МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра комп'ютерних наук та прикладної математики

"До захисту допущена"
Зав. кафедри комп'ютерних наук та
прикладної математики
д.т н., проф. Ю.В. Турбал
«_____» __2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

Проєктування та розробка програмного забезпечення FPV дрону з машинним зором на автопілотуванні

Виконав: Сіньков Іван Олександрович
(прізвище, ім'я, по батькові) (підпис)

група ІПЗ-41

Керівник:
к.т.н., доцент, доцент кафедри Мічута О.Р.
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2025

Національний університет водного господарства та природокористування

(повне найменування вищого навчального закладу)

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра **комп'ютерних наук та прикладної математики**

Освітньо-кваліфікаційний рівень **бакалавр**

Галузь знань **12 Інформаційні технології**
(шифр і назва)

Спеціальність **121 Інженерія програмного забезпечення**
(шифр і назва)

Спеціалізація –

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
д.т н., проф.
Ю.В.Турбал
"1" жовтня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сінькову Івану Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *"Проєктування та розробка програмного забезпечення FPV дрону з машинним зором на автопілотуванні"* керівник роботи Мічута Ольга Романівна, *к.т.н., доцент, доцент кафедри комп'ютерних наук та прикладної математики.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

затверджені наказом по університету від *"18" квітня 2025 року С №-463.*

2. Термін подання роботи студентом *1 червня 2025 року.*

3. Вихідні дані до роботи: *технології які необхідні для створення програмного рішення, компоненти для створення дрону, мікроконтролер на базі Raspberry Pi та список основних завдань та вимог.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *створення прототипу дрону, розробка програмного забезпечення для розпізнавання об'єктів та автопілотування. Провести тестування*

виправлення проблем.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *Мультимедійна презентація*

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	*Мічута О.Р., доцент*		
Розділ 2	*Мічута О.Р., доцент*		
Розділ 3	*Мічута О.Р., доцент*		
Розділ 4	*Мічута О.Р., доцент*		

7. Дата видачі завдання *1 жовтня 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапі роботи	Примітка
1	*Детальне вивчення питання FPV*	*01.10.24-15.10.24*	*виконав*
2	*Вивчення та дослідження актуальності машинного зору*	*15.10.24-27.11.24*	*виконав*
3	*Конструювання FPV*	*27.11.24-15.12.24*	*виконав*
4	*Прошивання мікроконтролера*	*15.12.14-24.12.24*	*виконав*
5	*Створення програмного забезпечення для FPV*	*24.12.24-15.03.25*	*виконав*
6	*Загальні висновки по роботі*	*15.03.25-20.04.25*	*виконав*
7	*Підготовка звіту кваліфікаційної роботи*	*20.04.25-21.05.25*	*виконав*
8	*Підготовка презентації*	*21.05.25-01.06.25*	*виконав*

Студент _____ **Сіньков І.О.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Мічута О.Р.**
(підпис) (прізвище та ініціали)

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 46с., 19 рисунки, 9 джерел.

Мета роботи: створення FPV дрону з інтеграцією готової моделі машинного зору та підключення пристрою до протоколу зв'язку MavLink також під'єднати до мікроконтролера для надсилання команд для автопілотування та визначення об'єктів з виконанням заданого алгоритму.

Об'єкт дослідження: системи з машинним зором та готові апаратні рішення в поєднання з технічними пристроями.

Предмет дослідження: методи проектування штучного інтелекту його використання, проектування пристроїв на мікроконтролерах Raspberry Pi, створення подібного типу дронів та їх впровадження, комунікація по протоколу MavLink.

Проведено глибоке дослідження пристроїв на схожих системах визначено їх переваги та недоліки в ході чого було визначено найбільш оптимальних хід роботи, визначено найкращій тип прошивки дрону для його стабільної роботи та можливості взаємодії з протоколом MavLink, також досліджено питання найкращого мікроконтролера для управління дроном було взято Raspberry Pi оскільки на його основі можливо побудувати програмне забезпечення яке матиме змогу комунікувати з польотним контролером дрону та надавати команди для виконання.

Ключові слова: FPV, машинний зір, автопілотування, протокол MavLink, ArduPilot, Raspberry Pi.

ВСТУП

В сучасному світі дана тема є доволі актуальна тим паче в нашій країні особливо у військовій сфері діяльності, оскільки дана система дрону може використовуватися у різних ситуація, як у розвідувальних роботах використовуючи машинний зір для знаходження та розпізнавання замаскованої техніки. Також у виконанні ураження цілі використовуючи дрони типу FPV в даному випадку виконується визначення цілі її захоплення і автоматичний політ для ураження. Дана технологія дозволяє обходити системи протидії перехоплення дронів системами РЕБ противника.

Для того, щоб розробити дане рішення, необхідно мати доволі глибокі пізнання у технічній побудові дронів та їх налаштування. Крім цього необхідно володіти навичками програмування мікроконтролерів на базі Raspberry Pi, оскільки даний типі мікроконтролерів програмується на Python з використання специфічниз бібліотек. Дана робота має на меті створення фінальної версії прототипу FPV дрону з машинним зором на автопілотуванні. Після чого буде можливо вивести проект з етапу прототипу до повноцінної моделі доопрацювання необхідне в частини відео передавання сигналу на приймач, оскільки на даному етапі відео передається на оперативну систему Raspberry Pi.

Робота над проектом включає в себе аналіз новітніх рішень у сфері дроно-конструювання з використанням машинного зору або схожих систем. Також вивчення можливостей створення унікального програмного забезпечення на базі Raspberry Pi. Дана робота продемонструє покращення існуючих рішень і запропонувати своє рішення проблематики в даній сфері.

Актуальність теми зумовлена нинішніми реаліями, даний проект має місце у військовій сфері де матиме велике використання, а саме має вирішити проблему перехоплення дронів різноманітними система РЕБ ворога та більш точної роботи на фінальній стадії роботи оператора саме в ураженні цілей.

Метою розробки пристрою є створення готового прототипу, який може чітко визначати предмети, захоплювати його та виконувати рух до предмета який

захоплений, також можливість коригування автоматичного польому від зміни позиції перебування об’єкта.

Завданням роботи є створення програмного коду для FPV, який буде надавати команди дрону за допомогою мікроконтролера Raspberry Pi та протоколу комунікації MavLink, дана робота має вирішити проблему перехоплення дрону системами РЕБ ворога.

Об’єкт дослідження дрони даного типу з використання схожих технологій, також програмні продукти, які використовують машинний зір у різних сферах.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТЕХНІЧНИХ РІШЕНЬ У СФЕРІ ДРОНІВ ТА АНАЛІЗ ТЕХНОЛОГІЙ МАШИННОГО ЗОРУ, АВТОПІЛОТУВАННЯ В ПОЄДНАННІ З FPV

1.1. Огляд сучасного розвитку безпілотних літальних апаратів

Сучасний етап розвитку безпілотних систем характеризується стрімким зростанням інтересу до даних технологій та їх впровадження в різних галузях людської діяльності. Популярність даного напряму технологій зумовлено у їх універсальності, доступності та легкості у використанні, що в свою чергу підвищує ефективність і безпеки у багатьох сферах їх використання.

Однієї з основних перспектив розвитку їх використання може стати у сільськогосподорській сфері особливо це актуальне в аграрних країнах в яких посіви це валова частина у їх економіці. Завдяки можливості точно моніторингу посівів внесення, добрив, та захисту рослин від шкідників, дрони можуть сприяти оптимізації роботи сільськогосподарських угідь, що в свою чергу знизить витрати на обробку та догляд врожаю та також відвищить значно врожайність.

Не менш важлива роль використання дронів у військовій сфері. В даному випадку їх потенціал є практично необмежений, можливість використання від доставки вантажів у небезпечні ділянки, в які логістично важко дістатися та є небезпечно. Розвідувальні місії для визначення розташування ворожих позицій, техніки і тому подібне з використанням технологій машинного зору, та навіть до нанесення точних ударів з використанням систем автопілотування, що не унеможливить перехоплення апарату та гарантуватиме більший шанс влучності в ціль. Дана технологій має можливість суттєво вберегти життя людей та збільшити кількість безпечного ураження цілей з більшою точність та меншою вартістю.

Варто зазначити, що можливі приклади використання дронів не зупиняється на використанні в даних двох сферах дана технологія знаходить місце використання також:

- логістика та доставка: доставлення важливих вантажів у важкодоступні місця;
- кінематографії: аерозйомка для створення вражаючих візуальних ефектів які неможливо виконати в інший спосіб;
- пошуково-рятувальних операціях: для вчасного виявлення людей, які потрапили під стихійне лихо або були загублені у віддалених місцях;
- геодезії та картографії: створення високоточних моделей місцевості і рельєфу у форматі 3D.

Таким чином використання безпілотних систем можливо успішно інтегрувати в різні сфери діяльності людей, що в свою чергу убезпечить та оптимізує доволі багато процесів, які робить людина. Подальші дослідження та розробка машинного зору та автопілотування FPV або схожих систем дронів відкриває нові можливості для їх застосування у житті людей [7].

1.2. Теоретичні основи машинного зору в поєднанні з FPV

Інтеграція машинного зору в моделі дронів FPV є доволі складною задачею, яка передбачає провадження додаткової системи обробки зображення. Оскільки базова побудова дрону не передбачає можливості внесення додаткового програмного забезпечення в основну прошивку, причиною цього слугує обмежений ресурс більшості доступних польотних контролерів які орієнтовані на внесення базових або більш розширених прошивань. Вирішенням проблеми з машинним зором в дроні можливо вирішити за допомогою додаткових мікроконтролерів, які можливо інтегрувати в основу дрону прикладом такого контролера є Raspberry Pi. Оскільки даний мікроконтролер є доволі простим у використанні та має ресурс для виконання програмного забезпечення, його можливо фізично підключити до дрону та створи змогу

отримувати відео потік та робити обробку для визначення об’єктів за допомогою готової моделі машинного зору.

Основними задачами машинного зору в FPV, що використовується можна виділить розпізнавання об’єктів під час польоту, спрощена навігація оскільки в дронах типу FPV недоцільно використовувати GPS системи для орієнтування в просторі, оскільки це можна вважати їх вразливою частиною перед системами РЕБ противника. Натомість систему машинного зору в поєднанні з військовими системами планування та орієнтації на місцевості такі як “КРОПИВА” та “ДЕЛЬТА” можливо поєднати. Прикладом цього можливо використати в визначенні ключових об’єктів в місцевості, що дозволить визначити орієнтовне знаходження дрона та дозволить оператору орієнтуватися в подальшому на місцевості.

Основний алгоритм який можливо впровадити в FPV є метод глибокого навчання моделі для подальшої її використанні та обробки в програмному забезпеченні. Архітектури які найкраще підходять це YOLO, SSD, Faster R-CNN, ці моделі задатні обробляти і визначати об’єкти в реальному часі виявляти та класифікувати різні об’єкти на, які вони були натреновані від людей до рослин для тренування моделі необхідно мати певний датасет зображень на, які планується розробляти програму для визначення [2].

1.3. Технології автопілотування для FPV

Принцип побудови системи для автопілотування включає в себе основний аспект такий як вибір прошивки для дрону, оскільки більшість прошивок не мають можливості налаштування дрону для автопілотування вони розраховані лише на управління оператором. Можна виділити три основні прошивки для дрону BetaFlight, INAV, ArduPilot перші дві орієнтовані на управління лише оператором вони є значно легшими у налаштуванні і налагодженні їх здебільшого використовують для більшості дронів FPV. Третій тип прошивки є в декілька разів складніше для прошивання, оскільки має врази більше тонких

налаштувань [1]. Проте саме даний тип прошивки може надати змогу підключити систему для автопілотування за допомогою протоколу комунікації мікроконтролера з дроном MavLink, або навіть у самому середовищі можна налаштувати чітке автопілотування по заданому маршруту.

Компоненти та технології які необхідні створення змоги для автопілотування дроном. Необхідно мати базову модель FPV від п’яти дюймової до 12 дюймової моделі, також мати мікроконтролер який матиме змогу виконувати роботу для автопілотування. Важливою частиною є це вірно об’єднати, щоб мікроконтролер мав змогу надсилати команди для дрону. Технології для виконання даного завдання це протокол комунікації MavLink він дозволяє мікроконтролеру безпосередньо втручатися в роботу дрону зчитувати дані і стан дрону в реальному часі та надавати команди для чіткого виконання. Саме на цьому побудовано основну архітектуру частини програми для автопілотування, програмне забезпечення для автопілотування створюється на мові програмування Python з використанням бібліотеки pymavlink [8].

Методи навігації для автопілотування дроном можуть бути, як із використання GPS, що значно спрощує визначення положення дрона, що дозволяє мати можливість автопілотування так і без його використання в даному випадку необхідний чіткій запрограмований алгоритм для автопілоту приклад використання камери для визначення руху в певну точку.

1.4. Інтеграція технологій машинного зору та автопілотування з FPV

Поєднання машинного зору та автопілотування в одну систему є над складним завданням оскільки потрібно чітко розуміти, що дана система повинна бути бездоганна бо найменша помилка може призвести до втрати апарату або його значного пошкодження. Для уникнення подібних ситуацій потрібно проводити випробування в тестових умовах, щоб переконатися, що двигуни апарату виконують команди коректно тобто крутяться з необхідною швидкі у відповідному напрямку.

Для об’єднання цих двох технологій в одну та підключення до FPV необхідно для початку об’єднати на рівні програмного коду на мікроконтролері [5]. Дана робота виконується модифікацією створених методів автопілотування та інтеграції в них частин методів машинного зору, цим сами дрон буде розуміти, що необхідно летіти не по заданому маршруту, а летіти до захопленого об’єкта. Після об’єднання програмного коду необхідно налаштувати протокол комунікації мікроконтролера та польотного контролера дрону за допомогою протоколу MavLink. Фінальна частина підключення потрібно фізично дрон підключити до мікроконтролера це виконується за допомогою вільних uart портів на польотному контролері до яких підключаються відповідні піни з мікроконтролера для обміну інформації.

1.5. Огляд існуючих рішень та технологій автопілотування FPV

На даний час існує обмежена кількість таких рішень у відкритому доступі, але все таки існують від готових дронів з автонаведением до тонких налаштування під час прошивання дрону. Дані рішення мають як і будь якій пристрій свої значні переваги та недоліки, але вони маю своє право на існування і покращення в подальшому.

1.5.1. FPV дрон “ПТАШКА 7”

Базовою платформою для “Пташка 7” є семидюймовий тип рами з надміцного вуглецевого волокна. Даний розмір був обраний, через його кращі характеристики ніх менші моделі, а саме кращу протидію вітру. Кращий за більші моделі, оскільки чим більший дрон тим потужніші потрібні двигуни, а вони в свою чергу потребують більшого акумулятора, що додає непотрібні зайві грами у вазі, на основі цих досліджень оптимальний варіант був обраний семидюймовий FPV, вигляд готової моделі на (рис. 1.1).

Рис. 1.1. Модель FPV “Пташка 7”

Однією з особливостей даної моделі є інтегрована система розпізнавання об’єктів у реальному часі та їх захоплення. Також використано орієнтування в просторі оцінка власного положення відносно оточення та візуальних особливостей місцевості.

Основною перевагою даного дронує захоплення цілі та її ураження з використанням автопілотування дана технологія унеможливлює його перехоплення зв’язку.

Недолік даної моделі знайдений лише один серед критичних він полягає в тому, що дрон не може коригувати рух до цілі тобто, якщо ціль буде рухатися дрон в режимі автопілотування її втратить з захоплення і буде втрачений оскільки немає інших функцій при режимі автопілотування.

1.5.2. Автодоведення

Дана технологія є доволі сумнівною через те, що це не є окремий пристрій чи програмне забезпечення, проте має також право на життя у певних ситуаціях. Автодоведення в дронах типу FPV встановлюється під час прошивання та активується тумблером на пульті керування за необхідності здебільшого активується під час близкого підльоту до техніки оскільки це убезпечить дрон він перехоплення систем протидії дронам. Виконує роль доведення дрону до цілі на вже заданій траєкторії.

Переваги даної технології її доволі легко додати до існуючої прошивки навіть до легких в налаштуваннях таких як BetaFlight або INAV, що робиться за декілька хвилин, також перевага це без використання сторонніх мікроконтролерів та програмних рішень.

Недолік, автодоведення не може змінити траєкторію польоту і не фіксує ціль це в свою чергу означає що якщо ціль відхилиться, дрон на це ніяким чином не реагує і оператор не матиме змоги втрутитися в роботу дрону, що означатиме втрату апарату.

РОЗДІЛ 2
РОЗРОБКА FPV З МАШИННИМ ЗОРОМ НА АВТОПІЛОТУВАННІ

2.1. Формулювання проблеми та методи дослідження

Проблема: створення прототипу FPV дрону з машинним зором на автопілотуванні вимагає тонкого поєднання двох типів технологій, а саме апаратного рішення з чітким програмним кодом та ефективної роботи технічної частини пристрою.

Методи застосовані для дослідження:

- аналіз створених рішень: детальний аналіз створених пристроїв з схожими технологіями виявлення їх реальних переваг та недоліків, щоб ідентифікувати в даній області найкращі рішення та обмеження з якими можна зіткнутися;
- поступове тестування: проведення тестування частинами кожного етапу проекту по окремості як апаратної части тобто зліт дрону та його налаштування в процесі так і програмно окремо роботу машинного зору з захопленням об’єктів, та автопілотування спочатку по певній траєкторії після чого до певного об’єкта.

2.2. Підбір та обґрунтування обраних компонентів

Для створення FPV з машинним зором на автопілотуванні необхідно для початку обрати копоненти які будуть відповідати якості і не будуть суперечити один з одним. Хоча на ринку існує доволі велика кількість компонентів для створення FPV про це не полегшує їх підбір спираючись на собівартість та якість компонентів було обрані наступні для FPV та обраний мікроконтролер:

1. Польотний контролер F405 V3 - основна частина управління дрону та об’єднання всіх компонентів в одну систему.

2. Регулятор обертів SpeedyBee BLS 50A - регулює обертання моторів їх швидкість та напрям обертання.
3. Приймач сигналу BAYCK ExpressLRS 2.4GHz Nano ELRS - призначений, для приймання сигналу з пульта керування.
4. Безколекторні мотори EMAX ECO || 2087 130KV - призначені для виконання польоту дрону та його руху.
5. Карбонова рама Mark4-7 295мм. - частина дрону до якої кріпляться всі елементи.
6. RadioMaster TX12 2.4GHz 16 CH - апаратура для керування дроном.
7. Камера FPV Waveshare RPi Camera (B) - елемент який знімає відео в реальному часі.
8. Мікроконтролер Raspberry Pi Zero 2 W - необхідна частина для запису програми яка виконує роль машинного зору та автопілотування.

Обраний стек компонентів був обраний після детального їх аналізу потужності і здатностей і також суми та доступності.

2.3. Вимоги до системи та постановка задачі

Основна вимога до системи, полягає у її чіткому виконанні, тобто система мусить виконувати завдання визначення об’єкта, чітке захоплення та автоматичний політ до цілі. Також реалізація відключення сигналу відео передавання при захопленні об’єкта та часткового відключення сигналу керування.

2.3.1. Функціональні вимоги для FPV дрону з машинним зором на автопілотуванні.

- Керування оператором:
    - дрон повинний забезпечувати стабільний політ під керуванням оператора;

- оператор мусить мати можливість повноцінного контролю дрону під час польоту.

- Визначення об’єктів в реальному часі:
  - система повинна виявляти та розпізнавати об’єкти у відеопотоці в реальному часі;
  - система мусить визначати рівень впевненості у виявленому об’єкті.
- Захоплення об’єктів:
  - оператор повинний мати змогу захопити виявлений об’єкт у відеопотоці;
  - система після захоплення повинна відстежувати захоплений об’єкт.
- Автономний політ до захопленого об’єкта:
  - дрон повинний самостійно летіти до зафіксованого об’єкта;
  - система повинна враховувати обмеження в висоті і швидкості польоту;
  - оператор мусить мати змогу відновити керування або перезапустити його.

Описані вимоги визначають основні можливості дрону та визначить подальшу розробку його апаратної та програмної частини.

2.3.2. Особливості експлуатації

Існують деякі обмеження, щодо його користування оскільки він не адаптований для передачі відео на велику відстань через відсутність приймача відео сигналу. Наданий час відеопотік передається, через мікроконтролер напряму на ноутбук з відкритою операційною системою Raspbian, також паралельно відео записується для змоги його перегляду після польоту. Інших особливостей експлуатації немає він, також виконує всі функції що і звичайний FPV дрон.

2.4. Опис та схема підключення компонентів

Підключення всіх компонентів виконувалося з точки зору стандартного розміщення та можливості розміщення на рамі даного типу для кращої роботи та уникнення створення шумів при передачі сигналів між компонентами.

Фізичне підключення більшості компонентів відбувалося на пряму до польотного контролера за допомогою uart портів, які на FPV визначені. Винятком підключення, є камера яка підключається на пряму до мікроконтролера це зумовлено тим щоб уникати затримки отримання та обробки відео потоку бо, якщо камеру підключити до польоника на пряму то існує можливість отримання відеопотоко з мінімальною затримкою яка може відіграти важливу роль в автопілотування дрона. Також виняток підключення є регулятор обертів, оскільки на польотному контролері існує спеціальний порт для підключення регулятора. Мотори підключені до регулятора обертів на відповідні піни в довільному порядку, оскільки їх обертання буде визначатися в подальшому. В подальшому всі підключенні компоненти визначаються відповідно у прошиванні дрона. Схема підключення компонентів зображена на (рис. 2.1) та (рис. 2.2).

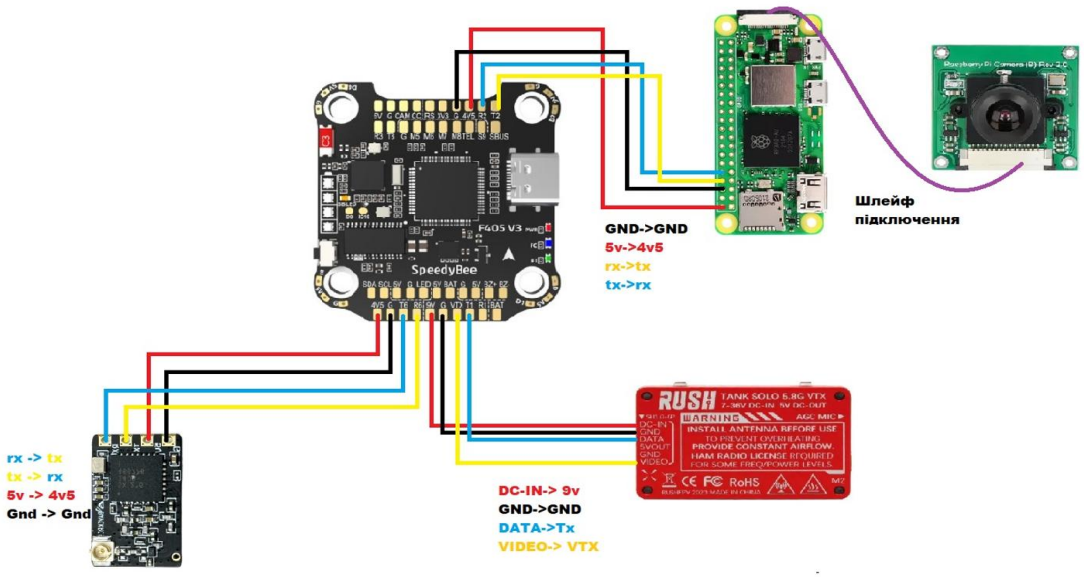

Рис. 2.1. Схема підключення компонентів до польотного контролера

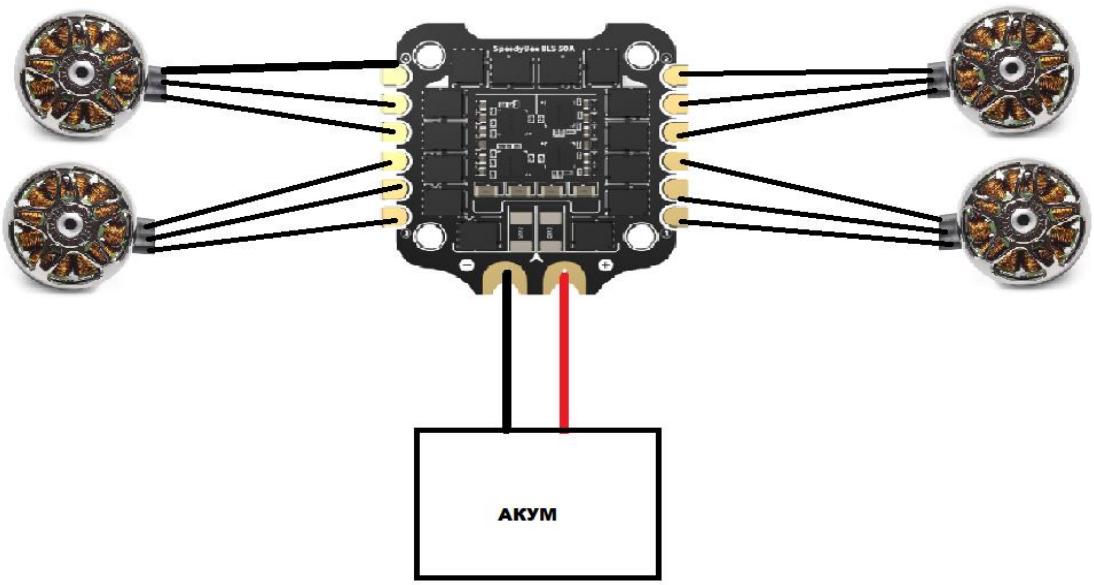

Рис. 2.2. Схематичне підключення моторів до регулятора обертів

Дана схематика підключення відповідає усім стандартним нормам що гарантує стабільну роботу апарату та легкості його налаштування.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА НАЛАШТУВАННЯ FPV

3.1. Налаштування мікроконтролера Raspberry Pi

Налаштування мікроконтролера є однією з найважливіших задач для побудови FPV з додатковим програмним забезпеченням, оскільки саме на цей елемент записується програмна частина роботи та виконує роботу комунікації з дроном.

Першочергова задача налаштування мікроконтролера це наявність micro-SD. Оскільки в мікроконтролері не передбачено вбудованої пам'яті це має як свій плюс так і мінус, мінусом є сам факт придбання додаткових частин для пристрою, безпосереднім плюсом є те, що саме ви визначає пам'ять яка буде на мікроконтролері. Після вирішенням даного питання з пам'яттю, необхідно завантажити на micro-SD операційну систему для Raspberry Pi для даного типу мікроконтролерів існує своє опереаційна система Raspbian вона своєю архітектурою нагадує систему Linux. Для завантаження системи необхідно підключити microSD до комп'ютера після чого за допомогою програми Raspberry Imager (рис. 3.1) здійснюється поетапне її встановлення це може зайняти деякий час. Налаштування для операційної системи є інтуїтивно зрозумілим необхідно лише обрати модель пристрою, операційну систему, та накопичувач для її запису.

Рис. 3.1. Середовище завантаження ОС Raspbian

Після успішного завантаження операційної системи на microSD необхідно її вставити в мікроконтролер Raspberry Pi. Після встановлення подати живлення і за допомогою Wi-Fi мережі вивести робочий стіл мікроконтролера (рис. 3.2) собі на екран. Також можливе підключення мікроконтролера до екрану за допомогою HDMI кабелю, якщо вдалося побачити робочій стіл мікроконтролера це означає що систему було успішно встановлено.

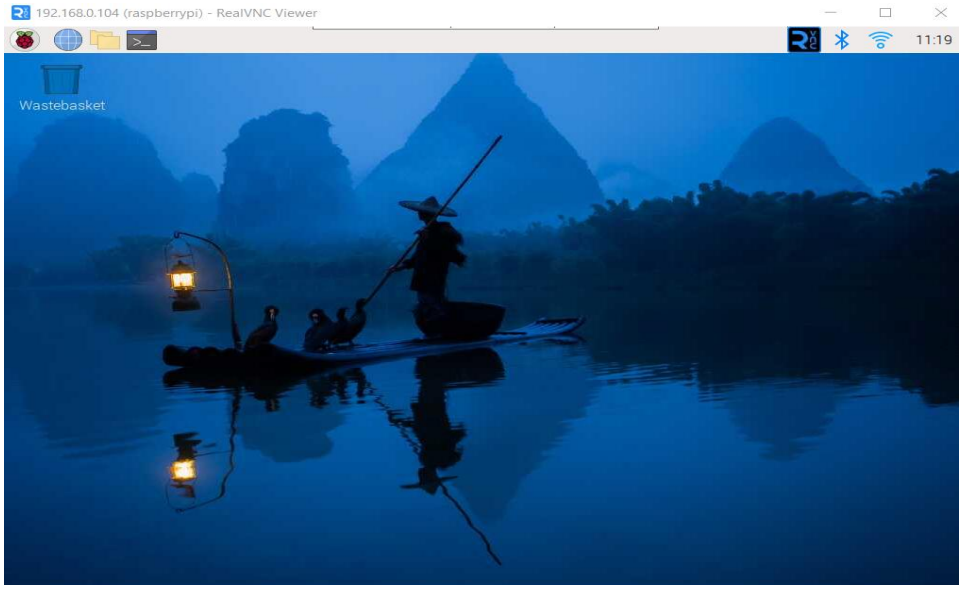

Рис. 3.2. Робочій стіл Raspberry Pi

Фінальною частиною налаштування є встановлення необхідних бібліотек для подальшої роботи з мікроконтролером для цього необхідно відкрити консоль в системі (рис. 3.3). Після її успішного відкриття необхідно ввести команди:

sudo apt install -y python3 python3-pip python3-opencv libopencv-dev

pip3 install numpy picamera2 PyQt5

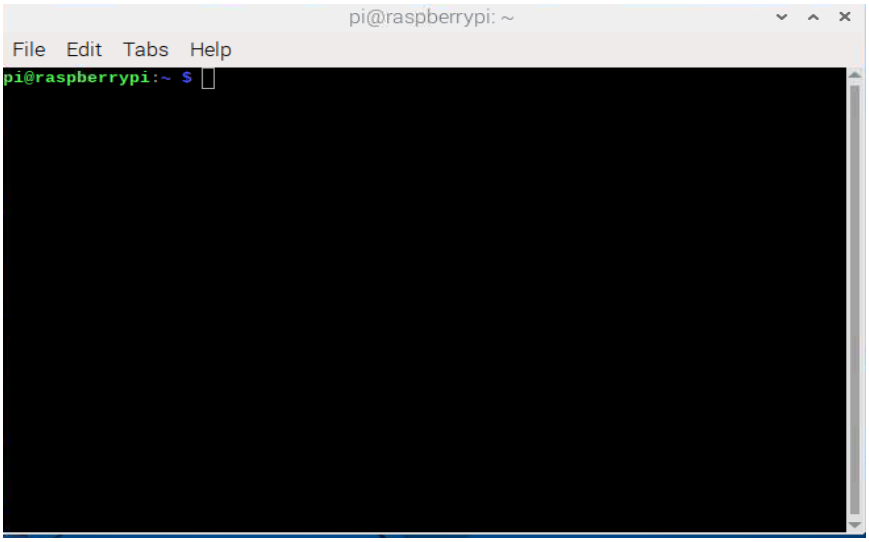

Рис. 3.3. Консоль Raspberry Pi

Дані команди автоматично протягом декількох хвилин встановлюються необхідні бібліотеки, для системи, щоб в подальшому на ній можна було без проблемно створювати програмне забезпечення. Для перевірки успіху їх встановлення можемо перевірити в консоли ввівши тестову базову команду для перевірки роботи камери *libcamera-hello*. Після виконання команди маєте побачити ряд ініціалізованих команд в консолі (рис. 3.4) це означатиме успіх встановлення всіх бібліотек та успішне налаштування системи для подальшого його використання.

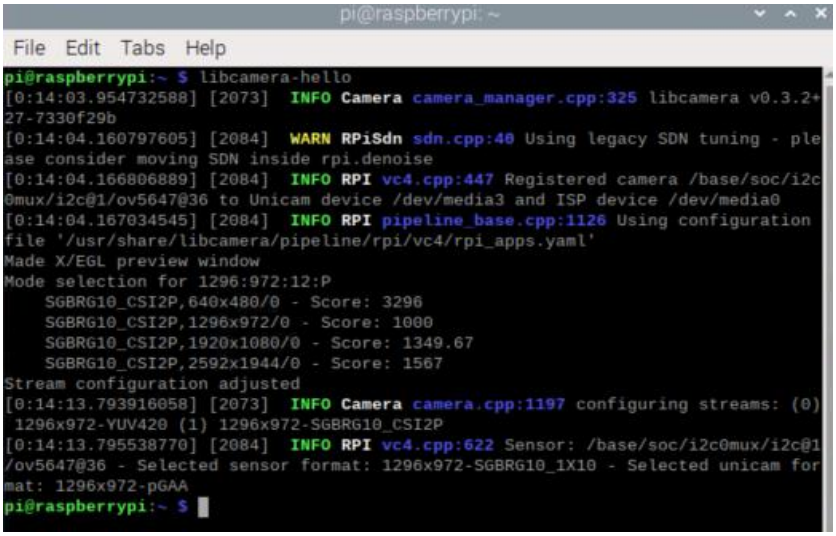

Рис. 3.4. Вивід команди libcamera-hello

3.2. Прошивання FPV

Частина прошивання дрону є дуже важливою, оскільки саме вона надає можливість керування дроном. На даному етапі спочатку потрібно визначити який тип прошивки необхідний для поставленого завдання оскільки кожна з них має свої переваги та недоліки, можна виділити три основні види прошивки:

- BetaFlight - дана прошивка є найпопулярнішою у прошиванні FPV оскільки вона доволі проста в усвоєні легко налаштовується інтуїтивно зрозумілий інтерфейс. Проте, якщо розробляти щось більше ніж базовий FPV з використання додаткових модулів або необхідність підключення до MavLink. Даний тип прошивки не підійде через те, що в ній не

передбачений такий функціонал вона немає змоги підключатися до систем комунікації з мікроконтролерами або іншими не базовим пристроями;

- INAV - даний тип прошивки є аналогом прошивки BetaFlight на мою думку дещо стабільніший і легший в тонких налаштуваннях керування. Також має змогу підєднуватися до протоколів комунікації такі, як MavLink, але з обмеженням, а саме буде лише можливість витягувати дані з дрону тобто спілкування з мікроконтролерами в одну сторону. Оскільки за допомогою мікроконтролера можна буде отримувати дані і стан, проте не передавати на виконання команд, отже цей тип прошивки підійде для базового вивчення методів комунікацій дрона з зовнішніми системами;
- ArduPilot - даний тип прошивки можна віднести до надскладних це пов’язано з складністю розуміння середовища прошивання. Оскільки існує доволі багато різноманітних тонких налаштувань кожного аспекту, з переваг це практично необмежені можливості у прошиванні. Даний тип прошивки можна використовувати в безпілотних системах для задання маршруту тобто після прошивання можна задавати автономний рух дрону проте з використанням GPS. Нарахунок комунікації з зовнішніми системами через протокол MavLink то комунікація виконується в дві сторони тобто мікроконтролер як отримує сигнали так і може передавати команди для виконання на цьому аспекті сконструйовано автопілотування.

Отже прошивання дрону виконувалося на ArduPilot. Для прошивання дрону даним прошиванням необхідно встановити і відкрити Mission Planner (рис. 3.5) після чого приєднати дрон в зібраному вигляді за допомогою кабелю Type-c до польотного контролера, а іншу сторону до комп’ютера з якого буде виконуватися прошивання.

Рис. 3.5. Середовище Mission Planner

Після успішного підключення можна перейти до налаштування прошивки а саме перейти в пункт **Setup** після чого відкриється дане вікно (рис. 3.6).

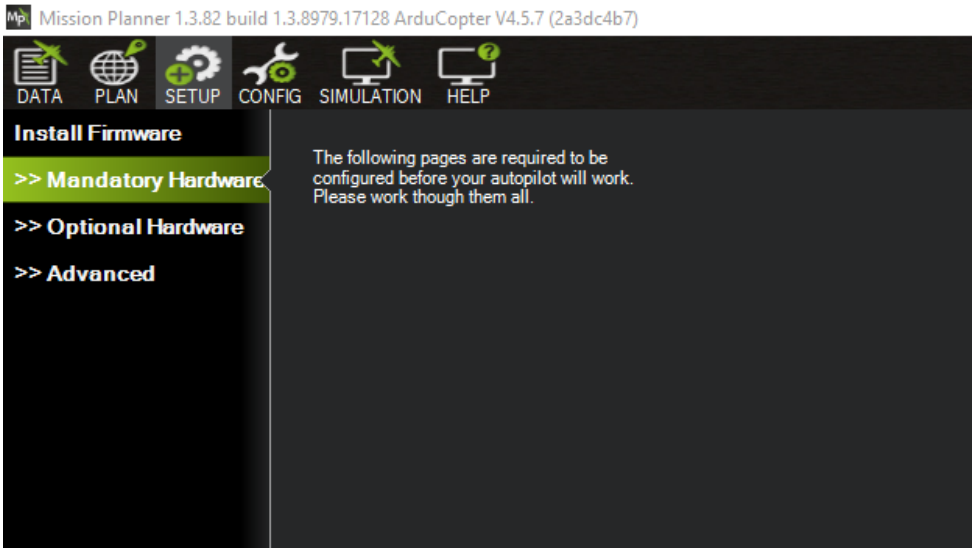

Рис. 3.6. Відкрите вікно Setup

У відкритому вікні необхідно буде відкрити Mandatory Hardware та налаштувати такі пункти:

Frame Type - в даному пункті необхідно обрати ваший тип рами це необхідно для визначення кількості моторів на дроні (рис. 3.7).

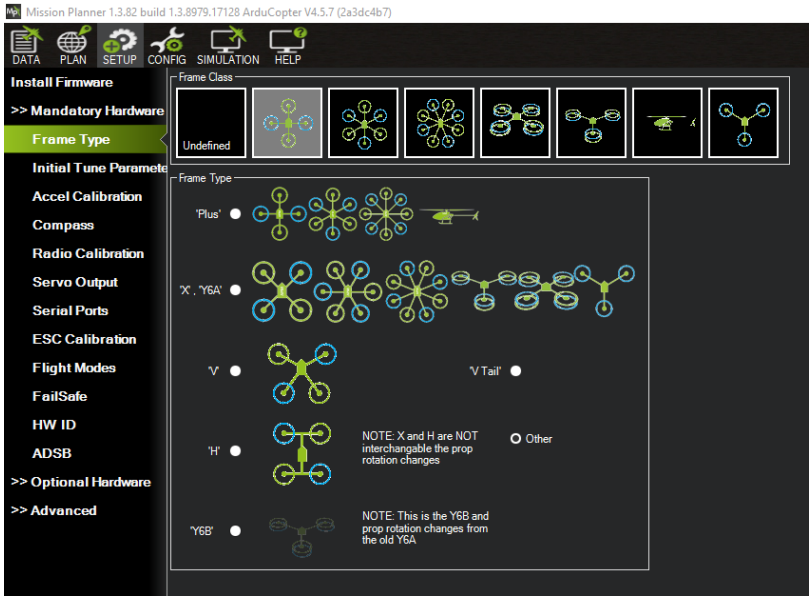

Рис. 3.7 Налаштування типу рами

Initial Tune Parameter - даний пункт необхідно налаштувати для того, щоб дрон розумів яка батарея на ньому стоїть та на які можливості вона розрахована (рис. 3.8).

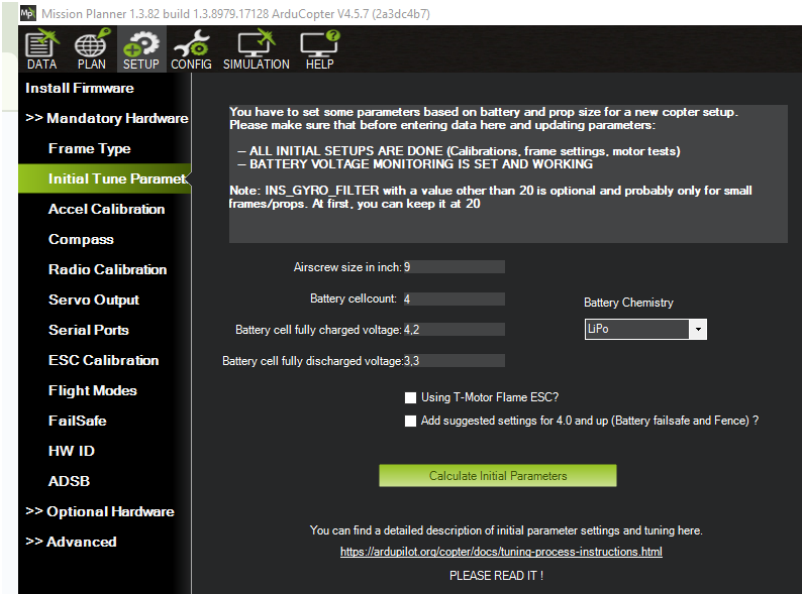

Рис. 3.8. Initial Tune Parameter

Radio Calibration - пункт для привязки пульта керування до дрону (рис. 3.9). Проте щоб все було успішно пульт все має бути коректно налаштований та мати модуль тої частоти яку має передавач на дроні в даному випадку

використовувалося частота 2.4GHz. Щоб виконати процес прив'язання потрібно натиснути на кнопку Calibrate radio після чого пуль буде успішно прив'язаний.

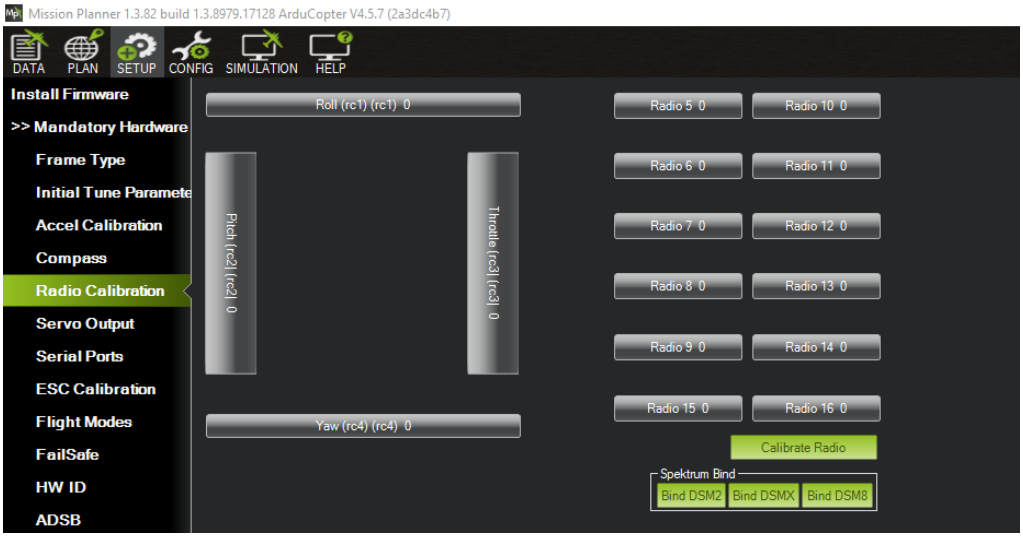

Рис.3.9. Розділ налаштування Radio Calibration

Serial ports - наступний етап налаштування це визначення портів до яких при прикріпленні елементи до польотного контроллера по uart. На даному етапі потрібно чітко знати, що до чого прикріплено та визначити це в налаштування (рис.3.10) на даному скрині видно що використовується два порти це uart 6 для приймача сигналу та uart 2 для мікроконтролера з протоколом MavLink.

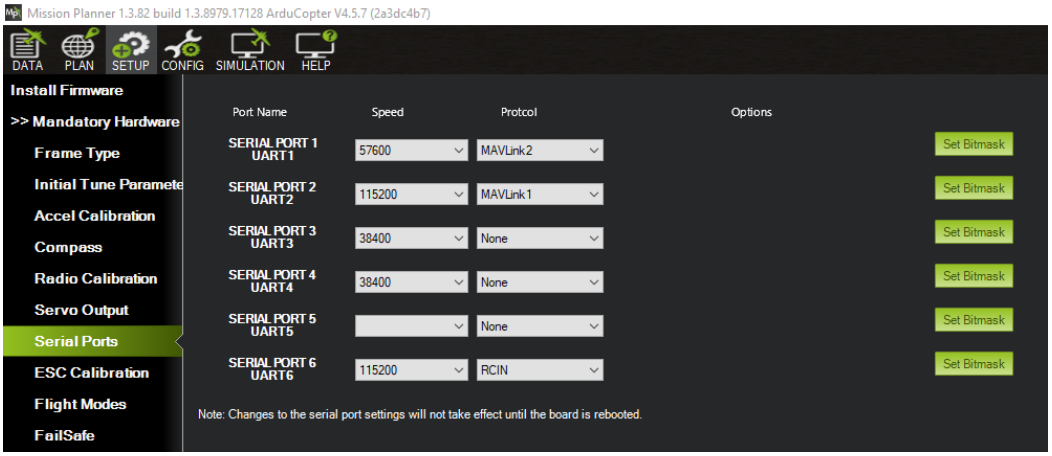

Рис. 3.10. Розділ налаштування Serial Ports

ESC Calibration - даний розділ налаштування необхідний для налаштування регулятора обертів для коректного керування моторами (рис.3.11). Дане налаштування є дуже специфічним оскільки щоб перепрошити вірно необхідно

виконати крім визначення налаштувань моторів ставленням значень потрібно ще виконати послідовність дій з підклбчення і відключенням акамулятора до тих пір поки польотний стек не видасть характерний подвійний сигнал що означатиме успішне завершення прошивання.

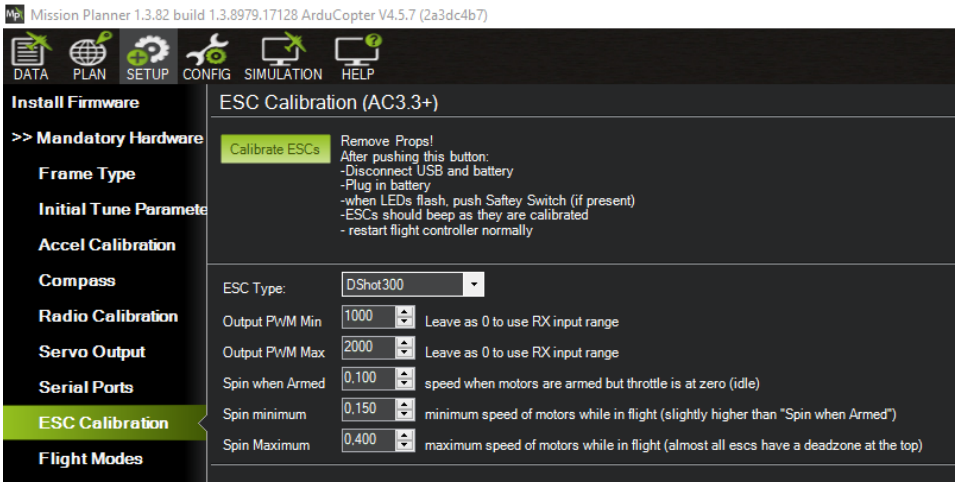

Рис. 3.11. Розділ налаштування ESC Calibration

Наданому налаштуванні завершується базові налаштування, проте необхідно далі налаштовувати під кожну модель окремо. У вікні Config - Full Parameter List (рис. 3.12) на даній сторінці можна як коригувати налаштовані параметри так і налаштувати за необхідності інші глибші налаштування прикладом такого можна зазначити вимкнення моторів при перевертані дрону. Також можливо налаштувати повернення додому, але для цього необхідно мати перевірені польотні контролери оскільки дана технологія є ненадійна.

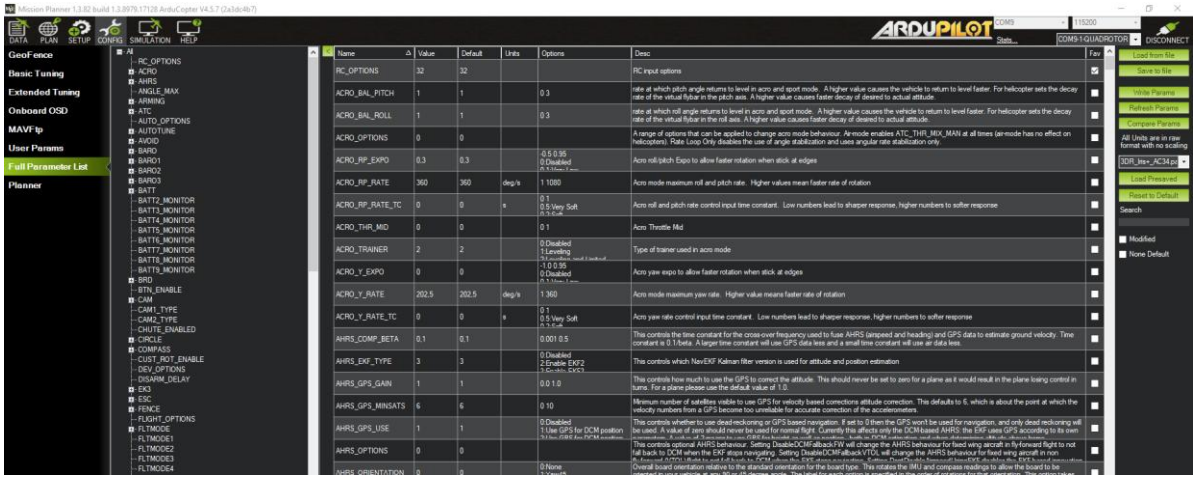

Рис. 3.12 Розширені налаштування

Підсумок показу налаштування FPV можна зробити, такий для кожної задачі необхідно використовувати відповідний тип прошивки яка орієнтована під певне використання. Варто також зауважити, що показано загальне налаштування для розуміння, про те вона виконується значно складніше з виникнення різноманітних проблем та нюансів в процесі.

3.3. Програмна реалізація машинного зору та автопілотування FPV

Для програмної реалізації даної задачі було використано мову програмування Python з стандартним бібліотеками та декількома специфічними. Чому саме ця мова програмування було обрана через її необмежений потенціал в різних сферах програмування, також дана мова ідеально підходить під мікроконтролер Raspberry Pi [6].

Опис бібліотек які використовувалися та для якої конкретної задачі:

- *sys* - дана бібліотека є стандартна у мові програмування Python надає доступ до до деяких змінних і функцій, які тісно взаємодіють з ітератором Python. Використовується для взаємодії з системними параметрами та функціями, керування стандартними потоками введення та виведення;
- *cv2 (OpenCV)* - бібліотека для використання комп'ютерного зору та надає багато функцій для обробки зображень та відео. Використовується для читання запису та обробки зображень та відео, відстеження руху, аналіз відео, детекція об'єктів та розпізнавання об'єктів;
- *numpy* - одна з найпотужніших бібліотек для великих обчислень в мові програмування Python. Використовується для багатовимірних масивів та матриць, математичних функцій для роботи з масивами також для генерації випадкових чисел;
- *threading* - дана бібліотека дозволяє створювати та керувати потоками даних, що в свою чергу дозволяє виконувати декілька задач одночасно. Використовується для реалізації паралелізму та конкретизації, виконання

задач в фоновому режимі по декілька за раз, покращення продуктивності та оптимізації програм які виконують по декілька задач одночасно;

- *time* - бібліотека дозволяє працювати з часом та надає відповідні функції. Використовується для вимірювання часу, визначення часу для затримки, форматування часу та дати, дана бібліотека дуже важлива в даному програмному забезпеченні;
- *picamera2* - дана бібліотека призначена лише для ініціалізації та роботи з камерами Raspberry Pi даний тип камери застосовується в даному проекті;
- *PyQt5* - це своєрідна збірка бібліотек в Python для створення графічних інтерфейсів користувача (GUI). Надає змогу класи для створення вікон, кнопок, текстових елементів, механізми для обробки подій таких як кліки миші та натискання на клавіатуру, підтримка для малювання графіки та відображення зображень;
- *pymavlink* - специфічна бібліотеra під використання з протоколом MavLink який використовується для зв'язку з безпілотним літальними засобами. Дана бібліотека надає засоби для кодування та декодування повідомлень MavLink, функції для відправки команд і отримання телеметричних даних від безпілотних апаратів. Використання даної бібліотеки в розробці дрона надало змогу реалізації автопілотування.

3.3.1. Реалізація машинного зору

Для інтеграції машинного зору в FPV дрон було використано готову модель формату SSD та дещо спрощено для кращої продуктивності. Lана модель була обрана після чіткого аналізу та тестування схожих моделей, які можуть працювати в реальному часі та аналізувати відео потік. Основним критерієм для моделі машинного зору сумісність з мікроконтролером Raspberry Pi оскільки даний мікроконтролер ідеально підходить для цієї задачі про те має свої системні обмеження які технічно розширити неможливо.

```
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
           "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
```

```
"dog", "horse", "motorbike", "person", "pottedplant",
"sheep", "sofa", "train", "tvmonitor"]
```

Масив *CLASSES* представляє собою масив назв для моделі машинного зору тобто описано назви всіх предметів які може виявити модель машинного зору дані класи є зменшені оскільки потрібно було обмежити кількість об'єктів для виявлення для кращої оптимізації.

```
self.net = cv2.dnn.readNetFromCaffe("deploy.prototxt",
"mobilenet_iter_73000.caffemodel")
```

Дана частина коду є частиною методу __init__ даний рядок з методу використовується для завантаження попередньо навченої моделі машинного зору, що включає в себе два файли, а саме *deploy.prototxt* представляє собою основу архітектури моделі та *mobilenet_iter_73000.caffemodel* є фалом вагом для моделі в поєднанні ці два файли разом представляють готову навчену модель.

```
image = self.picam2.capture_array()
        height, width = image.shape[:2]
        blob = cv2.dnn.blobFromImage(image, 1/255.0, (224, 224), (127.5,
127.5, 127.5), swapRB=True, crop=False)
        self.net.setInput(blob)
        detections = self.net.forward()
```

Дана частина коду перетворює відеопотік у формат який може сприймати модель для подальшого розпізнавання та визначення об'єктів.

- *image = self.picam2.capture_array()* - отримання вхідного відеопотоку з камери;
- *1/255.0* - масштабування пікселів зображення в діапазоні [0,1];
- *(224, 224)* - розмір під якій буде видозмінене зображення;
- *(127.5, 127.5, 127.5)* - середні значення пікселів які будуть підняті для кожного каналу пікселів зображення;
- *swapRB=True* - перемикає порядок каналів з BRG в RGB дане перемикання збільшує оптимізацію програми при роботі з видеопотоком.

```
for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            class_id = int(detections[0, 0, i, 1])
            x1 = int(detections[0, 0, i, 3] * width)
            y1 = int(detections[0, 0, i, 4] * height)
            x2 = int(detections[0, 0, i, 5] * width)
            y2 = int(detections[0, 0, i, 6] * height)
```

Даний цикл є основним у виявлення об'єктів і використанні моделі для їх знаходження:

- *confidence = detections[0, 0, i, 2]* - рядок для отримання рівня довіри або впевненості в даному об'єкті чим більша довіри тим точніше його визначення;
- *if confidence > 0.5:* - перевірка чи рівень довіри більший за значення 0.5 це допомагає відсіяти об'єкти які можуть визначатися помилково;
- *class_id = int(detections[0, 0, i, 1]*) - отримання класів визначених об'єктів та їх підписування в відеопотоці;
- *x1 = int(detections[0, 0, i, 3] * width)* - даний рядок і рядки під ним необхідні для визначення координат об'єктів, яких був визначений та динамічної їх перебудовувати координати при русі. Дана процедура знижує кількість кадрів в відео потоці проте це необхідно для коректної роботи програми [9].

Даний алгоритм ліг в основу програми для автопілотування.

3.3.2. Реалізація захоплення визначеного об'єкта та його трекінг

Дана задача була необхідна для розробки в подальшому автопілотування. Оскільки на даному етапі створюється алгоритм, який захоплювати визначений об'єкт та буде його супроводжувати при русі в реальному часі [3]. Для виконання даного алгоритму можливо використати готову бібліотеку opencv-contrib-python

проте на основі мікроконтролера Raspberry Pi це практично неможливо зробити. Оскільки дана бібліотека є над важкою і займає час на встановлення понад дві доби, проте дана вона значно б оптимізувала процес і збільшила кількість кадрів. Другий оптимальний варіант було об’єднання машинного зору і трекінгу, щоб після захоплення об’єкту машинний зір міг визначати, лише цей об’єкт це вирішило проблему з трекінгом проте навантажило мікронтролер, що прирівняло частоту кадрів до частоти яка використовується для пошуку об’єктів.

```
if best_detected:
            self.captured_object["bbox"] = best_detected["bbox"]
            x, y, w, h = self.captured_object["bbox"]
            label = CLASSES[self.captured_object["class_id"]]
            color = (255, 0, 0)
            cv2.rectangle(display_image, (x, y), (x + w, y + h), color, 3)
            cv2.putText(display_image, f"Tracking: {label}", (x, y - 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
            self.fly_to_object(x, y, w, h, width, height)
        else:

            x, y, w, h = self.captured_object["bbox"]
            label = CLASSES[self.captured_object["class_id"]]
            color = (255, 0, 0)
            cv2.rectangle(display_image, (x, y), (x + w, y + h), color, 3)
            cv2.putText(display_image, f"Lost track: {label}", (x, y - 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
```

Дана частина коду використовує код з пошуку об’єктів та вносить зміни для захоплення, а саме малює рамку червоного кольору для розуміння? що об’єкт захоплений після чого проводить його трекінг, якщо об’єкт був втрачений при трекінгу програма буде шукати лише об’єкт який був визначений останній після чого продовжить його трекінг.

```
if flight_value > FLIGHT_THRESHOLD and self.last_flight_value <=
FLIGHT_THRESHOLD and self.detection_enabled:
                QtCore.QMetaObject.invokeMethod(self,
"capture_object", QtCore.Qt.QueuedConnection)
            self.last_flight_value = flight_value
```

Дана умова використовує протокол MavLink для отримання і передачі сигналу з пульта керування оскільки зроблено так, якщо увімкнений 8 тумблер тоді програма використовує метод якій виконує логіку захоплення та трекінгу об'єктів *FLIGHT_THRESHOLD* дане значення оголошене на початку програми і визначає частоту 8 тумблера на пульті керування.

3.3.3. Реалізація алгоритму автопілотування

Для реалізації даного алгоритму необхідно мати повноцінний дрон, який є прошитий та підключений до мікроконтролера, також налогоджений протокол комунікації MavLink через те, що даний етап повинен виконувати звертання до дрону та надання команд на виконання через пуль керування який звертатиметься до мікроконтролера [4].

```
def fly_to_object(self, x, y, w, h, frame_width, frame_height):
    obj_cx = x + w // 2
    obj_cy = y + h // 2

    frame_cx = frame_width // 2
    frame_cy = frame_height // 2

    dx = obj_cx - frame_cx
    dy = obj_cy - frame_cy

    Kx = 0.5
    Ky = 0.5
    pitch = int(np.clip(Kx * dx, -1000, 1000))
    throttle = int(np.clip(500 + Ky * -dy, 0, 1000))
    if self.armed and self.master:
        self.master.mav.manual_control_send(
            self.master.target_system,
            pitch, 0, throttle, 0, 0
        )
```

Метод fly_to_object виконує роль автопілотування дрону після захоплення. Роль даного методу полягає в визначенні відхилення яке можливе при русі цілі та коригування моторів для цього в режимі реального часу.

- *obj_cx = x + w // 2 obj_cy = y + h // 2* - дані рядки в методі відповідають за визначення центру захопленої цілі;
- *frame_cx = frame_width // 2 frame_cy = frame_height // 2* - частина визначання центру кадру необхідно для з співставленням з центром захопленої цілі;
- *dx = obj_cx - frame_cx dy = obj_cy - frame_cy* - визначення відхілення відносно руху цілі використовується для коригування польоту;
- *pitch = int(np.clip(Kx * dx, -1000, 1000))* - рядок якій відповідає у русі дрону вперед це виконується за допомогою швидшого обертання моторів спереду відносносно задніх;
- *throttle = int(np.clip(500 + Ky * -dy, 0, 1000))* - відповідає за набір висоти для дрону;
- *if self.armed and self.master:* - умова для перевірки чи дрон заармений і чи зафіксована ціль після чого дрон виконуватиме коригування і політ до цілі автоматично.

```
if best_detected:
            self.captured_object["bbox"] = best_detected["bbox"]
            x, y, w, h = self.captured_object["bbox"]
            label = CLASSES[self.captured_object["class_id"]]
            color = (255, 0, 0)  # Червоний колір для захопленого об’єкта
            cv2.rectangle(display_image, (x, y), (x + w, y + h), color, 3)
            cv2.putText(display_image, f"Tracking: {label}", (x, y - 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
            # --- Автопілотування ---
            self.fly_to_object(x, y, w, h, width, height)
```

Умова *if best_detected:* в якій продемонстровано використання методу автопілотування. Дана умова використовується в захоплені цілі тобто реалізовано так, щойно ціль захоплена дрон починає автоматичний рух до цілі та перериває відео потік та часткове відключення від пульту керування, часткове

відключення для того, щоб оператор міг відновити керування у разі змінення завдання.

РОЗДІЛ 4
ТЕСТУВАННЯ ПРОБЛЕМИ ТА ЇХ ВИРІШЕННЯ

Тестування проводилося двохетапно перший етап був тестування проходив в тестових умовах при відсутності пропелерів на моторах другий етап проводиться в реальних умовах на вулиці, де було протестовано поведінку дрону при автопілотуванні.

4.1. Тестування в тестових умовах

Перший етап тестування був орієнтований на перевірку роботи частини програми виявлення об’єктів за допомогою машинного зору та їх детекції з фіксуванням. Також перевірка руху дрону за допомогою виведення швидкості моторів в реальному часі. Для початку тестування необхідно було підключити мікро комп’ютер до FPV та увімкнути протокол MavLink на мікроконтролері, що виконувалося при запуску консоні на Raspberry Pi та введення команди *python3 mavproxy.py --master=/dev/ttyAMA0 --baudrate 115200 --aircraft MyCopter* (рис. 4.1). Дана команда звертається до вбудованого скрипта *mavproxy.py* після чого пробує під’єднатися до серійного порту на дроні */dev/ttyAMA0* з швидкістю передачі даних *115200.*

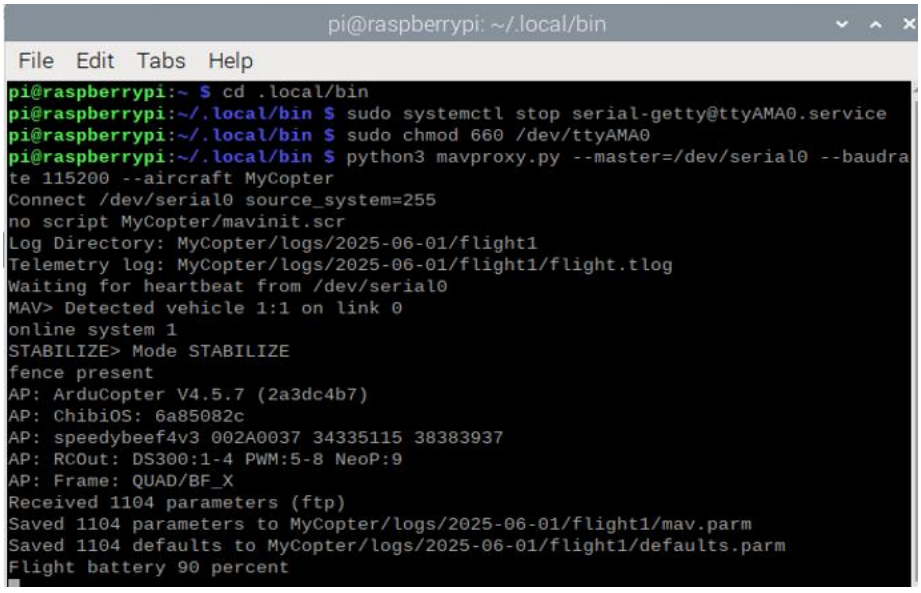

Рис. 4.1. Запуск комади для ввімкнення MavLink

Після запуску ми можемо бачити успішність запуску протоколу про це говорить рядок *online system 1* тобто протокол бачить наш пристрій і виводить базові параметри FPV, а саме тип прошивки, серійний номер польотного контролера, тип рами, та стан акумулятора.

Наступний крок тестування є запуск скрипта для перевірки роботи машинного зору та детекції предметів також тестовий політ без пропереперелерів. Після запуску скрипта запускається графічне вікно для виведення відеопотоку з камери FPV (рис. 4.2).

Рис. 4.2. Виведення відеопотоку та виявлення об’єктів

Як можемо бачити після запуску скрипта успішно відкрилося вікно відеопотоку де можемо побачити вже роботу частини програми з виявлення об’єктів. Було успішно виявлено об’єкт bottle з оцінкою впевненості 90% також можемо знизу бачити статус моторів які на даному етапі тестування нективні.

Наступний крок тестування це ARM моторів для перевірки успішної комунікації мікрокомпютера та FPV (рис. 4.3).

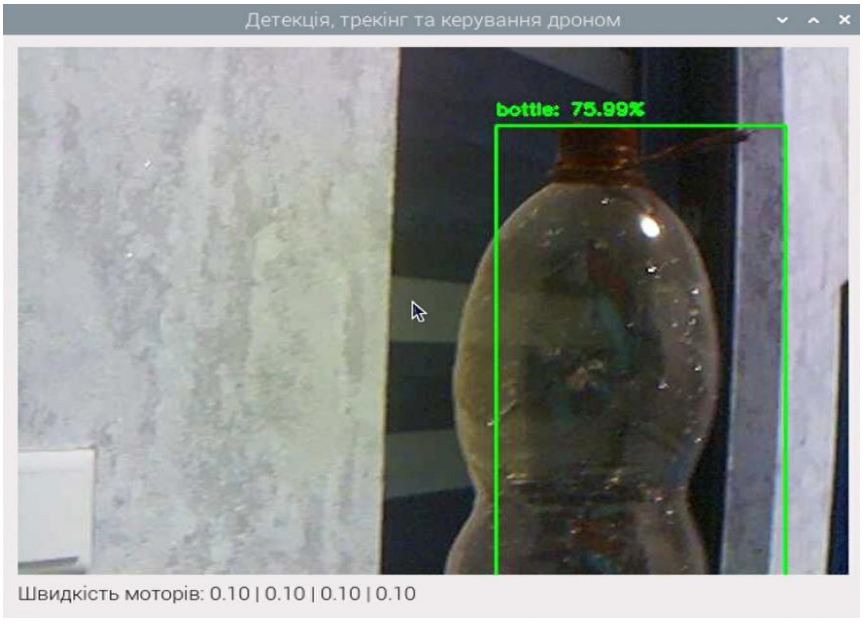

Рис. 4.3. ARM дрону при виконанні програми

Можемо бачити, що ARM є успішним оскільки швидкість моторів змінилася і тепер вони крутять на мінімальній потужності що свідчить що вдалося успішно передати команду з пульту на дрон через протокол комунікації.

Наступний етап це фіксація цілі та рух дрону вперед, що має бути висвітлено у швидкості руху моторів (рис. 4.4). Також в даному тестуванні перевірка чи захоплюється об’єкт при натисканні на 8 тумблер на пульті керування.

Рис. 4.4. Детекція об’єкта

Тестування детекції відбулося успішно, що може свідчити колір рамки який змінився з зеленого на червоний, також напис який свідчить, що об’єкт зафіксований. Також видно зміну рухів моторів які почали крутитися швидше і з різною швидкість, а саме перший на третій мають меншу швидкість ніж другий і четвертий, що значить, що дрон буде летіти вперед до об’єкта. Варто також зазначити в різниці швидкості в суміжних двигунів тобто 0.21 та 0.15 ці дані свідчать про те, що дрон намагаєть коригувати свій рух, щоб об’єкт був в центрі кадру. Дана поведінка проявляється, якщо об’єкт рухається мотори змінюють свою швидкість підлаштовуючись під рух, це може свідчить про успішне виконання головної і складної задачі щодо коригування під об’єкт який рухається.

4.2. Тестування в реальних умовах

Даний тип тестування є доволі небезпечний оскільки FPV буде винятково на автопілоті, що при наявності проблемних місць в коді може призвести до втрати пристрою. Дане тестування орієнтоване на перевірку FPV у вільному польоті, щоб пересвідчитися, що алгоритм автопілотуванні виконується коректно, що буде свідчити що робота всього проекту є виконаною. На (рис. 4.5) вигляд дрону при автопілотуванні на низькій висоті. Низька висота це перестрахування в безпеці, оскільки дані типи тестування необхідно проводити в спеціальних умовах до яких доступ, для тестування не маю тому було протестовано на вулиці на низькій висоті.

Рис. 4.5. Дрон під час автопілотування

На фото можемо бачити, що дрон самостійно знаходиться у повітрі та нахилений в сторону напрямку руху, дане тестування можна вважати успішним проте, щоб воно було повністю заверше необхідно проводити детальніше тестування в реальних умовах в спеціальних для цього місцях.

4.3. Проблеми та їх вирішення

При тестуванні в тестових і реальних умовах було виявлено ряд проблем які були неприпустимі при використанні FPV:

- проблема в невірному малювання рамки до об’єкта якій було виявлено тобто рамка малювалася лівіше від об’єкта;
- проблема в автопілоті через специфіку прошивки дрон на автопілоті замість польоту вперед рухався назад;
- проблема з детекцією програма захоплювала об’єкт проте не супроводжувала, дана проблема виникає при розробці;
- проблема в прикріплені мікроконтролера до FPV через обмежене місце на дроні;
- проблема в передачі відеосигналу, відео передається на обмежену відстань причиною цього є передача відеосигналу через мікроконтролер.

Наведенні помилки є неприпустимі для подальшої експлуатації FPV оскільки вони противоріччють основним поставленим задачам в розробці, проте для кожної з цих помилок було виявлено відповідне рішення:

- вирішення проблеми в обведені виявленого об’єкта було вирішено в коді, а саме зміщення координат правіше ніж виявляється, після чого подібних помилок не виникало;
- проблема автопілоту невірний рух було визначено, що для руху вперед необхідно змінити номери моторів які потрібно крутити швидше для вірного руху вперед;

- вирішення проблеми з детекцією що неможливо було супроводжувати виявлений об’єкт було вирішено додати до детекції частини коду виявлення об’єкта, що допомогло супроводжувати захоплений об’єкт;
- технічна проблема з розміщенням мікроконтролера була вирішена, дещо специфічною і не сильно правильно з точки зору безпеки, а саме зрозміщення на акамуляторі. Проте варто зазначити, що дане рішення використовується часто в реальних умовах для прикріплення і інших деталей не передбачених конструкцією на акумулятор;
- вирішення проблеми з передачею сигналу на більшу відстань можливо вирішити додавання базового елементу для FPV, а саме відео передавач. Проте для цього необхідно мати обладнання для отримання відеопотоку, а саме окуляри типу SkyZone, що допомагають бачити відео через відеопередавач. Проте знайдене рішення, а не його реалізація оскільки, щоб вирішити дану проблему необхідно значні кошти на відеопередавач типу цифрової моделі та окуляри для прийому сигналу.

Після вирішення проблем і при повторному тестуванні схожих помилок не виникало та FPV працював у штатному режимі успішно виконуючи завдання з виявлення, захоплення та автопілотування.

ВИСНОВКИ

В даній роботі було здійснено дослідження питання FPV дронів з використанням машинного зору в поєднанні з автопілотуванням. Також визначено їх актуальність в сучасних реаліях, виявлено їх переваги та недоліки. Проведено аналіз існуючих рішень як технічних так і програмних в сфері програмування FPV. Виявлено їх (рішень) ефективність у різних задачах. На основі отриманих результатів здійснено розробку прототипу, який може ідентифікувати об'єкт за допомогою машинного зору. Також реалізована можливість автопілотування до захопленого об'єкта.

В ході виконання роботи було:

1. Детально проаналізовано системи, які використовуються в FPV для детекції об'єктів, а саме було проаналізовано готові прототипи які вже є у виробництві. Також було визначено які можливі рішення використовується, окрім машинного зору, було досліджено можливість автопілотування FPV які вже є готові рішення та які є в них недоліки, які були дещо виправленні в ході виконання кваліфікаційної роботи та впровадженні інноваційні рішення.
2. Детально досліджено, які є варіанти створення інтеграції FPV з мікроконтролерами для запису на них програмного забезпечення. Також змоги надання команди дрону самостійно без втручання оператора. В процесі дослідження було визначено, що існує варіант з комунікації FPV з мікроконтролером за допомогою протоколу зв'язку MavLink. Також було досліджено вид прошивки, яка найкраще підійде під вирішення даної задачі та легшої інтеграції мікроконтролером.
3. Проведено розробку прототипу FPV з машинними зором на автопілотуванні. Де за основу було взято семидюймову модель FPV та мікроконтролер Raspberry Pi для запису програмного забезпечення. Важливим аспектом у роботі було дотримання основних вимог у конструюванні дрону та створення програмного забезпечення яке

гарантувало коректність роботи та безпеки при подальшій експлуатації дрону.

У підсумку, виконана робота демонструє реальну можливість створення FPV з інтегруванням стороннього програмного забезпечення та успішної роботи даного апарату. Крім цього робота можлива до розширення на рівні серійного виробництва в подальшому майбутньому. Проте для до цього необхідно використовувати іншу модель машинного зору та покращення компонентів збірки FPV. Також дана робота може слугувати доволі непоганим рішенням існуючих проблем в даному напрямку та запропонувати новий погляд на дані технології. Отже, дана робота не лише успішно досягла своїх цілей, але й закладає можливу основу для кращих та більш передових технологій в дроно будуванні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chen, C., Seff, A., Kornhauser, A., & Xiao, J. (2015). DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. IEEE ICCV. - застосування машинного зору в автономному русі (можна адаптувати для дронів).
2. Kim, H. (2021). Programming Computer Vision with Python: Tools and algorithms for analyzing images. O'Reilly Media.- базові інструменти комп'ютерного зору.
3. Korejo, I. A. (2021). Object Tracking for Drone Navigation using Computer Vision Techniques. *MSc Thesis, University of Gothenburg.* - відстеження об'єктів — ключова частина машинного зору на дроні.
4. PX4 Autopilot Documentation. https://docs.px4.io/ - офіційна документація автопілота PX4, одного з найпопулярніших у FPV.
5. Rafi, A., & Islam, R. (2021). Drone Programming: A Developer's Guide to Autonomous Drones Using Python and ROS. Apress. - реальні приклади автономного керування дроном.
6. TensorFlow Lite for Microcontrollers. - глибоке навчання на малопотужних пристроях (наприклад, Raspberry Pi Zero). https://www.tensorflow.org/lite/microcontrollers.
7. Valavanis, K. P., & Vachtsevanos, G. J. (2015). Handbook of Unmanned Aerial Vehicles. Springer. - всебічний ресурс про безпілотні літальні апарати, включно з автопілотуванням.
8. Yilmaz, A. (2020). Vision-Based Autonomous Navigation of UAVs. *Master's Thesis, Technical University of Munich.*- приклад академічної роботи по темі автономного управління дроном.
9. Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as Points. arXiv preprint arXiv:1904.07850.- новітній підхід у розпізнаванні об'єктів.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

РЕБ - скорочено від радіоелектрона боротьба

ARM - умовне позначення в запуску моторів FPV, дослівне значення озброєння

BRG - формати кольору

GPS - супутникова навігаційна система

FPV - дрон моделі first person vision

RGB - формат кольору для приймання моделі