспоживанням, що є критично важливим для автономних систем, де ресурси живлення обмежені. По-третє, UART-передача демонструє високу надійність в умовах реального використання, забезпечуючи стабільний обмін даними навіть у середовищах із електромагнітними завадами чи іншими перешкодами.

2.1.3 Вибір польотного контролера

Польотний контролер є одним із найважливіших і фундаментальних компонентів будь-якого безпілотної літального апарату. Він виконує функцію центрального обчислювального модуля, який забезпечує стабілізацію і керування польотом дрона. Контролер приймає та аналізує інформацію, що надходить з різноманітних датчиків, таких як гіроскопи, акселерометри, барометри, GPS-модулі та інші сенсори, що вимірюють положення, швидкість і орієнтацію апарата в просторі. На основі цих даних польотний контролер формує команди для виконавчих механізмів, таких як мотори та серводвигуни, які забезпечують необхідну реакцію дрона на зовнішні умови та керуючі сигнали оператора.

Коректність вибору польотного контролера має вирішальне значення для ефективності роботи безпілотного літального апарата. Від характеристик контролера залежить, наскільки стабільним, маневреним і надійним буде дрон під

час польоту, а також його здатність адаптуватися до змінних умов навколишнього середовища. Крім того, якісний контролер забезпечує точність виконання команд, зменшує ризик аварійних ситуацій та підвищує загальну безпеку польоту.

Польотні контролери класифікуються залежно від їх призначення, складності та функціональних можливостей [22]:

– базові контролери, які забезпечують стабілізацію польоту у простих безпілотних літальних апаратах із мінімальним набором сенсорів і базовими режимами керування;

– професійні контролери оснащені розширеним комплексом датчиків, мають потужніші процесори та підтримують додаткові функції, такі як автоматичне планування маршрутів, слідкування за об’єктами та системи безпеки, включаючи автоматичне повернення і уникнення перешкод;

– спеціалізовані контролери, розроблені для різних типів літальних апаратів: мультикоптерів, літаючих крил або гелікоптерів, які відрізняються алгоритмами стабілізації та підтримкою відповідних типів виконавчих механізмів.

Контролери поділяються також за типом програмного забезпечення: з відкритим кодом, що дозволяє модифікацію прошивки відповідно до потреб проєкту, та з закритим ПЗ, які пропонують готові рішення без можливості глибокого налаштування.

Під час вибору контролера польоту для FPV-дрона необхідно враховувати низку ключових параметрів, що впливають на його ефективність і сумісність із системою загалом. Насамперед, контролер має бути технічно сумісним із використовуваними сенсорами, двигунами, регуляторами швидкості, а також забезпечувати стабільну взаємодію з бортовим комп’ютером та телеметричною системою. Важливою є й обчислювальна потужність – сучасні контролери оснащуються високопродуктивними процесорами, що дозволяє оперативно обробляти сигнали і забезпечувати точне керування навіть у складних умовах. Не менш значущою є підтримка відкритого програмного забезпечення, такого як Betaflight, ArduPilot чи iNav, що дає можливість гнучко налаштовувати параметри

польоту відповідно до конкретних вимог. Оскільки вага та габарити важливі для FPV-дронів, контролер має бути максимально компактним і легким, аби не обмежувати льотні характеристики. Також бажаною є наявність вбудованих датчиків, зокрема гіроскопа, акселерометра, барометра та GPS, які підвищують стабільність польоту та дають змогу реалізувати автономні режими керування.

Matek Systems – це відомий виробник високоякісної електроніки для безпілотних літальних апаратів, зокрема FPV-дронів, що спеціалізується на розробці польотних контролерів, регуляторів напруги, сенсорних модулів, розподільників живлення та інших пристроїв [23]. Вироби Matek характеризуються продуманою інженерною конструкцією, стабільною роботою, високою якістю виготовлення та широкою підтримкою з боку глобальної спільноти розробників. Однією з визначальних переваг є повна сумісність з відкритими платформами програмного забезпечення, такими як ArduPilot, Betaflight та iNav, що забезпечує гнучкість у налаштуванні та адаптації системи під конкретні задачі.

З огляду на вищезазначене, для реалізації даного проєкту було обрано польотний контролер Matek F765-WSE, який є оптимальним вибором завдяки поєднанню високої продуктивності, широкої функціональності та надійності.

	F765-WSE
MCU	STM32F765VIH6
IMU	ICM-42688-P
Current sensor	220A
Barometer	DPS310
UARTs	6.5
CAN	1
USB	External Type-C
Size	44*29*14.5mm
Firmware	ArduPilot MatekF765SE
	INAV MatekF765SE

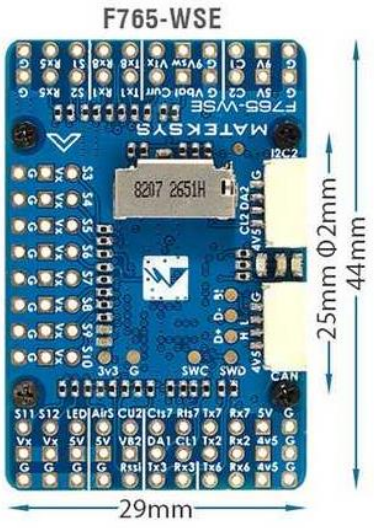


Рисунок 2.3 – Польотний Контролер Matek Flight Controller F765-WSE [24]

Даний контролер побудований на базі продуктивного 32-бітного процесора STM32F765, який має достатній обчислювальний ресурс для обробки телеметричної інформації, виконання алгоритмів стабілізації, навігації та інших завдань автономного керування в реальному часі [24].

Наявні розширені можливості підключення зовнішніх модулів, зокрема до п’яти UART-портів, а також підтримка інтерфейсів I2C, CAN, SBUS і PPM, що забезпечує високу гнучкість у конфігурації системи та сумісність з широким спектром сенсорів, GPS-приймачів, телеметричних модулів, компасів та іншого периферійного обладнання. Така інженерна відкритість дозволяє адаптувати польотний контролер до конкретних завдань і сценаріїв використання без необхідності в додаткових апаратних модифікаціях. Крім того, контролер повністю підтримує протокол MAVLink, який є стандартом для телеметричного зв’язку в екосистемі ArduPilot. Завдяки цьому забезпечується стабільна та ефективна інтеграція з бортовими або наземними обчислювальними системами, що критично важливо для передачі польотних даних, реалізації автономного керування та моніторингу в реальному часі.

Контролер оснащений вбудованими сенсорами, зокрема інерціальною вимірювальною системою, що включає гіроскоп і акселерометр, барометром, а також передбачає можливість підключення зовнішнього компаса для забезпечення точного просторового орієнтування.

Завдяки компактним габаритам і малій масі, контролер Matek F765-WSE є оптимальним варіантом для встановлення у конструкції FPV-дронів. Окрім цього, економічна доцільність вибору контролера є ще одним важливим аргументом: на відміну від дорожчих альтернатив, таких як Pixhawk, дана модель має значно нижчу вартість, водночас забезпечуючи повну сумісність з програмним забезпеченням ArduPilot і підтримку всіх ключових функцій, необхідних для реалізації складних сценаріїв автономного керування. Такий баланс між ціною, функціональністю та розмірами робить Matek F765-WSE практичним рішенням для широкого спектра застосувань.

2.2 Функціональні вимоги

Функціональні вимоги – це чітко сформульовані та формалізовані описи конкретної поведінки системи у відповідь на певні дії користувача, події або умови зовнішнього середовища [25].

Вони визначають, які саме функції має виконувати система для досягнення поставлених цілей, включаючи обробку інформації, прийом і виконання команд, генерацію відповідей та взаємодію між різними компонентами системи. Функціональні вимоги задають чіткі межі функціональності, описують алгоритми роботи, умови активації функцій, формат вхідних і вихідних даних, а також правила обробки помилок. Вони є основою для розробки програмного забезпечення, тестування та валідації, забезпечуючи відповідність кінцевого продукту очікуванням користувачів і замовників. Завдяки точному визначенню функціональних вимог можна уникнути неоднозначностей, помилок у розробці та забезпечити ефективну комунікацію.

Отож, функціональні вимоги до системи стабілізації дрона-ретранслятора визначають перелік обов’язкових дій, що забезпечать стабільну роботу в реальному часі. По-перше, система повинна підтримувати інтеграцію з RC-передавачем для ініціації та зупинки режиму стабілізації. Зчитування положення тумблера на каналі CH7 є обов’язковим: при значенні PWM > 1800 система повинна автоматично формувати команду запуску стабілізації, а при значенні PWM < 1200 – команду її зупинки.

По-друге, необхідно забезпечити стійкий обмін між польотним контролером та Raspberry Pi через інтерфейс UART. Польотний контролер повинен надсилати керуючі повідомлення у вигляді команд START_SCRIPT та STOP_SCRIPT, на які Raspberry Pi зобов’язаний реагувати негайно. Увесь час роботи система повинна утримувати активне прослуховування порту UART з метою гарантованого прийому команд.

Третя вимога стосується керування основним скриптом. Raspberry Pi, отримавши команду START_SCRIPT, повинен ініціювати запуск основного Python-скрипта, який виконує обробку відео та формування управляючих команд. У разі отримання STOP_SCRIPT, скрипт має бути коректно завершений за допомогою команди pkill -f.

Четверта функція передбачає обробку відеопотоку з аналогової камери, що надходить через пристрій AV-to-USB. Система повинна здійснювати виявлення та ідентифікацію об’єкта в потоці зображення (тепловий слід або контрастна мітка), а також розраховувати просторове зміщення об’єкта відносно центру кадру для подальшого коригування положення дрона.

П’ята вимога – це генерація команд стабілізації на основі отриманих даних про зміщення об’єкта. Система має формувати команди корекції кутів крену (roll) і тангажу (pitch) та передавати їх польотному контролеру через протокол MAVLink по UART. Це забезпечує точне утримання позиції дрона та адаптацію до зовнішніх впливів.

Шоста важлива функція полягає у підтримці роботи у симуляційному середовищі SITL ArduPilot. У режимі симуляції всі функції системи мають працювати ідентично реальному режиму, включаючи прийом команд, запуск і зупинку скриптів, обробку відеопотоку та надсилання команд стабілізації. Це дозволяє проводити тестування та налагодження без використання фізичного апарату.

Додатково, система повинна забезпечувати надійне логування всіх ключових подій і станів під час роботи, що спростить діагностику та оптимізацію алгоритмів. Також важливо передбачити механізми обробки помилок і аварійних ситуацій, зокрема коректне завершення роботи скриптів та безпечну зупинку стабілізації у разі втрати зв’язку або некоректних вхідних даних для забезпечення стабільної, безпечної і надійної роботи системи дрона-ретранслятора у різних умовах експлуатації.

2.3 Нефункціональні вимоги

Нефункціональні вимоги – це формалізовані характеристики, які визначають якісні аспекти функціонування системи, що не стосуються безпосередньо конкретної поведінки у відповідь на вхідні дії, але суттєво впливають на загальну ефективність, надійність, зручність використання та експлуатаційні властивості системи [25]. Вони окреслюють критерії, яких система має дотримуватися під час реалізації функціональних потреб. Це стосується продуктивності (наприклад, часу відповіді, швидкості обробки даних), здатності до масштабування, доступності, захищеності, стійкості, відмовостійкості, переносності, можливості зміни, відповідності стандартам та інших параметрів.

Нефункціональні вимоги часто формують основу для встановлення метрик якості системи, служать критеріями приймання та оцінки відповідності системи очікуванням користувачів, а також визначають технічні, регуляторні та експлуатаційні обмеження, у межах яких має відбуватися розробка та функціонування системи. Їхнє дотримання є критично важливим для досягнення високої якості програмного забезпечення, забезпечення користувацького досвіду та задоволення вимог безпеки, надійності та відповідності.

Однією з ключових нефункційних вимог є надійність. Система повинна стабільно функціонувати впродовж тривалого часу без збоїв, гарантуючи безперервне виконання усіх задач у режимі реального часу. Можливі відмови, зокрема збої в роботі бортового комп’ютера, втрата відеосигналу з тепловізора або порушення каналу зв’язку з польотним контролером, мають бути передбачені в архітектурі системи. У таких випадках повинні автоматично активуватись захисні сценарії, наприклад, утримання позиції, обмеження активних команд або примусова посадка дрона з мінімальними ризиками для навколишнього середовища й самого апарата.

Продуктивність є другою нефункційною вимогою. Усі ключові обчислювальні процеси, зокрема аналіз відеозображення, прийняття рішень та

передача керуючих команд на польотний контролер мають виконуватись із мінімальними затримками. У системах стабілізації, що реагують на просторові відхилення в режимі реального часу, загальна затримка обробки не повинна перевищувати 150 мс. Перевищення цього порогу здатне спричинити суттєве відхилення дрона від заданої траєкторії або положення, що унеможливлює точну фіксацію на об’єкті й може призвести до втрати керування в складних умовах.

Наступним аспектом є масштабованість системи, оскільки архітектура програмного забезпечення має забезпечувати можливість подальшого розширення без суттєвих змін у вже існуючому коді. Це включає підключення нових сенсорів, інтеграцію додаткових обчислювальних модулів, оновлення алгоритмів стабілізації або зміни логіки автономного керування. Важливо, щоб такі модифікації не вимагали повної перебудови архітектури, що забезпечує гнучкість системи та спрощує підтримку і розвиток проєкту в майбутньому.

Крім того, система повинна повинна характеризуватись можливістю перенесення програмного забезпечення, адже можливість запуску програмного забезпечення на інших апаратних платформах із операційною системою Linux без суттєвих змін у коді відкриває перспективи для використання розробленого ПЗ на різних моделях дронів або у альтернативних апаратних середовищах, що підвищує універсальність і довгострокову ефективність проєкту.

Сумісність із апаратними компонентами є обов’язковою: програмне забезпечення повинно коректно взаємодіяти з тепловізійною камерою, польотним контролером та іншими сенсорами через стандартизовані інтерфейси та протоколи, такі як UART, I2C, SPI та MAVLink. Також важливо, щоб середовище розробки підтримувало основні бібліотеки Python, зокрема OpenCV, MAVProxy, що забезпечить зручність реалізації алгоритмів обробки зображень і комунікації.

Зважаючи на обмеження по живленню безпілотного літального апарата, особливу увагу необхідно приділяти оптимізації програмних модулів, що запускаються на одноплатному комп’ютері Raspberry Pi. Всі компоненти програмного забезпечення повинні бути спроектовані таким чином, щоб

мінімізувати навантаження на центральний процесор, оперативну пам’ять та інші апаратні ресурси системи. Такий підхід важливий, оскільки надмірне енергоспоживання призводить до швидкого виснаження акумуляторної батареї, що безпосередньо обмежує тривалість автономного польоту безпілотника.

2.4 Вибір методів і середовища для реалізації програмного забезпечення

Керування дроном здійснюватиметься через подачу команд на польотний контролер за допомогою протоколу зв’язку MAVLink. Програмне забезпечення включає Linux для операційної системи, Python для обробки даних, а також ArduPilot SITL для симуляції та тестування алгоритмів автономного управління. Це забезпечить гнучкість для інтеграції всіх компонентів та їх тестування в умовах, наближених до реальних.

Для реалізації програмної частини проєкту як основний інструмент розробки було обрано мову програмування Python. Цей вибір зумовлений її високим рівнем абстракції, простотою синтаксису та широким спектром доступних бібліотек. Python є однією з найбільш популярних мов у сфері робототехніки та автоматизованих систем завдяки своїй здатності забезпечувати швидку інтеграцію з апаратним забезпеченням і підтримкою таких напрямів, як комп’ютерний зір (зокрема за допомогою бібліотек OpenCV), робота з телеметрією (наприклад, через pymavlink), обробка сигналів, а також реалізація мережевих протоколів для віддаленого моніторингу та керування.

Однією з ключових переваг мови програмування Python є її повна і широкомасштабна підтримка в середовищі операційної системи Linux, що, у свою чергу, розгортається на одноплатному комп’ютері Raspberry Pi Zero 2 W. Використання операційної системи Linux забезпечує стабільне та безперебійне функціонування вбудованих систем завдяки розширеним можливостям керування апаратними ресурсами, ефективному розподілу процесорного часу та оперативної пам’яті, а також підтримці багатозадачності та багатопоточності. Linux також надає

гнучкі інструменти конфігурування системи, що дозволяє адаптувати її під конкретні потреби проєкту. Важливою перевагою є широка сумісність із зовнішніми пристроями завдяки наявності великої кількості драйверів, що підтримують різноманітні сенсори, камери, модулі зв'язку та інші периферійні компоненти, необхідні для реалізації складних апаратних конфігурацій.

Крім того, Linux дозволяє ефективно розгортати фонові сервіси (демони), які забезпечують безперервний збір та обробку даних, обробку вхідних команд і реалізацію автономних сценаріїв керування системою [26]. Завдяки відкритій архітектурі операційної системи та широкому набору доступних інструментів адміністрування, можна реалізувати складні алгоритми моніторингу, діагностики та оновлення програмного забезпечення в реальному часі.

У рамках даного проєкту застосовуються низка ключових бібліотек мови програмування Python, що забезпечують необхідний функціонал для реалізації складних завдань автономного керування та обробки даних.

Бібліотека pymavlink [27] є офіційним клієнтом протоколу MAVLink, що надає можливості для двонаправленої комунікації з польотним контролером. Вона дозволяє здійснювати надсилання та прийом телеметричних повідомлень, що включають дані про стан дрона, параметри польоту, а також інформацію з різноманітних сенсорів. Завдяки цій бібліотеці забезпечується інтеграція з апаратним забезпеченням автономної системи, реалізується можливість дистанційного керування польотом та обробки команд у реальному часі, що є необхідних для підтримки стабільності і безпеки роботи системи.

А також бібліотека opencv-python, яка є Python-інтерфейсом до широко відомої платформи OpenCV (Open Source Computer Vision Library) [28], використовується для обробки зображень і відеопотоку. Вона забезпечує інструментарій для виконання базових операцій, таких як фільтрація, сегментація, виявлення об'єктів та аналіз зображень, отриманих з тепловізійної камери. Це дозволяє реалізувати алгоритми комп'ютерного зору у режимі реального часу,

задля швидкого та точного виявлення цілей, адаптивної реакції системи та подальшої обробки інформації для прийняття рішень в автономному режимі.

Обладнання для автономних безпілотних апаратів, зокрема FPV-дронів із ретранслятора ми них, зазвичай є дорогим. Проведення безпосередніх польотних тестів із використанням фізичних пристроїв не лише підвищує ризики пошкодження техніки, а й потребує значних часових та матеріальних ресурсів. Відтак, для забезпечення ефективного та безпечного процесу розробки програмного забезпечення надзвичайно важливим є застосування спеціалізованих платформ для тестування польоту, які дозволяють моделювати роботу автономних систем у віртуальному середовищі.

Існують різноманітні платформи для тестування польоту безпілотних літальних апаратів, серед яких Gazebo, JSBSim, FlightGear, а також спеціалізовані середовища на кшталт ArduPilot SITL (Software-In-The-Loop). Кожна з цих платформ має свої переваги:

– Gazebo забезпечує розширену фізичну симуляцію з високою точністю моделювання навколишнього середовища [29];

– JSBSim і FlightGear використовуються для авіаційних тренувань і наукових досліджень [30, 31];

– ArduPilot SITL пропонує можливість повної емуляції прошивки польотного контролера, що дозволяє перевіряти не лише фізичні, а й логічні аспекти роботи системи [19].

Для реалізації нашого проєкту було обрано платформу ArduPilot SITL з огляду на її специфічні переваги. ArduPilot є одним із найпоширеніших і найкраще підтримуваних open-source рішень для автономного керування дронами, що гарантує стабільність і широкий спектр функціональних можливостей. Окрім того, SITL дозволяє запускати повноцінну прошивку польотного контролера на звичайному комп’ютері, що забезпечує максимальну достовірність емуляції роботи реального апаратного забезпечення. Також ця платформа інтегрується з протоколом MAVLink, забезпечуючи тим самим можливість тестування і

налагодження комунікації між Python-програмами та системою керування польотом в умовах, максимально наближених до реальних.

Поєднання операційної системи Linux, мови програмування Python та симулятора ArduPilot SITL формує функціональне, гнучке та масштабоване середовище розробки, яке забезпечує повноцінну підтримку процесів моделювання, тестування та розгортання автономних керованих систем на базі Raspberry Pi у поєднанні з контролерами польоту, сумісними з протоколом MAVLink.

2.5 Висновки до другого розділу

У другому розділі було здійснено вибір технічних засобів і програмного середовища, необхідних для реалізації системи стабілізації дрона-ретранслятора на основі обробки зображення, сформовано цілісну концепцію побудови функціональної автономної системи.

На основі зіставлення технічних параметрів, сумісності, енергоспоживання та габаритів було визначено доцільність використання таких апаратних компонентів, як одноплатний мікрокомп’ютер Raspberry Pi Zero 2 W, що виступає в ролі обчислювального вузла системи; тепловізійна камера Foxeer Cat 3 Mini, яка дозволяє отримувати зображення в умовах низької видимості й передавати сигнал у режимі реального часу; а також польотний контролер Matek F765-WSE, здатний ефективно обробляти дані з сенсорів і реалізовувати стабілізаційні алгоритми завдяки потужному мікроконтролеру та підтримці протоколу MAVLink.

Було сформульовано функційні вимоги, що визначають основні можливості системи: стабілізація положення дрона у просторі, обробка відеопотоку з тепловізійної камери, передача телеметричної інформації та можливість тестування всіх функціональних модулів у програмному симуляторі.

Також були сформульовані нефункційні вимоги, які встановлюють якісні характеристики системи, що визначають її експлуатаційну придатність. Зокрема,

було встановлено вимоги щодо надійності, продуктивності, масштабованості, портативності, сумісності, енергоефективності та підтримуваності програмного забезпечення.

Для реалізації програмної частини проєкту було обґрунтовано вибір операційної системи Linux як надійної платформи для вбудованих систем з відкритим кодом, що забезпечує підтримку широкого спектра драйверів та дозволяє налаштовувати паралельне виконання сервісів. Основною мовою розробки обрано Python, яка має потужну екосистему бібліотек для комп'ютерного зору, обробки даних із сенсорів, роботи з телеметрією та мережевими інтерфейсами. Додатково, для забезпечення попереднього тестування системи, було обрано використання симуляційного середовища ArduPilot SITL, що дозволяє здійснювати перевірку алгоритмів автономного керування у безпечних віртуальних умовах без залучення фізичної апаратури.

РЕАЛІЗАЦІЯ СИСТЕМИ СТАБІЛІЗАЦІЇ ДЛЯ ДРОНА-РЕТРАНСЛЯТОРА

3.1 Принцип роботи системи стабілізації для дрона-ретранслятора на основі обробки зображення

Як уже зазначалося в попередніх розділах, у рамках реалізації даного проєкту було застосовано поєднання апаратних і програмних засобів, що у сукупності забезпечують реалізацію задачі стабілізації квадрокоптера з використанням тепловізійної камери.

З програмної точки зору основною мовою розробки є високорівнева мова Python, яка завдяки своїй простоті та широкому набору бібліотек значно спрощує взаємодію з апаратними компонентами через протокол зв’язку MAVLink. В апаратній частині системи було використано компактний і енергоефективний бортовий мікрокомп’ютер Raspberry Pi Zero 2 W, до якого підключено тепловізійну камеру Foxeer Cat 3 Mini, що виконує роль основного сенсора для візуального відстеження положення дрону відносно стаціонарного об'єкта.

Ключовим елементом апаратної системи є польотний контролер, який містить вбудовані інерційні сенсори – гіроскоп та акселерометр, а також барометр для визначення висоти. Через MAVLink-протокол, який виконує роль медіатора між Raspberry Pi та польотним контролером, відбувається обмін телеметричними даними. Зокрема, зчитуються поточні значення барометричної висоти та орієнтація дрона, яка представлена у вигляді кватерніона.

Кватерніони – це система гіперкомплексних чисел, яка утворює чотиривимірний векторний простір над полем дійсних чисел, вона була запропонована Вільямом Гамільтоном у 1843 році [32]. Завдяки своїм алгебраїчним властивостям, кватерніони широко застосовуються у комп’ютерній графіці, робототехніці, авіації та інших технічних галузях, де необхідно описувати обертання об’єктів у тривимірному просторі без ризику виникнення сингулярностей, таких як ефект ґімбального блокування. Це явище, що виникає під

час опису обертання об’єкта за допомогою системи з трьома ступенями вільності, наприклад, за допомогою Ейлерових кутів [33].

Суть ефекту полягає в тому, що при певному положенні одна з осей обертання вирівнюється з іншою, внаслідок чого втрачається один ступінь вільності. Це робить неможливим незалежне керування поворотом об’єкта в усіх трьох напрямках. В управлінні дроном, блокування обертання [34] може спричинити серйозні проблеми у навігації й стабілізації, оскільки система втрачає можливість правильно реагувати на зміни орієнтації. Саме тому кватерніони, які не схильні до цього ефекту, є кращим інструментом для опису обертання у тривимірному просторі.

Оскільки на вході з контролера ми отримуємо три осі, тобто значення тангажу, крену та рискання їх необхідно перетворити у кватерніон q = (w, x, y, z) за формулами перетворення Ейлера-Гамільтона:

$$
\begin{aligned}
w &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2},\\
x &= \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2},\\
y &= \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2},\\
z &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2},
\end{aligned}
\qquad (3.1)
$$

де ϕ – кут крену,

θ – кут тангажу,

ψ – кут рискання,

w – скалярна частина кватерніона,

x, y, z – векторна частина кватерніона.

Щоб уникнути накопичення похибок під час обчислень і забезпечити правильне представлення орієнтації, кватерніони після обробки треба

нормалізувати. Нормалізація полягає у приведенні довжини кватерніона до одиниці за допомогою поділу кожної компоненти на його норму:

$$\hat{q} = \frac{q}{\|q\|} = \frac{(w, x, y, z)}{\sqrt{w^2 + x^2 + y^2 + z^2}}, \tag{3.2}$$

де $\hat{q}$ – нормалізований кватерніон,

w – скалярна частина кватерніона,

x, y, z – векторна частина кватерніона.

Отриманий кватерніон разом із даними барометра синхронізується із результатами обробки зображення з тепловізійної камери. На зображенні аналізується положення заздалегідь визначеної стаціонарної теплової цілі, і фіксується її зсув відносно центру кадру зсув по горизонталі Δx та вертикалі Δy. Оскільки камера має обмежене кутове поле зору FOV (Field of View), кожен піксель зображення відповідає певному куту в просторі [36]. Це дозволяє виконати перетворення піксельного зсуву в кутові відхилення, які можуть бути використані для зміни крену та тангажу дрона:

$$\alpha_x = \Delta x \cdot \left(\frac{FOV_{hor}}{w}\right), \qquad \alpha_y = \Delta y \cdot \left(\frac{FOV_{ver}}{h}\right), \tag{3.3}$$

де α_x – кутове відхилення по горизонталі (крен),

α_y – кутове відхилення по вертикалі (тангаж),

FOV_{hor}, FOV_{ver} – кут огляду камери по горизонталі та вертикалі у радіанах,

w, h – роздільна здатність зображення по горизонталі та вертикалі.

Окрім того, кут α_y додатково інвертується, оскільки у зображеннях вісь у зазвичай направлена вниз, а позитивний тангаж навпаки означає підйом носа дрона вгору. На основі цих даних за допомогою пропорціонально-інтегрально-

диференціального (ПІД) закону регулювання формуються нові значення керуючих параметрів.

Для досягнення стабільного положення дрона використовується ПІД-регулятор – один із найбільш поширених алгоритмів у сфері автоматичного керування. Основна мета ПІД-регулятора полягає в мінімізації відхилення керованої величини, наприклад кута нахилу або висоти, від заданого значення, з урахуванням динамічних властивостей системи [37]. Це дозволяє досягати плавного і точного регулювання, уникати різких рухів і підтримувати бажане положення дрона в просторі.

Параметри ПІД-регулятора включають:

– коефіцієнт пропорційної складової, який визначає швидку реакцію системи на поточну помилку: чим більша помилка, тим більший вплив;

– коефіцієнт інтегральної складової, що відповідає за накопичення помилки в часі і усунення постійної похибки, яку не компенсує пропорційна частина;

– коефіцієнт диференціальної складової, що реагує на швидкість зміни помилки, допомагаючи зменшити коливання та згладити поведінку системи.

Тому функціональне призначення кожної складової можна підсумувати, як: пропорційна складова забезпечує швидку реакцію на величину помилки, інтегральна – усунення сталих похибок, а диференціальна – зменшує коливання, реагує на швидкі зміни помилки.

Отже, ПІД-регулятор обчислює керуючий сигнла *u(t)* за наступною формулою:

$$u(t) = K_P \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}, \qquad (3.4)$$

де $u(t)$ – керуючий сигнал, що подається на виконавчі механізми,

$e(t)$ – помилка між бажаним та фактичним значенням, обчислювана як їхня різниця,

K_P – коефіцієнт пропорційної складової (реакція на поточну помилку),

K_i – коефіцієнт інтегральної складової (накопичення помилки з часом),

K_d – коефіцієнт диференціальної складової (реакція на зміну помилки в часі).

Застосування ПІД-регулятора у контексті стабілізації дрона полягає у підтриманні рівноваги по всіх трьох осях просторового руху: тангажу (нахил вперед-назад) та крену (нахил вліво-вправо). Для кожної осі задається бажаний кут, який відповідає стабільному положенню, а поточні кути вимірюються за допомогою інерціального модуля. Помилка визначається як різниця між бажаним і фактичним значенням для кожної осі. ПІД-регулятори обробляють ці помилки окремо, генеруючи коригуючі сигнали, які змінюють тягу відповідних двигунів і приводять дрон у стабільне положення у просторі.

Також для стабілізації висоти дрона використовується барометричний датчик на польотному контролері Matek F765-WSE, що вимірює поточну висоту польоту. Похибка висоти визначається як різниця між цільовою висотою та поточним значенням барометра. На основі цієї помилки застосовується ПІД-регулятор, який формує коригувальне значення тяги. ПІД-контролер враховує пропорційну, інтегральну та диференційну складові помилки для забезпечення плавної і точної підтримки висоти дрона. Це дозволяє компенсувати зовнішні впливи, такі як вітер або незначні зміни навантаження.

Для запобігання виникненню непотрібних коливань і надмірних корекцій у системі стабілізації дрона впроваджується концепція так званої «мертвої зони» для помилок по нахилу (крен, тангаж) та по висоті. Ця мертва зона визначає граничний діапазон значень помилки, при якому регулятор не здійснює коригувальних дій. Інакше кажучи, якщо величина помилки знаходиться в межах встановленого малого порогу, система стабілізації вважає таке відхилення незначним і не реагує на нього, утримуючи команди керування на попередньому рівні.

Впровадження мертвої зони має низку важливих переваг, адже вона дозволяє усунути постійні дрібні коливання, викликані шумом датчиків або незначними, але швидкоплинними змінами положення дрона, що в іншому випадку призводили б

до безперервних реакцій контролера і, відповідно, до зайвих ривків і «тряски», що особливо важливо для завдань ретрансляції або тривалого висіння в повітрі.

На основі цього, відповідні команди формуються у форматі MAVLink-пакетів та передаються назад на польотний контролер для виконання, що дозволяє підтримувати стабільне положення дрону в повітрі над обраною фіксованою точкою, компенсуючи зовнішні збурення, зокрема пориви вітру чи інерційний дрейф. Для цього формується повідомлення, яке містить кватерніон бажаної орієнтації, значення тягової сили для утримання висоти та бітову маску, що визначає, які компоненти слід враховувати, а які навпаки ігнорувати. У разі потреби, команда може також включати цільові значення кутових швидкостей, що дає змогу точно керувати динамікою руху дрона навіть за наявності затримок чи нестабільностей в реакції апаратного регулятора.

3.2 Структурна схема та алгоритм роботи системи стабілізації для дрона-ретранслятора на основі обробки зображення

Розроблена система стабілізації для дрона-ретранслятора ґрунтується на взаємодії низки апаратних компонентів, що формують єдину функціональну архітектуру. Основною метою цієї структури є забезпечення стабільного положення дрона у просторі на основі аналізу відеосигналу з тепловізійної камери, а також своєчасна передача телеметричних даних і точний контроль руху в режимі реального часу. Для досягнення цих завдань система інтегрує різні модулі, які забезпечують безперервний збір, обробку та передачу інформації між собою, що дозволяє ефективно реагувати на зміни зовнішнього середовища та підтримувати високу стабільність польоту.

Як уже зазначалося, бортова система складається з кількох ключових компонентів, що взаємодіють між собою для забезпечення стабілізації та управління дроном:

– FPV-дрон, обладнаний польотним контролером Matek F765-WSE, який виконує завдання стабілізації, навігації та управління двигунами через електронні регулятори швидкості;

– одноплатний мікрокомп’ютер Raspberry Pi Zero 2 W, що виступає в ролі обчислювального вузла, відповідає за обробку відеопотоку, аналіз просторових зсувів цілі та формування керуючих команд для автопілота;

– відеоперетворювач аналогового сигналу у цифровий формат (AV в USB), який забезпечує оцифрування композитного відеосигналу CVBS та передачу отриманого потоку у цифровому вигляді до Raspberry Pi через інтерфейс USB;

– камера нічного бачення Foxeer Cat 3 Mini із оптичним модулем, яка генерує аналоговий відеосигнал, необхідний для подальшої обробки й аналізу зображень, спрямованих на точне наведення дрона на визначену ціль.

Ключові взаємодії між компонентами системи здійснюються наступним чином: Тому при виборі FPV-дрона необхідно враховувати такі технічні характеристики:

– двигуни та електронні регулятори швидкості (ESC), які мають бути достатньо потужними, щоб забезпечити стабільність дрона навіть із додатковим навантаженням;

– обмін даними між одноплатним мікрокомп’ютером Raspberry Pi Zero 2 W та польотним контролером Matek F765-WSE організовано через UART-інтерфейс із використанням протоколу MAVLink, що забезпечує двонаправлену передачу команд керування та телеметричних параметрів;

– зображення з камери надходить на Raspberry Pi через відеопередавач, що оцифровує зображення, де обробляється в реальному часі з використанням бібліотек комп’ютерного зору;

– результати обробки відеосигналу трансформуються у MAVLink-команди, які передаються з Raspberry Pi до польотного контролера для корекції положення дрона у просторі.

Відповідно до цього структурна схема системи виглядає наступним чином.

Рисунок 3.1 – Структурна схема системи

Дана структурна схема ілюструє взаємодію основних апаратних компонентів системи стабілізації дрона з використанням бортового комп’ютера Raspberry Pi Zero 2 W. Джерелом візуальної інформації виступає аналогова камера Foxeer Cat 3 Mini, яка передає відеосигнал у форматі. Через відсутність цифрового інтерфейсу камера не може бути безпосередньо під’єднана до Raspberry Pi, тому використовується зовнішній відеоконвертер, який оцифровує аналоговий сигнал і передає його через інтерфейс USB у вигляді цифрового потоку. Для забезпечення сумісності з операційною системою Raspberry Pi використовується USB Video Class-сумісний модуль захоплення відео, що дозволяє працювати з відеопотоком за допомогою стандартних засобів обробки зображень у Linux.

Raspberry Pi Zero 2 W виконує кілька критично важливих функцій. По-перше, він приймає цифровий відеопотік у реальному часі та здійснює попередню обробку зображення, зокрема для виявлення стаціонарних об’єктів, які можуть використовуватись як візуальні орієнтири. По-друге, мікрокомп’ютер отримує дані

телеметрії від польотного контролера Matek F765-WSE через протокол MAVLink. Зокрема, надходять значення кватерніону орієнтації дрона та барометричної висоти. По-третє, на основі поєднаної обробки зображення і телеметричних даних бортовий комп'ютер формує нові керуючі команди, які повертаються до польотного контролера також через MAVLink.

Польотний контролер Matek F765-WSE виконує збір і попередню обробку даних з вбудованих сенсорів, таких як інерційний модуля, барометр, та приймає команди як з RC-пульта, так і з бортового комп'ютера. Вихідними керуючими параметрами є значення крену, тангажу та тяги, які реалізуються через моторну підсистему дрона. Крім того, контролер передає поточний стан дрона бортовому комп'ютеру для подальшого аналізу та генерації коригувальних впливів.

Вся взаємодія між Raspberry Pi та польотним контролером здійснюється через протокол MAVLink, що забезпечує надійну двосторонню передачу повідомлень з телеметрією та командами.

Алгоритм роботи даної системи стабілізації складається з наступних ключових етапів. Перед початком роботи даної програми на постійній основі, слідуючи даному алгоритму вважається, що усі апаратні та програмні компоненти попередньо вже налаштовані (детальний опис налаштувань описаний у наступному пункті даного розділу), а також живлення під'єднано, Raspberry Pi увімкнено, дрон знаходиться у повітрі в режимі ручного керування.

Отож, спершу пілот активує режим стабілізації, перемкнувши тумблер на пульті. Польотний контролер приймає команду START_SCRIPT і передає її на Raspberry Pi через MAVLink. В цей час на Raspberry Pi працює демон-процес, який у фоновому режимі відстежує вхідні повідомлення. Отримавши даний сигнал активації, демон ініціює запуск основного скрипта.

Після запуску, основний python-скрипт створює власне з'єднання бортового мікрокомп'ютера з польотним контролером через MAVLink. Встановлюється частота оновлення телеметрії, перевіряється готовність до прийому команд типу SET_ATTITUDE_TARGET.

Тепер бортовий мікрокомп’ютер Raspberry Pi Zero 2 W отримує відеопотік із тепловізійної камери, що формує аналоговий сигнал, який надходить на пристрій через аналогово-цифровий перетворювач.

На наступному етапі виконується попередня обробка зображення, яка включає: фільтрацію шуму (зокрема за допомогою гаусового фільтра), нормалізацію рівнів яскравості та температурних значень у межах кадру для підвищення контрастності.

Після цього застосовується сегментація об'єктів за температурною інтенсивністю, яка дає змогу виділити області з аномальною тепловою активністю, зокрема для виокремлення контурів використовується порогова обробка.

На основі результатів сегментації система виконує вибір основної цілі, яка характеризується як найбільша або найбільш інтенсивна температурна аномалія у полі зору камери.

Далі обчислюється зсув цілі відносно центру кадру в пікселях по горизонталі Δx та по вертикалі Δy, на його основі розраховується кутовий зсув з урахуванням FOV камери. Даний зсув перетворюється у кут Ейлера, а потім у кватерніон-корекцію.

Через MAVLink зчитуються поточні значення орієнтації, тобто кватерніон та барометричної висоти. Формується новий цільовий кватерніон як добуток поточного на корекційний. Якщо висота зменшилась, то вносяться поправки до значення тяги.

Команда SET_ATTITUDE_TARGET з новими параметрами формується у форматі MAVLink та надсилається до польотного контролера. Контролер адаптує сигнали на ESC для стабілізації по крену, тангажу та тязі.

Потім система перевіряє, чи не надійшла команда завершити режим стабілізації STOP_SCRIPT, на цьому ітерація циклу завершується і якщо команди про припинення роботи не надходило, то ми повертаємось на початок циклу.

Окрім того, тепер ми не визначаємо ціль спочатку, а шукаємо на зображенні визначену раніше – здійснюємо трекінг стабільної цілі. Якщо все ж таки первинна

ціль втрачена, наприклад ціль вийшла за межі FOV і тепер необхідно перевизначити її, тоді запускається механізм повторного пошуку за алгоритмом описаним вище. При відновленні цілі даний цикл продовжується. А в разі ручного перехоплення управління (команда з пульта або failsafe), скрипт завершує роботу та повертає керування пілоту.

Графічне зображення алгоритму представлене на рисунку нижче.

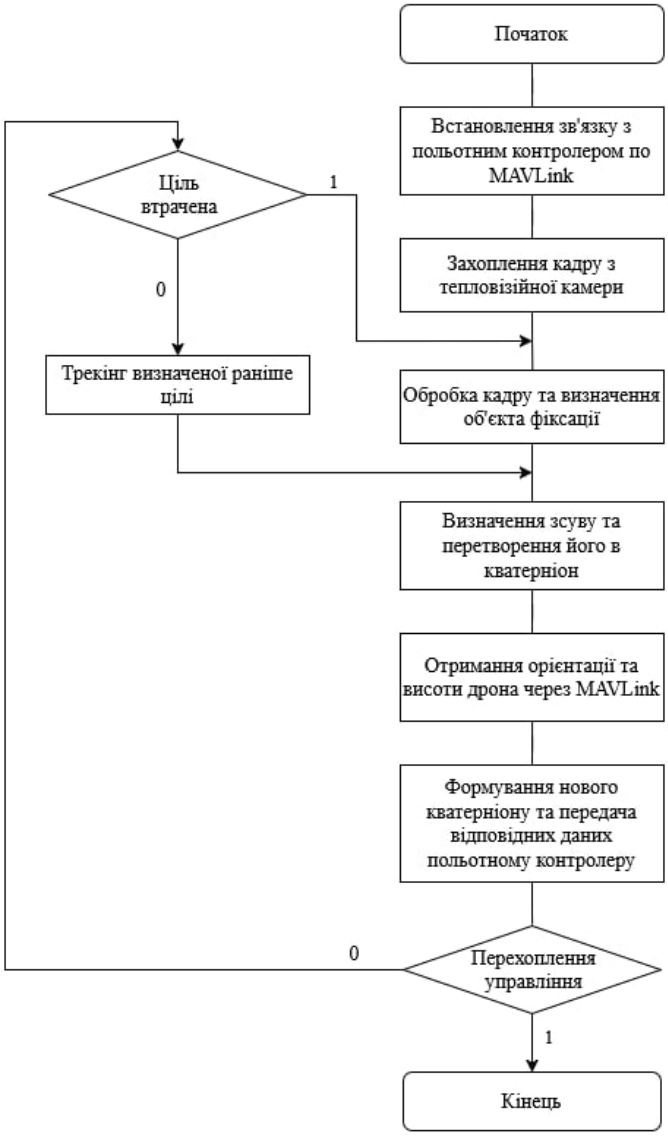


Рисунок 3.2 – Алгоритм роботи системи

Окрім того, обробка кадру в системі стабілізації дрона базується на двох ключових етапах. Перший етап полягає у визначенні цілі – об’єкта в кадрі, на якому

буде здійснюватися фіксація для подальшої стабілізації положення дрона. Другий етап – відстеження цієї раніше визначеної цілі протягом часу, що дозволяє підтримувати стабільне положення дрона відносно об’єкта навіть при його русі або зміні положення камери.

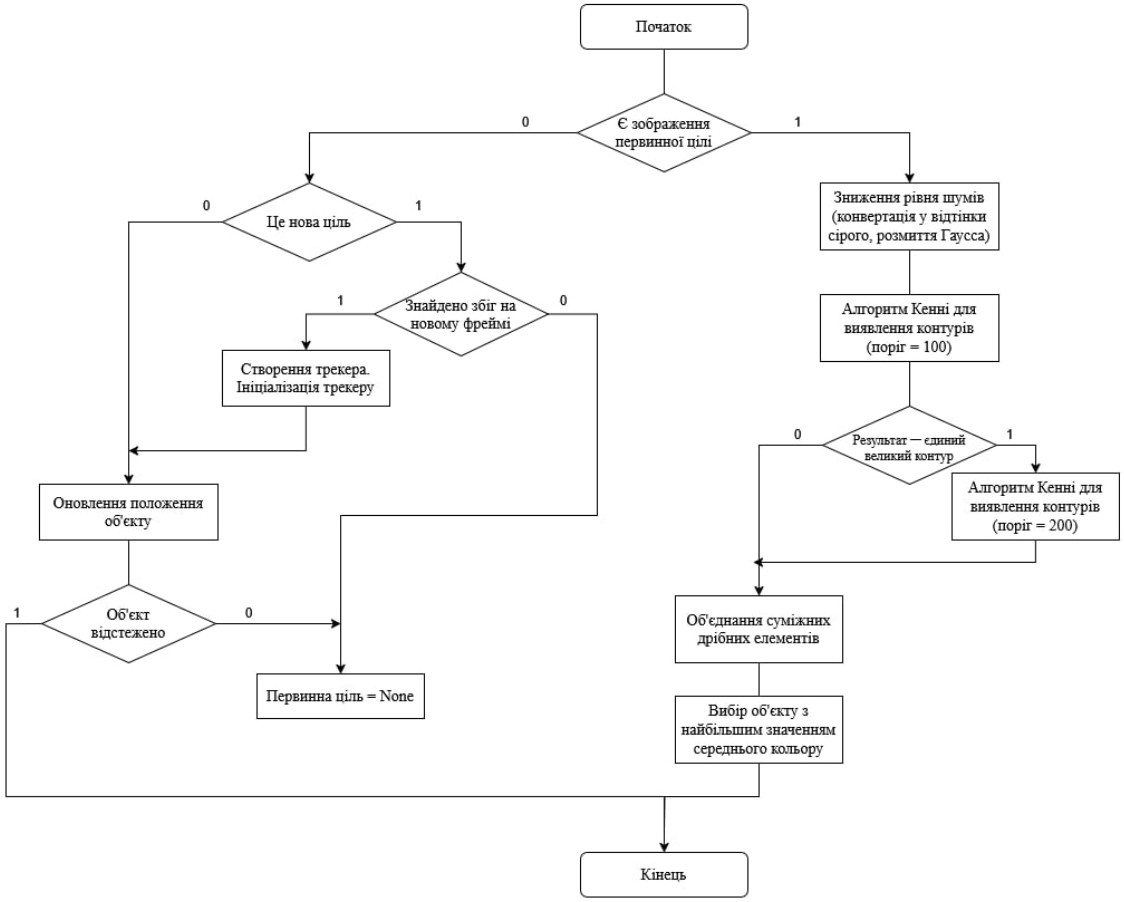


Рисунок 3.3 – Алгоритм обробки зображення

У процесі визначення первинної цілі обробка вхідного зображення здійснюється за кілька послідовних етапів. На першому етапі проводиться попередня обробка зображення з метою зниження рівня шумів. Для цього зображення конвертується у відтінки сірого, після чого застосовується гауссове розмивання, що дозволяє зменшити вплив неінформативних пікселів та локальних флуктуацій яскравості.

Другим етапом є виявлення контурів об’єктів за допомогою алгоритму Кенні. Через змінність зовнішніх умов, зокрема типу місцевості та добових коливань

температури, алгоритм використовує два значення порогу. Початково застосовується нижчий поріг для зменшення навантаження системи, бо якщо одразу взяти вищий поріг, а зображення буде достатньо контрастним, то на виході отримаємо замість визначення декількох масивних об’єктів, багато дрібних. І навпаки, якщо зображення сповнене нюансів, порогове значення треба збільшити. Отож, якщо при використанні нижчого порогового значення результат містить декілька об’єктів, він вважається задовільним. Якщо ж виявлено лише один об’єкт та його розміри займають майже все зображення, поріг підвищується.

Третій етап спрямований на зменшення кількості цілей, де контури, що розташовані достатньо близько один до одного, об’єднуються, оскільки потенційно вважаються єдиним об’єктом. Для цього визначаються умовні кола навколо кожного контуру і якщо кола перетинаються, то контури вважаються такими, що належать до одного об’єкта. Це дозволяє об’єднати просторово суміжні елементи в один узагальнений об’єкт.

Фінальним етапом є вибір основної цілі шляхом аналізу середнього кольору кожного з виявлених об’єктів на зображенні. Для цього використовується функція cv2.mean(), яка дозволяє обчислити середнє значення інтенсивності по каналах BGR (синій, зелений, червоний) для області, що відповідає кожному об'єкту. Оскільки вхідне зображення є тепловим, компонент червоного каналу (R) умовно приймається за індикатор температури. Об’єкт з найвищим середнім значенням у червоному каналі визначається як пріоритетна ціль, оскільки він, ймовірно, є найтеплішим чи найближчим, а отже й найбільш придатним для подальшого трекінгу.

Якщо ж первинна ціль вже визначена і необхідно лише фіксуватись на ній, тоді програма виконує алгоритм дій відслідковування, що складається з наступних етапів.

Якщо це перша ітерація циклу після визначення цілі, тоді спочатку застосовується cv.matchTemplate з метрикою TM_SQDIFF_NORMED для грубого визначення позиції шаблону. Потім із цієї області вирізається фрагмент, який

порівнюється з оригінальним шаблоном за допомогою SIFT: обчислюються дескриптори ключових точок і фільтруються збіги через BFMatcher та правило Lowe. Якщо знайдено достатню кількість «хороших» збігів, функція повертає координати; інакше – None, і програма шукає нову ціль.

Отож, якщо збіг знайдено, повертаємо координати відповідної області, яка описує положення об'єкта на кадрі. Для подальшого відстеження об'єкта створюємо трекер CSRT (Channel and Spatial Reliability Tracker) – це сучасний алгоритм, реалізований в OpenCV, який використовує просторову та канальну надійність для точного відстеження. Ініціалізуємо трекер на початковому кадрі, передаючи йому отримані координати об'єкта.

Надалі для оновлення положення об'єкта у новому кадрі використовується метод update трекера. Він приймає поточний кадр відео і повертає два значення: логічне значення, яке вказує, чи вдалося успішно відстежити об'єкт, та координати оновленої області об'єкта. Відповідно, якщо об’єкт було знайдено, виконуються подальші обчислення зміщення, які наведені в алгоритмі на рисунку 3.2. У випадку, коли функція повертає хибне значення вважається, що ціль втрачено, і запускається процес повторного вибору первинної цілі, описаний у наведеному вище алгоритмі.

3.3 Реалізація системи стабілізації для дрона-ретранслятора на основі обробки зображення

Зазначена у попередніх пунктах даного розділу архітектура є відкритою до масштабування та підтримує реалізацію як у фізичному середовищі, так і в симуляційному середовищі ArduPilot SITL.

Отож, для фізичної реалізації необхідно здійснити первинні налаштування польотного контролеру та бортового комп’ютеру, аби при вмиканні пілотом відповідного тумблера на пульті керування запускався алгоритм стабілізації та навпаки при вимиканні – контроль управління перехоплювався пілотом.

Спершу налаштуємо пульт керування (RC-передавач), для цього оберемо будь-який вільний тумблер, до прикладу SD, він буде виступати у ролі перемикача увімк/вимк для запуску основного скрипта. Виконаємо прив’язку даного тумблера до відповідного каналу, наприклад CH7, для цього перейдемо в меню налаштувань та вкажемо, що канал відображає приблизно 1000 при вимкненому положенні і 2000 при увімкненому, аби стани увімкнути/вимкнути чітко відрізнялись, щоб контролер міг правильно розпізнавати перемикання.

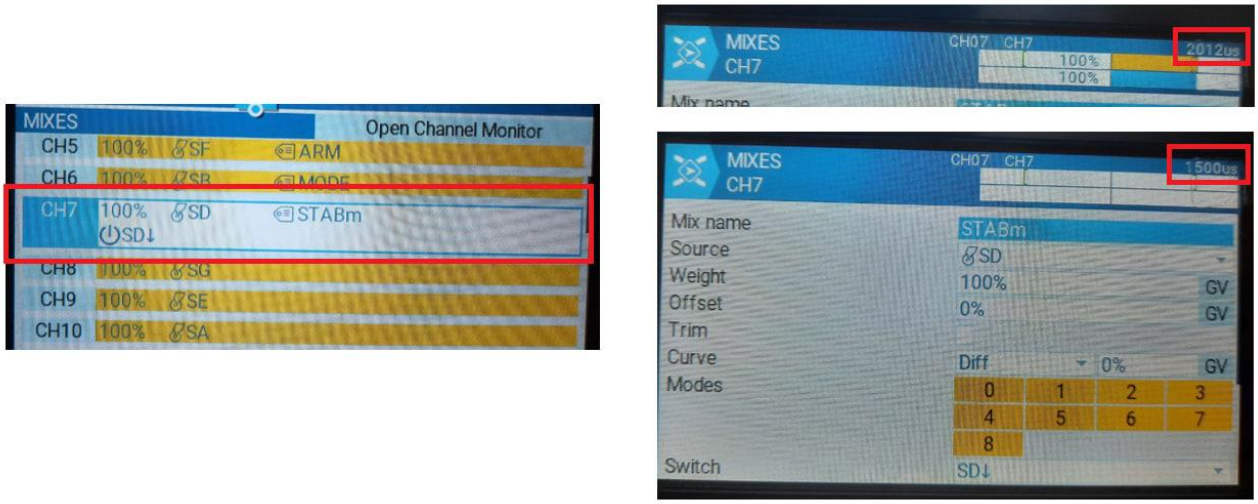


Рисунок 3.4 – Налаштування пульта керування

Наступним етапом є обробка сигналу польотним контролером. Налаштування польотного контролера Matek F765-WSE полягає у забезпеченні коректної взаємодії між RC-каналом, який отримує інформацію від пульта керування, та інтерфейсом UART для передачі команд на зовнішній пристрій – Raspberry Pi.

Для цього спочатку необхідно підтвердити наявність та правильне підключення каналу RC, що відповідає за тумблер на пульті (у нашому випадку це канал CH7). Контролер через Betaflight Configurator дозволяє контролювати параметри кожного RC-каналу, отримуючи сигнали PWM у діапазоні приблизно від 1000 до 2000, які інтерпретуються як логічні стани "вимкнено" та "увімкнено". Така двопозиційна логіка забезпечує просту та надійну активацію команд.

Далі налаштовується UART порт, наприклад UART2, який фізично з’єднується із приймачем Raspberry Pi. У Betaflight Configurator на вкладці "Ports" для відповідного UART включається функція Serial TX, що дозволяє польотному контролеру передавати дані у послідовному режимі. Встановлення правильного порту є критичним для успішного обміну командами.

Для реалізації логіки передачі команд використовується Lua-скрипт, завантажений у контролер. Даний скрипт періодично (з інтервалом 500 мс) опитує значення PWM на каналі 7 за допомогою функції rc:get_pwm(rc_channel). При переході стану до вищого за 1800 він відправляє команду "START_SCRIPT", сигналізуючи про активацію, а при переході до нижче за 1200 – команду "STOP_SCRIPT", що означає деактивацію. Окрім того, змінна last_state зберігає попередній стан, щоб уникнути повторної відправки команд без як такої зміни положення тумблера.

Команди передаються через відкритий UART порт, знайдений функцією serial:find_serial(2), що відповідає UART2. Дана реалізація дозволяє побудувати простий і надійний протокол керування зовнішнім пристроєм за допомогою RC-пульта, а використання Lua-скрипта безпосередньо на польотному контролері мінімізує затримки та ризики втрати сигналу між каналом керування та зовнішнім пристроєм, забезпечуючи інтегровану систему командного управління.

Окрім того, слід зазначити, що для досягнення повноцінної комунікації між польотним контролером і Raspberry Pi використовуються два різні UART-порти з різною функціональністю. Перший порт SERIAL2 призначений для передачі простих текстових команд від Lua-скрипта контролера до Raspberry Pi, що забезпечує швидке та пряме керування запуском чи зупинкою скриптів. Коли другий порт SERIAL1 використовується для обміну телеметричними даними та командами високого рівня через протокол MAVLink. Цей поділ дозволяє розділити канали командного керування та телеметрії, що підвищує надійність і гнучкість системи.

Останній етап – це обробка команди на Raspberry Pi та запуск по ній основного скрипта. Налаштування Raspberry Pi для роботи з польотним контролером Matek F765-WSE полягає у підготовці апаратних інтерфейсів і налаштуванні програмного забезпечення для прийому та обробки команд і телеметрії через UART-порти.

Спочатку потрібно звільнити послідовні порти Raspberry Pi від системних служб, які за замовчуванням можуть їх займати. Для цього в налаштуваннях Raspberry Pi у файлі config.txt відключимо серійну консоль і Bluetooth, щоб UART-порти стали доступними для власного використання. Це важливо, щоб уникнути конфліктів під час передачі даних з польотним контролером.

Далі фізично під’єднаємо GPIO-піни Raspberry Pi, що відповідають UART (TX, RX, GND), до відповідних UART-портів польотного контролера. Як вже зазначалось раніше, UART2 контролера, через який йдуть прості текстові команди, підключаємо до UART Raspberry Pi, що буде читати ці команди. UART1 використовуємо для MAVLink, тут Raspberry Pi приймає і відправляє телеметрію і команди через спеціальний протокол.

Для уникнення перевантаження системи та забезпечення ефективної роботи, логічно розділити функціонал на два скрипти: один – основний, що реалізує керування дроном, інший – «слухач», який постійно працює у фоновому режимі і очікує на сигнал запуску чи зупинки основного скрипту, він відкриває послідовний порт /dev/serial0 зі швидкістю 57600, постійно читає вхідні дані і реагує на команди "START_SCRIPT" та "STOP_SCRIPT". При отриманні команди запуску – запускає основний скрипт керування дроном, а при команді зупинки – припиняє його роботу.

Для автоматичного запуску цього «слухача» після завантаження системи створюється systemd-сервіс. Це гарантує безперервну роботу скрипта і своєчасне реагування на команди від польотного контролера без участі користувача. Файл сервісу містить налаштування для автозапуску, відновлення роботи у разі збою та логування.

Після створення та активації systemd-сервісу забезпечується постійний моніторинг порту і миттєвий запуск або зупинка основного процесу керування дроном. Завдяки цьому підвищується стабільність системи, розвантажується центральний скрипт від непотрібного очікування та створюється надійний канал комунікації між Raspberry Pi та польотним контролером.

Архітектура проєкту передбачає повну підтримку реалізації у симуляційному середовищі ArduPilot SITL, що є критично важливим з огляду на характер системи. Оскільки управління дроном, зокрема восьмидюймовою платформою з відповідною тягою та масою, повністю передається програмному забезпеченню, будь-яка помилка в алгоритмах або логіці керування може призвести до неконтрольованої поведінки апарата в реальних умовах. Це не лише створює ризик фізичного пошкодження компонентів, що є фінансово затратним, а й несе потенційну небезпеку для людей та середовища. Тому реалізація системи у симуляторі на ранньому етапі розробки є не просто зручним методом тестування, а необхідним кроком з погляду безпеки, практичності та економії ресурсів. ArduPilot SITL дозволяє перевірити роботу алгоритмів стабілізації, комунікації між модулями та реакцію на зовнішні збурення без фізичного втручання, що значно пришвидшує процес відлагодження системи і знижує загальний ризик на етапі переходу до реального апаратного середовища.

Отож, на поточному етапі розробки проєкт доцільно розділити на дві незалежні, але взаємопов’язані гілки: перша – це відстеження об'єкта за допомогою алгоритмів комп’ютерного зору, що відповідає за виявлення та позиціювання цілі у відеокадрі; друга – реалізація відповідного керування польотною платформою в симуляційному середовищі ArduPilot SITL на основі даних, отриманих з модуля відстеження. Такий поділ дозволяє паралельно розробляти та відлагоджувати ключові компоненти системи, зберігаючи чітку межу між обробкою зображення та реактивним управлінням дроном, що спрощує інтеграцію та тестування у подальшому.

Отож, аби перейти до реалізації керування дроном через комп'ютерне бачення, необхідно встановити базове програмне забезпечення зокрема, як вже зазначалось раніше, ArduPilot як основну платформу для симуляції та контролю польоту, а також Mission Planner як зручний графічний інтерфейс для моніторингу та первинного налаштування системи.

Першим кроком є встановлення ArduPilot у режимі SITL (Software-In-The-Loop), що дозволяє емулювати роботу польотного контролера без фізичного обладнання. Це відкриває можливість тестування команд, алгоритмів стабілізації та взаємодії з системою у безпечному програмному середовищі.

Наступним етапом є встановлення Mission Planner для візуалізації польотних параметрів у режимі реального часу. Він надає змогу спостерігати за орієнтацією дрона у 2D-проекції, контролювати статус системи, перемикати режими польоту, а також відображати телеметричні дані, що надходять через MAVLink.

Після встановлення ArduPilot SITL і Mission Planner виконаємо первинне налаштування симуляційного середовища, необхідне для подальшої роботи з системою керування дроном. На першому етапі в Mission Planner обираємо модель літального апарата – у нашому випадку це мультикоптер, зокрема квадрокоптер, що відповідає архітектурі платформи, яка буде використана у реальному експерименті. Далі налаштовуємо порти телеметрії, які реалізуються через віртуальні або фізичні UART-інтерфейси. Один з них призначається для обміну по MAVLink-протоколу з Raspberry Pi (SERIAL1), інший для передачі простих текстових команд з LUA-скрипта до Pi (SERIAL2).

Окремо виконується прив'язка каналів керування з RC-передавача, в Mission Planner конфігурується окремий тумблер на канал CH7, який у подальшому використовуватиметься для активації або зупинки алгоритму стабілізації. Також важливим кроком є активація підтримки скриптів LUA на польотному контролері: для цього в параметрах ArduPilot встановимо відповідні значення SCR_ENABLE = 1 та SCRIPTING_NAME, а також задамо номер порту для SERIALx_PROTOCOL = 28, що відповідає функції scripting.

Реалізація описаного в попередньому пункті алгоритму обробки зображення (рисунок 3.3) представлена на рисунках 3.5-3.7 нижче.

Рисунок 3.5 – Виявлення первинної цілі

Спершу виявляємо первинну ціль, як найтепліший об’єкт та зберігаємо зображення для створення подального відстеження його на наступний кадрах.

Рисунок 3.6 – Трекінг раніше визначеної цілі

Після виявлення первинної цілі на наступному кадрі, створюємо на її основі трекінг для подальшого відстеження.

Рисунок 3.7 – Виявлення цілі після втрати первинної

Якщо ж ціль була втрачена, то ми повертаємось на початок алгоритму та визначаємо нову ціль за первинну.

Тепер запустимо весь цикл роботи системи у ArduPilot Mission Planner: від виявлення цілі на зображенні до формування та передавання команд керування дроном у реальному часі.

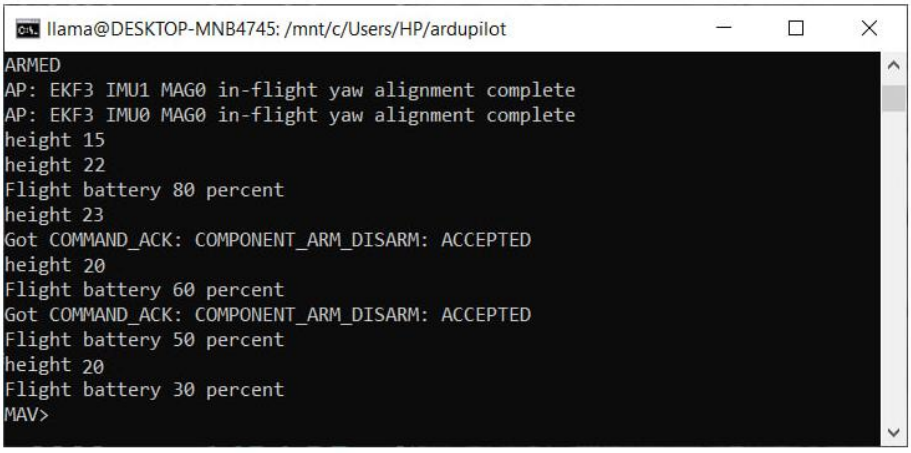


Рисунок 3.8 – Лог повідомлень ArduPilot

Як бачимо, швидкість та висота дрона залишаються стабільними, а сам апарат слідує за визначеною ціллю на зображенні. Для перевірки працездатності

системи була обрана рухома ціль, оскільки якщо дрон у симуляції здатен слідувати за об’єктом, що змінює своє положення, то в реальних умовах, коли ціль є статичною, але на дрон діють зовнішні чинники, наприклад, вітер, система забезпечить утримання положення апарата у фіксованій точці простору.

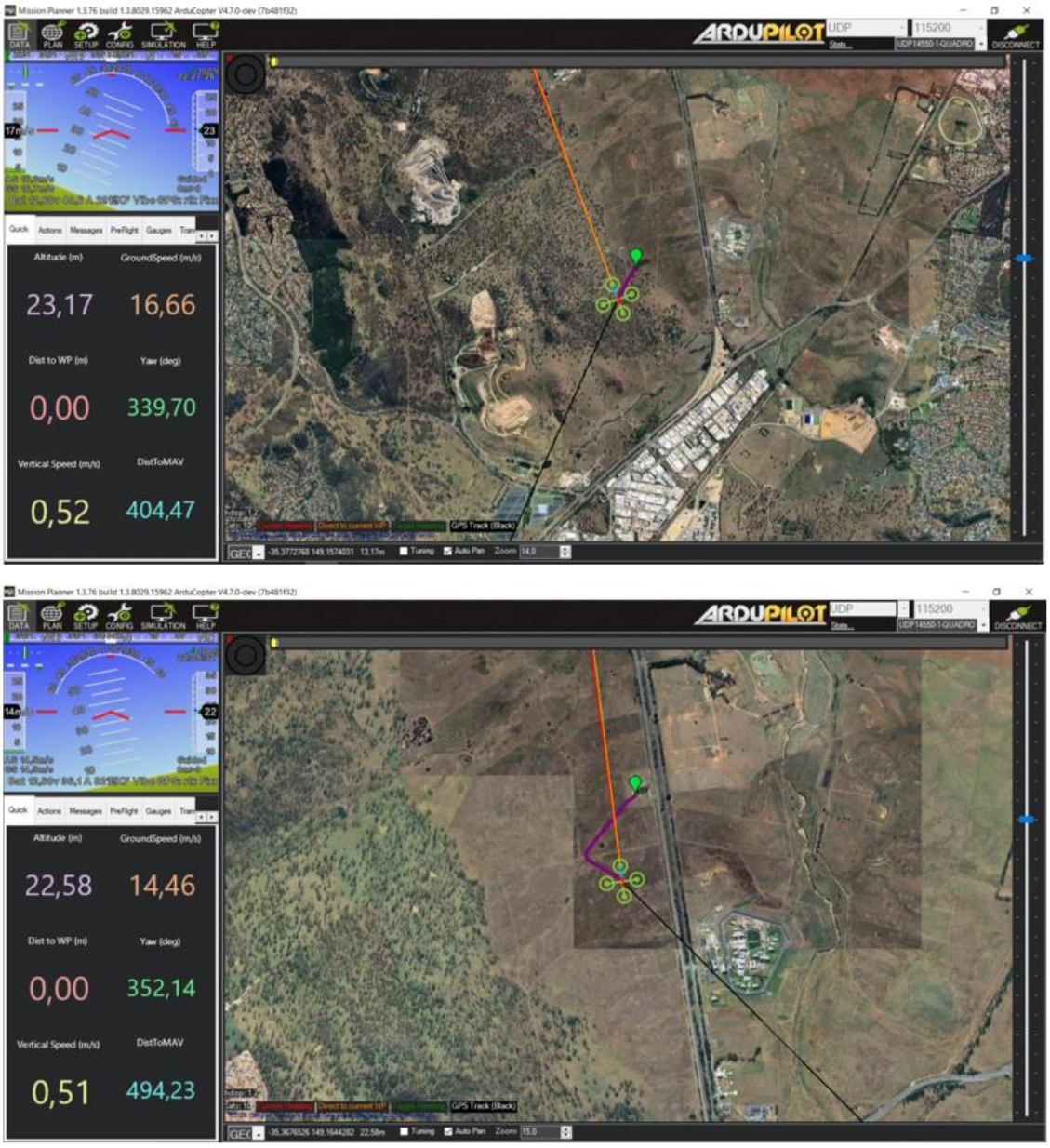


Рисунок 3.9 – Результат польоту у Mission Planner

3.4 Висновки до третього розділу

У даному розділі було детально розглянуто принцип роботи системи стабілізації дрона на основі візуального відстеження, описано загальну логіку взаємодії між апаратними та програмними компонентами, а також сформовано структурну схему проєкту.

Окрім теоретичного викладення, представлено практичну реалізацію ключових елементів системи: від налаштування RC-передавача, конфігурації параметрів польотного контролера та інтеграції з Raspberry Pi до реалізації скриптів, які забезпечують передачу керуючих команд. Зокрема, використання протоколу MAVLink через UART-інтерфейс та застосування LUA-скриптів на стороні польотного контролера дозволяє реалізувати стабільний двосторонній канал комунікації. Це забезпечує можливість запуску алгоритмів з обчислювального модуля в режимі реального часу з пульта керування автоматично під час польоту.

Також виконано тестування розробленої системи у симуляційному середовищі Mission Planner з використанням програмного автопілота SITL. Під час запуску перевірено повний цикл функціонування: від обробки відеосигналу та визначення координат цілі до формування та передачі команд типу SET_ATTITUDE_TARGET. В ході тестування спостерігалося стабільне підтримання висоти та швидкості, а зміни положення дрона відбувалися відповідно до зміщення об’єкта на зображенні. Це підтверджує коректну роботу реалізованого PID-регулятора по крену, тангажу та висоті, а також ефективну взаємодію між візуальним модулем і системою керування польотом.

ВИСНОВКИ

У даній роботі, спираючись на проведені теоретичні аналізи та практичні експерименти, була створена комплексна система стабілізації FPV-дрона-ретранслятора, що базується на обробці зображення, отриманого з термальної камери. Розроблена система забезпечує автономне підтримання стабільного положення безпілотного літального апарата за допомогою візуального трекінгу теплового зображення.

У першому розділі проєкту було здійснено комплексний аналіз сучасних підходів до стабілізації FPV-дронів, що дозволило обґрунтувати вибір найбільш ефективних технологій для реалізації даної системи. Сформовано модель апаратної частини, до якої входить бортовий комп'ютер, що відповідає за обробку зображень у реальному часі, польотний контролер для стабілізації польоту та обробки даних з інерційних сенсорів, а також тепловізійна камера як основний сенсор. Розглянуто програмну складову, зокрема можливості бібліотеки OpenCV, що забезпечує попередню обробку відеосигналу, фільтрацію шуму, виявлення цільових об'єктів і реалізацію алгоритмів оптичного потоку для оцнки переміщення FPV-дрона у просторі. Для реалізації комунікації між обчислювальними та керувальними модулями обґрунтовано використання протоколу MAVLink, який забезпечує високу надійність передавання даних, гнучкість і широку підтримку в середовищах безпілотних систем.

У другому розділі було проведено техніко-технологічне обґрунтування вибору апаратного та програмного середовища для реалізації проєкту. До складу апаратної частини увійшов одноплатний мікрокомп'ютер Raspberry Pi Zero 2 W, який виконує функції локального обчислювального вузла: саме на ньому реалізується програмна логіка обробки відео, виявлення цілей та генерації керуючих команд. Як сенсор обрано тепловізійну камеру Foxeer Cat 3 Mini, яка передає аналоговий відеосигнал у режимі реального часу, забезпечуючи роботу системи навіть за умов поганої видимості. Польотний контролер Matek F765-WSE

відповідає за стабілізацію дрона, обробку даних із сенсорів та прийом команд від Raspberry Pi. Також визначено перелік функційних та нефункційних вимог до системи. Основною мовою програмування для реалізації всіх алгоритмів обрано Python, а для попереднього тестування системи обрано симуляційне середовище ArduPilot SITL, яке дозволяє перевірити функціональність без ризику для фізичної апаратури.

У третьому розділі описано принцип роботи системи стабілізації дрона на основі обробки зображення, логіку взаємодії апаратних і програмних компонентів та структурну схему проєкту. Розглянуто практичну реалізацію: налаштування RC-передавача, конфігурацію польотного контролера, інтеграцію з Raspberry Pi і скрипти для передачі команд через MAVLink. Проведено тестування у симуляторі Mission Planner з автопілотом SITL, що підтвердило стабільне керування дроном та коректність роботи PID-регулятора і взаємодії між візуальним модулем і системою управління польотом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. FPV drone (first-person view drone). URL: https://www.techtarget.com-/whatis/definition/FPV-drone-first-person-view-drone (дата звернення: 10.03.2025).
2. Oscar Liang. What is an FPV Drone?. URL: https://oscarliang.com/fpv-drone-guide/#What-Is-An-FPV-Drone (дата звернення: 10.03.2025).
3. Resource info: Relaying. URL: https://rubyfpv.com/resource-info-relaying.php (дата звернення: 10.03.2025).
4. DJI Matrice 4T Quadrotor. URL: https://dji-kyiv.com/kvadrokopter-dji-matrice-4t-cp.en.00000546.02/ (дата звернення: 10.03.2025).
5. Oscar Liang. How to build an FPV drone. URL: https://oscarliang.com/how-to-build-fpv-drone/ (дата звернення: 10.03.2025).
6. Галицький О., Денисюк Д., Кожемяко Я., Квассай М. Метод стабілізації FPV-дрону за автоматично визначеною ціллю та подальше її спостереження // Computer Systems and Information Technologies. 2025. № 1. С. 36–41. https://doi.org/10.31891/csit-2025-1-4.
7. Flight controller. URL: https://oscarliang.com/flight-controller/ (дата звернення: 20.04.2025).
8. MAVLink Protocol. URL: https://mavlink.io/en/ (дата звернення: 20.04.2025).
9. Hohpe G., Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Pearson Education, 2012. 106 с. ISBN 978-0-133-06510-7.
10. Point-to-point. URL: https://en.wikipedia.org/wiki/Point-to-point (дата звернення: 20.04.2025).
11. About OpenCV. URL: https://opencv.org/about/ (дата звернення: 20.04.2025).
12. Python Tutorial. URL: https://docs.python.org/uk/3.13/tutorial/index.html (дата звернення: 20.04.2025).

13. ArduPilot Simulation. URL: https://ardupilot.org/dev/docs/simulation-2.html (дата звернення: 20.04.2025).

14. What Are Analog and Digital Signals. URL: https://www.tutoroot.com/blog/what-are-analog-and-digital-signals-definition-difference-examples/ (дата звернення: 22.04.2025).

15. Foxeer Mini CAT 3 1200TVL FPV Camera. URL: https://www.foxeer.com/foxeer-mini-cat-3-1200tvl-0-00001lux-starlight-fpv-camera-g-320 (дата звернення: 25.04.2025).

16. Одноплатний комп'ютер. URL: https://artline.ua/uk/blogs/odnoplatnyy-kompyuter-miniatyurnyy-gigant-tekhnologiy/amp (дата звернення: 25.04.2025).

17. Arduino Introduction Guide. URL: https://www.arduino.cc/en/Guide/Introduction/ (дата звернення: 25.04.2025).

18. Orange Pi Official Site. URL: http://www.orangepi.org/ (дата звернення: 25.04.2025).

19. Raspberry Pi Boards Overview. URL: https://itmaster.biz.ua/directory/kits-nabory/raspberry-pi-boards.html (дата звернення: 25.04.2025).

20. Raspberry Pi Zero Microcomputer. URL: https://networkdiscount.com.ua/ua-/p2013954706-mikrokompyuter-raspberry-zero.html?srsltid=AfmBOoox_aNZtmTEAQBydvenhMk1bp21IucicCmyclr3sEjgC2VQ7_iP (дата звернення: 25.04.2025).

21. Raspberry Pi Zero 2 W Microcomputer. URL: https://evo.net.ua/mikrokomputer-raspberry-pi-zero-2-w/?srsltid=AfmBOoqT10SLPiqiiaCt91yTRPyn2hFP90KCWuRJk17O9UqYc3P4rQ9Q (дата звернення: 25.04.2025).

22. Flight Controller for Drone. URL: https://onefpv.com/blog/flight-controller-drone?srsltid=AfmBOorMF_RXOtUN4jdF1RK8Ai8QevJfQKsrbBI5nmuq4U6GHGnFa26V (дата звернення: 25.04.2025).

23. Matek Systems Official Website. URL: https://www.mateksys.com/ (дата звернення: 25.04.2025).

24. Matek F765-WSE Manual. URL: https://www.mateksys.com-/downloads/Manual/F765-WSE_Manual.pdf (дата звернення: 25.04.2025).

25. Functional vs Non-functional Requirements. URL: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/ (дата звернення: 25.04.2025).

26. Why Linux Is the Better Choice for Developers and Programmers. URL: https://mangohost.net/blog/why-linux-is-the-better-choice-for-developers-and-programmers/ (дата звернення: 25.04.2025).

27. pymavlink Developer Documentation. URL: https://www.ardusub.com-/developers/pymavlink.html (дата звернення: 25.04.2025).

28. OpenCV Python Tutorial. URL: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html (дата звернення: 25.04.2025).

29. PX4 Gazebo Simulation. URL: https://docs.px4.io/main/uk/sim_gazebo_gz/ (дата звернення: 25.04.2025).

30. PX4 JSBSim Simulation. URL: https://docs.px4.io/main/uk/sim_jsbsim/ (дата звернення: 25.04.2025).

31. PX4 FlightGear Simulation. URL: https://docs.px4.io/main/uk/sim_flightgear/ (дата звернення: 25.04.2025).

32. Berthold K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. URL: http:/www.cs.ucr.edu/~vbz/resources/quatut.pdf (дата звернення: 10.05.2025).

33. Grubin C. Derivation of the quaternion scheme via the Euler axis and angle. *Journal of Spacecraft and Rockets*. 1970. Vol 7 No 10. Pp 1261–1263. DOI 10.2514/3.30149.

34. HowStuffWorks. Gimbal: How Gimbals Work. URL: https://web.archive.org/web/20210718072730/https:/science.howstuffworks.com/gimbal.htm (дата звернення: 10.05.2025).

35. Carino J., Abaunza H., Castillo P. Quadrotor quaternion control. 2015 International Conference on Unmanned Aircraft Systems *ICUAS*. *IEEE* 2015. DOI 10.1109/icuas.2015.7152367.

36. Zhang Y., Rudnicky A. A practical camera model for pose estimation in UAVs: Proceedings of the International Conference on Unmanned Aircraft Systems (*ICUAS*). 2018.

37. O'Dwyer A. Handbook of PI and PID Controller Tuning Rules. 3rd ed. London: Imperial College Press, 2009. ISBN 978-1-84816-242-6.

38. Carino J., Abaunza H., Castillo P. Quadrotor quaternion control: 2015 International Conference on Unmanned Aircraft Systems (*ICUAS). IEEE*, 2015. DOI: 10.1109/icuas.2015.7152367.

39. Communicating with Raspberry Pi via MAVLink – Dev documentation. URL: https://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html (дата звернення: 10.05.2025).

40. MAVLink Common Message Definitions. URL: https://mavlink.io/en/messages/common.html (дата звернення: 10.05.2025).

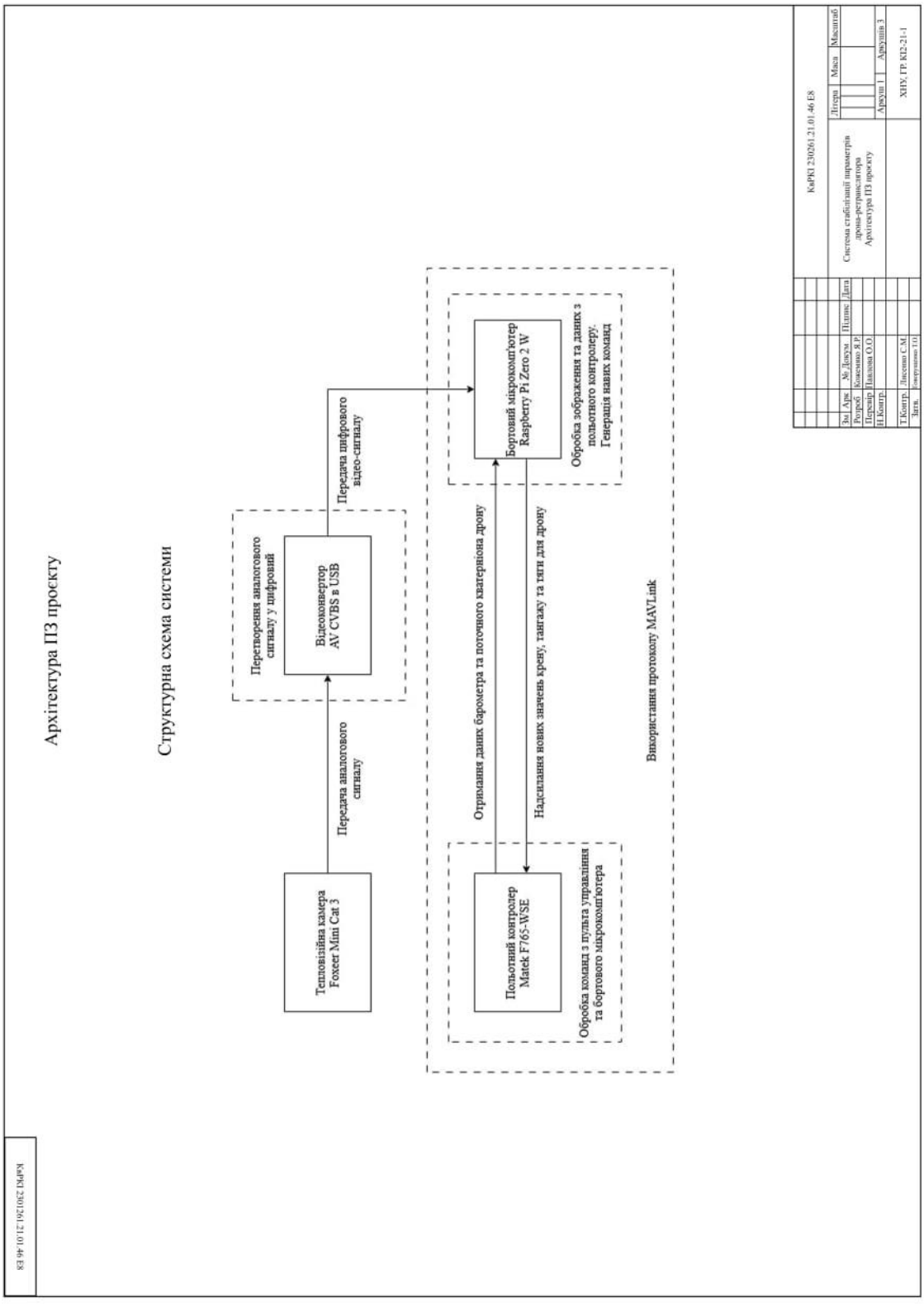
41. Singh J. B., Jaison B. Gyro-stabilized camera control in drones for military applications // IOP Conference Series: Materials Science and Engineering. 2021. Vol. 1012, No. 1, p. 012017. IOP Publishing.

42. UA DYNAMICS WARFARE TECHNOLOGIES. Теорія і практика застосування безпілотних літальних апаратів (дронів) / Посібник, створений ветеранами бойових дій, 2022. 125 с.

Додаток А

(обов’язковий)

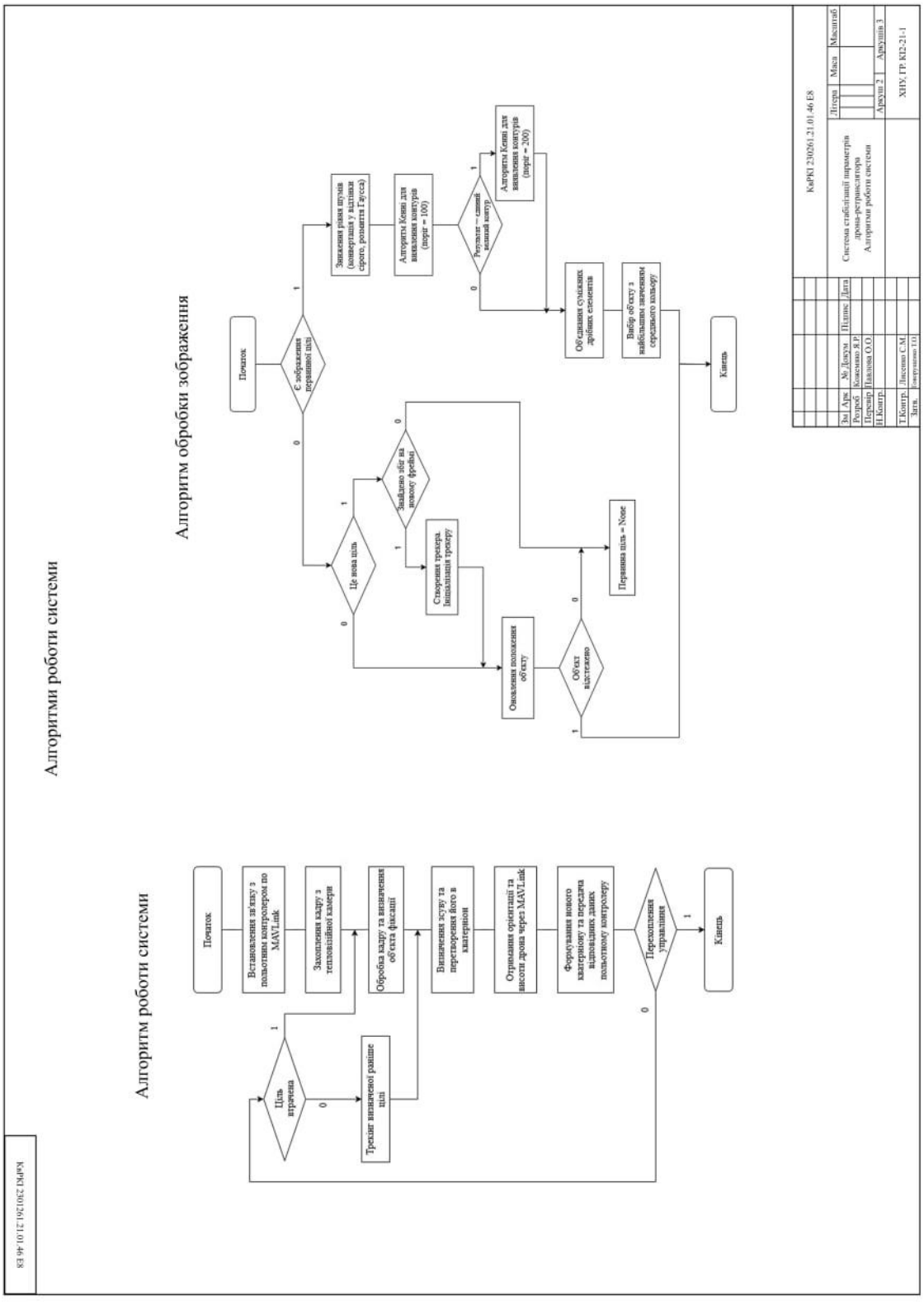
КОПІЯ КРЕСЛЕННЯ «АРХІТЕКТУРА ПЗ ПРОЄКТУ»



Додаток Б

(обов’язковий)

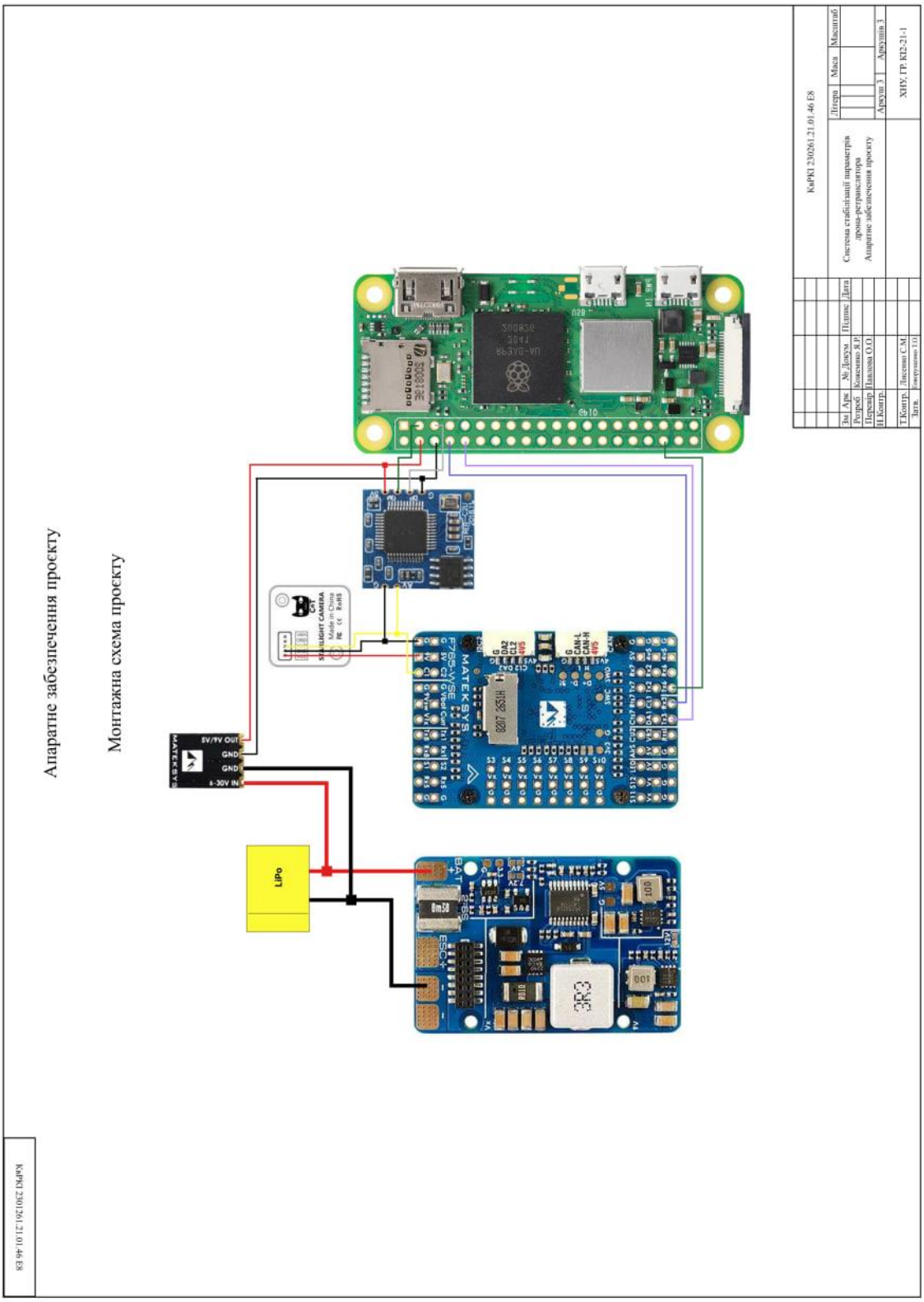
КОПІЯ КРЕСЛЕННЯ «АЛГОРИТМИ РОБОТИ СИСТЕМИ»



Додаток В

(обов’язковий)

КОПІЯ КРЕСЛЕННЯ «АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ПРОЄКТУ»



Додаток Г

ЛІСТИНГ ПРОЄКТУ

Лістинг canny_meth.py

```
import cv2 as cv
import numpy as np
import random as rng
import math

def first_circle_intersection(contours):
    point1, r1 = cv.minEnclosingCircle(contours[0])
    contours_to_extend = contours[0]
    contours_update = []

    for i in range(1, len(contours)):
        point2, r2 = cv.minEnclosingCircle(contours[i])
        distance_between_centers = math.sqrt((point2[0] -
point1[0])**2 + (point2[1] - point1[1])**2)
        if (distance_between_centers <= r1 + r2):
            contours_to_extend = np.vstack((contours_to_extend,
contours[i]))
            point1, r1 = cv.minEnclosingCircle(contours_to_extend)
        else:
            contours_update.append(contours[i])

    contours_update.append(contours_to_extend)
    return contours_update

def match_intersection_circles(contours):
    contours_update = contours
    for i in range(len(contours)):
        contours_update = first_circle_intersection(contours_update)
    return contours_update
```

```
def find_all_intersections(contours, len_prev):
    new_contours = match_intersection_circles(contours)
    iter = 0    while(True):
        if (len_prev == len(new_contours)):
            if (iter < len_prev):
                new_contours                                    =
match_intersection_circles(new_contours)
                iter += 1
            else:
                break
        else:
            len_prev = len(new_contours)
            iter = 0
            new_contours = match_intersection_circles(new_contours)

    return new_contours

def thresh_callback_variable(image, threshold):
    src_gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    src_gray = cv.blur(src_gray, (3,3))

    canny_output = cv.Canny(src_gray, threshold, threshold * 2)
    cv.imshow('Canny`s', canny_output)
    contours,   _   =   cv.findContours(canny_output,   cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)

    contours_poly = [None]*len(contours)
    for i, c in enumerate(contours):
        contours_poly[i] = cv.approxPolyDP(c, 3, True)

    drawing = image.copy()
    contours_processed    =    find_all_intersections(contours_poly,
len(contours_poly))
```

```
    return contours_processed

def thresh_callback(image):
    frame_height, frame_width = image.shape[:2]
    contours_processed = thresh_callback_variable(image, 100)

    if len(contours_processed) == 1:
        x, y, w, h = cv.boundingRect(contours_processed[0])
        height_ratio = h / frame_height
        width_ratio = w / frame_width
        print(height_ratio, width_ratio)

        if height_ratio >= 0.7 and width_ratio >= 0.7:
            print("t-200")
            return thresh_callback_variable(image, 200)

    print("t-100")
    return contours_processed
```

Лістинг detection_one_cam.py

```
import numpy as np
import imutils
import cv2
from math import sqrt
import numpy


# середнє значення кольору (теплоти) виділеного об'єкту
def object_color(img, c):
    x, y, w, h = cv2.boundingRect(c)
    cropped_img = img[y:y+h, x:x+w]
    avg_color = cv2.mean(cropped_img)[:3]

    return avg_color
```

```
# найтепліший
def find_the_brightest_obj(img, cnts):
    position = None
    obj_color = -1

    for i in range(len(cnts)):
        current_obj_color = sum(object_color(img, cnts[i]))     #(r +
g + b) / 3  або просто sum([r, g, b])

        if current_obj_color > obj_color:
            obj_color = current_obj_color
            position = i

    return position

# картинка найсвітлішого об'єкту
def detect(image, cnts):
    radius = 51
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gauss_gray = cv2.GaussianBlur(gray, (radius, radius), 0)
    maxLoc = cv2.minMaxLoc(gauss_gray)[3]  # find the brightest area
in the image
    position = find_the_brightest_obj(gauss_gray, cnts)

    return position
```

Лістинг template_match.py

```
import cv2 as cv
import numpy as np
from comparasion import images_have_matches

def find_match(image, aim_img):
    img = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    template = cv.cvtColor(aim_img, cv.COLOR_BGR2GRAY)
```

```
    w, h = template.shape[::-1]

    result = cv.matchTemplate(img,template,cv.TM_SQDIFF_NORMED)
    top_left                =                cv.minMaxLoc(result)[2]
# top_left = min_loc при SQDIFF
    aim_shifted           =           img[top_left[1]:top_left[1]+h,
top_left[0]:w+top_left[0]]     # crop image to get exectly aim
    is_match = images_have_matches(aim_img, aim_shifted)
    if is_match:
        return top_left[0], top_left[1], w, h
    else:
        return None
```

Лістинг template_match.py

```
import cv2 as cv
def images_have_matches(image1, image2):
    img1 = cv.cvtColor(image1,cv.IMREAD_GRAYSCALE)
    img2 = cv.cvtColor(image2,cv.IMREAD_GRAYSCALE)

    sift = cv.SIFT_create()
    des1 = sift.detectAndCompute(img1,None)[1]
    des2 = sift.detectAndCompute(img2,None)[1]

    if des1 is None or des2 is None:
        return False

    bf = cv.BFMatcher()
    matches = bf.knnMatch(des1,des2,k=2)

    good_matches = []
    for i, pair in enumerate(matches):
        try:
            m, n = pair
            if m.distance < 0.7*n.distance:
                good_matches.append(m)
```

```
            except ValueError:
                pass

        return len(good_matches) > 0
```

Лістинг angles_conversion.py

```
from pymavlink.quaternion import QuaternionBase
from pid_controller import PID
import math


MAX_ANGLE_RAD = math.radians(35)  # Максимальний нахил

def limit_angle(angle_rad):
    return max(-MAX_ANGLE_RAD, min(MAX_ANGLE_RAD, angle_rad))

def euler_to_quaternion(roll, pitch, yaw):
    cy = math.cos(yaw * 0.5)
    sy = math.sin(yaw * 0.5)
    cp = math.cos(pitch * 0.5)
    sp = math.sin(pitch * 0.5)
    cr = math.cos(roll * 0.5)
    sr = math.sin(roll * 0.5)

    w = cr * cp * cy + sr * sp * sy
    x = sr * cp * cy - cr * sp * sy
    y = cr * sp * cy + sr * cp * sy
    z = cr * cp * sy - sr * sp * cy
```

```
    return QuaternionBase([w, x, y, z])

def pixels_to_angles(delta_x, delta_y, width=480, height=304, fov_h_deg=90, fov_v_deg=60):
    fov_h = math.radians(fov_h_deg)
    fov_v = math.radians(fov_v_deg)

    angle_per_pixel_x = fov_h / width
    angle_per_pixel_y = fov_v / height

    delta_angle_x = delta_x * angle_per_pixel_x
    delta_angle_y = delta_y * angle_per_pixel_y

    roll = delta_angle_x
    pitch = -delta_angle_y  # інверсія тангажу

    return roll, pitch

def normalize_quaternion(q):
    norm = math.sqrt(sum(x*x for x in q))
    return [x / norm for x in q]

def calculate_new_quaternion(q_current, dx, dy, pid_roll, pid_pitch, dt):
    delta_roll, delta_pitch = pixels_to_angles(dx, dy)

    roll, pitch, yaw = q_current

    pid_roll.setpoint = limit_angle(roll + delta_roll)
    pid_pitch.setpoint = limit_angle(pitch + delta_pitch)
```

```
    smooth_roll = roll + pid_roll.update(roll, dt)
    smooth_pitch = pitch + pid_pitch.update(pitch, dt)

    q_target_normalized = normalize_quaternion(q_target.q)

    return q_target_normalized
```

Лістинг pid_controller.py

```
class PID:
    def __init__(self, kp, ki, kd, setpoint=0.0, integral_limit=None):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.setpoint = setpoint
        self.last_error = 0.0
        self.integral = 0.0
        self.integral_limit = integral_limit

    def update(self, measurement, dt):
        error = self.setpoint - measurement
        self.integral += error * dt
        if self.integral_limit is not None:
            self.integral = max(min(self.integral, self.integral_limit), -self.integral_limit)
        derivative = (error - self.last_error) / dt if dt > 0 else 0.0
        self.last_error = error
        return self.kp * error + self.ki * self.integral + self.kd * derivative
```

Лістинг mavlink_commands.py

```
from pymavlink import mavutil
from angles_conversion import euler_to_quaternion
import time
import math


def send_attitude(master, quat, thrust=0.6):
    type_mask = 0b00000111  # Ігнорувати body rates
    master.mav.set_attitude_target_send(
        int(time.monotonic() * 1000),
        master.target_system,
        master.target_component,
        type_mask,
        quat,
        0, 0, 0,
        thrust
    )


def receive_attitude(master):
    while True:
        msg = master.recv_match(type='ATTITUDE', blocking=True)
        if msg:
            roll = msg.roll
            pitch = msg.pitch
```

```
        yaw = msg.yaw
        # q = euler_to_quaternion(roll, pitch, yaw)
        return roll, pitch, yaw
```

Лістинг test.py

```
import cv2
import numpy as np
from canny_meth import thresh_callback
from detection_one_cam import detect
from template_match import find_match
from mavlink_commands import send_attitude, receive_attitude
from angles_conversion import calculate_new_quaternion
from pymavlink import mavutil
import time
import math
from pid_controller import PID


cap = cv2.VideoCapture(0)

pid_roll = PID(kp=1.5, ki=0.1, kd=0.5, integral_limit = 10)
pid_pitch = PID(kp=1.5, ki=0.1, kd=0.5, integral_limit = 10)
last_time = time.time()

aim = None
frame_match = None
match_counter = 1
```

```
match_result = None
center = (240, 152)

# ====================== CONNECTION ==========================
master = mavutil.mavlink_connection('udp:127.0.0.1:14550')
master.wait_heartbeat()
print("Heartbeat received")

mode = 'GUIDED'
mode_id = master.mode_mapping()[mode]
master.mav.set_mode_send(
    master.target_system,
    mavutil.mavlink.MAV_MODE_FLAG_CUSTOM_MODE_ENABLED,
    mode_id
)

while True:
    hb = master.recv_match(type='HEARTBEAT', blocking=True)
    if hb.custom_mode == mode_id:
        print("Режим GUIDED активовано")
        break

master.mav.command_long_send(
    master.target_system, master.target_component,
    mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM,
    0, 1, 0, 0, 0, 0, 0, 0
)
print("Армування дрона...")
master.motors_armed_wait()
```

```
print("Дрон збройований")
# ====================== CONNECTION ========================
while (cap.isOpened()):
    ret, frame = cap.read()

    if ret == True:
        # create-find aim
        if aim is None:
            contours_processed = thresh_callback(frame) # len > 0 == цілі існують (є виражені об'єкти)
            print("no target")
            if len(contours_processed):
                position = detect(frame, contours_processed)
                x1, y1, w1, h1 = cv2.boundingRect(contours_processed[position])
                aim = frame[y1:y1+h1, x1:w1+x1]
                cv2.rectangle(frame, (x1, y1), (x1 + w1, y1 + h1), (255, 0, 0), 2)
                match_counter = 1
                print("detected")

        # search existed aim
        else:
            if match_counter == 1:
                match_result = find_match(frame, aim)
                if match_result is None:
                    aim = None
                    continue
                tracker = cv2.legacy.TrackerCSRT_create()
                tracker.init(frame, match_result)
                match_counter += 1
```

```
            success, match_result = tracker.update(frame)
            if success:
                x, y, w, h = map(int, match_result)
                aim_center = (x+w//2, y+h//2)
                cv2.circle(frame, center, 2, (0,255,0), -1)
                cv2.circle(frame, aim_center, 2, (255,0,0), -1)
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

                shift_x, shift_y = (aim_center[0] - center[0], aim_center[1] - center[1])
                        quaternion_current = receive_attitude(master)
                current_time = time.time()
                dt = current_time - last_time
                last_time = current_time
                quaternion_target = calculate_new_quaternion(quaternion_current, shift_x,
shift_y, pid_roll, pid_pitch, dt)
                send_attitude(master, quaternion_target)
            else:
                aim = None
                print("lost")


        cv2.imshow('Frame', frame)
        if aim is not None:
            cv2.imwrite("image.jpg", frame)

        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
```

```
    else:
        break

cap.release()
```

Лістинг uart_listener.py

```
import serial, subprocess
ser = serial.Serial('/dev/serial0', 57600, timeout=1)
while True:
    line = ser.readline().decode('utf-8').strip()
    if line == "START_SCRIPT":
        subprocess.Popen(['python3', '/home/pi/fc_control.py'])
    elif line == "STOP_SCRIPT":
        subprocess.run(['pkill', '-f', 'fc_control.py'])
```

Протокол аналізу звіту подібності експертом

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

Автор: Ярослава КОЖЕМЯКО

Співавтор:

Назва: Кожемяко_Система стабілізації параметрів дрона-ретранслятора

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:2%

Коефіцієнт подібності 2:0.8%

Мікропробіли: 6

Заміна букв: 4

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-06 03:32:29.0

Після аналізу Звіту подібності констатую наступне:

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

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

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

Обгрунтування:

2025-06-06 Доцент Андрій Нічепорук

Дата експерт

Thu Jun 05 18:05:32 EEST 2025, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 13.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 12%**

ID: 243754 Title: БКР Система стабілізації параметрів дрона-ретранслятора Added in a DB: 2025-06-05 Authors: Ярослава КОЖЕМЯКО Heads: Ольга ПАВЛОВА Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	106172	685	14366 (14%)	120 (18%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes
240699	Title: Звіт з ПДП Система стабілізації для дрона-ретранслятора на основі обробки зображенн Added in a DB: 2025-05-01 Authors: Кожемяко Я.Р. Heads: Павлова О.О. Consultants: Opponents:	13544 (13.0%)	114 (17.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Кожемяко Ярослава Романівна

Тема: Система стабілізації параметрів дрона-ретранслятора

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 70

1. Короткий зміст роботи та прийнятих рішень: метою кваліфікаційної роботи є розробка системи стабілізації для дрона-ретранслятора на основі обробки зображення.

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

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: в першому розділі кваліфікаційної роботи проведено дослідження предметної області (проаналізовано існуючі системи стабілізації для дронів) та виконано постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено вибір компонентів та апаратно-програмне середовище для виконання завдання. В третьому розділі кваліфікаційної роботи виконано апаратну та програмну реалізацію для дрона-ретранслятора на основі обробки зображення.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи:

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

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

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Бурлаток Леонід Петрович, фіз-мат наук, професор, зав. каф. ІПЗ

"05" 06 2025 р. (підпис)

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Дениса ЛЮБОВЕЦЬКОГО
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-22-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та зберження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

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

05.06 2025 року

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

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система стабілізації параметрів дрона-ретранслятора

Автор: Ярослава КОЖЕМЯКО

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Ольга ПАВЛОВА, доктор філософії

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

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

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності чотирьохрозряних двійкових кодів, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
5) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 2% і адресується до 42 першоджерел; та системою Anti-Plagiarism складає 13%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Ольга ПАВЛОВА

Гарант ОП _____ Андрій НІЧЕПОРУК

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА