Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня
магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Глоби Віталія Юрійовича*
	(ПІБ)
академічної групи	*121М-22-2*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Розробка та дослідження ефективності впровадження програмного забезпечення на мові програмування Python для безпілотних літальних апаратів*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	Проф. Бердник М.Г.			
Рецензент				
Нормоконтролер	Доц. Гуліна І.Г.			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-22-2* *Глобі Віталію Юрійовичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка та дослідження ефективності впровадження програмного забезпечення на мові програмування Python для безпілотних літальних апаратів*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227 -с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес розробки програмного забезпечення на мові програмування Python для безпілотних літальних апаратів.

Предмет досліджень – методи розробки та впровадження розробленого програмного забезпечення на мові програмування Python для безпілотних літальних апаратів.

Мета НДР – дослідження та вдосконалення системи БПЛА шляхом впровадження в нього розробленого програмного забезпечення мовою прорамування Python.

Вихідні дані для проведення роботи – вимоги до функціональності БПЛА та до програмного забезпечення, методика оцінки ефективності.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень запропоновано методи покращення системи безпілотного літаючого апарата за допомогою впровадження в нього програмного забезпечення для розпізнавання об'єктів.

Практична цінність результатів полягає у тому, що запропоновані в роботі моделі і методи підвищать використання БПЛА в різноманітних галузях.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання програмного забезпечення для БПЛА.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та вимог, постановка задачі	09.10.2023-20.10.2023
Розробка архітектури ПЗ, алгоритмів та вихідного коду. Збірка БПЛА в макетному вигляді для подальшого впровадження розробленого ПЗ	20.10.2023-20.11.2023
Проведення тестів готової моделі БПЛА із ПЗ. Заповнення Звітності та аналіз ефективності впровадження ПЗ для БПЛА мовою програмування Python	20.11.2023-05.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним, завдяки зниженню витрат на розробку ПЗ та стимулюванню зросту даної промисловості та кількості робочих місць у ній.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки тому, полегшенню роботи розробників та працівників у різних сферах.

Завдання видав _____ Бердник М.Г.
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Глоба В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *26.12.2023 р.*

РЕФЕРАТ

Пояснювальна зписка: 82 стор., 48 рис., 1 таблиця, 2 додатки, 50 джерел.

Об’єкт дослідження: процес розробки програмного забезпечення на мові програмування Python для безпілотних літальних апаратів.

Предмет дослідження: методи розробки та впровадження розробленого програмного забезпечення на мові програмування Python для безпілотних літальних апаратів.

Мета роботи: дослідження та вдосконалення системи БПЛА шляхом впровадження в нього розробленого програмного забезпечення мовою прорамування Python.

Методи дослідження. Були використані аналітичні методи при аналізі різноманітної інформації та моделювання та конструюсвання при розробці ПЗ та моделі БПЛА.

Новизна отриманих результатів визначається тим, що запропоновано методи покращення системи безпілотного літаючого апарата за допомогою впровадження в нього програмного забезпечення для розпізнавання об’єктів.

Практична цінність результатів полягає у тому, що запропоновані в роботі моделі і методи підвищать використання БПЛА у різноманітних галузях.

Область застосування. Розроблена модель БПЛА із ПЗ може використовуватись у таких сферах як: воєнна галузь, рятувальні операції, доставка товарів, медицина та багато інших.

Значення роботи та висновки. Розробка ПЗ для БПЛА за допомогою мови програмування Python значно спрощує процес розробки та витрати часу, що проаналізовано та показано в даній роботі.

Прогнози щодо розвитку досліджень. Модель БПЛА, розроблена в кваліфікаційні роботі, може бути покращена методом оптимізації програмного коду та використання більш потужної апаратної частини.

Список ключових слів: БПЛА, PYTHON, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, PIXHAWK CUBE, RASPBERRY PI, NVIDIA JETSON, РУШІЙ.

ABSTRACT

Explanatory note: 82 pages, 48 figures, 1 table, 2 applications, 50 sources.

Object of research: the process of developing software in Python for unmanned aerial vehicles.

Subject of research: methods of development and implementation of developed software in the Python programming language for unmanned aerial vehicles.

Purpose of Master's thesis: research and improvement of the UAV system by introducing into it the software developed in the Python programming language.

Research methods. Analytical methods were used in the analysis of various information and modeling and construction in the development of software and UAV models.

Originality of research is determined by the fact that the methods of improving the system of an unmanned aerial vehicle by introducing software for object recognition into it are proposed.

Practical value of the results is that the models and methods proposed in the work will increase the use of UAVs in various fields.

Scope of application. Field of application. The developed UAV model with software can be used in such fields as: the military industry, rescue operations, delivery of goods, medicine and many others.

The value of the work and conclusions. The development of software for UAVs using the Python programming language greatly simplifies the development process and time consumption, which is analyzed and shown in this paper.

Research forecast and development. The UAV model developed in the qualification work can be improved by optimizing the software code and using more powerful hardware..

Keywords: UAV, PYTHON, SOFTWARE, PIXHAWK CUBE, RASPBERRY PI, NVIDIA JETSON, THRUSTER.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БПЛА – безпілотний літаючий апарат;

ПЗ – програмне забезпечення;

ПК – польотний контролер;

ТТХ – тактико технічні характеристики;

ШІ – штучний інтелект;

ESC – Electronic Speed Controller;

FPS – Frames Per Second;

FPV – First Point of View;

OSD – On Screen Display;

UAV – Unmanned Aerial Vehicle.

ЗМІСТ

ВСТУП

За останні 20 років індустрія безпілотних літаючих апаратів стрімко розвивалася в провідних країнах світу. Із роками БПЛА зменшувались в розмірах, системи керування та навігації удосконалювались, а ціна знижувалась і ось тепер ми можемо побачити застосування даних апаратів в різноманітних сферах нашого життя: фото/відеозйомка – за допомогою БПЛА в кіноіндустрії з'явився ще один інструмент для отримання гарних кадрів із повітря, значну роль дрони відіграли в аграрній сфері для автоматизованого процесу обробки полів. Також БПЛА виявилися корисними для пошуково-рятувальних операцій, у нагляді за лісами, для різноманітних досліджень атмосфери та змін у ландшафті, сфері безпеки та багатьох інших.

Так як Україна – це аграрна країна, то й найбільшого застосування для БПЛА знайшли саме в цій сфері, на другому місці за популярністю застосування була фото/відеозйомка. Після початку повномасштабної війни проти росії усе змінилося і тепер кожного дня українці донатять свої гроші в благодійні фонди, волонтерам та відомим блогерам на FPV дрони-камікадзе, адже за майже 2 роки війни вони довели свою ефективність у цій сфері.

Із кожним днем технології розвиваються і з'являються дрони різних модифікацій із новим функціоналом. Популярні не тільки дрони-камікадзе, а й БПЛА розвідники, за допомогою яких, у реальному часі можна відстежувати пересування ворога, або корегування союзної артилерії.

Впровадження систем штучного інтелекту або комп'ютерного зору сильно розвинуло та дало нові можливості в застосуванні дронів. Із такою тенденцією розвитку даної технології, ми у майбутньому можемо очікувати нових нестандартних та креативних рішень від розробників, аби застосування її було все в більших сферах нашого життя.

Мета дослідження даної роботи є дослідження та вдосконалення системи БПЛА шляхом впровадження в нього розробленого програмного забезпечення мовою прорамування Python.

Завдання дослідження. Для досягнення даної мети були поставлені наступні задачі:

1. Проаналізувати історію та тенденцію розвитку БПЛА.

2. Здійснити детальний аналіз найпопулярніших існуючих рішень.

3. Докладно дослідити застосування різних мов програмування в розробці програмного забезпечення для безпілотних літальних апаратів

4. Розібратися в апаратних та програмних рішеннях впровадження ПЗ для БПЛА.

5. Розробити тестувальний макет майбутньої моделі БПЛА.

6. Створити програмне забезпечення для неї.

7. Зібрати готову модель із впровадженим ПЗ.

8. Провести тести.

Об’єкт дослідження: процес розробки програмного забезпечення на мові програмування Python для безпілотних літальних апаратів.

Предмет дослідження: методи розробки та впровадження розробленого програмного забезпечення на мові програмування Python для безпілотних літальних апаратів.

Методи дослідження: За допомогою аналітичного методу були проаналізовані література та ринок для виявленнях сучасних підходів до розробки ПЗ для БПЛА. Методи моделювання та конструювання були використані для розробки програмного забезпечення та створення робочої моделі БПЛА.

Наукова новизна: запропоновано методи покращення системи безпілотного літаючого апарата за допомогою впровадження в нього програмного забезпечення для розпізнавання об’єктів.

Практична значення: полягає в тому, що запропоновані в роботі моделі і методи покращення БПЛА підвищать їх використання в галузях доставки товарів, фото/відеозйомки, охоронної та воєнної галузі, аграрної та багатьох інших галузях нашого життя.

Особистий внесок автора:

1. Наукові результати роботи отримані автором самостійно.
2. Вибір методів дослдіжень і технологій реалізіцій.
3. Розробка робочої моделі БПЛА.
4. Розробка теоретичної частиви, у якій досліджено БПЛА та методи розробки та вровадження ПЗ для них.
5. Проведення тестуваннь розробленої моделі із ПЗ.
6. Оцінка отриманих результатів.

Структура і обсяг роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і двох додатків. Загальний обсяг роботи становить 82 сторінки, із них 67 сторінок основного тексту, у тому числі 48 рисунків, 1 таблиця і список використаних джерел із 50 найменувань на 6 сторінках.

РОЗДІЛ 1
АНАЛІЗ СУЧАНОЇ ІНДУСТРІЇ БПЛА ТА ІНСТРУМЕНТІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НИХ

1.1 Історія створення та розвитку БПЛА

У 1890-х роках, коли йшла гонка обзроєннь між США, Великобританією, Францією, Німеччиною, Іспанією та Японією американський винахідник Нікола Тесла почав працювати над концептом першого в світі бездротовогу безпілотного апарату (рис. 1.1).

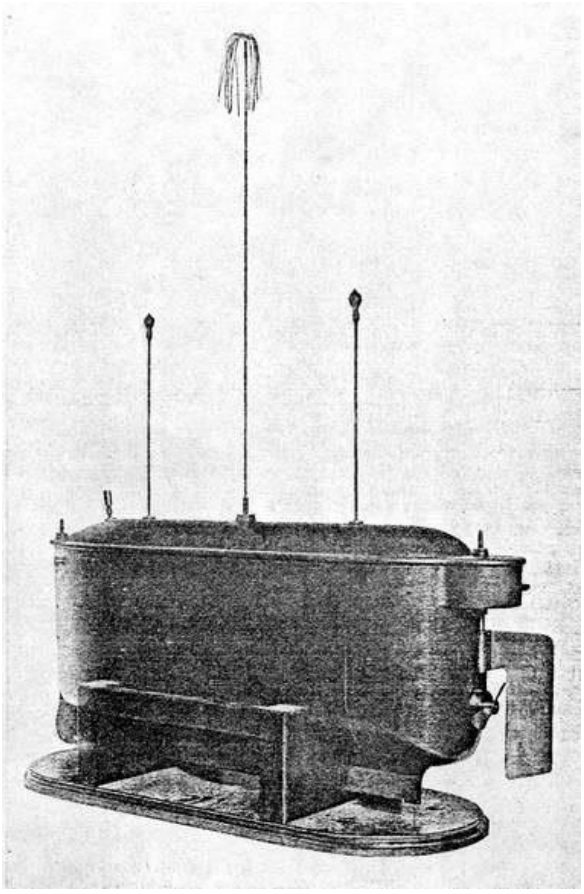

Рис. 1.1. Прототип радіокерованого човна Н.Тесли

Ним був “Devil automata” (автоматичний диявол) (рис. 1.1), як сам називав винахідник, торпедний катер завдовжки 1,2 м., що живився від акумуляторної батареї [7]. Команди він отримував від розподільного пульта, у вигляді коробки

із важелями. Нікола Тесла вірив, що даний винахід може врятувати багато життів, але військові США визнали винахід занадто важким у керуванні[2].

У часи Другої світової війни країни часто використовували некеровані безпілотні апарати, які несли вибухову частину на собі. Яскравим прикладом такого апарату є ФАУ-1 (рис. 1.2), який стояв на обзроєнні у німецької армії[39].

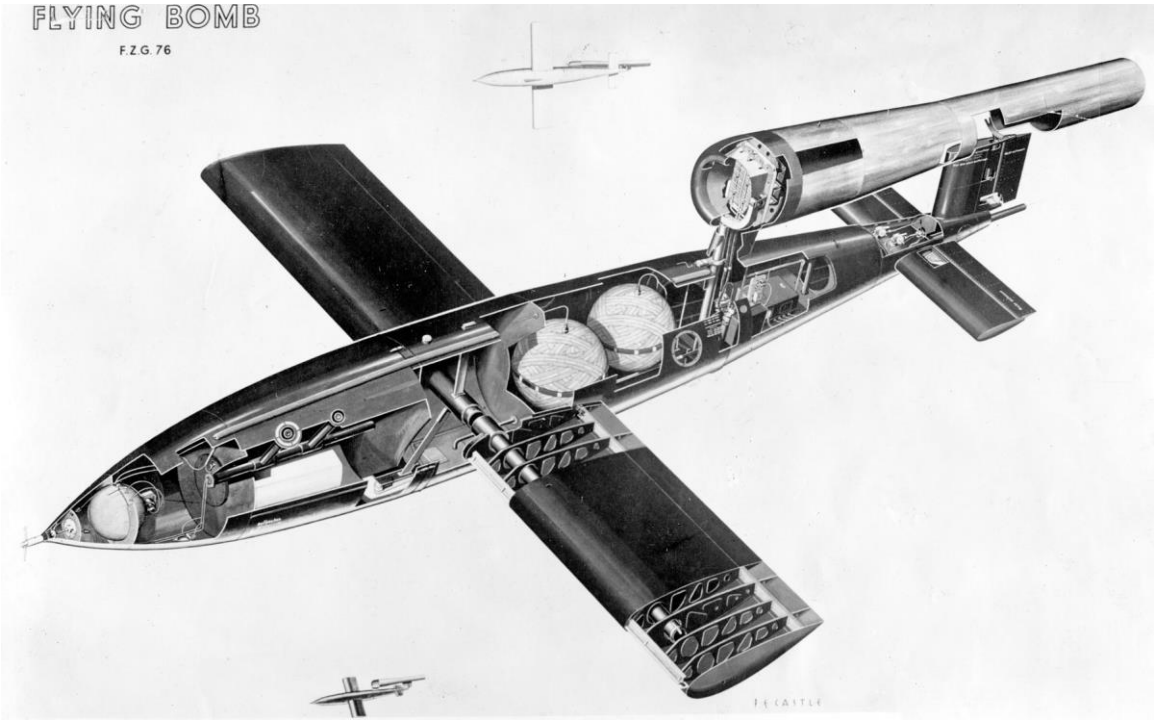

Рис. 1.2. Літак-ракета ФАУ-1

Це приклад першого в світі використання даної технології людиною. За призначенням він був застосований уперше в липні 1944 року під час обстрілу Лондона німецькою армією[28]. Запускався із катапульти з землі, або ж з літака, після запуску автоматично летів до цілі та вибухав. ФАУ-1 відіграв значну роль у ході війни та подальшому розвитку безпілотної авіації. Пізніше були випущені модифікації ФАУ-2, ФАУ-3 та ФАУ-4.

Після цього вчені пішли далі та працювали над безпілотними апаратами, якими можна буде керувати під час польоту, щоб дало змогу підлаштовуватись під конкретні ситуації та збільшити область застосування.

Британці мали в себе на озброєнні надзвичайно успішний літак для навчальних польотів DH.82 Tiger Moth[18]. Британія та країни співдружності

використовували його протягом усієї Другої світової війни. На його базі інженери-розробники вирішили зробити безпілотний радіокерований літак-мішень DH.82 Queen Bee (рис. 1.3).

Рис. 1.3. Офіцери британських ВВС керують DH.82 Queen Bee

Даний вид літака британці використовували для тренування власних сил протиповітряної оборони. Для того, щоб зробити її доступною та економічно ефективною для навчання зенітних сил та пілотів, модель була виготовлена на базі існуючого літака із заміненням матеріалу корпусу на дерево. Успіх даної моделі вказав на потенціал використання безпілотної техніки у військовій справі[34, 33].

Після закінчення Другої світової війни у світі почався розвиток технологій, за допомогою яких дрони почали використовувати не тільки у воєнних цілях, а ще й для різноманітних накових досліджень[28,44], тестувань та розвідки (рис. 1.4).

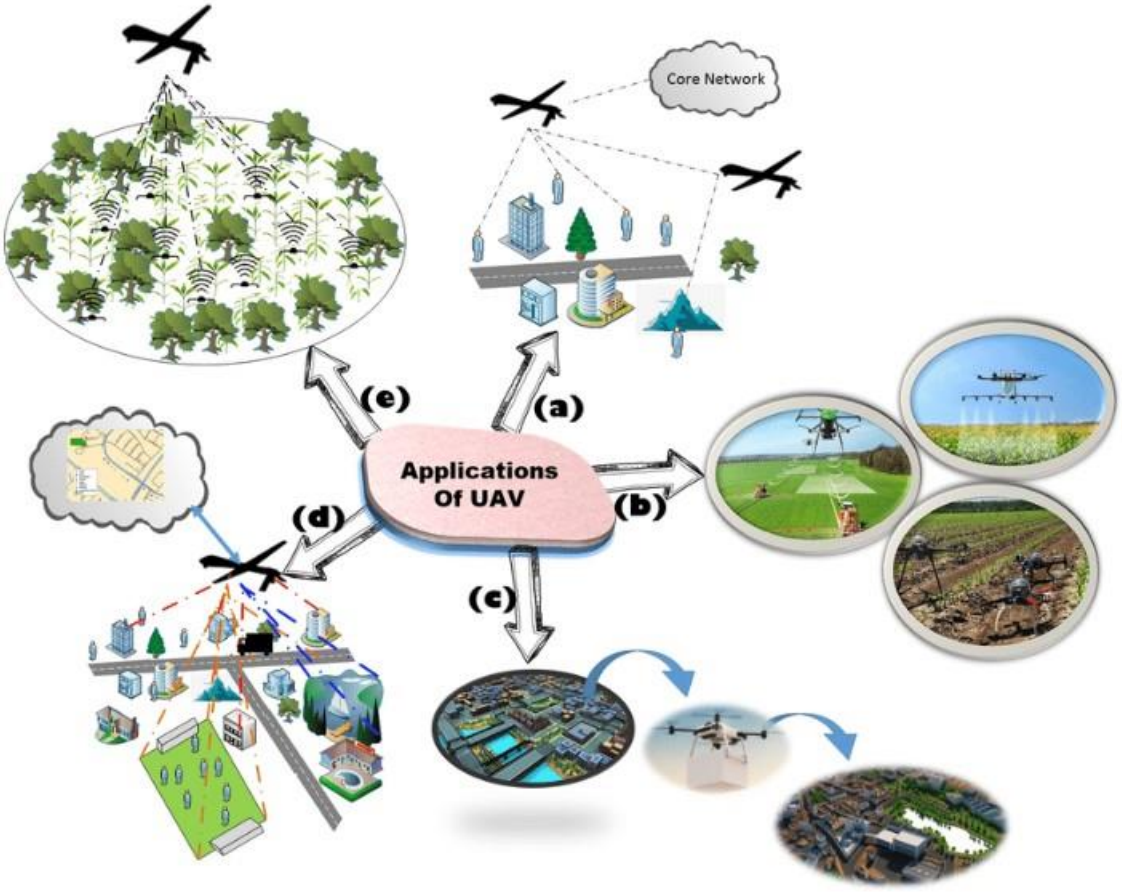

Рис. 1.4. Використання БПЛА в різних сферах

Зараз же для нас дрон не є чимось дивним, або якоюсь секретною воєнною розробкою. Кожен, хто забажає, може піти в магазин та купити його. На рис 1.5 ми можемо побачити на скільки сильно розвиток технології змінив вигляд дрона, у порівнянні із першим БПЛА[22].

Рис. 1.5. Приклад сучасного БПЛА

1.2 Основні види сучасних БПЛА, їх класифікація та область застосування

Із моменту свого виникнення та до сьогодення безпілотні літальні апарати зазнали великих еволюційних змін та вдосконалень. Люди проявили інтерес до цієї технології та намагалися впровадати її в різні сегменти сучасного суспільства. Із кожним десятиліттям видів дронів з'являлося все більше і більше, виходячи із цього, людям потрібно було створювати різні класифікації для них, за допомогою яких можна було б розділяти літальні апарати за певними категоріями[23,35].

Як можна побачити з рис. 1.6 виділяють 4 види дронів: одноротрні, мультирорні, фіксовані крила та гібридні.

Рис. 1.6. Види дронів

Одноротрні дрони – мають у своїй конструкції один ротор, який розташовується вертикально по центру апарата[30]. Зазвичай мають вигляд вертольотного типу, завдяки цієї конструкції мають велику маневреність, що дозволяє легко змінювати напрямок їх руку та використовувати в специфічних місцях.

Багатороторні дрони – до багатороторних дронів відносять дрони, які мають на борту від 4 роторів[29]. У залежності від їх кількості можна розрізнити декілька основних підвидів: квадрокоптери (4 ротори), гексакоптери (6 роторів), октокоптери (8 роторів). Найпопулярнішими серед них є квадрокоптери. У порівнянні із одноторними дронами мають кращу стабільність польоту та маневреність, можуть переносити на собі більше корисного навантаження. Недоліками в порівнянні із одноторними є їх розміри та більша витрата енергії, оскільки потрібно живити мінімум 4 двигуни[48].

Фіксовані крила – представляють собою літальні апарати літакового типу, які мають жорстке фіксоване крило в своїй конструкції. На відміну від одноторних та багатороторних дронів фіксовані крила не мають вертикальних роторів, ротори в них розташовуються горизонтально[45,24]. За допомогою своїх планувальних властивостей літака, вони можуть забезпечувати більшу тривалість польоту та вантажопідйомність із збереженням енергоефективності. Фіксоване крило через свої конструктивні особливості має ряд певних недоліків:

– дороговнизна виготовлення – аби корпус літака мав потрібну міцність та на великій швидкості його не розривало , корпус виготовляють із композитних матеріалів, які мають велику вартість. Також це не швидкий процес, що може тривати від 2-3 днів до тижня, у порівнянні із багатороторними дронами, для яких на виготовлення корпуса достатньо пару хвилин на порізку із листа карбону, або текстоліту;

– складність керування – для того, аби успішно керувати апаратом даного типу, пілот повинен мати високу кваліфікацію, адже крило потребує значну площу злітно-посадкової смуги, має нижчу стабільність польоту, ніж багатороторні системи та має декілька особливостей у процесах запуску та посадки;

– низька ремонтнопридатність – цей пункт виплаває із першого, оскільки, щоб пересадити авіоніку із одного корпусу в інший витратиться багато часу та грошей.

Гібридні дрони – являють собою суміш фіксованого крила та мультироторного дрона. У їх конструкції присутнє фіксоване крило, чотири вертикальних ротори та один горизонтальний ротор[17]. Аби прибрати недолік фіксованого крила в складності взльоту та посадки, до його конструкції було додано 4 вертикальних ротори, за допомогою яких можна було виконувати вертикальний взліт та посадку, а політ виконувався б за допомогою горизонтального ротору[15]. Перехід із вертикального на горизонтальний політ відбувається плавно без додавання в процес керування складного функціоналу. Це підвищило виживаємість моделей та надало нових можливостей у використанні бпла.

Проаналізувавши багато інформації можна виділити наступні основні класифікації БПЛА: за керованістю, за призначенням, за типом використання та за тактико технічними характеристиками[35,23].

За керованістю дрони поділяють на:

– безпілотні некеровані – повністю відсутнє керування під час польоту;

– безпілотні автоматичні – даний вид літальних апаратів має вбудований автопілот та не потребує втручання оператора, але така змога є на випадок екстрених та непередбачуваних ситуацій[18];

– безпілотні дистанційно-пілотовані – політ апарата відбувається повністю під прямим керуванням оператора.

Найбільшого застосування зараз набули дрони саме останнього типу, через свою нескладність у виготовленні та експлуатації.

За призначенням можна виділити наступні категорії :

– ударні БПЛА – головна задача полягає у знищенні різноманітних ворожих цілей. На борту в літального апарата розміщується вибухівка різного типу, яка під час зіткнення із ціллю детонує[14];

– розвідувальні БПЛА – призначаються для ведення розвідки та корегування союзного вогню. Обов’язково повинні бути обладнані гарною

камерою із оптичним збільшенням для забезпечення ведення завдань на великих висотах, аби вони не були помічені противником;

– транспортні БПЛА – дрони, які здатні переносити на собі велику вагу на певну відстань. Деякі екземпляри даного виду можуть транспортувати до 180 кг на відстань 70 км[23,35]. Зазвичай для них використовують багатороторні системи для збільшення стабільності та підйомної ваги;

– гоночні БПЛА – використовуються для проведення FPV гонок на спеціально розроблених трасах із перешкодами. Мають малі габаритні розміри для досягнення максимальної швидкості та маневреності. Оператор керує таким дроном у спеціальних FPV окулярах, що дозволяє йому відтворювати більш точно рухи та маневри[31];

– мультифункціональні БПЛА – вид БПЛА, який можна використовувати для різноманітних задач, адже вони поєднують в собі декілька видів дронів та мають можливість модернізовуватись під певний вид діяльності[1].

За типом використання БПЛА поділяють на:

– військового призначення – використовуються військами для ведення розвідки та бою;

– цивільного призначення – для використання в цивільних сферах життя;

– наукового призначення – для проведення наукових дослідженнь.

Як можна побачити на рис 1.7 БПЛА також класифікують за їх тактико технічними характеристиками. Тільки за даним графіком можна побачити наскільки багато існує варіантів БПЛА, їх видів та характеристик[36].

Рис. 1.7. Класифікація дронів за їх ТТХ

Окрім очевидної воєнної галузі застосування, для якої БПЛА і створювались, розвиток технологій дав можливість людям використовувати їх не тільки в ній, а ще й в цивільних та наукових цілях[43,36]. За допомогою дронів вдалось поліпшити деякі задачі та час на їх виконання, вирішити практичні завдання, які раніше не вдавалось розв’язати по причині її специфічності або відсутності альтернативних засобів для вирішення.

На рис. 1.8. зображено декілька сфер нашого життя, в яких знайшлось місце застосуванню безпілотним літальним апаратам. Присутні сфери як розважального характеру, так і наукових досліджень та рятувальних операцій[49].

Рис. 1.8. Приклад сфер застосування БПЛА

У сфері медицини дрони використовують у важкодоступних або небезпечних для здоров'я людини місцях, таких як зони техногенних катастроф або епідемій. Дрони можуть доставляти обладнання для моніторингу за цими зонами, за допомогою яких учені збирають корисну для себе інформацію із приводу поширення захворювань або для прогнозування епідемій та кризових ситуацій[21]. Також, як і в сфері доставки, дрони можуть доставляти медицинські препарати в певні місця за короткий проміжок часу, що може врятувати людське життя.

Більшого застосування даний винахід набув у агрокультурі, а саме:

– моніторинг стану полів – за допомогою мультиспектральних датчиків та інфрачервоних камер аграрії можуть проводити аналіз стану рослин та вчасно виявляти захворювання, стресові стани рослин та інши фактори, які впливають на родючість;

– оптимізація процесу розпилення добрив та зрошення – за допомогою даних, зібраних БПЛА, фермери можуть бачити, у яких ділянках поля не

вистачає добрива, або воно потребує зрошення. Цей аспект також сильно впливає на урожайність та економію ресурсів[47];

– боротьба зі шкідниками – за допомогою безпілотників фермери можуть розпилювати різні речовини для боротьби зі шкідниками із більшою продуктивністю та з економією на продовольчому та людському ресурсі;

– картографування полів – безпілотний апарат може здійснювати фото/відеозйомку поля з висоти для побудови детальних карт полів. Ці карти допомогають фермерам у інтелектуальному управлінні полями.

За допомогою БПЛА відсоток успішних рятувальних операцій набагато зріс. Дрон, обладнаний камерою із нічним баченням або тепловізором, набагато продуктивніший у пошуку людей уночі[47]. Рятувальники тепер можуть досліджувати значні території за менший проміжок часу, що грає дуже важливу роль у порятунку, бо час іде на хвилини. Із сьогоденних прикладів ми бачимо, як часто за допомогою дронів доставляють їжу або ліки для людей, до яких безпечно дістатись іншим видом транспорту просто неможливо. Також за допомогою них рятувальники можуть оцінювати стан ситуації, масштаб руйнувань або місцевості, в якій буде проводитись порятунок та побудувати подальший план дій.

У кіноіндустрії режисери отримали змогу надавати глядачам кіно із гарними краєвидами, маштабними епічними сценами та сцени у важкодоступних місцях без використання комп’ютерної графіки. У документальних фільмах тепер можна використовувати правдоподібні знімки реальних місць, до яких без дрона людина не змогла б дістатися.

На рис. 1.9 зображений професійний дрон Aerigon MK II, який використовують для роботи із великими промисловими камерами.

Рис. 1.9. Дрон Aerigon MK II для зйомок кіно

Даний дрон приймав участь у зйомках багатькох першокласних фільмах, таких як “Месники: ера Альтрона”, “Спектр”, “Місія нездійсненна: нація ізгоїв” та багатьох інших. Дрони не тільки облегшують зйомку, а й дозволяють отримувати унікальні кадри та візуальні образи, які допомогають глядачу відчувати масштабність подій та підкреслюють динаміку сцен[6].

Важливу роль використання безпілотних апаратів відіграло в різноманітних наукових дослідженнях. Вони були використані в екологічних, археологічних, геологічних, кліматичних, океанографічних та багатьох інших дослідженнях для аналізу певних параметрів у деяких зонах та зборі різноманітних даних про них.

Ці приклади показують наскільки сильно така розробка, як БПЛА полеглує певні процеси в різноманітних галузях та покращує якість кінцевого продукту або явища[37].

1.3 Апаратна складова основних видів БПЛА

Розібравшись із видами БПЛА та проаналізувавши тендецію їх використання були визначені найпопулярніші типи безпілотників, якими є багатороторний дрон модифікації квадрокоптер із 4 роторами та фіксоване крило. Саме для цих двох типів бпла проведемо аналіз апаратної частини для визначення майбутньої моделі літаючуго апарата для впровадження програмного забезпечення.

На рис. 1.10 зображено приклад безпілотного літального апарата типу “квадрокоптер” та його апаратної частини.

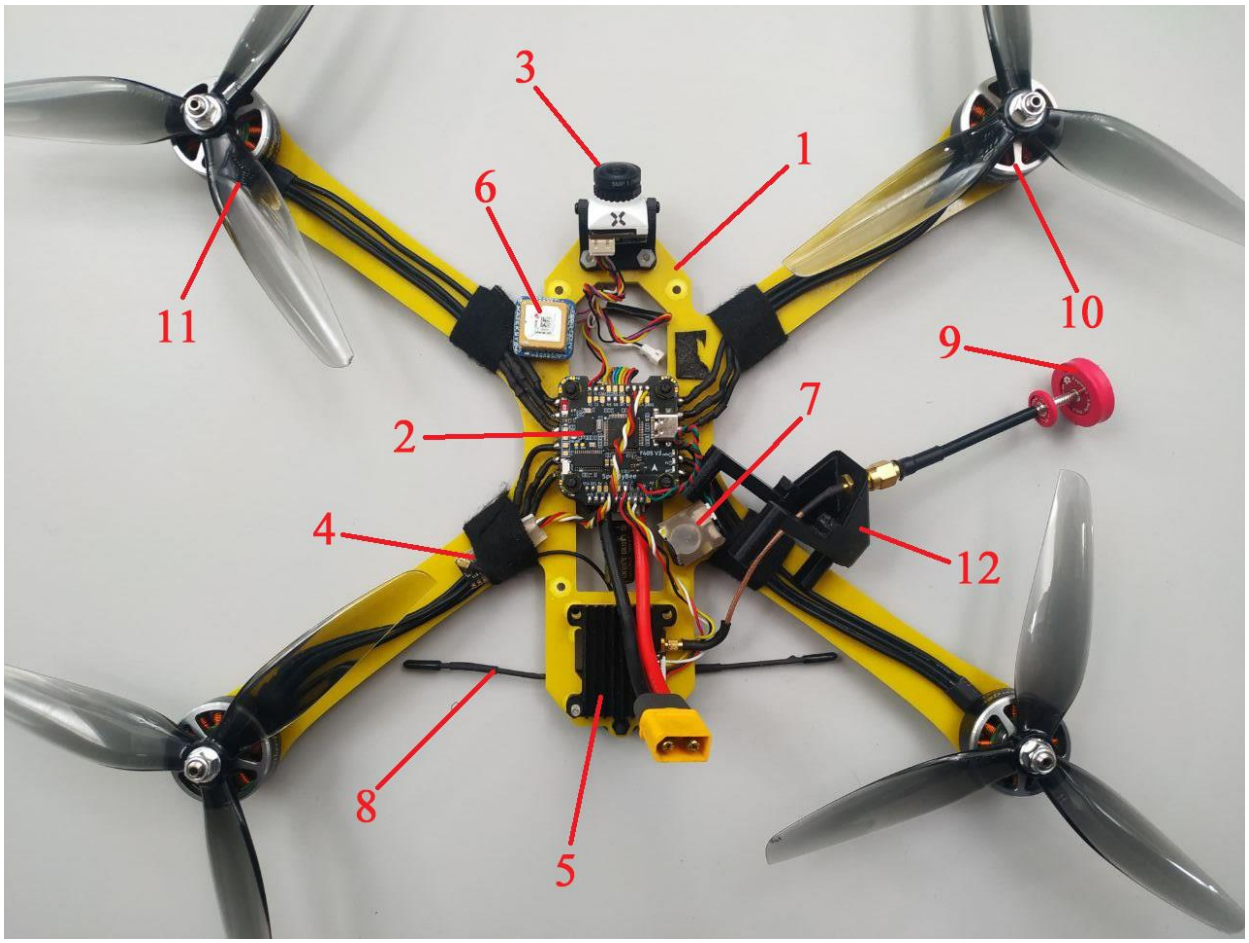

Рис. 1.10. Апаратна частина квадрокоптера

До його апаратної частини входять наступні комплектуючі:

1. Рама – виготовлена із текстоліту, зачасту використовують карбонові рами, які мають більший запас міцності, але при цьому вони коштують дорожче.

За типом рами – це unibody рама типу “X” подовженого вигляду із незйомними лучами, яка є самою популярною. Також існують рами типу “H” та типу “квадрат”.

2. Стек (польотний контролер + регулятор обертів) – зачасту для того, щоб зекономити корисне місце для розташування комплектуючих, якого на дронах і так мало, використовують польотні стеки, які об’єднують у собі відразу дві комплектуючі – польотний контролер та регулятор обертів. Польотний контролер – це центральний елемент управління дроном, він має вбудовані датчики, такі як акселерометр, барометр та магнітометр, за допомогою яких дрон стабілізується під час польоту, проводить логування всіх параметрів з датчиків та записує на накопичувач. Регулятор обертів (ESC) тісно пов’язаний із польотним контролером, він призначений для контролю швидкості, методом подачі певного значення струму на мотори. Команди на подачу він отримує від польотного контролера, назад же він надає до ПК телеметрію про кількість обертань кожного двигуна та температуру. ESC є важливим компонентом, який забезпечує захист від перевантажень, безпеку та стабільність використання дрона.

3. Камера – на більшості моделях використовується аналогова камера, як на рис., оскільки коштують вони набагато дешевше та мають меншу затримку передачі відео, ніж цифрові, що може відігравати важливу роль у певних галузях, наприклад у FPV гонках, де швидкість передачі відео повинна бути максимальною. Цифрові ж камери мають високу якість зображення та розширення відео. Деякі види мають розширений функціонал із автоматичним фокусуванням, запис відео та фото на влаштований накопичувач, багато налаштувань кольоропередачі та інше. Якщо на дроні використовується аналогова камера, то і відеопередавач повинен бути аналогового формату, із цифровими системами так само.

4. Радіоприймач – компонент дрона, який отримує команди з пульта керування із передавачем та передає їх через дріт на польотний контролер. Головним параметром даної деталі є частота, на якій він працює. Наразі є

популярними наступні частоти: 433 мГц, 868-915 мГц та 2.4 ГГц. Щоб система успішно працювала приймач і передавач повинні працювати на одній частоті.

5. Відеопередавач – за допомогою нього картинка передається від камери до оператора із відеоприймачем. Так само, як і камери, існують аналогові та цифрові відеопередавачі із тими самими перевагами та недоліками. Як і у радіоприймача, він має свої частоти, найпопулярнішими зараз є: 1.2 ГГц та 5.8 ГГц. Якщо підключити відеопередавач напряму до камери, то передаватися на приймач буде чиста картинка із камери, якщо ж цю систему підключити через польотний контролер то на зображення камери можна накласти OSD, яке виводить різноманітну корисну інформацію: висоту польоту, gps координати, кількість обертання двигунів і т.п..

6. GPS модуль – за допомогою цього модуля польотний контролер може отримувати свої координати в просторі та виконувати автоматичні польоти по місіям без втручання пілота, або ж у випадку, коли зникне зв’язок керування, так званий failsafe, дрон зможе повернутись у точку взльоту, попередньо записавши її координати.

7. Buzzer – являє собою п’єзодинамік, який може видавати звуки, щоб можна було легше знайти літальний апарат, у випадку його непередбачуваного падіння.

8. Антена радіоприймача – антена усенаправленого типу, існують також направлені антени, за допомогою яких можна добитись зв’язку на більшій дистанції, але на дрони зачасту ставлять усенаправлену, бо дрон летить не тільки в одному напрямку, а постійно маневрує і радіочастотні хвилі потрібні з усіх сторін. Частота антени повинна співпадати із частотою радіоприймача.

9. Антена відеопередавача – так само як і антена радіоприймача, ця антена усенаправлена, але має іншу частоту – 5.8 ГГц, як і відеопередавач.

10. Двигуни – у сучасних дронах використовуються безколекторні двигуни, які прийшли на заміну колекторним. На відміну від них, безколекторні двигуни мають більший коефіцієнт корисної дії, оскільки немає тертя між колекторами. Для своєї роботи вони використовують систему магнітів, які

прикріплені до статора та ротора та за допомогою електронного регулювання мотор починає свій рух. Вони мають дуже високу швидкість обертання, яка залежить прямопропорціонально від напруги батареї. У кожного двигуна є свій власний kv – коефіцієнт обертання на 1 вольт в хвилину, тобто, якщо двигун має 1300 kv та живиться батареєю із напругою 25 V, то в хвилину даний мотор буде виконувати 32 500 обертів.

11. Пропелери – елемент дрона, який відповідає за генерацію тяги, що необхідна для здійснення польоту. Також мають ряд певних характеристик, які сильно впливають на стабільність та продовжуваність польоту. Від кількості лопатей залежить швидкість та стабільність, чим більше лопатей, тим він стабільніший, але повільніший. Також важливими є діаметр та нахил лопаті, діаметр впливає на вагу підйому, а нахил на швидкість. Виробляються із різних матеріалів, таких як: карбон, пластик, дерево та полікарбонат. На мультироторних системах зазвичай використовують пропелери із полікарбонату, оскільки вони не дорогі та мають потрібну міцність.

12. 3Д моделі для кріплення антен – виготовляються із пластику на 3д принтері, слугують для більш зручного монтажу певних компонентів та для економії місця.

13. Акумуляторна батарея – важливий та невід’ємний компонент БПЛА, від якого залежить час польоту та швидкість. Виготовляють батареї зазвичай або з liion або з lipo акумуляторів. Батарея із liion компонентів має меншу вартість та вагу при тій самій ємності, але на відміну від lipo мають невелику струмовіддачу. Зазвичай використовують саме liion збірки акумуляторів, бо вони дешевші та спаявши паралельно декілька збірок можна отримати більшу ємність, що забезпечить збільшення часу польоту.

На рис. 1.11 зображено приклад безпілотного літального апарата типу “фіксоване крило” та його апаратної частини.

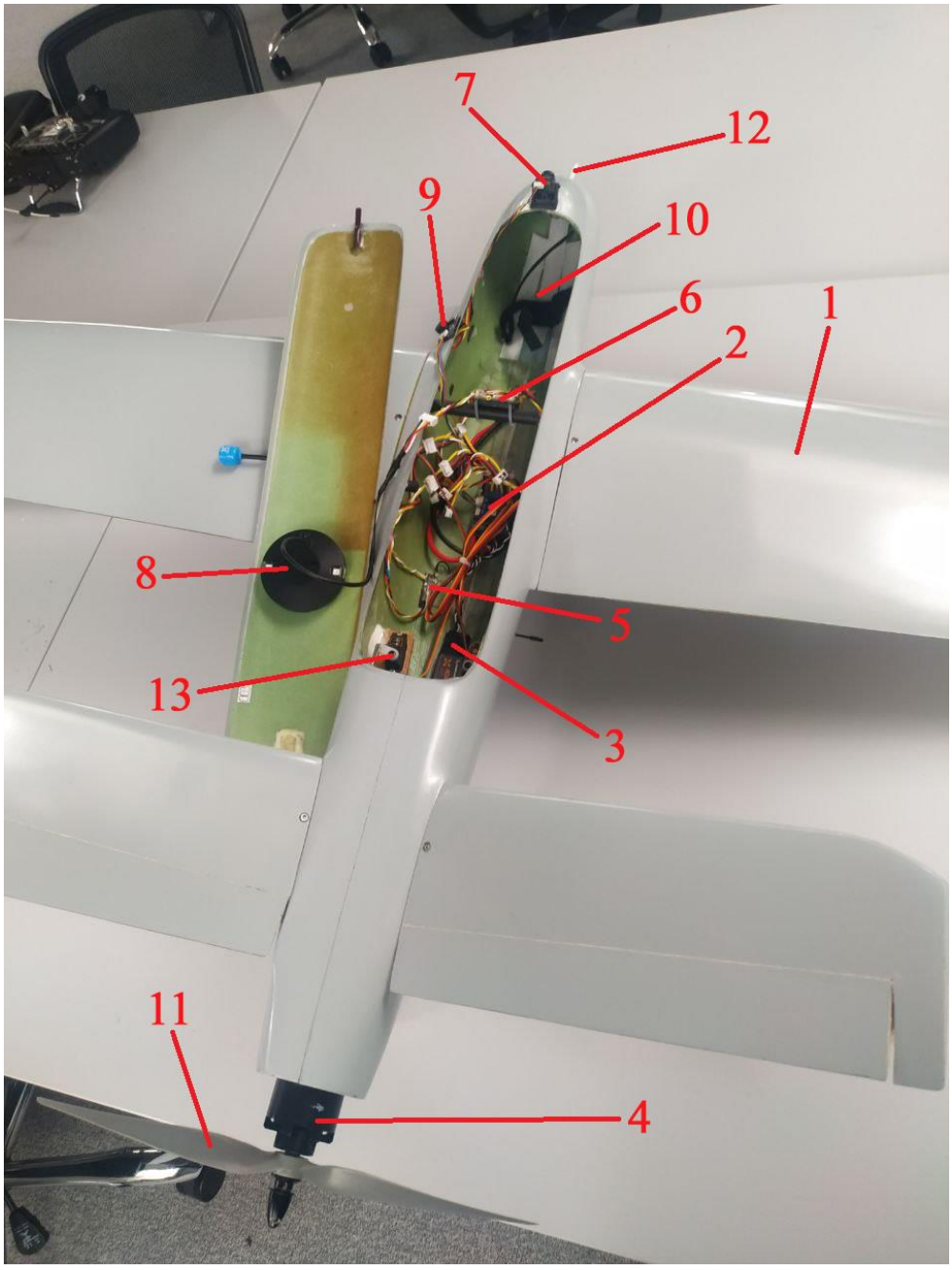

Рис. 1.11. Апаратна частина літака

Як і багатороторні системи фіксовані крила мають майже ті самі компоненти: ПК, ESC, передавачі і т.п., але є декілька певних відмінностей, які розберемо за допомогою рис. 1.11:

1. Корпус – зазвичай виготовлений із композитних матеріалів для міцності, але бувають моделі із дерева, пінопласта, піноплєкса, картона і т.п.. Корпус відіграє дуже важливу роль у керованості моделі, її стабільності та

тривалості польоту. Він повинен бути виготовлений із дотриманням усіх правил та пропорцій.

2. Польотний контролер – майже нічим не відрізняється від польотних контролерів, що використовуються на мультироторних системах, окрім кількість виходів PWM портів.

3. Регулятор обертів – на літаках використовують регулятор обертів, як окремий компонент, а не як польотний стек, оскільки використовується менше двигунів (1 або 2) із більшим споживанням струму.

4. Двигун – на дронах літакового типу використовуються більші двигуни із меншим kv для збільшення тяги апарата.

5. Радіоприймач із антеною.

6. Відеоприймач із антеною.

7. Камера – літак завдяки своїм розмірам та більш стабільного польоту дозволяє використовувати так звані gimbal камери із стабілізацією та зумом, які за розмірами у декілька разів більші ніж ті, що на квадрокоптерах. За допомогою них можна отримувати найякіснішу картинку із високою роздільною здатністю.

8. GPS модуль.

9. Buzzer.

10. Акумуляторна батарея.

11. Пропелер – у дорогих моделях літаків використовують пропелери, виготовлені із дерева, які надають більшої ефективності та стабільності під час польоту.

12. Датчик повітряної швидкості – цей датчик використовує принцип динамічного тиску для виміру швидкості повітря. Використовується в навігаційних режимах для корегування швидкості літального апарата.

13. Сервоприводи – завдяки ним виконується механічне керування літаком, через PWM вихід на них поступає команда від польотного контролера і виконується рух елеронів – для керування креном літака, елевонів – для керування кутом атаки та обертанням навколо осі та кермом висоти – для керування тангажом.

1.4 Існуючі рішень у розробці та впровадженні ПЗ для БПЛА

У залежності від конкретних вимог та задач до безпілотника та його програмного забезпечення розробники використовують різні мови програмування та підходи.

Одним із інструментів для розробки програмного забезпечення для дронів є відкритий фреймворк , що включає в себе багато інструментів та бібліотек. Як можна побачити на рис.1.12 дана ОС використовується не тільки для розробки ПЗ для дронів, а й для різноманітних роботів, роботизованих систем, лідарів і т.п..

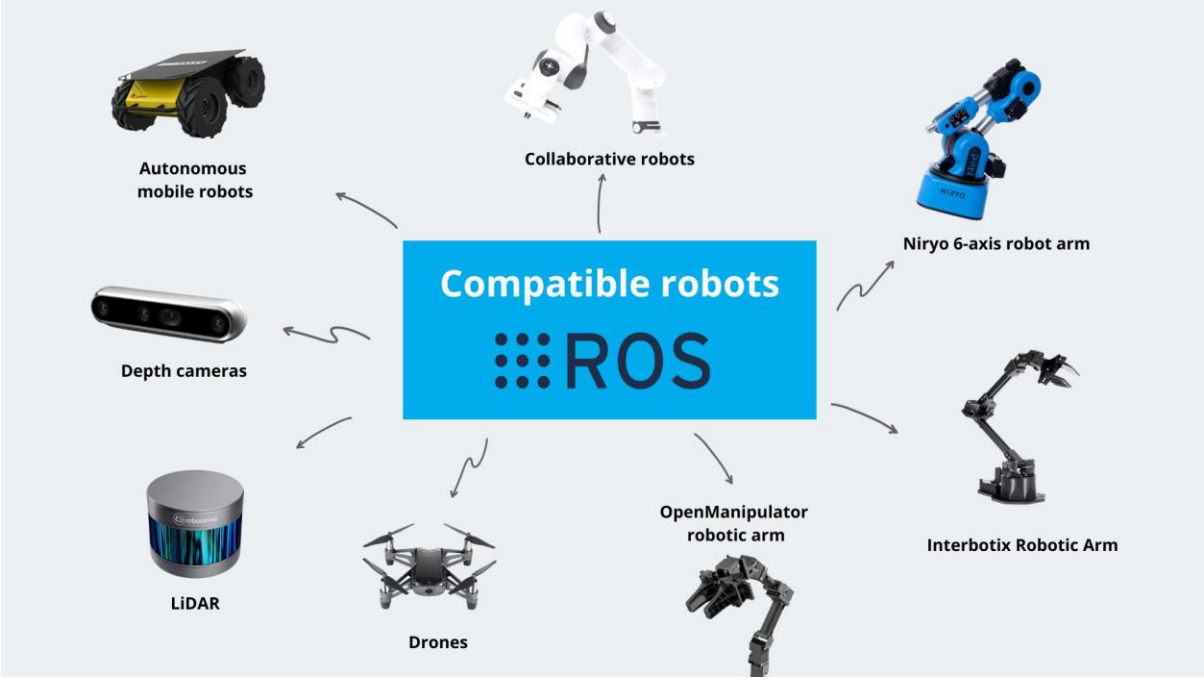

Рис. 1.12. Фреймворк ROS та сфери його використання

ROS підтримує розробку на різних обчислювальних платформах, що додає їй універсальності у використанні[19]. Важливим чинником, завдяки якому розробники надають свою перевагу саме цьому інструменту для розробки, є те, що даний фремворк містить у собі багато популярних бібліотек, які широко використовуються, а саме: OpenCV – бібліотека, яка містить алгоритми комп’ютерного зору та для обробки зображень, PCL – бібліотека для роботи із

хмари точок 3Д геометрії, Ogre – об’єктно-орієнтований графічний рушій, Orocos – бібліотека для керування роботами. ROS підтримує розробку на мовах програмування C++ та Python[27].

Також даний фреймворк є широковикористовуваним у розробці ПЗ для БПЛА, оскільки:

– ROS можна інтегрувати із платформаю для автопілота дрона PX4;

– ROS має стандартний набір алгоритмів та пакетів для навігації та планування маршрутів та місій;

– пакет MAVROS (Mavlink & ROS) забезпечує використання протоколу телеметрії Mavlink для обміну даних між дроном та наземною станцією;

– функціонал даного фреймворку підтримує симуляції для тестування розробленого ПЗ без використання реальної моделі дрона, що підвищує продуктивність даного процесу.

Незважаючи на великий обсяг інструментів та бібліотек для розробки, дана система має декілька недоліків та проблем із якими зіштовхнеться розробник під час використання, а саме:

– дана система потребує значних ресурсів від апаратної частини, оскільки має велике нагромадження інструментів та високий рівень абстракції, що може стати проблемою для безпілотників, оскільки їх ресурс обмежений простором, де можна розмістити апаратну частину, та великими енергозатратами;

– ROS не має вбудованих механізмів для забезпечення безпеки програмної частини, що змушує витрачати багато ресурсів на самостійну розробку та впровадження даних систем[32];

– поріг входження у дану ОС є завеликим для новачків та невеликих проєктів. Можуть виникнути проблеми із встановленням та налаштуванням різноманітних пакетів, бо процес не інтуїтивний, також система має складний процес відкладки, через велику кількість взаємопов’язаних вузлів.

Іншим засобом для розробки та вровадження ПЗ для БПЛА є інструментарій від усесвітньо відомої компанії з виробництва БПЛА та різноманітних систем для них DJI (рис. 1.13). Даним інструментарієм є набір для розробки ПЗ DJI SDK[11]. Для роботи із ним доведеться використовувати мову програмування Java.

Рис. 1.13. Логотип компанії DJI

Перевагами даного інструментарію є:

– підтримка роботи із багатьма моделями DJI – даний інструментарій може допомогти із розширенням можливостей ваших дронів DJI, таких як: Mavic, Phantom Spark та інші;

– за допомогою DJI SDK можна керувати польотною системою апарата, вносити зміни в обмеження по певним даним, розробляти автономні місії та інші операції;

– також за допомогою неї можна збирати дані з різих сенсорів, які вбудовані в БПЛА для подальшого аналізу;

– можливість керувати апаратом віддалено – за допомогою даного набору інструментів розробники можуть створювати застосунки для віддаленого керування дронами.

Із недоліків даної системи можна відмітити наступні:

– головним недоліком даного інструментарію є те, що використовувати його можна тільки із апаратами компанії DJI, із дронами від усіх інших виробників дану систему використати ніяк не вдасться;

– хоч DJI SDK і має багатий інструментарій, але велика частина із нього або платна, або може не підтримуватись певними дронами через свою функціональну та апаратну особливість;

– нестабільні оновлення та ліцензійні обмеження – майже кожне нове оновлення інструментарію DJI SDK та вбудованого ПЗ працює нестабільно та може призвести до некоректної поведінки дрона. Також можуть виникнути проблеми із використанням розробленого ПЗ за допомогою даного набору через ліцензійні обмеження.

Ще одним можливим рішенням із розробки програмного забезпечення для безпілотників є поєднання Matlab та Simulink (рис. 1.14).

Рис. 1.14. Логотип Matlab & Simulink

Поєднання Matlab та графічного середовища Simulink також може бути використане для розробки та покращення програмного забезпечення для БПЛА.

За допомогою даних середовищ аналізу та розробки ПЗ можна створювати застосунки для аналізу даних польоту, різноманітних даних взятих із сенсорів, створювати моделі польотів та навігації[12,20]. Але через велику кількість недоліків даного поєднання, таких як: потреба у великій кількості ресурсів, висока вартість ліцензії програмного продукту, обмежена швидкість та закритість коду, даний інструментарій не користується великим попитом серед розробників для створення програмного забезпечення для безпілотників.

1.5 Постановка задачі

Апаратна частина моделі БПЛА повинна задовольняти таким умовам:

1. Корпус повинен бути із міцного матеріалу для надійності моделі.
2. Акумуляторна батарея повинна бути великої ємності (від 4000 мАг) для забезпечення довгої тривалості роботи.
3. Безпілотник повинен мати на борту наступні датчики та сенсори: барометр, магнітометр, GPS модуль та п’єзоелемент (buzzer).
4. Камера повинна забезпечувати гарну якість зображення для корректної роботи алгоритмів розпізнання об’єктів.
5. Повинна використовуватись надійна система радіокерування, така як: Crossfire або ELRS.

Програмна частина моделі БПЛА повинна задовольняти наступним умовам:

1. Алгоритми розпізнавання повинні визначати наступні об’єкти : люди, машини та дрони.
2. Розпізнавання об’єктів повинно успішно працювати від двох метрів.
3. Активація функцій розпізнавання обличчя та об’єктів повинні виконуватись із пульта керування.
4. Можливість виконувати автоматичні місії без втручання пілота;

5. Оброблене відеозображення алгоритмами комп’ютерного зору повинно передаватись на персональний комп’ютер оператора та підтримувати частоту зображення в 20 FPS.

1.6 Висновки

У першому розділі було проведено дослідження історії створення та розвитку БПЛА, розглянуті перший прототип безпілотного літаючого апарата “Devil automata”, розробником якого був Нікола Тесла, перші некеровані літаючі апарати, які використовувались для нанесення ударів по ворожій техніці та першого у світі бездротового радіокерованого безпілотника, який Британія використовувала для навчання своїх зенітних військ.

Також були проаналізовані види сучасних БПЛА, а саме: одноротрні та багатороторні дрони, фіксовані крила та гібридні дрони, які поєднують у собі декілька видів. Також були розібрані основні класифікації дронів: за керованістю, за призначенням, за типом використання та тактико технічними характеристиками. Дослідивши обсласть застосування даної технології, було виявлено, що застосування їй знайшлось у багатьої головних сферах нашого життя. БПЛА стали не тільки воєнною зброєю чи розвагою для дітей, вони ще й здатні оптимізувати деякі процеси та покращити кінцевий продукт у певних галузях.

Провівши аналіз апаратних складових найпопулярніших видів БПЛА, якими є : мультироторний дрон типу квадрокоптер та фіксоване крило, були виявлені основні відмінності між ними. Даний аналіз допоміг у визначенні майбутньої моделі, яка буде використовуватись для впровадження розробленого програмного продукту.

Розглянувши та дослідивши існуючі рішення по розробці та впровадженню ПЗ для БПЛА були виявлені переваги та недоліки наступних інструментів розробки: фреймворку ROS, інструментарію від компанії DJI для їхніх дронів та інтеграцію Matlab & Simulink.

На останок була проведена постановка задачі, за допомогою якої були розроблені умови, яким повинні задовольняти апаратна та програмна частини розробленої моделі БПЛА.

РОЗДІЛ 2
АНАЛІЗ АПАРАТНИХ І ПРОГРАМНИХ РІШЕНЬ ДЛЯ БПЛА

2.1 Системи керування БПЛА Ardupilot

Ardupilot – це апаратно-програмна система із відкритим кодом, що використовується для керуванням різними аспектами польоту безпілотників майже усіх видів (рис. 2.1). Реліз відбувся у 2009 році тоді парою ентузіастів, зараз же Ardupilot має велику спільноту розробників, яка постійно вдосконалює свій програмний продукт[3].

Рис. 2.1. Різновиди системи Ardupilot

На рис. 2.1 зображені приклади різновидів системи Ardupilot, яка може підтримувати різні автопілоти для БПЛА, такі як: літаки, фіксовані крила, одноротрні системи, усі різновиди мультироторних систем, трекінгові системи, машинки на радіокеруванні, лодки і т.п. Саме через такий великий список підтримуваних систем Ardupilot зібрала навколо себе величезну спільноту користувачів, які кожного дня знаходять їй нове використання.

Із табл. 1.1, яка наведена нижче, ми можемо побачити, що Ardupilot переважає свої аналоги майже по всім параметрам порівняння. Він має ширший спект підтримуваних апаратів, до яких входять: квадрокоптери, гексакоптери, окотокоптери, планери, гелікоптери, мультикоптери, підводні дрони, автономні транспортні засоби та антенні трекери, аналоги ж підтримують максимум конфігурації квадрокоптерів, гексакоптерів октокоптерів та планерів[26].

Також на відміну від аналогів він має відкритий програмний код, що дозволяє розробникам мати більш гнучкий інструментарій для розробки, який можна модернізувати і впровадити власне програмне забезпечення. Також це дає перевагу в тому, що ти не залежиш від розробників та їх можливих заборон та обмежень для комерційного використання розробленого продукту.

Хоч і завдяки всім порівнюваним системам можна змінювати різні налаштування автопілота БПЛА, Ardupilot все ж має більш тонкі налаштування. Планувальник місій відразу містить уже кілька готових сценаріїв місії, які можна підлаштовувати під власний БПЛА, а саме: сценарій для розрошення різноманітних добрив для поля, сценарій для проведення аерофотозйомки та ще декілька сценаріїв. Також важливим фактором є підтримка інерційної системи навігації БПЛА[3].

Із недоліків у порівнянні із аналогами можна виділити тільки “не дружелюбний” інтерфейс застосунку та вискорий поріг входження в систему у порівнянні із аналогами, через що у користувачів може виникнуть проблема із пошуком потрібних функцій та налаштувань, особливо таких важливих як: калібрування усіх датчиків, налаштування місій і т.п..

Таблиця 1.1

Порівняльна таблиця системи Ardupilot із її аналогами за різноманітними параметрами

№ п/п	Параметр для порівняння	Ardupilot	Betaflight	Inav
1	К-ть підтримуваних видів автопілота	9	3	4
2	Наявність навігаційних режимів польоту	+	+	+
3	Наявність планувальника для автономних польотів	+	–	+
4	Відкритий програмний код системи	+	–	–
5	Можливість впровадження стороннього ПЗ	+	–	–
6	Тонкі налаштування конфігурації автопілота	+	–	–
7	Підтримка телеметрії	+	+	+
8	Можливість віддаленого керування БПЛА	+	–	+
9	Точне визначення позиції за допомогою додаткових сенсорів	+	–	–
10	Підтримка інерційної системи навігації	+	–	–
11	Підтримка gimbal камер	+	–	–
12	Інтуїтивно зрозумілий для користування інтерфейс застосунку	–	+	+
13	Велика спільнота користувачів	+	+	+
14	Оцінка складності входження в систему (де 1 – дуже легко, 5 – дуже складно)	4	1	2

Для користування системою керування Ardupilot потрібен графічний застосунок Mission Planner (рис. 2.2). Він допомагає взаємодіяти із

функціональним набором системи Ardupilot та відносно зручно налаштовувати конфігурації різних сумісних автопілотів.

Рис. 2.2. Логотип застосунку Mission Planner

На рис. 2.3 можемо побачити вкладку застосунку “Data”, яка відкривається після запуску програми і є головним її екраном. На цій вкладці розташовано 3 екрани із різноманітною інформацією, зверху зліва розташовані інші вкладки застосунку, справа зверху парамерти та методи підключення до автопілота та кнопка під’єднатись.

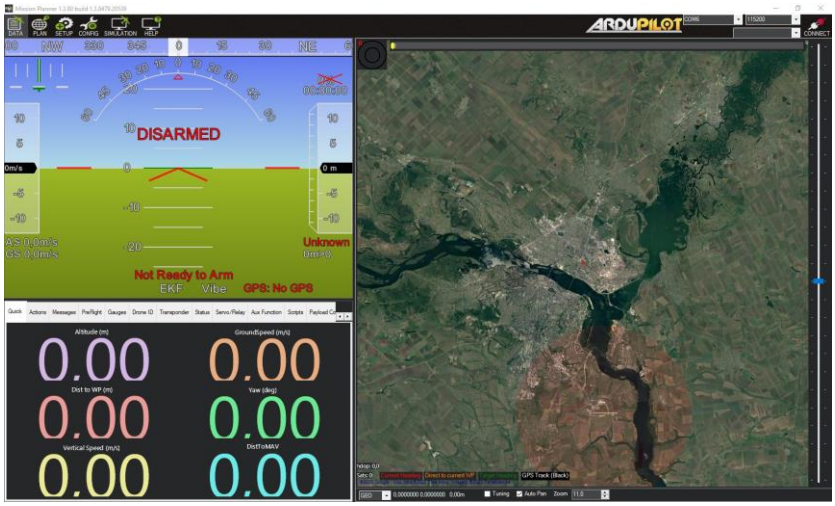

Рис. 2.3. Головний екран застосунку Mission Planner

Дана вкладка призначена для використання під час польоту на БПЛА, оскільки у нас на екрані відображаються дані телеметрії з борта: його висота, швидкість відносно землі, дистанція до точки назначення, градуси крену та

тангажу та інші, також, маючи систему GPS на борту ми можемо в реальному часі відстежувати в якій точці зараз знаходиться наш апарат. За допомогою даної вкладки також можна проводити передпольотну перевірку БПЛА, аналізувати повідомлення від польотного контролера про помилки, які не дають заармити модель, продивлятися статус усіх датчиків та сенсорів, які підключені до автопілота, проводити логування даних та їх аналіз та багато іншого.

Ardupilot підтримує різні способи підключення до літального апарата (рис. 2.4), як дротові так і бездротові. Розглянемо кожен із них:

– COM (Serial) – це тип підключення до автопілота через послідовний порт за допомогою серійного з'єднання. Це стандартний метод підключення до усіх польотних контролерів для подальшого їх налаштування. Потрібен USB кабель для підключення через даний метод;

– TCP (Transmission Control Protocol) – бездротовий метод підключення до автопілота, який використовує TCP мережу для взаємодії Ardupilot та польотним контролером;

– UDP (User Datagram Protocol) – також бездротовий протокол передачі даних, який працює без встановлення з'єднання та відправляє пакети із даними, не очікуючи відповіді, за допомогою чого досягається велика швидкість, у порівнянні із TCP, але страждає надійність та безпека;

– UDPCI (UDP Control Interface) – протокол, який використовує UDP, але із додатковим інтерфейсом для керування БПЛА;

– WebSocket – протокол для взаємодії веб-застосункам із системою через інтернет.

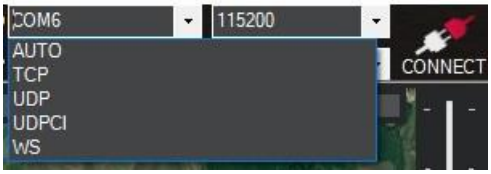

Рис. 2.4. Варіанти підключення до автопілота

Для створення, налаштування та виконання автоматичних польотних місій потрібно перейти на вкладку "Plan" (рис. 2.5). Ні один застосунок не

зрівняється із функціоналом планувальника місій Ardupilot`a. Користувач може назначати точки маршруту для майбутньої місії методом натискання на певну ділянку на мапі, або увівши координати потрібної точки. Також дані маршрутні точки користувач може налаштовувати, змінюючи такі параметри: як висота польоту, швидкість, орієнтація БПЛА та інше. При досягненні кожної із точок можна обрати дію, яку виконає безпілотник: змінити висоту, пришвидшиться, зробити знімок камерою або ж запустити сервопривід, за допомогою якого відкриється ємність із якосюь рідиною для розпилення.

Рис. 2.5. Приклад автоматичної місії польоту

На рис. 2.5 зображено приклад місії для обстеження певного регіону, який було обмежено границями (червоними мітками на мапі) та здійснення аерофотозйомки в певних точках даного регіону. Це один із прикладів примітивної місії для БПЛА, які підтримує система Ardupilot.

Для змінення конфігурації польотного контролера, взаємодії із його апаратною частиною, яка до нього під’єднана, використовують вкладки “Setup” та “Config”, які зображені на рис. 2.6 та рис. 2.7 відповідно. Тут відбувається

вся головна робота із автопілотом, а саме: калібрування датчиків авітопілота зручним способом, назаначення різної периферію на відповідні UART порти польотного контролера, налаштування польотних режимів, вибір функцій, які будуть виконувати сервоприводи та багато іншого.

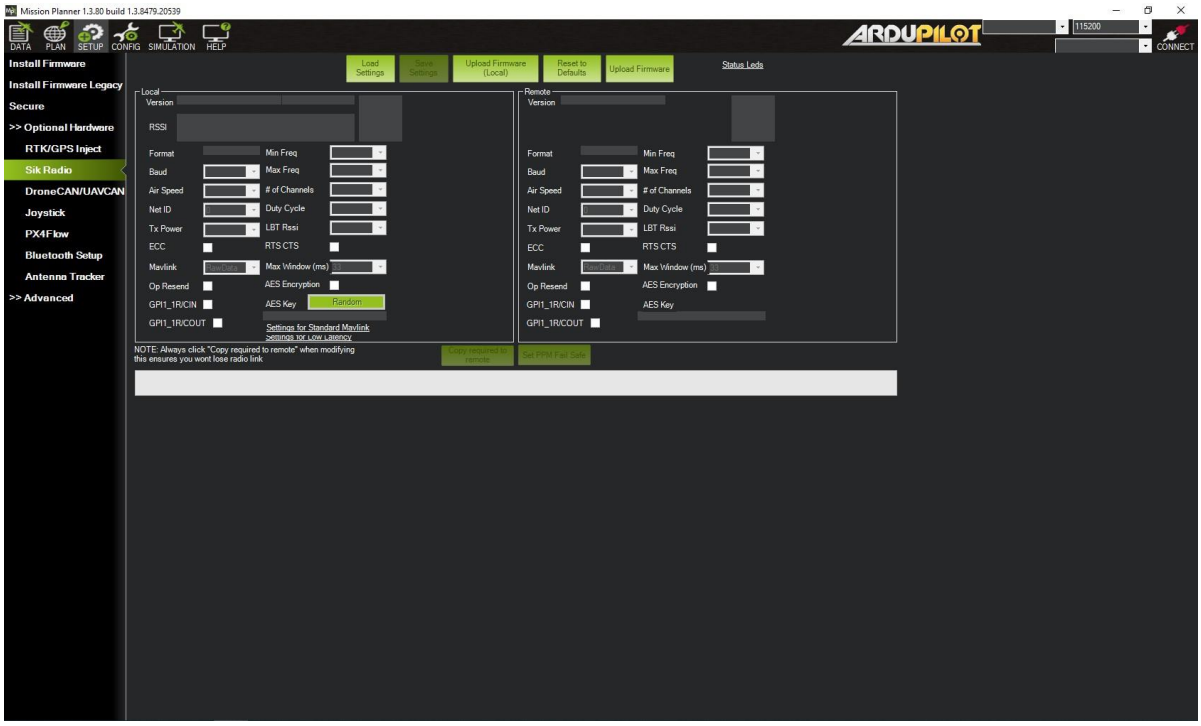

Рис. 2.6. Вкладка “Setup”

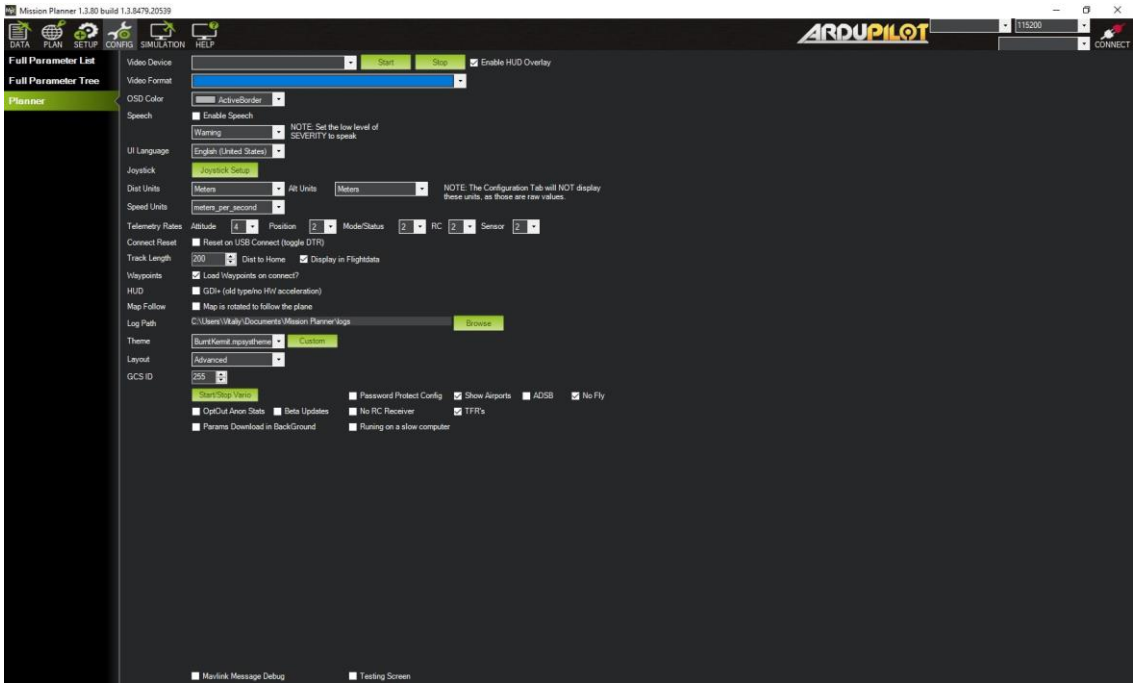

Рис. 2.7. Вкладка “Config”

Ще однією корисною функцією Ardupilot є режим симуляції (рис. 2.8), у якому можна змоделювати політ безпілотного літаючого апарата без фактичного польоту. Цей функціонал дуже сильно підвищує продуктивність налаштування бпла та розробки ПЗ для нього, оскільки:

– можна обрати конфігурцію свого БПЛА із усіма сенсорами та датчиками, які на нього встановлені та не ризикуючи апаратом провести тестовий польот;

– швидкий спосіб перевірки програмного забезпечення при його розробці – є дуже важливим аспектом, при розробці ПЗ, оскільки розробник може перевіряти його, не відходячи від свого робочого місця та не витрачаючи час на поїздку на тестувальний майданчик;

– моделювання аварійних ситуацій та аналіз поведінки БПЛА в них – допомагає розробнику та оператору змоделювати можливі сценарії певних аварійних ситуацій під час польоту, для розуміння поведення БПЛА в них та прийняття відповідних рішенньь для їх вирішення;

– симуляції використання датчиків та сенсорів – можна протестувати БПЛА із комплектуючими, яких фізично не має в наявності у розробника.

Для впровадження програмного забезпечення для БПЛА Ardupilot містить тонкі налаштування серійних портів польотного контролера безпілотника, що надає змогу підключити до нього такий пристрій як Raspberry Pi, або ж Nvidia jetson для обробки програмних алгоритмів.

Застосунок Mission Planner містить вбудований термінал (рис. 2.8), за допомогою якого розробник може взаємодіяти із розробленими скриптами, налаштуваннями свого безпілотника та надавати доступ до додаткових налаштувань.

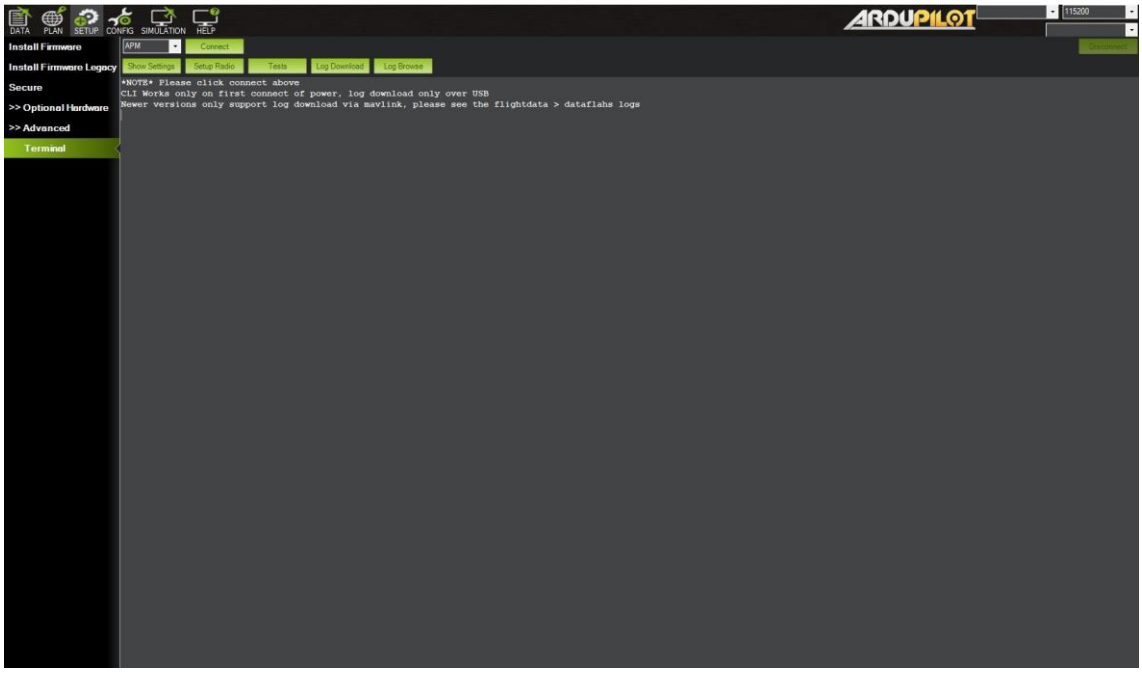

Рис. 2.8. Вбудований термінал Mission Planer

Також у системі Ardupilot є змога налаштуванню радіоканалів не тільки для керуванням дрона, а ще й для активації власних розроблених скриптів, за допомогою яких можна активовувати алгоритми розпізнавання обличчя чи інші алгоритми. Ardupilot вирішує поставленні задачі до функціональності майбутного продукту, а саме: забезпечить стабільний політ за допомогою правильного та тонкого налаштування автопілота та PID регулятора, забезпечить керування та активацію функції розпізнавання обличчя зі звичайного пульта керування TX16S, а також впровадить у БПЛА автоматичні місії.

2.2 Raspberry Pi, як головний рушій програмного продукту

Raspberry Pi – серія одноплатних комп’ютерів розроблених британським фондом Raspberry Pi Foundation[4]. Даний комп’ютер має низьку вартість та малі габаритні розміри, що зробило його чудовим рішенням для використання в робототехніці, навчанні школярів програмуванню та електронній інженерії[25]. Не дивлячись на маленькі розміри даного апаратного рішення, він має напрочуд великий запас потужності, за допомогою якої можна розробляти наступні речі:

– пристрої IoT для віддаленого моніторингу;
– створення розумних роботів;
– автономні транспортні речі;
– система розумного дому;
– система моніторингу сну;
– Web сервіси;
– робот-пилосос;
– Desktop комп’ютер;
– системи для автоматизації поливу рослин;
– медіацентри.

І це тільки малий список проєктів, які можна реалізувати за допомогою Raspberry Pi (рис. 2.9).

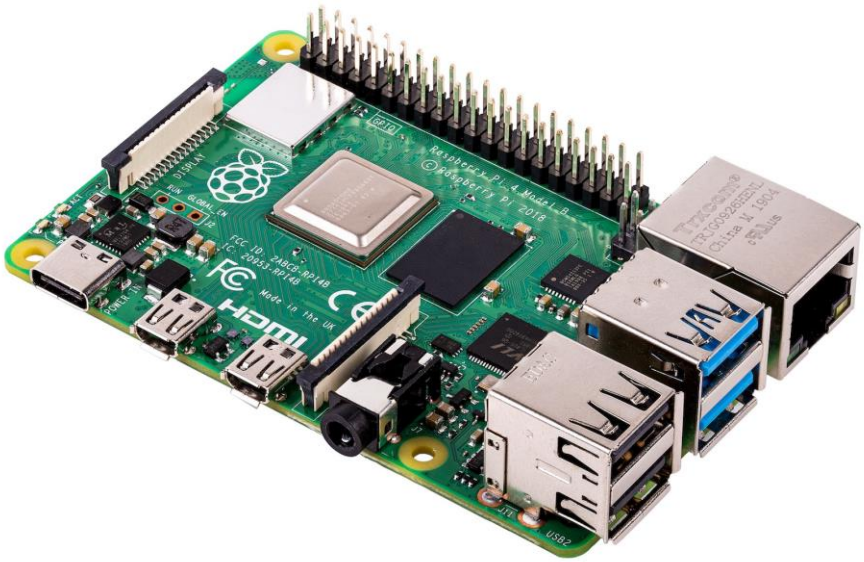

Рис. 2.9. Raspberry Pi 4

Наразі існують наступні модифікації Raspberry Pi:

– Raspberry Pi model B – був випущений у 2012 році;
– Paspberry Pi model B+ – був випущений із покращеним дизайном у 2014 році;

– Raspberry Pi 2 – мав 32-розрядний 4 ядерний процесор 900 МГц, випущений у 2015 році;

– Raspberry Pi 3 Model B – із покращеним процесором та новими інтерфейсами, вийшов у 2016 році;

– Raspberry Pi 3 Model B+ – випустився в 2018 році із потужнішим Ethernet та процесором;

– Raspberry Pi 4 Model B – випущений із рядом покращень та підтримкою до 8 ГБ оперативної пам’яті в 2019 році;

– Raspberry Pi 400 – випустили у 2020 році, яка мала потужність на 20% більше, ніж попередня версія;

– Paspberry Pi 5 – заанансований у 2023 році, за своєю потужністю повинен переганяти Raspberry Pi 4 майже в два рази.

Розберемо технічні характеристики одноплатного комп’ютера Raspberry Pi 4 Model B+, який буде використаний для реалізації поставлених задач:

1. Центральний процесор – 4-х ядерний ARM Cortex-A72 із тактовою частотою 1.5/1.8 ГГц.

2. Графічний процесор – вбудований VideoCore VI з тактовою частотою 500 МГц. Підтримує OpenGL ES 3.1 та Vulkan 1.2.

3. Оперативна пам’ять – 2 ГБ, 4 ГБ або 8 ГБ LPDDR4, у залежності від моделі;

4. Бездротові технології – Wi-fi 802.11ac та Bluetooth 5.0 BLE;

5. Порти та роз’єми: 2 x USB 3.0, 2 x USB 2.0, GPIO порти, 2 x micro HDMI, 3,5 mini jack, microSD;

6. Живлення – USB Type-C 5V 3A.

Для розпізнавання обличчя в реальному часі потрібні значні обчислювальні ресурси для обробки великих масивів даних. Одноплатний комп’ютер Raspberry Pi 4 має недостатню потужність для обробки такої кількості даних із нормальною частотою кадрів, що недопустимо для БПЛА[5].

Для усунення даного недоліку допоможе використання апаратного прискорення за допомогою USB прискорювачів, таких як: Coral Edge TPU (рис. 2.10) або Intel Movidius Neural Compute Stick (рис. 2.11).

Рис. 2.10. Coral Edge TPU

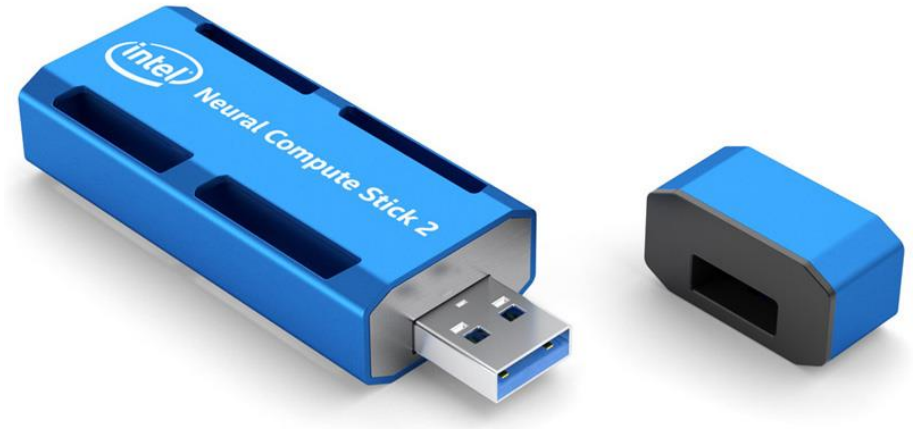

Рис. 2.11. Intel Movidius Neural Compute Stick

Дані пристрої під’єднуються до Raspberry Pi 4 через інтерфейс USB 3.0, що дозволяє використовувати прискорювач для інференції на повну потужність, та знизити навантаження на одноплатний комп’ютер, оскільки математичні навантаження візьмуть на себе USB прискорювачі. Вони мають наступні технології для збільшення потужності:

1. Використання спеціальних гіперпараметрів та структур для швидкого виконання завдань, що зіграє велику роль у процесі глибокого навчання моделі.

2. Добре пристосовані до парелельного виконання тензорних операцій, що дозволить досягнути високої ефективності масового паралелізму.

3. За допомогою методу квантування зменшується кількість бітів, що зменшує вимоги до пам’яті та підвищує ефективність системи.

Для створення кінцевого продукту буде використана зв’язка одноплатного комп’ютера Raspberry Pi із польотним контролером Pixhawk Cube (рис. 2.12).

Рис. 2.12. Pixhawk Cube Orange

Pixhawk Cube – автопілот, призначений для використання із різними видами БПЛА та дронів для виконання керування та стабілізації. Використовує ефективні алгоритми для проведення стабілазації дрона, має багато вбудованих сенсорів, за допомогою яких отримує інформацію про стан апарата під час

польоту. За допомогою модульної системи підключення комплектуючих до польотного контролера є дуже зручним у використанні.

Для роботи із даним автопілотом Raspberry Pi підключатиметься до нього за допомогою UART порта.

UART – протокол обміну даними між пристроями, обмін виконується за допомогою спеціальних сигналів тактового сигналу. Часто використовується в контексті мікропроцесорів та мікроконтролерів, UART порт використовує два сигнали: TX (transmit) для передачі даних та RX (receive) для отримання даних. Даний спосіб передачі даних забезпечує її надійність та швидкість у реальному часі.

UART має наступні характеристики та особливості:

– асинхронність – дані відправляються без глобального тактового сигналу, замість цього в обох пристроях повинен бути однаковий baud rate;

– біти даних – UART підтримує різні розміри бітів даних для передачі даних, від 5 до 9 бітів;

– біти парності – допомогають вивляти помилки під час передачі даних;

– стоп-біти – біти, які йдуть після біта даних для того, щоб пристрій, який приймає дані міг підготуватися до прийому наступного біта даних;

– швидкість передачі даних – вимірюється в бодах (bps), зазвичай стандартними значеннями є 38 400 bps 57 600 bps та 115 200 bps.

2.3 Використання мови програмування Python та його бібліотек для розробки та оптимізації пограмного коду для комп’ютерного зору

Python – це високорівнева, інтерпретована мова програмування (рис. 2.13), основними особливостями якої є простий та лаконічний синтаксис, високий рівень абстракції, підтримка об’єктно-орієнтовано програмування та велика спільнота розробників.

Рис. 2.13. Логотип мови програмування Python

Мова програмування Python та її багатий інструментарій та велика кількість різноманітних бібліотек допоможе ефективно розробити програмне забезпечення для безпілотного літаючого апарата.

Одним із важливих аспектів, який зіграв на користь Python при виборі мови програмування для виконання даної задачі, виявилася підтримка мультипроцесорної обробки даних[10]. Так як Raspberry Pi має обмежену потужність, тому потрібно якомого краще оптимізувати використання усіх ресурів. Для даного завдання відмінно підійде пакет multiprocessing. Принцип даного пакету можна розглянути за допомогою рис. 2.14 та рис. 2.15.

Рис. 2.14 Однопроцесорна обробка даних

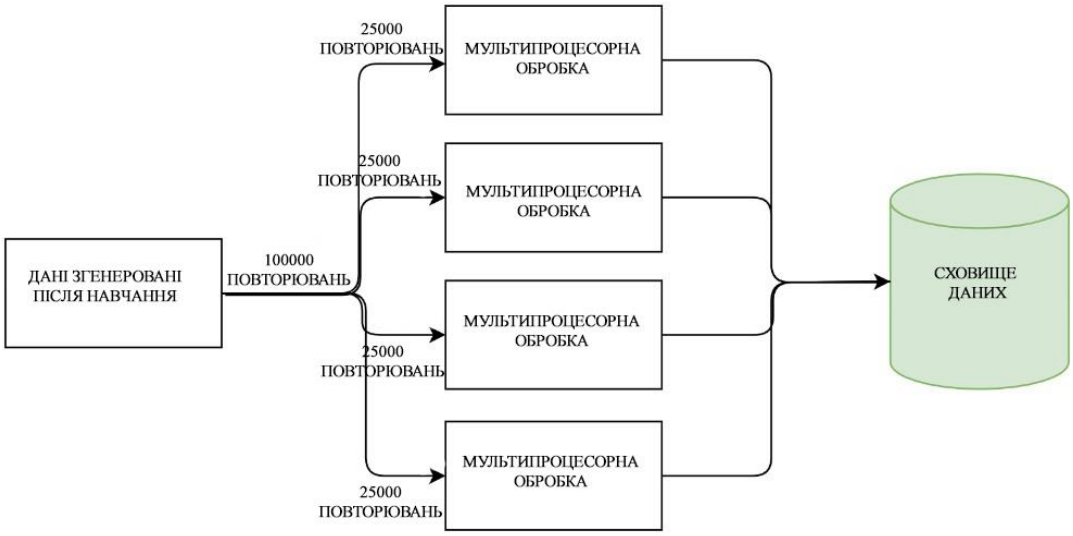

Рис. 2.15 Мультипроцесорна обробка даних

На даних рисунках зображені процеси однопроцесорної та мультипроцесорної обробки даних. Можна побачити, що мультипроцесорний підхід до обробки даних використовує декілька ядер процесору для розділення вихідних даних та обробки їх незалежно один від одного, що підвищить процес обробки відеозображення в реальному часі.

Multiprocessing використовує розділену пам'ять для обміну даними між процесами. Кожен процес має свою власну область пам'яті.

На рис. 2.16 основний клас для створення процесів. Кожен об'єкт Process представляє окремий процес.

```
from multiprocessing import Process

1 usage
def my_function():
    print("Hello from a process!")

if __name__ == "__main__":
    process = Process(target=my_function)
    process.start()
    process.join()
```

Рис. 2.16. Основний клас Process для створення процесів

Модуль multiprocessing дозволяє спільно використовувати пам'ять між процесами, використовуючи об'єкти, такі як Value та Array (рис. 2.17).

```
from multiprocessing import Process, Value

1 usage
def increment(count):
    for _ in range(1000):
        count.value += 1

if __name__ == "__main__":
    count = Value("i", 0)
    processes = [Process(target=increment, args=(count,)) for _ in range(4)]

    for process in processes:
        process.start()

    for process in processes:
        process.join()

    print("Counter:", count.value)
```

Рис. 2.17. Розділення пам’яті між процесами

Pool дозволяє створювати паралельні обчислення за допомогою пула процесів (рис. 2.18).

```
from multiprocessing import Pool

1 usage
def square(x):
    return x * x

if __name__ == "__main__":
    data = [1, 2, 3, 4, 5]
    with Pool() as pool:
        result = pool.map(square, data)
```

Рис. 2.18. Паралельність функцій

Іншою бібліотекою для розробки програмного забезпечення для БПЛА є OpenCV – відкрита бібліотека для комп’ютерного зору, яка надає широкий інструментарій для обробки зображення в реальному часі[13,16]. Має широкий спектр функцій, включаючи фільтрацію, виявлення об’єктів, розпізнавання обличчя та силуетів людей, обробка відео та багато іншого.

TensorFlow – відкрита програмна бібліотека для машинного та глибокого навчання, розроблена командою Google (рис.2.19). Вона надає широкий інструментарій для створення та навчання штучних нейронних мереж, що використовуються в різних задачах, таких як класифікація зображень, розпізнання мови та багато іншого[38].

Рис. 2.19 Бібліотека TensorFlow

Основна особливість TensorFlow полягає у визначенні обчислювальних графів, де вузли представляють операції, а ребра – дані, які проходять між цими операціями. Такі графи можуть бути оптимізовані та використовуватись на центральних та графічних процесорах.

Гнучка архітектура дозволяє легко відтворювати обчислення на різноманітних платформах – ЦП, ГП та TPU.

TensroFlow Lite – інструментальний набір, розроблений компанією Google, для перетворення на менший, більш портативний та ефективний формат моделі TensorFlow для виконання на мобільних або малопотужних пристроях. Для виконання моделі потрібне спеціальне середовище виконання, а також вхідні дані, які передаються в модель, повинні бути у форматі тензору. Даний підхід може значно підвищити ефективність роботи з моделями машинного навчання на пристроях, у яких обмежений ресурс обчислення.

Coral Edge TPU API – бібліотека для класифікації зображень та виявлення об’єктів, створена на основі API TensorFlow Lite і абстрагує код, необхідний для вхідних і вихідних тензорів.

Ключові модулі API, які використовуються у даній бібліотеці:

– classificationEngine – виконує класифікацію зображень;

– detectionEngine – виконує виявлення об’єктів.

Обидві системи є підкласами BasicEngine та підтримуют усі формати зображнь, включаючи JPEG, PNG, BMP та інші.

MobileNetSSD – комбінована модель для визначення об’єктів у реальному часі, яка поєднує в собі два ключових елементи: MobileNet – для ефективної роботи на мобільних та малопотужних пристроях та SSD (Single Shot Multibox detector) – для швидкого та точного визначення об’єктів. Основні її характеристики:

– MobileNet – архітектура для глибокого навчання, спроєктована для ефективної роботи по зберіганню ресурсів обчислювальної системи. За допомогою зменшеної кількості операцій та параметрій, вона ідеально підходить для мобільних та вбудованих пристроїв;

– SSD – архітектура для об’єектного визначення, яка використовує один об’єктний прогін даних для прогнозування класів та координат об’єктів, що робить процес визначення більш швидким та ефективним.

2.4 Висновки

У другому розділі була розглянута система керування безпілотними літальними апаратами Ardupilot. Були виявлені її головні переваги відносно аналогічних систем, а саме: можливість впровадження стороннього ПЗ до автопілота, наявність тонких налаштувань конфігурації польотного контролера, система має відкритий програмний код та розширеший функціонал планувальника місій, який містить багато різноманітних сценаріїв проведення польоту та налаштувань. Із недоліків виявився тільки складний інтерфейс для користування та великий поріг входження в систему. Загалом система Ardupilot надає великий набір інструментів для ефективної розробки безпілотного літаючого апарата із власним програмним забезпеченням.

Також було розібране апаратне рішення, щодо впровадження розробленого ПЗ для БПЛА, яким виявився міні одноплатний комп'ютер Raspberry Pi 4 Model B. Досліджений процес розвитку даної серії одноплатних комп'ютерів, яка налічує 8 різних модифікацій, які відрізняються технічними характеристиками. Були розглянуті технічні характеристики Raspberry Pi 4. Проаналізована проблематика у використанні даного комп'ютера при вирішенні поставлної задачі, а саме нестача оброблювальної потужності при обробці великих даних у реальному часі та розброблений метод для її вирішення із використання апаратного прискорення за допомогою USB прискорювачів таких як Coral Edge TPU або Intel Movidius Neural Compute Stick.

Була проаналізована взаємодія автопілота Pixhawk Cube, який призначений для керування БПЛА, із Raspberry PI та метод роботи між ними за допомогою протоколу даних UART, його головні особливості та характеристики роботи.

Були досліджені мова програмування Python та пакет multiprocessing, за допомогою якого можна оптимізувати та збільшити ефективність обробки алгоритмів комп'ютерного в реальному часі за допомогою використання багатоядерної обробки процесу. Також була розглянута бібліотека комп'ютерного зору OpenCV, бібліотека машинного навчанння TensorFlow, інструментарій TensorFlow Lite для покращення ефективності роботи машинного навчання на малопотужних пристроях та комбіновану модель для визначення об'єктів у реальному часі MobileNetSSD.

РОЗДІЛ 3
РОЗРОБКА БПЛА ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розроблена модель БПЛА

Апаратна частина була спеціально підібрана під поставлені задачі та повністю їм задовільняє. На рис. 3.1 можна побачити розроблену модель безпілотника для впровадження в неї власного програмного забезпечення.

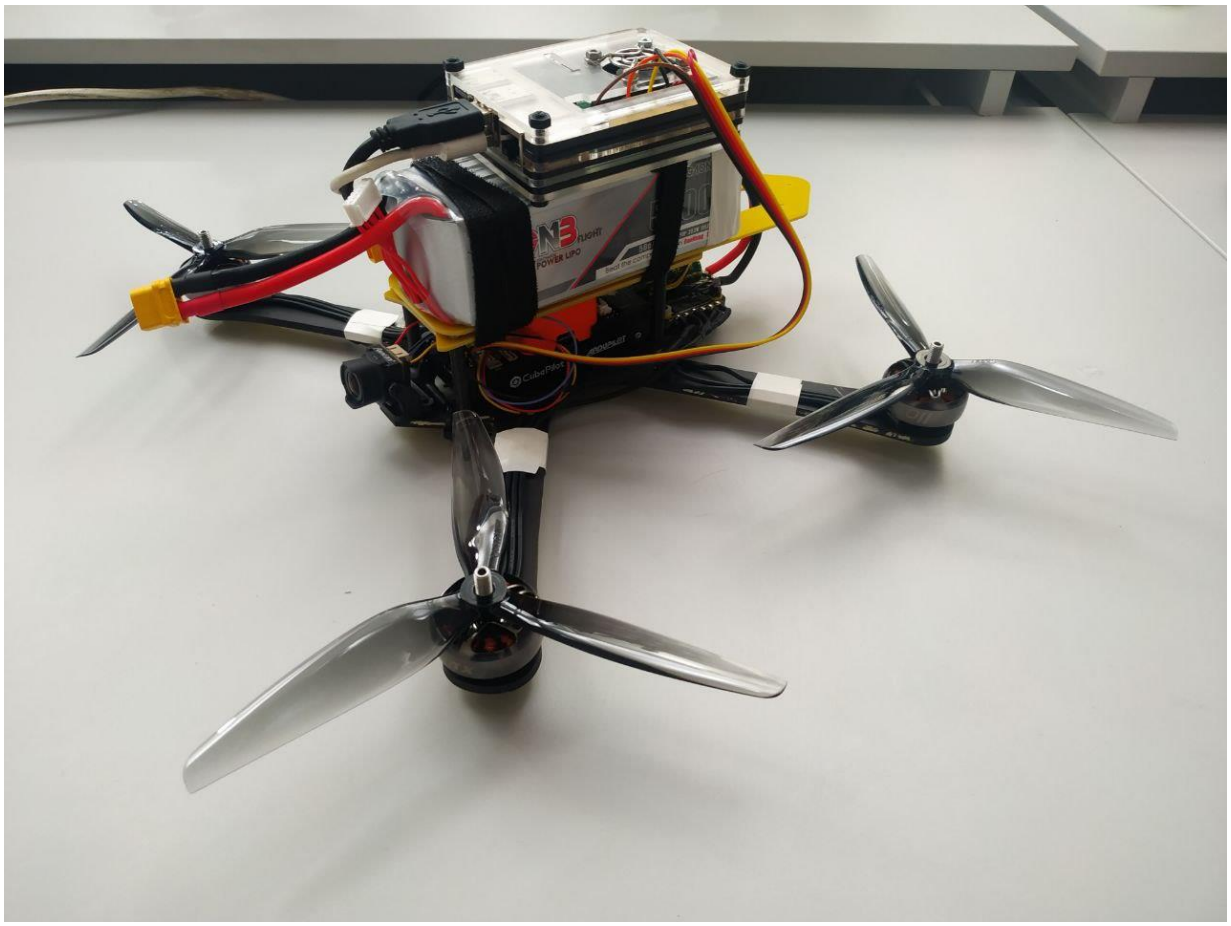

Рис. 3.1. Розроблена модель БПЛА

Для її розробки були використані наступні комплектуючі:

1. Рама – 7 дюймова, виготовлена із текстоліту для здешевлення моделі.
2. Польотний контролер – Pixhawk Cube Orange.
3. Регулятор обертів – SpeedyBee BLS 50A.

4. Камера + відеопередавач – цифрова система Avatar HD nano kit із комплектними антенами для відеопередавача.

5. Радіоприймач – CRSF 868 МГц із комплектною антеною.

6. GPS модуль – Matek m10q-5883.

7. Двигуни – Emax exo II 2807 1300kv.

8. Пропелери – HQProp 7x3.5x3 із полікарбонату.

9. Акумуляторна батарея – lipo 6s 5600 мАг.

10. Одноплатний ком’ютер для обробки алгоритмів Raspberry Pi 4 Model B+.

11. USB пришвидчувач для Raspberry Pi Coral Edge TPU.

12. Знижувальний перетворювач напруги DC-DC LM2596S для забезпечення потрібного струму для Raspberry.

За допомогою перелічених апаратних компонентів безпілотний літальний апарат буде забезпечений довгою тривалістю польоту, завдяки високопродуктивним двигунам та lipo акумуляторної батареї на 5600 мАг із підтримкою великою струмовіддачі, що дозволить мати в запасі велику потужність, не втрачаючи при цьому в енергоефективності.

Використання польотного контролера Pixhawk Cube забезпечить стабільність апарата під час польоту за допомогою своєї багатої кількості сенсорів і датчиків та корректного їх калібрування. За допомогою точного позиціонування, яке підтримує GPS модуль Matek дрон буде мати можливість виконувати складні автоматичні місії польоту.

На рис. 2.3 показаний процес взаємодії основних апаратних частин, які використовуються на БПЛА. Оператором взаємодіє із БПЛА через спеціальний пульт (рис. 3.3) із багатьма тумблерами, для активації дрона та алгоритмів обробки зображення. Raspberry pi взаємодіє із камерою дрону через спеціальний шлейф із адаптером для підключення до неї. Результат роботи алгоритмів обробки Raspberry Pi передає через бездротовий зв’язок wifi на персональний комп’ютер оператору.

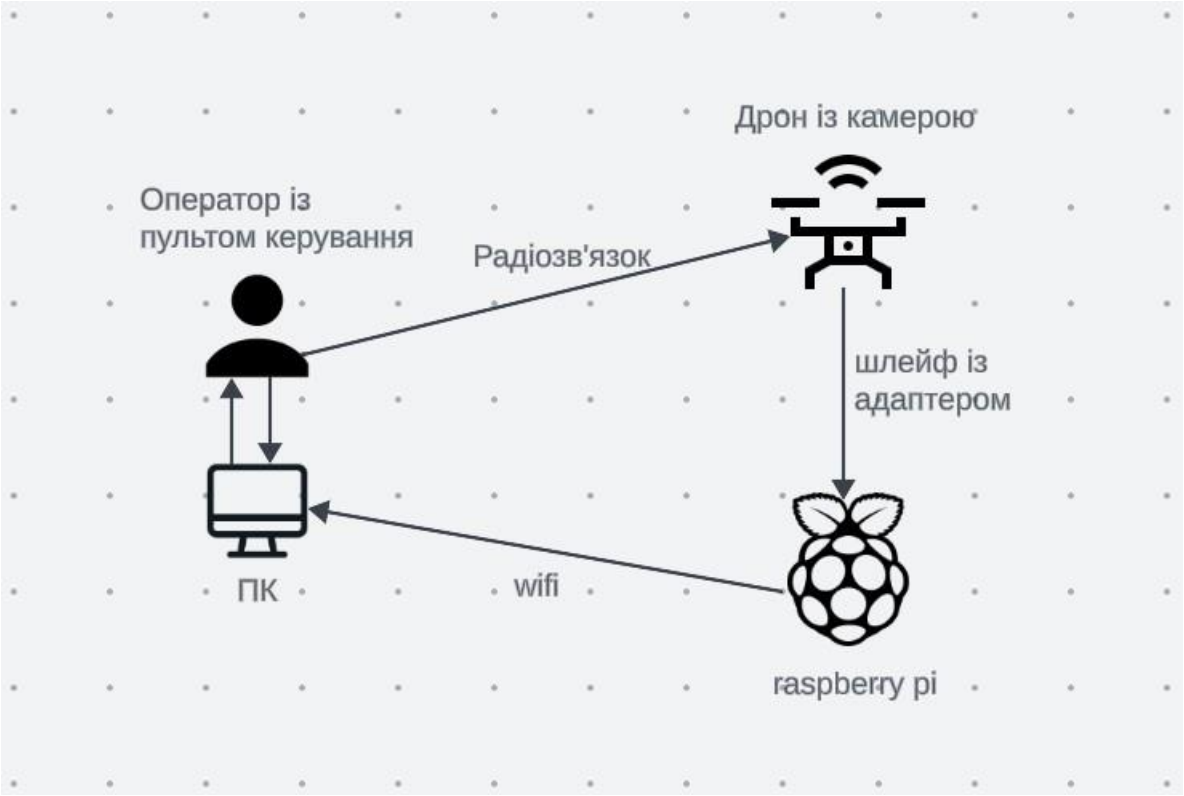

Рис. 3.2 Процес взаємодії апаратної частиною БПЛА та оператора

Рис. 3.3. Пульт для керування БПЛА і активації алгоритмів

Розглянемо діаграму послідовностей принципу користування розробленим продуктом (рис. 3.4). Щоб почати користуватися дроном, оператору потрібно: (1) активувати БПЛА (заармити) за допомогою тумблера на пульті керування, з якого буде передана команда на апарат для запуску і він буде готовий для звичайного користування (2). Щоб користуватися розробленими алгоритмами для визначення об’єктів потрібно активувати інший тумблер на пульті керування (3), після чого буде передана команда на польотний контролер(4) і він запустить скрипт для запуску raspberry pi (5). Коли оператор захоче увімкнути алгоритм для визначення об’єктів, він повинен натиснути третій тумблер для його активації (6), команда з пульта керування передасться на дрон (7) і алгоритм запуститься на rapspberry (8), після чого результат виконання алгоритма буде переданий на персональний комп’ютер оператора (9), де він зможе в реальному часі відслідковувати його роботу.

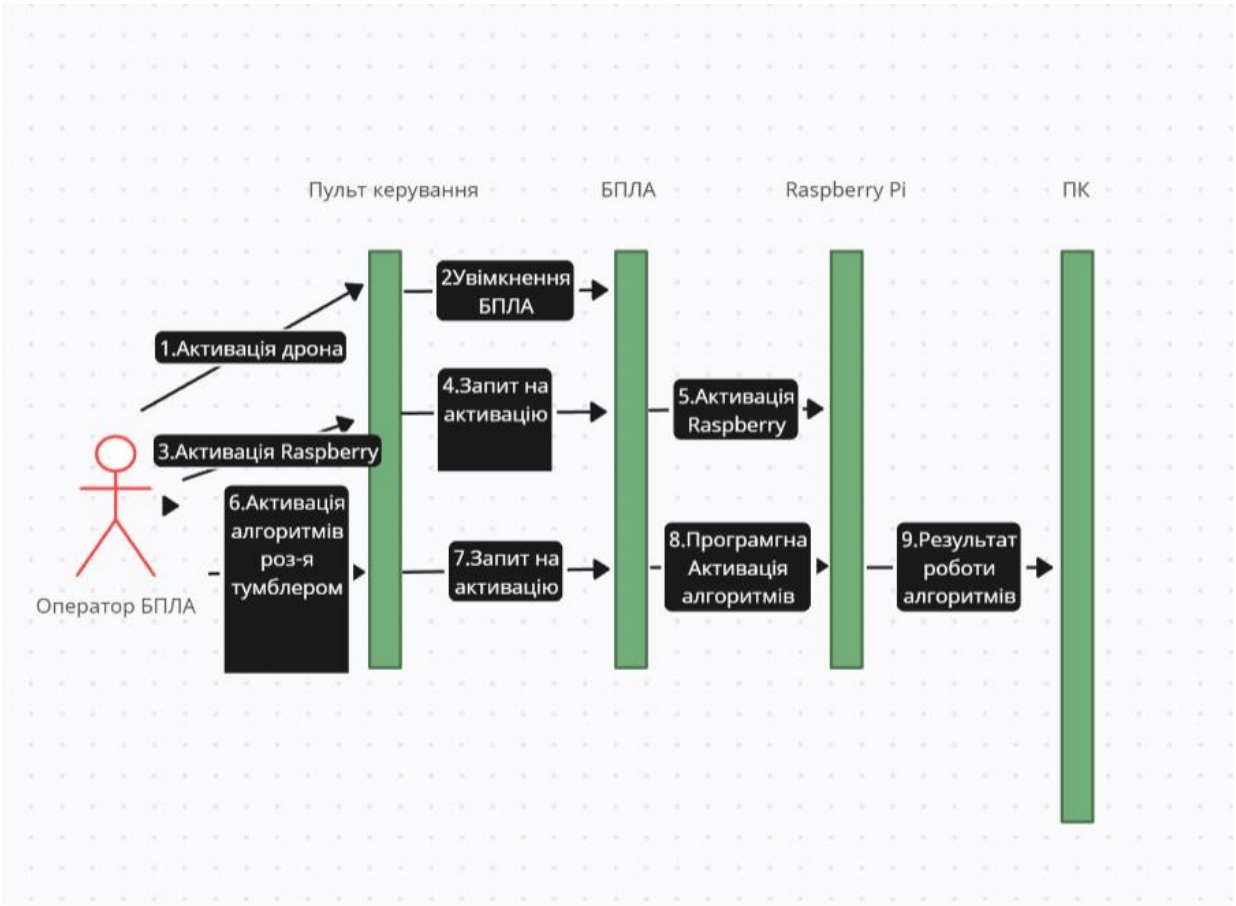

Рис. 3.4. Діаграма послідовності процесу керування дроном

3.2 Програмна частина розробленої моделі БПЛА

На рис. 3.5 можна побачити архітектуру розробленого проєкту із розпізнавання об’єктів. Для зручності показана в середі розробці pycharm, дані файли ж були перенесені на raspberry.

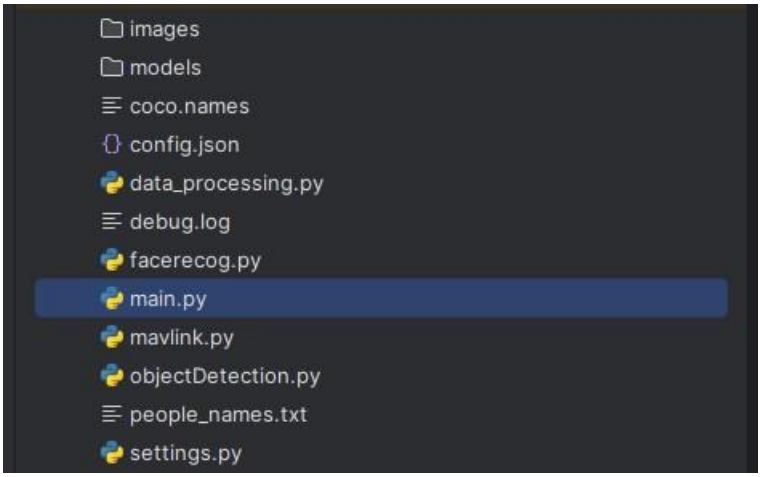

Рис. 3.5. Архітектура розробленого проєкту

Coco.names – файл, для зберігання назв класів, які використовуються в наборі даних COCO. Даних набір даних найпопулярніший для вирішення завдань комп’ютерного зору. У ньому містяться такі імені як: person, car, drone та інші.

Config.json – файл для зберігання налаштувань, у якому зберігаються шляхи до файлів, налаштування підключення по mavlink, розмір зображень для обробки та інші.

Папки images та models зберігають зображення для навчання та моделі.

Debug.log – використовується дле зберігання логів роботи програми, для подальшого їх аналізу.

Facerecog.py – файл, який містить алгоритми розпізнавання обличчя, використовуючи камеру raspberry.

Main.py – файл із головною логікою програми.

Mavlink.py – файл для підключення протоколу mavlink для передачі телеметрії.

ObjectDetection.py – файл, що містить алгоритми для розпізнавання об’єктів.

People_names.txt – текстовий файл із іменами людей, на яких була навчена модель розпізнавання обличчя.

Навчені моделі були завантажені з офіційного репозиторію TensorFlowHub та збережені в директорію models (рис. 3.6).

```python
import os
import requests


model_urls = {
    "ssd_mobilenet_v2_face": "https://tfhub.dev/tensorflow/lite-model1/ssd_mobilenet_v2/1/metadata/2?lite-format=tflite",
    "ssd_mobilenet_v2_drone": "https://tfhub.dev/tensorflow/lite-model2/ssd_mobilenet_v2/1/metadata/2?lite-format=tflite",
    "ssd_mobilenet_v2_cup": "https://tfhub.dev/tensorflow/lite-model3/ssd_mobilenet_v2/1/metadata/2?lite-format=tflite",
    "ssd_mobilenet_v2_person": "https://tfhub.dev/tensorflow/lite-model4/ssd_mobilenet_v2/1/metadata/2?lite-format=tflite",
    "ssd_mobilenet_v2_car": "https://tfhub.dev/tensorflow/lite-model5/ssd_mobilenet_v2/1/metadata/2?lite-format=tflite",
}


models_folder = "models"
os.makedirs(models_folder, exist_ok=True)


for model_name, model_url in model_urls.items():
    model_path = os.path.join(models_folder, f"{model_name}.tflite")


    response = requests.get(model_url)
    with open(model_path, 'wb') as model_file:
        model_file.write(response.content)
```

Рис. 3.6. Завантаження моделей у форматі .tflite

Для захоплення відеозображення з камери, яка піключена до raspberry була імпортована бібліотека picamera, за допомогою якої відеопотік ініціалізується та передається в pirgbarray (рис. 3.7).

```python
camera = PiCamera()
camera.resolution = (640, 480)
raw_capture = PiRGBArray(camera, size=(640, 480))


time.sleep(0.1)


for frame in camera.capture_continuous(raw_capture, format="bgr", use_video_port=True):
    image = frame.array
```

Рис. 3.7. Захоплення відео із камери

Було оброблене захоплене відеозображення (рис. 3.8) із камери raspberry pi та за допомогою файлу моделі MobileNetSSD був виконаний алгоритм для визначення об’єктів на даному зображенні.

```
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.2:
            class_id = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([image.shape[1], image.shape[0], image.shape[1], image.shape[0]])
            (startX, startY, endX, endY) = box.astype("int")


            label = f"Class: {class_id}, Confidence: {confidence:.2f}"
            cv2.rectangle(image, (startX, startY), (endX, endY), (0, 255, 0), 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(image, label, org: (startX, y), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, color: (0, 255, 0), thickness: 2)


    cv2.imshow( winname: "Object Detection", image)
    raw_capture.truncate(0)

    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break


cv2.destroyAllWindows()
```

Рис. 3.8. Обробка зображення та виявлення об’єктів на ньому

Розроблений безпілотний літальний апарат може працювати в двох режимах: як звичайний безпілотник, керування якого відбувається з пульта керування та відеозображення передається в FPV окуляри, і як дрон із комп’ютерним зором, оброблене відеозображення якого передається на комп’ютер.

Дані два режими можуть працювати одночасно і оператор буде отримувати два зображення: зображення, яке пройшло через польотний контролер, де було накладено OSD на нього для відображення різноманітної інформації, після чого передано на відеоприймач, приклад даного зображення на рис. 3.9, та інше зображення, яке було оброблено алгоритмами комп’ютерного зору для розпізнавання об’єктів та передано в обробленому вигляді на комп’ютер оператора (рис. 3.10).

Рис. 3.9. Зображення з відеокуляр

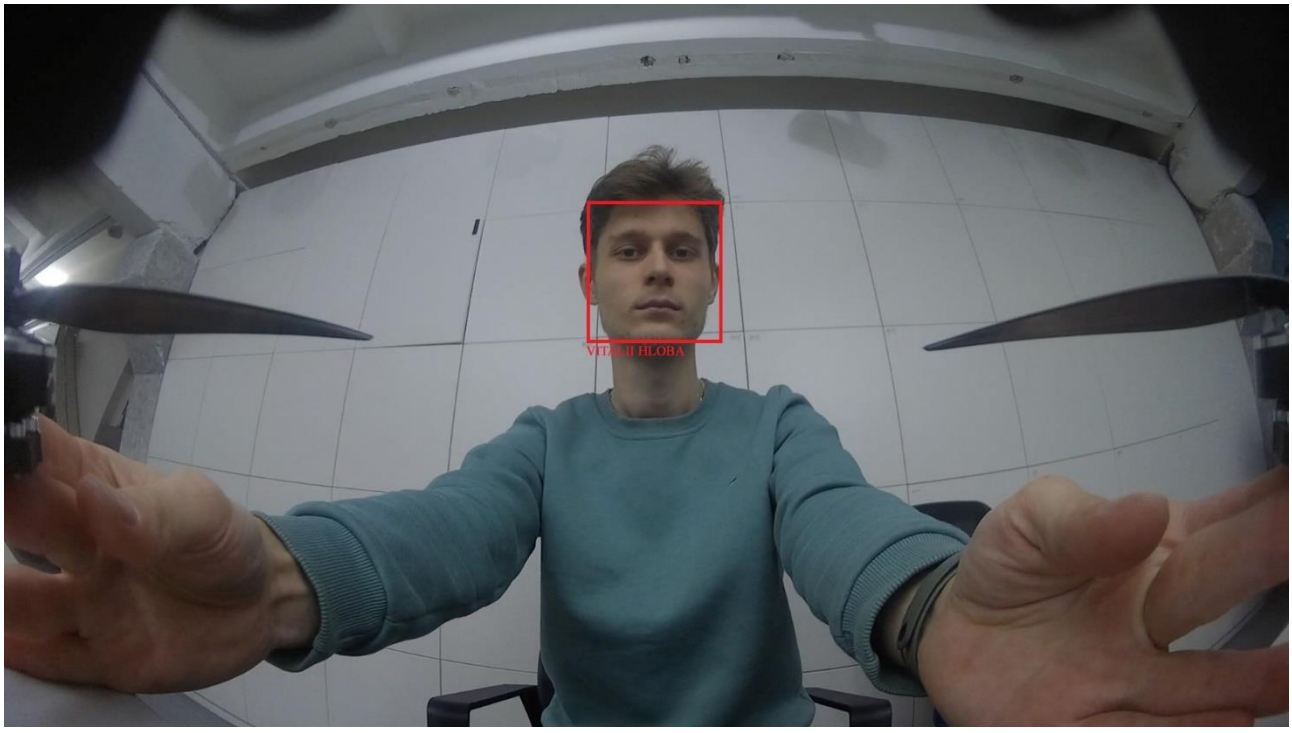

Рис. 3.10. Оброблене відео після алгоритму

3.3 Дослідження розробленої моделі БПЛА та роботи її алгоритмів

Розпізнавання людини, коли дрон знаходиться на висоті, людина йде, алгоритм визначення відпрацьовує добре та знаходить людини серед багатьох речей навколо неї (рис.3.11).

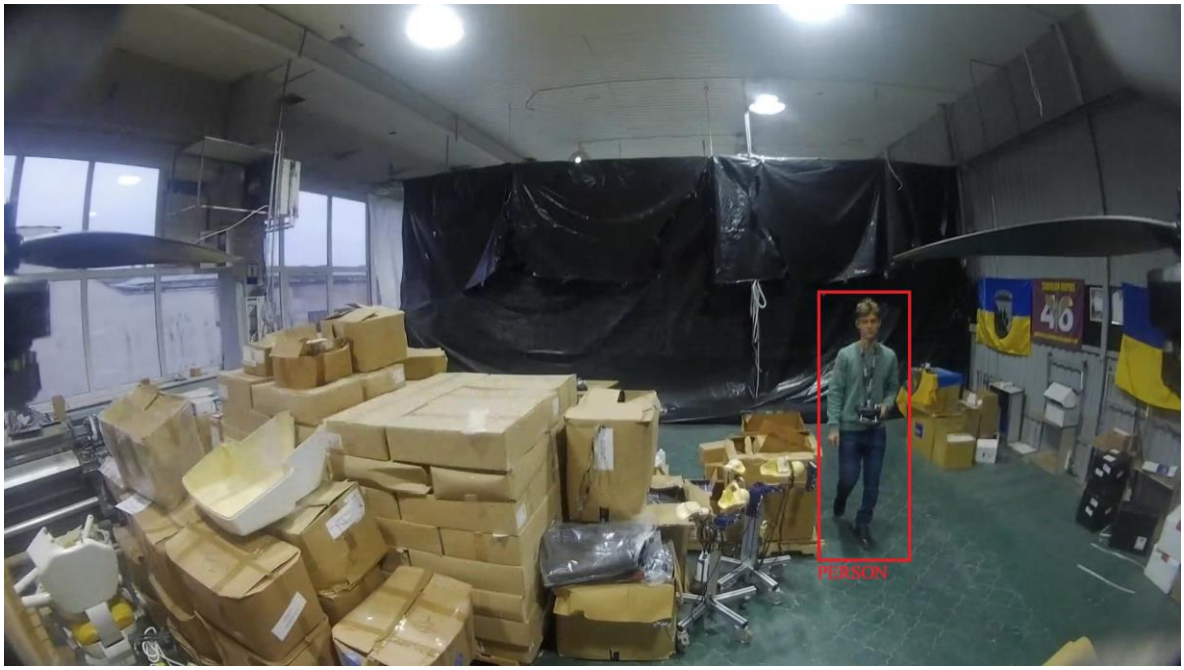

Рис. 3.11. Розпізнавання людини з висоти

Розпізнавання людини зі спини, процес проводиться із наявністю інших об’єктів у кадрі. Алгоритм розпізнавання працює добре та зміг розпізнати на зображенні людину (рис. 3.12).

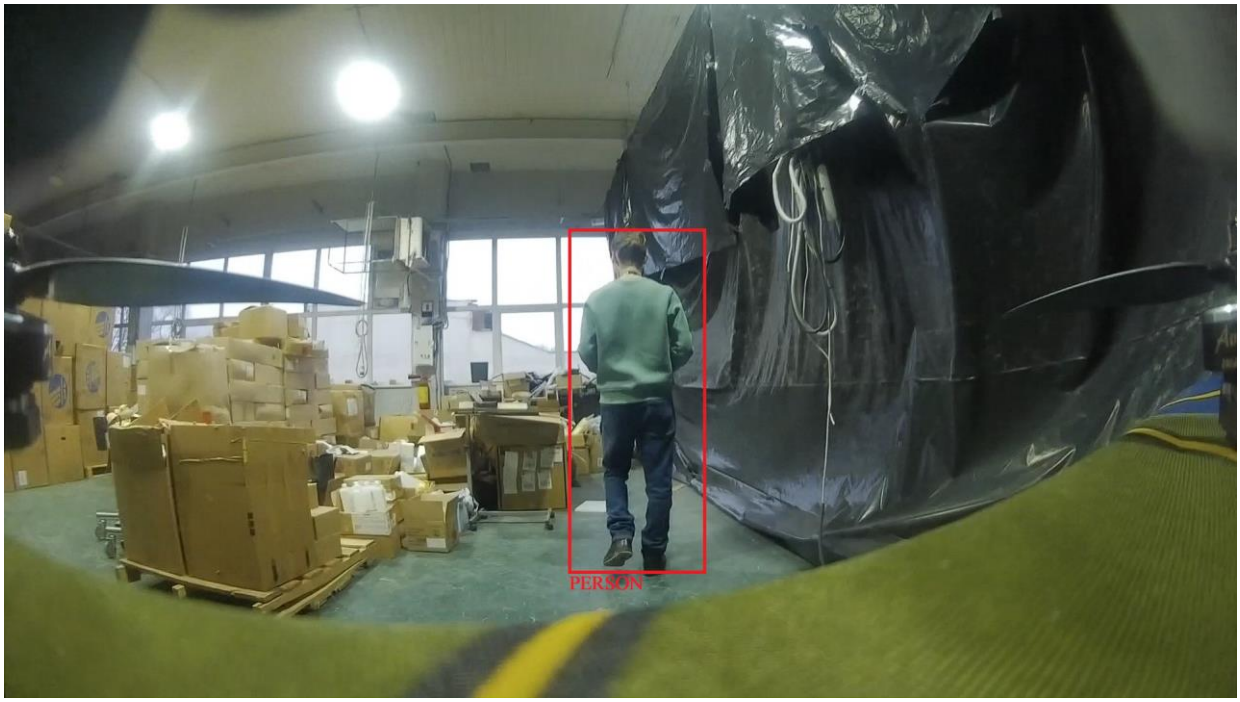

Рис. 3.12. Розпізнавання зі спини

На рис. 3.13 відбувається процес тестування алгоритму розпізнавання в умовах поганої освітленості приміщення. Алгоритм спрацьовує в цих умовах погано, розпізнає тільки зблизька. Потрібно навчати модель із більшою кількість даних та в різних умовах, включаючи погане освітлення, для вирішення даної проблеми.

Рис. 3.13. Розпізнавання в поганому освітленні

Процес розпізнання автомобіля на зображенні, навчена модель гарно справляється із даною задачею (рис. 3.14).

Рис. 3.14. Розпізнавання машини

На рис. 3.15 показаний процес розпізнання такого об’єкта, як квадрокоптер.

Рис. 3.15. Розпізнавання дрона

3.4 Висновки

У третьому розділі була розглянута розроблена модель БПЛА, яким являється безпілотний літальний апарат із впровадженим програмним забезпеченням в нього для розпізнавання об’єктів. Був перелічений список апаратних комплектуючих, які були використані при розробці даного проєкта, вибір апаратної частини був обгрунтований та повністю задовільняє поставленим задачам перед нею. Були розглянуті процес роботи апаратної частини та діаграма послідовностей процесу керування БПЛА оператором за допомогою пульта керування.

Також була розглянута програмна частина проєкту, яка була інтегрована в розроблену модель БПЛА та архітектура проєкту. Розглянуті методи роботи із Raspberry pi та її камерою за допомогою бібліотек TensorFlow Lite та Coral Edge TPU та навченої моделі для розпізнання MobileNetSSD, яка добре підходить для роботи із малопотужними пристроями та може забезпечувати визначення об’єктів в реальному часі. За допомогою даних бібліотек та використання апаратного прискорення вдалось досягти вихідного зображення в 20 кадрів на секунду.

Були показані результати роботи алгоритмів розпізнання таких об’єктів як: людина, машина, дрон. Уцілому алгоритм працює добре та справляється із розпізнаванням даних моделей. Розпізнавання людини можливе під час польоту дрона та із-за спини. Через недостатню кількість даних для навчання, алгоритм погано справляється із розпізнаванням у погано освітленому приміщенні. Процес розпізнавання таких об’єктів як дрон та машина відбувається задовільно.

ВИСНОВКИ

Розроблена модель БПЛА із впровадженим програмним забезпеченням на мові програмування Python під час тестів проявила себе добре та задовольнила всі поставлені задачі перед апаратною та програмною частиною. Даний безпілотник здатний розпізнавати людей та машини під час польоту із невеликої дистанції, також може проводити розпізнавання дронів. За допомогою використання одноплатного комп'ютера Raspberry Pi разом із апаратним прискорювачем Coral Edge TPU та оптимізованих бібліотек для нього, таких як: Tensor Flow Lite та MobileNetSSD були досягнуті високі показники продуктивності даної моделі в розпізнанні об'єктів.

Актуальність даної розробленої моделі визначається підвищенням попиту на БПЛА, особливо на апарати, які містять вбудовані рішення зі штучного інтелекту або комп'ютерного зору. Дана модель може нести велику практичну цінність, так як може бути впроваджена у різноманітні сфери нашого життя, такі як: доставка товарів, рятувальні операції, аграрна та воєнна сфера та багато інших, якщо модель буде удосконалена та будуть вирішені всі наявні її недоліки.

Під час аналізу ефективності впровадження програмного забезпечення мовою програмування Python для БПЛА було виявлно, що дана мова програмування має великий інструментарій для цього та відмінно підходить для даного завдання, а саме містить великий вибір бібліотек комп'ютерного зору, таких як OpenCV, TenserFlow, TensorFlow Lite, Coral Edge TPU API та інші із готовими, навченими моделями, які оптимізовані під вибрані апаратні рішення в проєкті та повністю розкривають їх потужність.

Під час виконання кваліфікаційної роботи були виконані всі поставлені задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 6 Hour Endurance Electric VTOL UAV for Mapping, Survey & Surveillance.
– [Електронний ресурс] – Режим доступу: https://www.unmannedsystemstechnology.com/2022/04/6-hour-endurance-electric-vtol-uav-for-mapping-survey-surveillance/.

2. Applications of Unmanned Aerial Vehicles. – [Електронний ресурс] – Режим доступу: https://encyclopedia.pub/entry/25512.

3. ArduPilot Copter. – [Електронний ресурс] – Режим доступу: https://ardupilot.org/copter/index.html.

4. Beginner's Guide: How to Get Started With Raspberry Pi. – [Електронний ресурс] – Режим доступу: https://www.pcmag.com/how-to/beginners-guide-how-to-get-started-with-raspberry-pi.

5. Communicating with Raspberry Pi via MAVLink. – [Електронний ресурс] – Режим доступу: https://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html.

6. De Havilland DH 82A Tiger Moth. – [Електронний ресурс] – Режим доступу: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/197392/de-havilland-dh-82a-tiger-moth/.

7. De Havilland Tiger Moth & Queen Bee. – [Електронний ресурс] – Режим доступу: https://www.baesystems.com/en/heritage/de-havilland-tiger-moth---queen-bee.

8. Different Types of Drones and Uses (2023 Full Guide). – [Електронний ресурс] – Режим доступу: https://www.jouav.com/blog/drone-types.html.

9. Drones in human detection. – [Електронний ресурс] – Режим доступу: https://www.barcelonadronecenter.com/2023/05/24/drones-in-human-detection-2/.

10. Face Detection and Recognition using OpenCV and Python. Tejashree Dhawle, Urvashi Ukey, Rakshandha Choudante.

– [Електронний ресурс] – Режим доступу: IRJET_V7I10219-libre.pdf (d1wqtxts1xzle7.cloudfront.net).

11. Face Recognition on Drones. Hwai-Jung Hsu, Kuan-Ta Chen. – [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/300655333_Face_Recognition_on_Drones.

12. Face recognition using Viola-Jones depending on Python. Khansaa Dheyaa Ismael, Stanciu Irina. – [Електронний ресурс] – Режим доступу: Sebuah Kajian Pustaka: (researchgate.net).

13. Face Recognition with Python. Philipp Wagner. – [Електронний ресурс] – Режим доступу: https://online.datasport.pl/logos/reg4791.pdf.

14. Global UAV Drones Market Analysis. – [Електронний ресурс] – Режим доступу: https://www.researchdive.com/8348/unmanned-aerial-vehicle-uav-drones-market.

15. Hassija, V.; Saxena, V.; Chamola, V. Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory. Comput. Commun. 2019, 149, 51–61.

16. Intelligent UAV Deployment for a Disaster-ResilientWireless Network. Hassaan Hydher, Dushantha Nalin K. Jayakody, Kasun T. Hemachandra, Tharaka Samarasinghe – [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/343898624_Intelligent_UAV_Deployment_for_a_Disaster_Resilient_Wireless_Network.

17. Introductory Chapter: Drones. George Dekoulis. – [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/326050429_Introductory_Chapter_Drones.

18. Kettering Aerial Torpedo “Bug”. – [Електронний ресурс] – Режим доступу: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/198095/kettering-aerial-torpedo-bug/.

19. Maturana, D., Mery, D., and Soto, A. Face recognition with local binary patterns, spatial pyramid histograms and naive bayes nearest neighbor classification. 2009 International Conference of the Chilean Computer Science Society (SCCC) (2009), 125–132.

20. N. Davis, F. Pattaluga, and K. Panetta. Facial recognitionusing human visual system algorithms for robotic and UAVplatforms. In Proceedings of TePRA 2013, pages 1–5, 2013.

21. The six-inch Black Hornet ‘microdrones’. – [Електронний ресурс] – Режим доступу: https://www.telegraph.co.uk/world-news/2022/08/24/britain-donate-black-hornet-microdrones-ukraine-spy-missions/.

22. Types of drones and their advantages. – [Електронний ресурс] – Режим доступу: https://www.embention.com/news/types-of-drones-and-their-advantages/.

23. Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends. Syed Agha Hassnain Mohsan, Nawaf Qasem Hamood Othman, Yanlong Li, Mohammed H. Alsharif, Muhammad Asghar Khan. – [Електронний ресурс] – Режим доступу: file:///C:/Users/%D0%92%D0%B8%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D1%8F/Downloads/s11370-022-00452-4.pdf.

24. Unmanned vehicles. Handbook 2010 / Shephard press. – Burnham, 2010. – 145 p.

25. What is a Raspberry Pi? – [Електронний ресурс] – Режим доступу: https://opensource.com/resources/raspberry-pi.

26. What is ArduPilot? – [Електронний ресурс] – Режим доступу: https://ardupilot.org/.

27. Zhao, W., Chellappa, R., Phillips, P., and Rosenfeld, A. Face recognition: A literature survey. Acm Computing Surveys (CSUR) 35, 4 (2003), 399–458.

28. Ачасов А.Б., Тітенко Г.В. Щодо використання БПЛА для оцінки стану посівів/ Вісник ХНУ імені В.Н. Каразіна серія «Екологія». – 2015. – №13. – С. 2.

29. Безпілотні літальні апарати контейнерного старту: сучасний стан і напрямки досліджень. Збруцький О.В., Масько О.М., Сухов В.В./ НТУУ «Київський політехнічний інститут», м. Київ – [Електронний ресурс] – Режим доступу: https://ela.kpi.ua/bitstream/123456789/2826/1/63-64.pdf.

30. Безпілотні літальні апарати. Б. П. Книш, Д.С. Попіль. – [Електронний
ресурс] – Режим доступу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/20930/4903.pdf?sequence=3.

31. Бондар О. П. Про стратегію враження ворожих БПЛА / О. П. Бондар // Системи та технології, № 2 (64), 2022. С. 14-18.

32. Використання OpenCV і Face Recognition в системах розпізнавання облич на одноплатних комп'ютерах типу Raspberry Pi – [Електронний ресурс] – Режим доступу: https://evergreens.com.ua/ua/articles/open-cv-face-recognition.html.

33. Використання дронів в надзвичайних ситуаціях. Тараба М.О. – [Електронний ресурс] – Режим доступу: https://sci.ldubgd.edu.ua/jspui/bitstream/123456789/10128/1/Problems%20and%20prospects%20of%20development%20of%20life%20safety%20system.pdf#page=311.

34. Голубничий Д.Ю., Третяк В.Ф., Рубан І.В. Класифікація безпілотних літальних апаратів. Системи озброєння і військова техніка. 2007. № 1(9). С. 61-67.

35. Грицан П. А., Лехіцкий Т.В. (2017). Класифікація безпілотних літальних апаратів. Комп'ютерно – інегровані технології. Випуск 27.стр. 16-18.

36. Дрони (бпла), їх роль у сучасному житті та екології. О.Т. Мехед, Защепкіна Н.М. – [Електронний ресурс] – Режим доступу: https://ela.kpi.ua/bitstream/123456789/53580/1/Page_128-130.pdf.

37. Дрони в сільському господарстві. Ростовський І.Р., Мікла І.А., Галич
І.В. – [Електронний ресурс] – Режим доступу: https://repo.btu.kharkov.ua/bitstream/123456789/14953/1/Innovatisiini%20rozrobky%20v%20ahrarnii%20sferi_T%201_2018_185.pdf.

38. Дрони на базі нейронних мереж у агропромисловості. Горбачов Я., Тимчук С. О. – [Електронний ресурс] – Режим доступу: https://elartu.tntu.edu.ua/bitstream/lib/25026/2/MSNK_2018v1_Horbachov_Y-Drons_based_on_neural_networks_41.pdf.

39. Дубов Д. В. Нові покоління технологій подвійного призначення, як інноваційні детермінанти розвитку сфери Національної безпеки та оборони / Д. В. Дубов // Стратегічні пріоритети. – К., 2014. – Вип. 4 (33). – С.106–113.

40. Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2020). Тринадцята міжнародна науково-практична конференція 19-20 травня 2020 р., Київ, Україна. – К.: НАУ, 2020. – 305 с.

41. Класифікація задач управління групою тактичних безпілотних літальних апаратів. Гримуд А.Г. – [Електронний ресурс] – Режим доступу: https://www.viti.edu.ua/files/zbk/2020/1_1_2020.pdf.

42. Методичні рекомендації до виконання кваліфікаційних робіт здобувачами другого (магістерського) рівня вищої освіти спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки» / Б.І. Мороз, О.В. Іванченко, О.В. Реута, О.С. Шевцова; М-во освіти і науки України, Нац. техн. ун-т "Дніпровська політехніка". – Дніпро : НТУ «ДП», 2022. – 87 с.

43. Окремі проблеми нормативного забезпечення БПЛА. Зуєва В. О., Куліш А. В. – [Електронний ресурс] – Режим доступу: Зуєва В.О., Куліш А.В. ОКРЕМІ ПРОБЛЕМИ НОРМАТИВНОГО ЗАБЕЗПЕЧЕННЯ БПЛА.PDF (nau.edu.ua)

44. Погляд із минулого в майбутнє: безпілотні літаючі дрони як елемент евакуації поранених у медичній службі Збройних сил України. Гуменюк К.В., Горошко В.Р. – [Електронний ресурс] – Режим доступу: file:///C:/Users/%D0%92%D0%B8%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D1%8F/Downloads/emergadminojs,+22-27.pdf.

45. Система керування БПЛА. Синєглазов В.М., Іщенко В.С. – [Електронний ресурс] – Режим доступу: https://conf.ztu.edu.ua/wp-content/uploads/2016/04/tezyikt-2016pdf.pdf#page=157.

46. Стеценко О. О. Космічні системи інформаційного забезпечення безпілотних засобів різного призначення: підручник / О. О. Стеценко, Ю. Г. Даник, М. С. Пастушенко. – К.: МО України, 2004. – 298 с.

47. Тези доповідей VIII Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології – 2016» (22–23 квітня 2016 р.). – Житомир : ЖДТУ, 2016. – 288 с.

48. Техніка авіаційна військової призначеності. Апарати літальні безпілотні. Основні терміни, визначення понять і класифікація: ДСТУ В 7371:2013 / Міністерство економічного розвитку і торгівлі України [Наказ № 1010 від 22.08.2013]. – К., 2014. – С. 2.

49. Філяшкін М. К. Алгоритм роботи адаптивної системи керування бічним рухом безпілотних літальних апаратів на етапі виходу на задану лінію шляху / Філяшкін М. К. Закордонець А. А. // Електроніка та системи керування. – 2009. - №2(20), С. 103-109.

50. Шулежко В. В. Основні напрямки розвитку та застосування безпілотних літальних апаратів: підручник / В. В. Шулежко. – К.: МО України, 2013. – 65 с.

Додаток А

КОД ПРОГРАМИ

Файл objectDetection.py

```
import io
import re
import time
import os
import shutil
from tflite_runtime.interpreter import load_delegate

from annotation import Annotator

import numpy as np
import picamera

from PIL import Image

from tflite_runtime.interpreter import Interpreter


import cv2

CAMERA_WIDTH = 1280
CAMERA_HEIGHT = 960

def main():
    ifEdgeTPU_1_else_0 = 1

    labels = load_labels('coco_names.txt')

    if ifEdgeTPU_1_else_0 == 1:
        interpreter =
Interpreter(model_path='models/ssd_mobilenet_v2_face_quant_postprocess_edgetpu.tflite',
                                    experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
    else:
        interpreter =
Interpreter(model_path='models/ssd_mobilenet_v2_face_quant_postprocess.tflite')

    interpreter.allocate_tensors()
    _, input_height, input_width, _ = interpreter.get_input_details()[0]['shape']

    person_numb = 1
    count_images = 0

    if os.path.isdir('scand_people') == False:
        os.mkdir('scand_people')

    if os.path.isdir('scand_people/' + str(person_numb)) == False:
        os.mkdir('scand_people/' + str(person_numb))
        os.mkdir('scand_people/' + str(person_numb) + '/png')
```

```
            os.mkdir('scand_people/' + str(person_numb) + '/npy')
        else:
            shutil.rmtree('scanned_people/' + str(person_numb))
            os.mkdir('scand_people/' + str(person_numb))
            os.mkdir('scand_people/' + str(person_numb) + '/png')
            os.mkdir('scand_people/' + str(person_numb) + '/npy')

        with picamera.PiCamera(
            resolution=(CAMERA_WIDTH, CAMERA_HEIGHT), framerate=30) as camera:
            camera.rotation = 270
            camera.start_preview()
            try:
                stream = io.BytesIO()
                annotator = Annotator(camera)
                for _ in camera.capture_continuous(
                        stream, format='jpeg', use_video_port=True):
                    stream.seek(0)
                    image_large = Image.open(stream)
                    image = image_large.convert('RGB').resize(
                        (input_width, input_height), Image.ANTIALIAS)
                    start_time = time.monotonic()
                    results = detect_objects(interpreter, image, 0.9)
                    elapsed_ms = (time.monotonic() - start_time) * 1000
                    # print(image.size)

                    annotator.clear()
                    annotate_objects(annotator, results, labels)
                    annotator.text([5, 0], '%.1fms' % (elapsed_ms))
                    annotator.update()

                    ymin, xmin, ymax, xmax, score = get_best_box_param(results,
CAMERA_WIDTH, CAMERA_HEIGHT)

                    if score > 0.99:
                        # print(ymin, " ", xmin, " ", ymax, " ", xmax)
                        # print(image_large.size)
                        img = np.array(image_large)
                        # print("img: ", img.shape)
                        img_cut = img[ymin:ymax, xmin:xmax, :]
                        print(img_cut.shape)
                        img_cut = cv2.resize(img_cut, dsize=(96, 96),
interpolation=cv2.INTER_CUBIC).astype('uint8')
                        img_cut_pil = Image.fromarray(img_cut)
                        img_cut_pil.save(
                            'scanned_people/' + str(person_numb) + '/png/img_' + str(count_images) +
'.png')
                        np.save('scanned_people/' + str(person_numb) + '/npy/img_' +
str(count_images), img_cut)
                        count_images = count_images + 1
                    stream.seek(0)
                    stream.truncate()
```

```
        finally:
            camera.stop_preview()
def load_labels(path):

    with open(path, 'r', encoding='utf-8') as f:
        lines = f.readlines()
        labels = {}
        for row_number, content in enumerate(lines):
            pair = re.split(r'[:\s]+', content.strip(), maxsplit=1)
            if len(pair) == 2 and pair[0].strip().isdigit():
                labels[int(pair[0])] = pair[1].strip()
            else:
                labels[row_number] = pair[0].strip()
    return labels
def set_input_tensor(interpreter, image):
    tensor_index = interpreter.get_input_details()[0]['index']
    input_tensor = interpreter.tensor(tensor_index)()[0]
    input_tensor[:, :] = image
def get_output_tensor(interpreter, index):
    output_details = interpreter.get_output_details()[index]
    tensor = np.squeeze(interpreter.get_tensor(output_details['index']))
    return tensor

def detect_objects(interpreter, image, threshold):

    set_input_tensor(interpreter, image)
    interpreter.invoke()

    boxes = get_output_tensor(interpreter, 0)
    classes = get_output_tensor(interpreter, 1)
    scores = get_output_tensor(interpreter, 2)
    count = int(get_output_tensor(interpreter, 3))

    results = []
    for i in range(count):
        if scores[i] >= threshold:
            result = {
                'bounding_box': boxes[i],
                'class_id': classes[i],
                'score': scores[i]
            }
            results.append(result)
    return results

def get_best_box_param(results, CAMERA_WIDTH, CAMERA_HEIGHT):

    best_boxvalue = 0
    xmin = 0
    xmax = 1
    ymin = 0
    ymax = 1
    for obj in results:
```

```
        if obj['score'] > best_boxvalue:
            best_boxvalue = obj['score']
            ymin, xmin, ymax, xmax = obj['bounding_box']
            if xmin < 0:
                xmin = 0
            if xmax > 1:
                xmax = 1
            if ymin < 0:
                ymin = 0
            if ymax > 1:
                ymax = 1
            xmin = int(xmin * CAMERA_WIDTH)
            xmax = int(xmax * CAMERA_WIDTH)
            ymin = int(ymin * CAMERA_HEIGHT)
            ymax = int(ymax * CAMERA_HEIGHT)
    # print("score: ", best_boxvalue)
    return ymin, xmin, ymax, xmax, best_boxvalue

def annotate_objects(annotator, results, labels):

    for obj in results:

        ymin, xmin, ymax, xmax = obj['bounding_box']
        xmin = int(xmin * CAMERA_WIDTH)
        xmax = int(xmax * CAMERA_WIDTH)
        ymin = int(ymin * CAMERA_HEIGHT)
        ymax = int(ymax * CAMERA_HEIGHT)

        annotator.bounding_box([xmin, ymin, xmax, ymax])
        annotator.text([xmin, ymin],
                       '%s\n%.2f' % (labels[obj['class_id']], obj['score']))

if __name__ == '__main__':
    main()
```

Файл facerecog.py

```
import io
import re
import os
import time
from tflite_runtime.interpreter import load_delegate

from annotation import Annotator

import numpy as np
import picamera

from PIL import Image
from PIL import ImageDraw
import cv2

import pyttsx3

engine = pyttsx3.init()
```

```
from tflite_runtime.interpreter import Interpreter

CAMERA_WIDTH = 1280
CAMERA_HEIGHT = 960


def main():
    ifEdgeTPU_1_else_0 = 1

    labels = load_labels('coco_labels.txt')
    people_lables = load_labels('people_labels.txt')

    if ifEdgeTPU_1_else_0 == 1:
        interpreter = Interpreter(model_path='models/ssd_mobilenet_v2_face_quant_postprocess_edgetpu.tflite',
                                  experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
    else:
        interpreter = Interpreter(model_path='models/ssd_mobilenet_v2_face_quant_postprocess.tflite')

    interpreter.allocate_tensors()
    _, input_height, input_width, _ = interpreter.get_input_details()[0]['shape']

    if ifEdgeTPU_1_else_0 == 1:
        interpreter_emb                                                                                    =
Interpreter(model_path='models/Mobilenet1_triplet1589223569_triplet_quant_edgetpu.tflite',
                                      experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
    else:
        interpreter_emb = Interpreter(model_path='models/Mobilenet1_triplet1589223569_triplet_quant.tflite')

    interpreter_emb.allocate_tensors()

    with picamera.PiCamera(
            resolution=(CAMERA_WIDTH, CAMERA_HEIGHT), framerate=30) as camera:

        camera.rotation = 270
        camera.start_preview()
        try:
            stream = io.BytesIO()
            annotator = Annotator(camera)
            for _ in camera.capture_continuous(
                    stream, format='jpeg', use_video_port=True):
                stream.seek(0)
                image_large = Imge.open(stream)
                image = image_large.convert('RGB').resize(
                    (input_width, input_height), Image.ANTIALIAS)
                start_time = time.monotonic()
                reslts = detect_objects(interpreter, image, 0.5)
                elapsed_ms = (time.monotonic() - start_time) * 1000

                annotator.clear()
                annotte_objects(annotator, results, labels)
                annotator.text([5, 0], '%.1fms' % (elapsed_ms))
                anntator.update()

                ymin, xmin, ymax, xmax, score = get_best_box_param(results, CAMERA_WIDTH,
CAMERA_HEIGHT)

                if score > 0.96:
                    img = np.array(image_large)
                    img_ct = img[ymin:ymax, xmin:xmax, :]
                    img_ct = cv2.resize(img_cut, dsize=(96, 96), interpolation=cv2.INTER_CUBIC).astype('uint8')
                    img_ct = img_cut.reshape(1, 96, 96, 3) / 255.
                    emb = img_to_emb(interpreter_emb, img_ct)
```

```
                get_person_from_embedding(people_lables, emb)

            stream.seek(0)
            stream.truncate()

        finally:
            camera.stop_preview()

def get_person_from_embedding(people_lables, emb):

    num_emb_check = 20
    path = 'scand_people/'
    folder = os.listdir(path)
    folder = sorted(folder)
    averages = np.zeros(len(folder))
    folder_number = 0
    start = time.time()
    for folder in folder:
        average_one_person = 0
        # print(folder)
        files = os.listdir(path + folder + '/embeddings')
        files = sorted(files)
        checked = 0
        for file in files:
            emb2 = np.load(path + folder + '/embeddings' + '/' + file)
            # print(emb.shape)
            norm = np.sum((emb - emb2) ** 2)
            average_one_person = average_one_person + norm
            # print(norm)
            checked = checked + 1
            if checked == num_emb_check:
                break
        average_one_person = average_one_person / num_emb_check
        averages[folder_number] = averages[folder_number] + average_one_person
        folder_number = folder_number + 1
    who_is_on_pic = 0
    lowest_norm_found = 10
    run = 0
    end = time.time()
    print("time for detection: ", end - start)
    for average in averages:
        run = run + 1
        if average < 0.6 and average < lowest_norm_found:
            lowest_norm_found = average
            who_is_on_pic = run
        print(average)
    print("person on pic: ", people_lables[who_is_on_pic])
    if who_is_on_pic > 0:
        engine.say('Hello ')
        engine.say(str(people_lables[who_is_on_pic]))
        engine.runAndWait()

def load_labels(path):

    with open(path, 'r', encoding='utf-8') as f:
        lines = f.readlines()
        labels = {}
        for row_number, content in enumerate(lines):
            pair = re.split(r'[:\s]+', content.strip(), maxsplit=1)
            if len(pair) == 2 and pair[0].strip().isdigit():
                labels[int(pair[0])] = pair[1].strip()
            else:
```

```
            labels[row_number] = pair[0].strip()
    return labels

def set_input_tensor(interpreter, image):

    tensor_index = interpreter.get_input_details()[0]['index']
    input_tensor = interpreter.tensor(tensor_index)()[0]
    input_tensor[:, :] = image

def get_output_tensor(interpreter, index):

    output_details = interpreter.get_output_details()[index]
    tensor = np.squeeze(interpreter.get_tensor(output_details['index']))
    return tensor

def set_input_tensor_emb(interpreter, input):

    input_details = interpreter.get_input_details()[0]
    tensor_index = input_details['index']
    scale, zero_point = input_details['quantization']
    input_tensor = interpreter.tensor(tensor_index)()[0]
    input_tensor[:, :] = np.uint8(input / scale + zero_point)


def img_to_emb(interpreter, input):

    set_input_tensor_emb(interpreter, input)
    interpreter.invoke()
    output_details = interpreter.get_output_details()[0]

    emb = interpreter.get_tensor(output_details['index'])
    scale, zero_point = output_details['quantization']
    emb = scale * (emb - zero_point)
    return emb

def detect_objects(interpreter, image, threshold):

    set_input_tensor(interpreter, image)
    interpreter.invoke()

    # Get all output details
    boxes = get_output_tensor(interpreter, 0)
    classes = get_output_tensor(interpreter, 1)
    scores = get_output_tensor(interpreter, 2)
    count = int(get_output_tensor(interpreter, 3))

    results = []
    for i in range(count):
        if scores[i] >= threshold:
            result = {
                'bounding_box': boxes[i],
                'class_id': classes[i],
                'score': scores[i]
            }
            results.append(result)
    return results


def get_best_box_param(results, CAMERA_WIDTH, CAMERA_HEIGHT):

    best_boxvalue = 0
    xmin = 0
```

```
    xmax = 1
    ymin = 0
    ymax = 1
    for obj in results:
        if obj['score'] > best_boxvalue:
            best_boxvalue = obj['score']
            ymin, xmin, ymax, xmax = obj['bounding_box']
            if xmin < 0:
                xmin = 0
            if xmax > 1:
                xmax = 1
            if ymin < 0:
                ymin = 0
            if ymax > 1:
                ymax = 1
            xmin = int(xmin * CAMERA_WIDTH)
            xmax = int(xmax * CAMERA_WIDTH)
            ymin = int(ymin * CAMERA_HEIGHT)
            ymax = int(ymax * CAMERA_HEIGHT)

    return ymin, xmin, ymax, xmax, best_boxvalue

def annotate_objects(annotator, results, labels):

    for obj in results:

        ymin, xmin, ymax, xmax = obj['bounding_box']
        xmin = int(xmin * CAMERA_WIDTH)
        xmax = int(xmax * CAMERA_WIDTH)
        ymin = int(ymin * CAMERA_HEIGHT)
        ymax = int(ymax * CAMERA_HEIGHT)

        annotator.bounding_box([xmin, ymin, xmax, ymax])
        annotator.text([xmin, ymin],
                       '%s\n%.2f' % (labels[obj['class_id']], obj['score']))

if __name__ == '__main__':
    main()
```

Додаток Б

ПЕРЕЛІК ДОКУМЕНТІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	
Кваліфікаційна робота Глоба.doc	Пояснювальна записка роботи. Документ Word.
Кваліфікаційна робота Глоба.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація Глоба.ppt	Презентація роботи