НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих

комп'ютерних систем

До захисту допущено:

Завідувач кафедри

_____Віталій РОМАНКЕВИЧ

«___»_____20___р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Комп'ютерна система донаведення FPV-дрона на ціль»

Виконав (-ла):
студент IV курсу, групи КВ-11
Гультяєв Дмитро Антонович _____

Керівник:
доцент кафедри СПІСКС, к.т.н., доцент,
Петрашенко Андрій Васильович _____

Консультант з нормоконтролю:
доцент кафедри СПІСКС, к.т.н., доцент,
Клятченко Ярослав Михайлович _____

Рецензент:
доцент кафедри ПЗКС, к.т.н., доцент,
Юрчишин Василь Якович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики
Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
«___»_____20___р.

ЗАВДАННЯ

на дипломний проєкт студента

Гультяєва Дмитра Антоновича
(прізвище, ім'я, по батькові)

1. Тема проєкту Комп'ютерна система донаведення FPV-дрона на ціль, керівник проєкту Петрашенко Андрій Васильович, к.т.н., доцент, затверджені наказом по університету від «___»_____20___р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз існуючих рішень та обґрунтування теми дипломного проєкту
2) Вибір програмних та апаратних засобів системи
3) Реалізація та опис розробленої системи
4) Тестування розробленої системи

1. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

1) Структурна схема роботи системи
2) Алгоритм роботи механізму захоплення та супроводження цілі
3) Алгоритм спрямування БПЛА на захоплену ціль
4) Алгоритм побудови та передачі пакетів команд керування

2. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к. т. н., доцент каф. СП і СКС		

3. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	15.11.2024	
2.	Розроблення та узгодження технічного завдання	30.01.2025	
3.	Аналіз існуючих рішень	01.04.2025	
4.	Підготовка матеріалів першого розділу дипломного проекту	10.04.2025	
5.	Підготовка матеріалів другого розділу дипломного проекту	23.04.2025	
6.	Підготовка матеріалів третього розділу дипломного проекту	07.05.2025	

7.	Підготовка матеріалів четвертого розділу дипломного проекту	15.05.2025	
8.	Підготовка графічної частини дипломного проекту	20.05.2025	
9.	Оформлення документації дипломного проекту	26.05.2025	
10.	Попередній огляд матеріалів диплому на кафедрі	31.05.2025	

Студент _____ Дмитро ГУЛЬТЯЄВ
(підпис) (ініціали, прізвище)

Керівник проєкту _____ Андрій ПЕТРАШЕНКО
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (86 с., 48 рис., 5 додатків, ___аркушів презентації).

Об’єкт розробки – створення системи донаведення FPV-дрона на ціль.

Розроблена система має такі можливості: захоплювати бажану ціль за допомогою графічного прицілу на екрані пілота, супроводжувати захоплену ціль засобами комп’ютерного зору, за командою пілота активувати систему автоматичного управління БПЛА для подальшого враження захопленої цілі.

В ході розробки:

- Проведено аналіз вже існуючих рішень у сфері БПЛА
- Вибрано та проаналізовано найоптимальніші програмні та апаратні засоби, необхідні для розробки системи.
- Побудована структура реалізації системи донаведення FPV дрона на ціль
- Створений та програмно реалізований алгоритм захоплення, супроводження (трекінгу) цілі
- Розроблена програмна реалізація протоколу зв’язку між одноплатним комп’ютером та польотним контролером БПЛА.
- Створений та програмно реалізований алгоритм генерування команд керування FPV-дроном на основі положення захопленого об’єкта.

Впровадження цієї системи допоможе значно підвищити відсоток уражень ворожих цілей FPV дронами, що дасть перевагу у введенні бойових дій в умові дії ворожих РЕБ систем.

Ключові слова: система донаведення на ціль, FPV-дрон, БПЛА, протидія РЕБ системам, комп’ютерний зір, raspberry pi, автоматичне керування БПЛА.

ABSTRACT

The qualification work includes an explanatory note (86 p., 48 figure, 5 supplements, ___arch presentation).

The object of development is the creation of a system for guiding the FPV drone to the target.

The disaggregated system has the following capabilities: to bury the target using an additional graphic sight on the pilot's screen, to guide the buried target using the computer view, and to activate the UAV's automatic control system at the pilot's command for further attack of a buried target.

During the development:

- An analysis of existing solutions in the UAV sphere was carried out The most optimal software and hardware necessary for developing the system have been selected and analyzed.
- The structure of the implementation of the system for guiding the FPV drone to the target was created
- Creation and software implementation of an algorithm for storage, support (tracking) of goals
- The software implementation of the communication protocol between a single-board computer and a multi-UAV controller has been developed.
- Creations and software implementations of an algorithm for generating commands for flying an FPV drone based on the position of the buried object.

The implementation of this system will help to significantly advance the attack on hostile targets using FPV drones, which will give advantage to the introduction of combat operations in the minds of hostile electronic warfare systems.

Keywords: target targeting system, FPV drone, UAV, countermeasures for electronic warfare systems, computer star, raspberry pi, automatic UAV control.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.002 ТЗ	Комп'ютерна система	4		
			донаведення FPV-дрона			
			на ціль			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Комп'ютерна система	2		
			донаведення FPV-дрона			
			на ціль			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Комп'ютерна система	86		
			донаведення FPV-дрона			
			на ціль			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Структурна схема роботи	1		
			системи			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Гультяєв Д.А.			Комп'ютерна система донаведення FPV-дрона на ціль **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А. В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В. О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.006 Д2	Алгоритм роботи	1		
			механізму захоплення та			
			супроводження цілі			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Алгоритм спрямування	1		
			БПЛА на захоплену			
			ціль			
			Схема алгоритм			
	А4	ІАЛЦ.045490.008 Д4	Алгоритм побудови та	1		
			передачі пакетів команд			
			керування			
			Схема алгоритм			
	А4	ІАЛЦ.045490.009 Д5	Комп'ютерна система	9		
			донаведення FPV-дрона			
			на ціль			
			Текст програмного коду			
		Диск CD-ROM	Текст пояснювальної			
			записки.			
			Графічний матеріал			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Комп'ютерна система донаведення FPV-дрона на ціль».

Галузь застосування: військові ударні місії

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є збільшення відсотку влучань FPV-дронів у ворожі цілі, в тому числі під час дії ворожих РЕБ систем.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- Сумісність з програмним забезпеченням Betaflight
- Можливість захоплення бажаної цілі на екрані пілота
- Можливість супроводження захопленої цілі
- Можливість отримувати дані з польотного контролера про поточне положення дрона у просторі

- Можливість формування команд керування на основі поточного та бажаного положення дрону
- Можливість передавати команди керування до польотного контролера дрона засобами протоколу MSP

5.2 Вимоги до апаратного забезпечення

- Процесор: Arm Cortex-A53 @ 1GHz
- Оперативна пам’ять: 512 Мб
- Живлення: 5V
- Підтримка UART
- Наявність аналогового відеовихіду
- Наявність роз’єму USB 2.0 чи USB 3.0
- Підтримка microSD карток об’ємом до 128 Гб

5.3 Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Linux
- Підтримка бібліотек OpenCV та Pi4J

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.11.2024
2.	Розроблення та узгодження технічного завдання	30.01.2025
3.	Аналіз існуючих рішень	01.04.2025
4.	Підготовка матеріалів першого розділу дипломного проекту	10.04.2025
5.	Підготовка матеріалів другого розділу дипломного проекту	23.04.2025
6.	Підготовка матеріалів третього розділу дипломного проекту	07.05.2025
7.	Підготовка матеріалів четвертого розділу дипломного проекту	15.05.2025
8.	Підготовка графічної частини дипломного проекту	20.05.2025
9.	Оформлення документації дипломного проекту	26.05.2025
10.	Попередній огляд матеріалів диплому на кафедрі	31.05.2025

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.004 ПЗ	Комп'ютерна система	86		
			донаведення FPV-дрона			
			на ціль			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Структурна схема роботи	1		
			системи			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Алгоритм роботи	1		
			механізму захоплення та			
			супроводження цілі			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Алгоритм спрямування	1		
			БПЛА на захоплену			
			ціль			
			Схема алгоритм			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Гультяєв Д.А.			Комп'ютерна система донаведення FPV-дрона на ціль **Відомість технічного проекту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А. В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В. О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.008 Д4	Алгоритм побудови та	1		
			передачі пакетів команд			
			керування			
			Схема алгоритм			
	А4	ІАЛЦ.045490.009 Д5	Комп'ютерна система	9		
			донаведення FPV-дрона			
			на ціль			
			Текст програмного коду			
		Диск CD-ROM	Текст пояснювальної			
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БПЛА — безпілотний літальний апарат

РЕБ — радіоелектронна боротьба

FPV — first person view

ППРЧ — псевдовипадкове перелаштування робочої частоти

ШІ — штучний інтелект

FPS (frames per second) — частота кадрів

OSD (On-Screen Display) — показники від датчиків дрону на екрані пілота

ROI (Region of interest) — певна область в зображенні чи відео

ОЗП — оперативний запам’ятовувальний пристрій

ВСТУП

Російсько-українська війна є однією із наймасштабніших, а можливо і наймасштабнішою війною в Європі після Другої світової війни. Особливо [3] гаряча фаза війни почалася з 2022 року і це докорінно змінило політичну та економічну ситуацію у світі, а також, як і будь-який інший масштабний військовий конфлікт, ця війна започаткувала нові підходи та способи введення війни, в тому числі за допомогою новітніх технічних засобів.

Одним із основних технічних засобів, що докорінно змінив характер російсько-української війни та підхід до ведення сучасних бойових дій є БПЛА. На даний момент БПЛА на полі виконують значну кількість задач:

- Збирання розвідувальних даних, спостереження за позиціями ворога, патрулювання лінії розмежування, корегування вогню артилерійських розрахунків, збір картографічних даних
- Знищення ворожої піхоти, техніки, укріплень тощо
- Виконання функцій ретранслятора, тобто передачі з землі сигналів у повітряний простір певного сектору, для забезпечення якісного зв’язку з дронами, що знаходяться в тій місцевості
- Доставка на поле бою боєприпасів, води, харчів, а також евакуація поранених
- Виконання мінування шляхом доставки і закладання мін в певному секторі

Особливо популярними і широко використовуваними на полі бою стали FPV-дрони. До початку повномасштабної війни 2022 року БПЛА цього типу, як правило, використовувалися в перегонах, сфері знімання відеоконтенту тощо. Проте згодом дрони цього типу знайшли застосування і у військовій сфері, через те що вони дають можливість доставляти боєприпаси на великі

відстані і вражати необхідні цілі з великою точністю, порівняно з артилерійськими засобами.

Сучасні технології в сфері комп'ютерного зору, машинного навчання, одноплатних комп'ютерів дають нам можливість створювати системи, що допоможуть збільшити точність ураження ворожих цілей засобами БПЛА, в тому числі FPV-дронами в умовах дії різноманітних завад. У цьому контексті важливою стає розробка систем, що зможуть забезпечити автономність польоту дрону, виявлення, супроводу та ураження цілей, при цьому використовуючи обмежені апаратні ресурси.

Метою даного дипломного проекту є створення системи донаведення FPV-дрона на ціль, що забезпечить можливість захоплення, супроводу та подальшого знищення цілі. Практичне значення цієї системи полягає у збільшенні ефективності, точності та автономності виконання FPV-дроном ударних місій, що може дати значну перевагу на полі бою в умовах сучасних бойових дій.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Актуальність проблеми

В попередньому розділі було зазначено, що FPV-дрони стали новим ефективним засобом для виконання ударних місій в сучасній війні. Проте пілоти цього різновиду БПЛА стикаються з різноманітними викликами при використанні FPV-дронів в бойових задачах.

На даний момент майже всі FPV-дрони керуються вручну за допомогою команд оператора, і це вимагає від нього високої кваліфікації, вміння швидко реагувати на зміну положення БПЛА у просторі, а також особливої концентрації під час виконання фінального етапу польоту ударної місії, а саме влучання та знищення ворожої цілі.

Перешкодою для оператора також стає активний розвиток і використання РЕБ систем, що спрямовані на виявлення, глушіння та дезорганізацію роботи засобів керування FPV-дроном. Для керування БПЛА данного типу, пілоту необхідно використовувати два види сигналу, які поширюються у просторі засобами радіохвиль:

- Відеосигнал, який надсилається дроном на окуляри чи екран пілота, для того щоб він міг розуміти поточне положення літального апарату у просторі та розуміти куди йому варто скерувати БПЛА. На даний момент розповсюдженою практикою є використання аналогового відеосигналу частотою 5.8 ГГц для цих цілей.
- Сигнал керування, який пілот передає за допомогою пульта керування на дрон. На даний момент переважно використовуються частоти від 300 до 1000 МГц.

Відповідно, ворожий РЕБ може створювати перешкоди для передачі та прийому відповідного сигналу. У випадку відеосигналу, РЕБ системи можуть створювати потужний шум або працювати точково, глушачи лише окремі частоти, що призводить до погіршення якості відеосигналу, а згодом спричиняє повну втрату зображення, що в результаті призводить до втрати розуміння положення дрона у просторі і згодом до втрати дрону. Також РЕБ може діяти на сигнал керування, шляхом створення радіоелектронних перешкод, які не дають можливість дрону отримати сигнал керування, або шляхом перехоплення керування, що призводить до втрати можливості передавати відповідні команди на дрон, що в результаті спричіняє падіння БПЛА.

Проте РЕБ системи для протидії FPV-дронів мають, як правило, невеликий радіус дії, і встановлюються ворогом в районі скупчення особового складу, на броньованій техніці, складах, логістичних вузлах тощо. Через те що РЕБ системи даного типу мають невеликий радіус дії, то часто оператор може побачити необхідну ціль візуально в свої окулярах, отримавши картинку з дрону. Проте при наближенні до неї з метою її ураження, він з великою вірогідністю втратить відеосигнал, а дрон — сигнал керування від оператора.

Ще одним викликом використання FPV-дрона є обмеженість поширення радіохвиль у просторі та їх затухання на відстані. Радіохвилі під час свого поширення стикаються з різними об'єктами: будівлі, дерева, гори, інші радіохвилі тощо. В результаті сигнал слабшає, особливо це характерно для відеосигналу частотою 5.8 ГГц. Цей ефект стає особливо помітним, при спробі оператора знизитися для враження ворожої цілі. Це призводить до втрати відеосигналу і втраті БПЛА.

Через перераховані вище проблеми статистика влучання FPV-дрона у ціль складає 20-30%, тобто із 10 дронів, що були запущені для знищення цілі, лише 2-3 дрони досягнуть і вразять її.

Отже, можна зробити висновок, що для підвищення ефективності застосування FPV-дронів критично важливо зосередитися на вирішені проблем, які постають для пілота на кінцевому етапі польоту, а саме зниження, наближення до цілі та її знищення.

Таким чином розробка комп'ютерної системи донаведення на ціль є актуальним питанням, яке має значний практичний інтерес для підвищення ефективності застосування FPV-дронів в сучасній війні. Розроблена система зможе по команді пілота фіксувати і супроводжувати необхідну ціль, після чого перехоплювати керування у пілота та вражати ціль. Перевагами використання такої системи є:

- Підвищення точності враження цілі, так як автоматизоване супроводження цілі забезпечує мінімізацію помилок керування від оператора
- Автономні алгоритми керування БПЛА на основі захопленої цілі забезпечують роботу FPV-дрона навіть за умов активної роботи РЕБ систем та слабкого відеосигналу та сигналу керування

Отже, метою створення системи донавдення на ціль для FPV-дрона є вирішити або послабити вплив проблем, що були описані в даному підрозділі, таким чином збільшивши ефективність застосування даного типу БПЛА по ворожим цілям.

1.2 Огляд існуючих рішень

Виробники БПЛА щороку впроваджують дедалі більше інноваційних рішень в свої вироби, особливо це стосується технологій в основі яких лежать алгоритми комп'ютерного зору та штучний інтелект. Це стало

можливо завдяки оптимізації наявних алгоритму комп'ютерного зору, а також розвитку одноплатних комп'ютерів, які можна помістити майже в будь-який дрон, але при цьому вони забезпечать достатню кількість потужності, щоб коректно працювати з наявними алгоритмами комп'ютерного зору та навіть зі штучним інтелектом. зокрема ці технології впроваджуються у військові дрони для їхнього використання у тактичних бойових операціях, розвідці, евакуаційних місіях тощо.

Серед виробників цивільних дронів найбільших успіхів з впровадження комп'ютерного зору в свої продукти досягла компанія DJI. Ця компанія є одним із найбільших виробників комерційних безпілотників. Моделі DJI FPV та DJI Avata мають розширені можливості управління та стабілізації, але їх системи наведення обмежуються напівавтоматичними функціями, такими як ActiveTrack та APAS (Advanced Pilot Assistance System).

- ActiveTrack дозволяє дрону слідкувати за об'єктом, але не дає можливості спрямування БПЛА на ціль.
- APAS допомагає уникати перешкод, що використовується для можливості автономних польотів на дронах даної команії, але ця система більше орієнтована на зйомку, а не на використання в бойових операціях.

Одним із найпередовішим комерційним дроном у сфері автономного керування та відстеження цілі є Skydio 2+.

Рисунок 1.1 - Skydio 2+

Цей БПЛА завдяки шести ширококутним камерам та обчислювальним потужностям свого процесора може самостійно будувати 3D-карту оточення в реальному часі та обирати оптимальний маршрут для польоту.

Вбудовані алгоритми автономного керування Skydio Autonomy Engine використовують штучний інтелект та алгоритми комп’ютерного зору, щоб оцінювати траєкторію руху та передбачати поведінку цілі, на основі того, що дрон здатний ідентифікувати та слідувати за рухомими об'єктами, навіть якщо вони тимчасово виходять із поля зору, тобто він здатен аналізувати контекст і передбачає подальше переміщення цілі, що робить його надзвичайно ефективним у динамічних умовах.

Особливу увагу слід приділити розробкам американської компанії AeroVironment, а саме мова йде про Switchblade 300 та Switchblade 600.

Switchblade 300 і Switchblade 600 — це баражуючі боєприпаси за класифікацією, але у російсько-українській війні отримали статус БПЛА. Ці технічні засоби розроблені для швидкого та точного ураження цілей на полі бою. Вони поєднують функції розвідки, спостереження та ураження ворожих цілей, забезпечуючи при цьому високу точність.

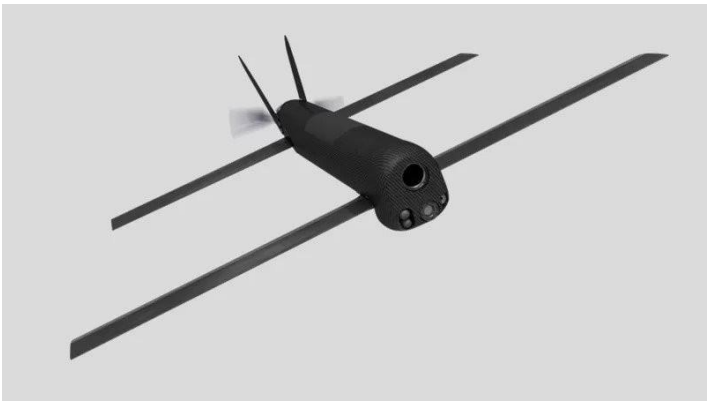


Рисунок 1.2 - Switchblade 300

Головна особливість Switchblade є інтеграція дрона і боєприпасу в одному пристрої [1]. Вони можуть барражувати(зависати у повітрі), власне

тому за класифікацією вони є баражуючими боєприпасами, після цього ідентифікувати необхідну ціль і виконати удар по цій цілі з високою точністю.

Switchblade 300 та 600 використовують сучасні технології наведення та управління, завдяки яким і досягається висока точність ураження навіть по рухомих цілям.

Повний алгоритм польоту цього дрона складається з таких пунктів:

- БПЛА прямує до цілі за допомогою системи GPS та попередньо заданих координат ворожої цілі. Або дрон прямує до цілі шляхом ручного керування пілотом.
- Після підльоту до цілі Switchblade використовує вбудовані алгоритми комп'ютерного зору для фіксації та відстеження цілі.
- Завдяки вбудованому штучному інтелекту після фіксації цілі БПЛА здатен розпізнати тип цієї цілі (людина, техніка, броньована техніка)
- Якщо ціль є рухомою дрон здатен самостійно завдяки математичним алгоритмам розрахувати траєкторію та швидкість руху цілі. Ця інформація буде використана при безпосередньому уражені цілі
- Після підтвердження оператора на ураження цілі, БПЛА переходить в режим автоматичного керування і починає «заходити» на ціль. При цьому він завдяки ШІ здатен розпізнати слабке місце цієї цілі, наприклад: якщо Switchblade атакує бронетехніку, він може націлитися в місце, де розташований двигун чи боєприпаси в броньованій техніці.

Отже, на даний момент, Switchblade 300 та Switchblade 600 мають одну з найрозвиненіших систем донаведення серед баражуючих боєприпасів та БПЛА в світі.

Всім відомий іранський дрон SHAHED-136, ймовірно, теж оснащений системами донаведення, що дозволяє йому самостійно орієнтуватися в

просторі та наводитися на ціль. Точні технічні характеристики на даний момент залишаються засекреченими, проте ми можемо припустити, що Shahed-136 ймовірно використовує камеру або тепловізор для кінцевого наведення на ціль [2]. Це може бути реалізовано кількома способами:

- Дрон може використовувати звичайну камеру, яка аналізує зображення в видимому спектрі для розпізнавання об'єктів, наприклад на основі різниці контрастів.
- ІЧ-наведення або, простіше кажучи, тепловізор для визначення теплових контрастів, для атаки на об'єкти, що випромінюють тепло.

Також варто згадати про баражуючий боєприпас, який можна віднести до класу БПЛА, а саме IAI Harop, розроблений ізраїльською компанією Israel Aerospace Industries (IAI).

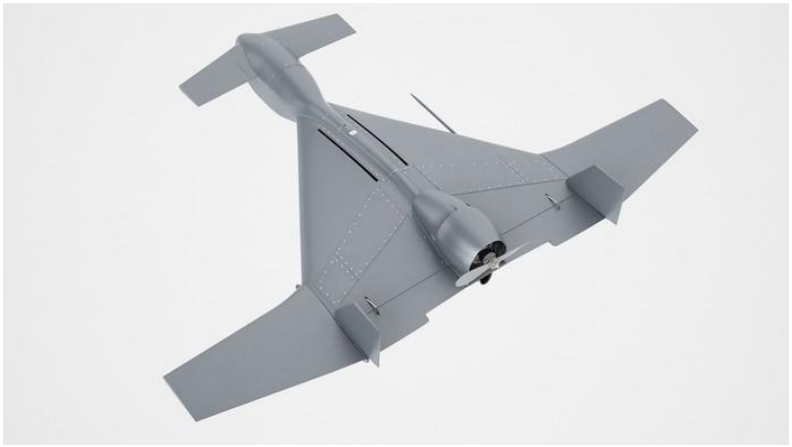


Рисунок 1.3 - IAI Harop

Цей дрон-камікадзе створений для ураження цілей за допомогою потужної оптичної та тепловізійної системи наведення, що дозволяє йому працювати в будь-який час доби та за складних погодних умов. Harop оснащений електрооптичною/інфрачервоною (EO/IR) камерою, яка дозволяє оператору здійснювати візуальний пошук цілей та точне наведення перед атакою. Така система забезпечує високу точність удару, оскільки оператор

отримує чітке зображення цілі у видимому та інфрачервоному спектрах, що особливо важливо для розпізнавання об’єктів, замаскованих або частково прикритих.

Нагор може функціонувати як у автономному режимі, так і під керуванням оператора. У разі автономного використання дрон патрулює заданий район, шукаючи ворожі об’єкти за допомогою своїх сенсорів. Як тільки ціль виявлена, Нагор виконує точне наведення та здійснює атаку, використовуючи вбудовану боєголовку. Водночас оператор може взяти управління на себе, що дозволяє не тільки змінити ціль, а й перенацілити дрон у польоті або навіть скасувати удар у разі необхідності.

Ще однією важливою особливістю цієї системи є тепловізійна підтримка, що дає можливість Нагор діяти в умовах обмеженої видимості, наприклад, уночі або під час сильного задимлення. Інфрачервоні сенсори дозволяють розрізняти теплові сигнатури живої сили, транспортних засобів і навіть працюючої електроніки, що значно підвищує ефективність розпізнавання ворожих об’єктів.

1.3 Аналіз недоліків розглянутих рішень

Аналізуючи всі розглянуті дрони з системами оптичного та тепловізійного донаведення, можна виділити кілька ключових недоліків, які є характерними для майже всіх розглянутих моделей дронів вище.

Перш за все, слід звернути увагу на затримку обробки даних, яка є критичною для високошвидкісних FPV-дронів та баражуючих боєприпасів. Більшість дронів використовують потужні алгоритми комп’ютерного зору та нейромережі для аналізу зображення, проте ці процеси потребують значних обчислювальних ресурсів. Наприклад, Skydio 2+ має розвинену систему автономного керування, але навіть вона іноді демонструє затримку реакції при складних маневрах або в умовах недостатнього освітлення. Аналогічна

проблема властива й AeroVironment Switchblade 300/600 – ці дрони здатні самостійно визначати та знищувати цілі, але їх ефективність значно залежить від швидкості обробки інформації, що може призводити до помилок у реальних бойових умовах.

По-друге, всі системи, які були перераховані вище, є закритого типу і, як правило, накладають обмеження на гнучкість налаштувань. Головною проблемою є саме те, що програмне забезпечення в цих БПЛА майже не піддається налаштуванням. На даний момент, в російсько-українській війні майже кожен розрахунок БПЛА індивідуально налаштовує та змінює програмне забезпечення і поведінку дрона під конкретну задачу та конкретний напрямок фронту, де вони виконують поставлену задачу. Особливо це стосується FPV дронів, основна перевага яких, в цивільному та військовому використанні, є саме можливість гнучко налаштовувати їх. Окрім програмного забезпечення, військовослужбовці часто виконують апаратні зміни в дроні заради підвищення ефективності виконання бойового завдання, проте в системах, що були описані вище такої можливості немає і скоріше за все апаратні зміни призведуть до виведення з ладу БПЛА.

По-третє, як було описано в попередньому пункті, системи донавдення, які були розглянуті вище є закритого типу, а отже не можуть бути легко інтегровані на інші зразки БПЛА. В сучасній війні старі безпілотні системи швидко втрачають актуальність, проте з’являються нові рішення, особливо це стосується апаратної частини дрону, таким чином важливою перевагою систем донавдення є саме швидка і проста інтеграції її на різні безпілотні літальні апарати.

Вчетверте, одним із вагомих недоліків розглянутих безпілотних літальних апаратів є їхня вартість, яка значною мірою впливає на можливість їхнього широкого використання в бойових умовах в Україні. Для прикладу Switchblade 300/600, який використовується як баражуючий боєприпас, має

орієнтовну вартість 6000 доларів за одиницю, що означає, що його використання у великих кількостях малоймовірно, через обмежений військовий бюджет України. Більш потужна версія, Switchblade 600, яка має покращене озброєння та можливості наведення, коштує вже близько 50000 доларів, що значно обмежує її масове застосування. Щодо більш складних бойових систем, таких як IAI Harop, ціна є ще вищою. За даними на 2009 рік, Індія придбала 10 дронів за 100 мільйонів доларів, що становило близько 10 мільйонів за одиницю. А отже, для України використання подібних дорогих дронів, як Harop чи Switchblade 600, є недоцільним через високу вартість і варто розглядати можливість створення альтернативної бюджетної системи донавдення для FPV дрона.

1.4 Вибір технологій для розробки власної системи

В попередньому розділі був проведений аналіз недоліків існуючих систем донаведення для БПЛА і на основі цього аналізу можна зробити висновок, що основними недоліками є: відсутність гнучких налаштувань, обмежена можливість інтегрувати системи донавдення на інші БПЛА, в тому числі і FPV, а також висока ціна.

Виходячи з перелічених недоліків, основною задачею є створення універсальної системи, яка зможе бути гнучко налаштована під задачі оператора, і яка буде бюджетною для подальшого встановлення її на FPV дрони.

Основою системи, що буде задовольняти поставлені задачі, є одноплатний комп’ютер, оскільки він об'єднує в собі достатню обчислювальну потужність для виконання складних алгоритмів обробки зображень у реальному часі, а також можливість підключення периферійних пристроїв, таких як камери. До того ж, побудова системи на основі

одноплатного комп’ютера дозволить нам отримати незалежну систему, яку можна буде легко встановлювати на різні FPV дрони.

Для захоплення зображення буде використовуватися звичайна цифрова камера з необхідною роздільною здатністю та кількістю мегапікселів, також основною характеристикою для вибору камери стане field of view, яке вимірюється в градусах.

Частина, яка буде відповідальна за фіксацію і супроводження бажаної цілі буде побудована на основі комп’ютерного зору без використання штучного інтелекту. Це пов’язано з тим, що штучний інтелект вимагає значних обчислювальних ресурсів від апаратного забезпечення, а отже з’явиться необхідність використовувати більш потужніші, складніші та дорожчі одноплатні комп’ютери, що призведе не тільки до збільшення ціни системи, а також до збільшення споживання батареї літального апарату і відповідно зменшення його радіусу дії та часу польоту. При цьому використання ШІ значно не збільшить ефективність роботи запропонованої системи.

Отже, комбінація одноплатного комп’ютера, комп’ютерного зору та цифрової камери створює міцну основу для розробки високоточної системи донаведення FPV-дрона, при цьому задовольняючи поставлені задачі.

2. ПРОГРАМНІ ТА АПАРАТНІ ЗАСОБИ СИСТЕМИ

2.1 Основи будови FPV дрона

FPV-дрони (First Person View дрони) — це безпілотні літальні апарати, оснащені камерами та системами передачі відеосигналу в реальному часі, керування БПЛА цього типу здійснюється вручну за допомогою пульта керування. Вони використовуються для різних задач, включаючи спортивні змагання, аерозйомку, моніторинг територій, але на даний момент вони найактивніше використовуються у військових операціях. Основна особливість FPV-дронів полягає в тому, що оператор отримує зображення з камери дрона через спеціальні окуляри або монітор, що дозволяє керувати польотом у реальному часі з високою точністю.

Рис. 2.1 — FPV дрон з боєприпасом

У військових операціях FPV-дрони зазвичай керуються вручну, що дає можливість оператору швидко адаптуватися до змінної бойової обстановки. Для цього використовуються радіопередавачі (пульти керування), які забезпечують передачу команд польоту з мінімальною затримкою. Завдяки сучасним системам зв’язку, таким як ExpressLRS або TBS Crossfire, можна забезпечити стабільний сигнал навіть на значних відстанях, що дозволяє

операторам працювати на безпечній відстані від ворога. З ручним керуванням ФПВ дрон здатен отримувати команди на відстані до 20-40 км, в залежності від рельєфу.

Базовий FPV дрон має такі компоненти:

- Рама
- Двигуни
- Пропелери
- ESC (електронний регулятор швидкості)
- FC (польотний контролер)
- Камера
- Відеопередавач
- Приймач керування
- Батарея

Рама є основою конструкції FPV-дрона та визначає його міцність та вагу. Вона зазвичай виготовляється з вуглепластику, що забезпечує високу жорсткість і низьку вагу, що критично важливо для бойових дронів, задача яких нести високу вагу при цьому маючи максимально можливий радіус дії.

Двигуни FPV-дронів є компонентом, що створює крутний момент та оберти для пропелерів. Двигуни цього типу дронів є, як правило, безколекторними і забезпечують високу ефективність, потужність і довговічність. Завдяки своїй конструкції, вони працюють на високих обертах і можуть швидко змінювати швидкість обертання для точного керування польотом. Це є важливим фактором, через те що, саме чотири двигуни на квадрокоптері впливають на утримання та зміну позиції, а також орієнтацію дрону у просторі. Через те що ФПВ дрон не має керуючих поверхонь, які наприклад є у БПЛА літакового типу, двигуни повинні максимально швидко та точно змінювати оберти, щоб забезпечувати бажану поведінку дрону.

ESC (електронний регулятор швидкості) керує швидкістю обертання двигунів, отримуючи сигнали від контролера польоту. Він перетворює команди, що надходять з польотного контролера, на вихідну керуючу напругу на кожен двигун, задаючи двигуну певну швидкість обертання. Таким чином регулятор швидкості забезпечує плавне регулювання тяги двигунів, що дає нам змогу чітко контролювати FPV дрон.

Камера є основним сенсором FPV-дрона, який передає зображення оператору або системі автоматичного наведення. Як правило, на FPV дронах використовуються аналогові камери з високою кількістю горизонтальних ліній, за це відповідає значення TVL, і воно, зазвичай, складає від 1000 до 1500 одиниць. У бойових дронах можуть використовуватися камери з інфрачервоним спектром або нічні камери, які працюють в умовах сутінок.

Відеопередавач (VTX) є ключовим компонентом FPV-дрона, який забезпечує передачу відеосигналу з камери на наземну станцію або FPV-окуляри оператора. Основний частотний діапазон для FPV-дронів – 5.8 ГГц. Проте у військових умовах можуть використовуватися інші частоти, такі як 1.2 ГГц, 2.4 ГГц або 3.3 ГГц, оскільки нижчі частоти мають кращу проникаючу здатність через перешкоди, такі як будівлі чи рослини. Також ваговим фактором для військового застосування є потужність відеопередавача, чим вона вище, тим більший радіус застосування БПЛА можна досягти.

Приймач керування приймає команди від пульта оператора та передає їх контролеру польоту. У FPV-дронах для військових застосувань використовують системи з великою дальністю дії та низькою затримкою, такі як ExpressLRS або TBS Crossfire. Проте є і інші системи, які забезпечують вищу РЕБ захищеність, наприклад завдяки використанню ППРЧ.

Польотний контролер (FC) є центральним та одним із найважливіших компонентів FPV дрону, без якого не можливе функціонування цього типу БПЛА. Контролер обробляє дані від датчиків, виконує стабілізацію FPV дрона та реалізує команди оператора. Він отримує інформацію про поточний стан дрона, обчислює коригувальні команди та надсилає їх до електронних регуляторів швидкості (ESC), що керують двигунами.

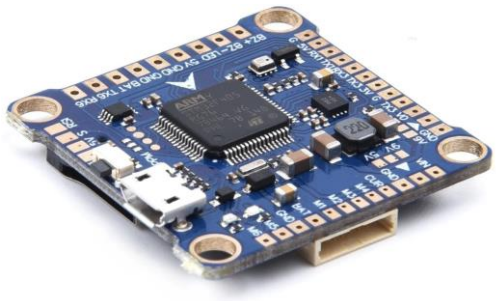


Рис 2.2 — Польотний контролер

Основу польотного контролера складає мікропроцесор або мікроконтролер. Найпоширенішими є мікроконтролери сімейства STM32, а саме серії F4, F7 та H7. Найпопулярнішими є польотні контролери на базі мікропроцесорів сімейства F4, а саме F405. Ці контролери мають достатню кількість пам’яті для зберігання прошивки, а також вони здатні достатньо швидко проводити необхідні обчислення для забезпечення мінімальної затримки в керуванні FPV дрона.

На польотному контролері встановлюються різноманітні датчики необхідні для визначення позиції дрона у просторі та обрахунку необхідних поправок для видачі відповідних команд на двигуни БПЛА. Одним із

найважливішим із них є інерціальний вимірювальний блок (IMU), що складається з гіроскопа та акселерометра.

- Гіроскоп є основним датчиком без якого не можливий політ FPV дрона. Цей датчик визначає кутові швидкості дрона навколо трьох осей, тобто він дозволяє визначати, наскільки швидко змінюється орієнтація дрона по цим трьом осям [7]. Ця інформація є основою для розрахунку польотним контролером необхідних поправок для стабілізації або зміни положення дрону у просторі.
- Акселерометр вимірює лінійне прискорення дрона вздовж трьох осей та дозволяє визначати його нахил відносно горизонту. Дані з цього датчику критично важливі для стабілізаційних режимів польоту, таких як «Angle Mode» або «Horizon Mode», де контролер автоматично утримує дрон у горизонтальному положенні.

Ще одним із основних датчиків є барометричний датчик, або альтиметр, який використовується для визначення висоти дрона на основі зміни атмосферного тиску, що дозволяє реалізовувати функції утримання висоті. Також на дронах можна зустріти магнітометри для визначення азимуту та GPS модулі для обрахунку координат положення БПЛА.

Для взаємодії зі всіма переліченими датчиками польотний контролер використовує різні протоколи в залежності від типу сигналів та необхідної швидкості отримання їх, нижче буде наведений перелік основних протоколів.

SPI (Serial Peripheral Interface) – це високошвидкісний протокол зв’язку, який застосовується для підключення пристроїв, що потребують швидкої передачі даних. В FPV дронах до таких пристроїв належить гіроскоп, акселерометр та флеш пам’ять для запису статистики польоту. SPI використовує окремі лінії для даних (MOSI та MISO), синхронізації (SCLK) і вибору пристрою (CS). Основною перевагою SPI є висока швидкість (до десятків Мбіт/с) і низькі затримки, що робить його ідеальним для датчиків по

типу гіроскопа, отримані значення з якого є критичними для стабільного польоту БПЛА.

I2C (Inter-Integrated Circuit) – це двопровідний послідовний протокол зв’язку, який використовується для обміну даними між політним контролером і другорядними датчиками, до прикладу: барометр, магнітометр та GPS модуль. Він складається з ліній SDA (дані) і SCL (синхронізація), що дозволяє підключати кілька пристроїв до однієї шини. I2C є енергоефективним і простим у реалізації, але має відносно низьку швидкість передачі даних (до 3.4 Мбіт/с), тому його зазвичай використовують для обміну невеликими за обсягом і некритичними для польоту даними.

UART (Universal Asynchronous Receiver-Transmitter) – це асинхронний послідовний протокол, який є основним в комунікації між польотним контролером та різними пристроями в FPV-дронах, а саме: приймачів керування, відеопередавачів, GPS модулів, телеметрії з регулятора швидкості тощо. Він передає дані по двох лініях (TX – передача, RX – прийом) без необхідності окремої синхронізації.

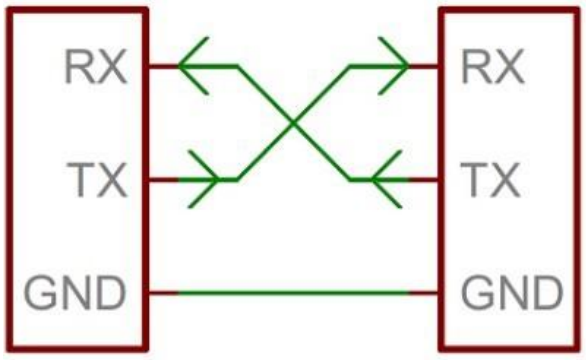


Рис 2.3 - Схема з’єднання пристроїв за допомогою протоколу UART

Швидкість UART може варіюватися від 9600 біт/с до понад 1 Мбіт/с, що дозволяє ефективно передавати команди керування та діагностичні дані.

Важливою особливістю є підтримка таких протоколів, як MSP, Crossfire та SmartAudio, які забезпечують взаємодію дрона з багатьма системами. Також UART є основним протоколом для комунікації між одноплатним комп'ютером та польотним контролером FPV дрона.

Також варто в контексті цього розділу згадати про прошивки, які використовуються для FPV дронів. На даний момент існує велика кількість різноманітних прошивок для польотних контролерів, наприклад: Betaflight, INAV, ArduPilot, KISS тощо. Саме Betaflight здобула неабияку популярність серед цивільних власників FPV дронів, а найголовніше ця прошивка користується великою популярністю серед військових, а отже систему донавдення доцільно інтегрувати саме з прошивкою Betaflight.

Betaflight здобув популярність через широкий спектр підтримки апаратного забезпечення, включаючи сучасні сенсори, протоколи зв'язку та алгоритми стабілізації. Прошивка підтримує високочастотні оновлення IMU (до 8 кГц), що дозволяє швидко аналізувати та коригувати положення дрона в просторі. Крім того, Betaflight пропонує розширені фільтраційні алгоритми, які зменшують вплив вібрацій на сенсори, забезпечуючи більш стабільний політ, що критично важливо для військового застосування.

Ще однією важливою функцією Betaflight є підтримка різних протоколів зв'язку, включаючи UART для телеметрії та керування, а також DShot для цифрового управління ESC. Завдяки використання цих протоколів дрон більш ефективно споживає батарею, а відповідно має більший радіус застосування.

Також варто згадати систему налаштування параметрів в цій прошивці, яка дозволяє змінювати такі характеристики, як коефіцієнти PID-регулятори, криві газу тощо. Це особливо важливо для військових FPV-дронів, оскільки налаштування можна адаптувати під конкретні умови бойових завдань, вагу боєприпасу та побажання пілота.

Отже, на основі отриманих знань про базові принципи апаратного та програмного забезпечення FPV-дрона можна побудувати архітектуру власної системи донавдення на ціль для подальшої успішної інтеграції її у екосистему даного типу БПЛА.

2.2 Алгоритми захоплення та тренінгу цілі

У контексті розробки системи донавдення для FPV-дронів захоплення та тренінг цілі є ключовими етапами для забезпечення роботи системи. FPV-дрони, через специфіку їх використання в бойових умовах, повинні швидко і надійно фіксувати та утримувати ціль.

Узагальнено, усі методи захоплення та тренінгу цілі можна поділити на дві основні категорії: традиційні алгоритми, що базуються на ознаках, та сучасні, які використовують глибоке навчання.

Традиційні підходи спираються на ручне виділення ознак зображення. Наприклад, метод HOG (Histogram of Oriented Gradients) аналізує орієнтації градієнтів пікселів для побудови дескрипторів, які є характерними для форми об'єкта. Такий підхід був широко використаний для задач виявлення пішоходів або транспортних засобів. Іншим класичним методом є каскади Хаара (Haar Cascades), які дозволяють швидко виявляти об’єкти на основі простих прямокутних ознак, об’єднаних у каскад слабких класифікаторів. Хоча такі методи є ефективними та можуть працювати в реальному часі на пристроях з обмеженими ресурсами, вони мають низьку стійкість до змін освітлення, фону або орієнтації об'єкта.

Сучасні підходи використовують згорткові нейронні мережі (CNN), які автоматично визначають релевантні ознаки зображення на основі великої кількості прикладів. Наприклад, популярні архітектури для детекції об'єктів, як-от YOLO (You Only Look Once) або SSD (Single Shot Multibox Detector), дозволяють не лише точно локалізувати об'єкт, але й класифікувати його в

режимі реального часу. Такі моделі здатні враховувати контекст сцени, мають високу стійкість до завад і змін зовнішніх умов, але вимагають більшої обчислювальної потужності або оптимізації.

В контексті розробки системи донавдення для FPV-дрона за основу будуть взяті традиційні методи захоплення та трекінгу об’єкту. Це пов’язано з двома основними причинами:

- Сучасні підходи захоплення та трекінгу об’єкту вимагають значні за потужність апаратні ресурси. Звідси необхідні дорожчі та більші за розміром одноплатні комп’ютери, використання яких є недоцільним на стандартних FPV-дронах
- Задача системи донаведення не полягає у розпізнаванні та ідентифікації цілі, тобто задача системи зафіксувати об’єкт згідно команди пілота. Тому використання сучасних підходів на основі ШІ є недоцільним.. Проте підхід з використанням традиційних методів дає нам можливість забезпечити мінімальну затримку для захоплення та трекінгу бажаної цілі.

Отже, для того, щоб визначити який алгоритм чи метод трекінгу об’єкту доцільно використати, варто розглянути декілька популярних алгоритмів з їх перевагами та вадами.

Алгоритм MOSSE (Minimum Output Sum of Squared Error) є одним із найшвидших методів для трекінгу об'єктів у комп'ютерному зорі. Його принцип базується на використанні кореляційного фільтра, який створюється на основі шаблону об'єкта з першого кадру. Цей фільтр потім використовується для пошуку об'єкта в кожному новому кадрі. Для досягнення високої швидкості обчислень MOSSE використовує перетворення Фур'є, що дозволяє обробляти зображення в частотній області, замість того щоб порівнювати кожен піксель безпосередньо, що значно прискорює процес.

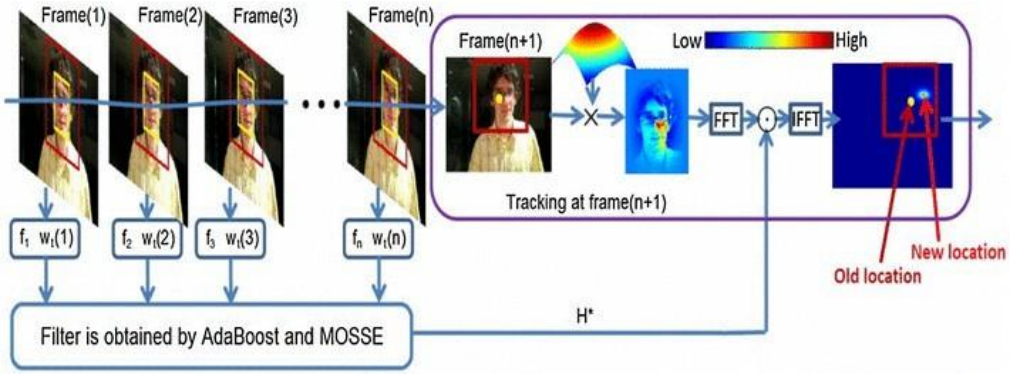


Рис. 2.4 - Принцип роботи алгоритму MOSSE

MOSSE добре працює в умовах, коли об'єкт стабільний, тобто не змінює форму і розмір, не перекривається іншими об'єктами, а також не обертається. Однак при великих змінах об'єкта або його перекритті з іншими об'єктами MOSSE може втратити точність. Тому цей алгоритм відзначається високою швидкістю, але має обмежену точність в складних сценах.

KCF (Kernelized Correlation Filter) є покращеною версією MOSSE. Він використовує ядерні методи для обробки зображень, що дозволяє працювати не лише з простими ознаками (як у MOSSE, наприклад, яскравість пікселів), а й з більш складними ознаками, такими як HOG (Histogram of Oriented Gradients), що дозволяє кращим чином відслідковувати контури і форму об'єкта. KCF використовує методи, що дозволяють підвищити точність при пошуку об'єкта в кадрі, навіть коли об'єкт має схожий колір з фоном або коли фон складний. Цей метод працює швидко, оскільки також використовує частотну область, але є більш точним за MOSSE завдяки використанню складніших ознак. Однак він потребує більше ресурсів для обробки даних і може бути повільнішим за MOSSE, особливо на слабких системах.

CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability) — це ще більш складний метод, який додає до кореляційного фільтра концепцію надійності каналів і просторової регуляризації. Це дозволяє визначити, які частини зображення є найбільш надійними для

відстеження об'єкта, зменшуючи вплив шуму та змін фону. CSRT більш стійкий до змін форми об'єкта, обертання, зміни розміру та часткового перекриття.

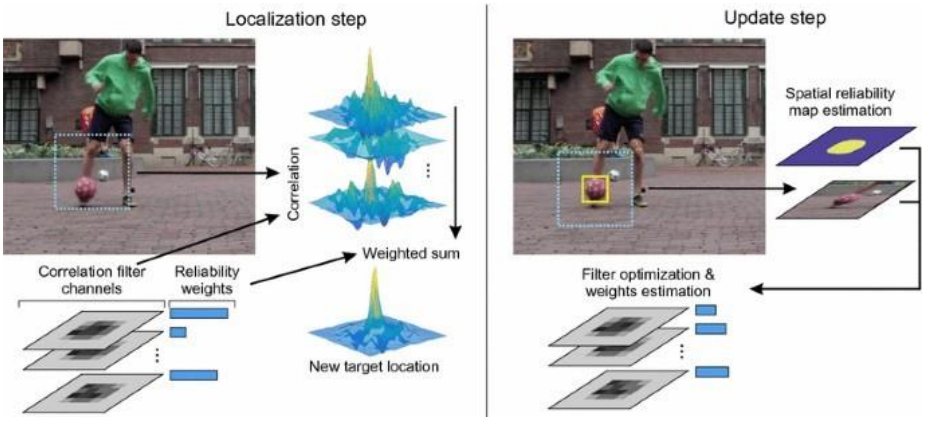


Рис. 2.5 - Принцип роботи алгоритму CSRT

Алгоритм здатний досягати високої точності навіть у складних сценах, де об'єкти можуть частково зникати або змінювати свою позу. Однак цей метод вимагає більше обчислювальних ресурсів, ніж KCF, і може бути повільнішим, тому він менш підходить для реального часу на слабких пристроях.

Staple (Structural and Textural Appearance Model) є комбінованим методом, що використовує одночасно структурні ознаки об'єкта та текстурні (кольорові) ознаки. Це дозволяє трекеру залишатися на об'єкті, навіть коли він змінює форму, але зберігає стабільний колір. Staple поєднує переваги методу кореляційного фільтра з кольоровими гістограмами, що дозволяє точніше відстежувати об'єкти на складних фонах і в умовах, коли об'єкт зазнає значних змін у формі чи положенні.

Рис 2.6 — Приклад роботи алгоритму Staple в складних умовах

При цьому Staple не так швидко працює, як MOSSE, але є більш точним і гнучким, зокрема для відстеження об'єктів, які мають яскраво виражений колір, що дозволяє досягати кращих результатів у складних ситуаціях, коли об'єкти змінюють свій вигляд.

Усі ці алгоритми мають свої сильні та слабкі сторони. MOSSE забезпечує високу швидкість, але може втратити об'єкта на складних фонах або при значних змінах об'єкта. KCF дає кращу точність завдяки використанню складніших ознак, але потребує більше ресурсів. CSRT є дуже точним і стійким до змін, але більш ресурсоємним і повільнішим, а Staple забезпечує хорошу точність.

На основі теоретичних відомостей та практичних експериментів з перерахованими вище алгоритмами, було визначено, що найкраще для задачі донавдення FPV-дрона на ціль підходить алгоритм MOSSE. Експериментальним шляхом було встановлено, що допрацювавши цей алгоритм під задачу донаведення на ворожу ціль, можна отримати гарну точність трекінгу цього об’єкту. Основний недолік цього алгоритму, що він погано працює, коли об’єкт починає змінювати форму. Практичний досвід використання подібних систем на полі бою показав, що при фіксуванні цілі переважно вона надалі не змінює свою форму, а отже недолік алгоритму нівелюється. Проте використання алгоритму MOSSE дозволить нам не

перевантажить апаратну частину системи. Також алгоритм MOSSE є найшвидшим із перерахованих алгоритмів, а отже він може максимально швидко реагувати на зміну положення цілі, що є критичним під час застосування FPV-дронів в бойових умовах.

2.3 Вибір апаратних засобів для реалізації системи

Загальні вимоги до апаратного забезпечення комп'ютерної системи донаведення FPV-дрона будуються на основі того, що система повинна забезпечити стабільну та продуктивну роботу. Насамперед системі необхідно мати високу продуктивність, достатню для обробки відеопотоку в реальному часі. Це означає, що одноплатний комп'ютер має бути здатним з високою частотою зчитувати кадри з камери (приблизно 30-40 FPS), застосовувати алгоритми комп'ютерного зору, в нашому випадку алгоритм трекінгу, а згодом генерувати команди для польотного контролера дрону.

При цьому надзвичайно важливим є компактність одноплатного комп'ютера та його споживання батареї FPV-дрона, оскільки живлення одноплатного комп'ютера буде забезпечуватися саме нею. Звідси велике споживання енергії системою донавдення може призвести до значного зменшення радіусу дії БПЛА. Також апаратне забезпечення не повинно суттєво збільшувати вагу дрона, адже збільшення ваги дрону призведе до того, що або зменшиться радіус дії цього БПЛА, або зменшиться вага корисного навантаження для ураження цілі.

Також критичною вимогою є наявність різноманітних інтерфейсів для взаємодії з польотним контролером дрона та взаємодії з камерою для отримання зображення. Для взаємодії між одноплатним комп'ютером та польотним контролером, як правило використовується протокол UART. Для взаємодії між одноплатним комп'ютером та камерою використовуються інтерфейси CSI або USB. Ще однією важливою вимогою до одноплатного

комп’ютера є наявність аналогового відеовиходу з сигналом протоколу PAL або NTCS. Це пов’язано з тим, що оброблене відео одноплатний комп’ютер буде передавати на вхід відеосигналу польотного контролера. При цьому переважна більшість FC підтримує тільки аналоговий вхідний відеосигнал.

Також великою перевагою буде наявність вбудованого графічного процесора (GPU). Завдяки цьому центральний процесор одноплатного комп’ютера буде менше навантажуватися задачами обробки зображення, а отже ми отримаємо кращу швидкодію системи.

Одним із найпопулярніших виробників одноплатних комп’ютерів є компанія Raspberry Pi Foundation з серією комп’ютерів Raspberry Pi відповідно. Вибір саме цього виробника зумовлений тим, що Raspberry Pi є надійними одноплатними комп’ютерами з підтримкою широкого спектра бібліотек комп’ютерного зору та бібліотек для програмного коду на різних мовах програмування. Для того, щоб обрати модель комп’ютера під задачу донавдення FPV-дрона на ціль, розглянемо декілька найпопулярніших моделей цієї лінійки.

Raspberry Pi 4 — це флагманська модель у лінійці Raspberry Pi, що надає значно вищу продуктивність порівняно з попередніми версіями. Вона обладнана чотириядерним процесором ARM Cortex-A72 з тактовою частотою 1.5 ГГц, що забезпечує достатню обчислювальну потужність для завдань комп’ютерного зору, трекінгу об’єктів тощо.

Оперативна пам’ять доступна у конфігураціях 2 ГБ, 4 ГБ або 8 ГБ LPDDR4, що дозволяє обробляти відеопотік високої роздільності.

Raspberry Pi 4 має такі інтерфейси:

- 2 порти USB 3.0 та 2 порти USB 2.0,
- дві HDMI-лінії,
- 40-контактний GPIO (включно з UART, I2C, SPI),

- роз’єм CSI для підключення камер (наприклад для камери Raspberry Pi Camera Module V2)
- TV-OUT (аналоговий відеовихід)

Рис 2.7 — Raspberry Pi 4

Також Raspberry Pi 4 має вбудований графічний процесор, що дає нам можливість ефективніше працювати з обробкою зображень.

Споживання Raspberry Pi 4 складає приблизно 3-5Вт. Розміри цього одноплатного комп’ютера складають 88мм*58мм.

Хоч ця модель є одною із найпродуктивніших моделей в модельному ряді Raspberry Pi, я пропоную розглянути менш потужнішу модель, а саме Raspberry Pi Zero 2 W.

Raspberry Pi Zero 2W — це мініатюрний одноплатний комп’ютер, проте він має доволі непогану продуктивність завдяки новітньому чипу. Основою Zero 2W є чотириядерний процесор ARM Cortex-A53 (SoC Broadcom BCM2710A1) з частотою 1 ГГц. Це дозволяє Zero 2W виконувати базові обчислення, попередню обробку відео або невимогливі моделі комп’ютерного зору.

Zero 2W має 512 МБ оперативної пам’яті, що є мінімальним для запуску повноцінної системи на базі Linux, але цілком достатнім для легких задач, які не потребують обробки великих обсягів відео або одночасного виконання кількох моделей комп’ютерного зору.

Raspberry Pi Zero 2W має такі інтерфейси:

- роз’єм CSI для підключення камер,
- mini-HDMI,
- micro-USB OTG для підключення USB-периферії,
- GPIO (40 контактів), що підтримує UART, I2C, SPI тощо,
- TV-OUT (аналоговий відеовихід)

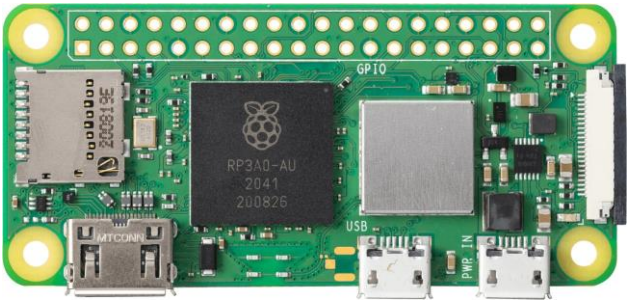


Рис 2.8 — Raspberry Pi Zero 2W

Також Raspberry Pi Zero 2W має вбудований графічний процесор, що дає нам можливість ефективніше працювати з обробкою зображень. Проте на відміну від Raspberry Pi 4 цей одноплатний комп’ютер має набагато менші розміри, а саме 65мм*30мм, та споживання на рівні 1-2 Вт.

Емпіричним методом було встановлено, що для задачі трекінгу цілі на основі алгоритма MOSSE продуктивності Raspberry Pi Zero 2W цілком достатньо. Також Zero 2W згідно наведених вище характеристик має всі необхідні інтерфейси для реалізації системи донаведення. Великою перевагою використання Raspberry Pi Zero 2W є її розміри, що дасть нам можливість розмістити її на рамі майже будь-якого FPV-дрона, а також низьке

споживання, що забезпечить нам більший радіус дії нашого БПЛА. А отже, використання саме Raspberry Pi Zero 2W є найдоцільнішим в якості одноплатного комп'ютера для реалізації системи донавдення на ціль.

Ще одним ключовим елементом системи донавдення є камера, яка повинна відповідати ряду вимог для ефективної роботи системи.

По-перше, камера повинна бути цифровою, а не аналоговою. В минулих розділах було вказано, що в FPV-дронах використовуються аналогові камери, проте Raspberry Pi підтримує роботу тільки з цифровими камерами.

По-друге, камера має під'єднуватися до одноплатного комп'ютера за допомогою інтерфейсу USB. Використання USB замість більш швидкого та ефективного інтерфейсу CSI пов'язано з програмними обмеженнями. Камери, які під'єднані до Raspberry Pi через інтерфейс CSI працюють на основі бібліотеки libcamera, яка натомість не підтримується переважною більшістю бібліотек комп'ютерного зору. На противагу цьому, USB камери працюють на основі програмного інтерфейсу v4l2, який підтримується майже всіма бібліотеками комп'ютерного зору.

По-третє, камера має мати роздільну здатність не менше 1280*720p, що є подібним за якістю картинки на аналогову камеру з кількістю горизонтальних ліній 1200 TVL. Також на такій роздільній здатності камера має мати можливість захоплювати картинку з частотою не менше 30 кадрів на секунду

Вчетверте, важливо підібрати камеру, яка буде мати схожу величину кута огляду з аналоговими камерами, які встановлюються на FPV-дрони. Це пов'язано з тим, що пілоти звикли керувати БПЛА саме з такою величиною кута огляду і зміна цієї величини може призвести до зменшення точності керування пілотом. Величина цього кута в залежності від виробника камери складає від 100 до 120 градусів.

Отже, на основі перелічених вище вимог для реалізації системи донаведення була обрана USB камера на основі матриці OV9732 з роздільною здатністю 1280*720p та кутом огляду (FOV) 100 градусів.

Рис 2.9 — USB камера з матрицею OV9732

2.4 Протоколи зв'язку апаратних засобів

Як було зазначено вище основною частиною апаратного забезпечення системи донавдення стане одноплатний комп’ютер. Тепер постає задача налагодити комунікацію між одноплатним комп’ютером та польотним контролером дрону, для цього необхідно використати спеціальний програмний протокол. Найпоширенішим протоколом для вирішення цієї задачі є MAVLink.

MAVLink (Micro Air Vehicle Link) — це двійковий протокол обміну повідомленнями між безпілотником і зовнішніми системами, такими як наземні станції або одноплатні комп’ютери на борту. Використовуючи MAVLink, одноплатний комп’ютер може як зчитувати інформацію від польотного контролера, так і передавати йому команди. Наприклад, комп’ютер може отримувати дані про положення дрона в просторі , телеметрію, або ж контролювати сам політ — наприклад, наказати дрону перейти у певний режим, взлетіти, летіти в задану точку чи змінити швидкість. Також одноплатний комп’ютер може імітувати RC-сигнали —

тобто формувати команди керування так, ніби вони йдуть від звичайного пульта.

На технічному рівні MAVLink є дуже компактним: кожне повідомлення має чітку структуру з заголовком, полем довжини, ID системи, типом повідомлення, полем даних і контрольним сумуванням. Усе це дозволяє протоколу бути ефективним навіть на повільних лініях зв’язку типу UART. MAVLink може працювати через UART, USB, а також по мережі — через TCP або UDP. Наприклад, якщо дрон має Wi-Fi або Ethernet, можна спілкуватися з польотним контролером по UDP навіть без прямого дротового з'єднання.

MAVLink є дуже ефективним і функціональним протоколом, проте він не підтримується повною мірою програмним забезпеченням Betaflight, яке було обране, як основне, для польотного контролера в нашій системі. Тому доцільним буде використати старіший протокол MSP, який є менш функціональним, але при цьому швидшим ніж MAVLink, а також цей протокол повною мірою підтримується програмним забезпеченням Betaflight.

MSP (MultiWii Serial Protocol) — це бінарний протокол, створений для передачі даних між польотним контролером і зовнішніми пристроями через послідовне з'єднання (UART). Його розробили для проєкту MultiWii, але протокол виявився настільки гнучким і простим, що став стандартом де-факто у Betaflight/iNav-екосистемі. Його основна мета — забезпечити низькорівневий доступ до польотного контролера. MSP дозволяє зчитувати телеметрію, змінювати налаштування дрона в повітрі, подавати RC-команди на польотний контролер дрону. Для взаємодії з MSP можна використовувати UART-порт на польотному контролері, і підключати до нього Raspberry Pi за допомогою інтерфейсу GPIO на вже згаданому одноплатному комп’ютері.

Основна особливість роботи протоколу MSP, те що він ніколи не надсилає щось самостійно. У кожному випадку потрібно зробити запит, щоб

отримати або встановити дані. Кожне отримане повідомлення підтверджується, навіть якщо в ньому немає даних.

Повідомлення для запиту та повідомлення для встановлення даних мають однакову структуру, а саме:

$M[direction][data length][code][data][checksum]

- 0 - 7 біт — преамбола. ASCII символ «$»
- 8 - 15 біт — преамбола. ASCII символ «M»
- 16 - 23 біт — [direction]. ASCII символ «<» у випадку, якщо ми надсилаємо запит. Повідомлення з відповіддю буде мати ASCII символ «>» в цьому полі. У разі, якщо повідомлення є помилковим чи невідомим для системи поле [direction] буде мати ASCII символ «|».
- 24 - 31 біт — [data length]. У випадку, якщо ми хочемо встановити якісь значення і передати певні дані, то data length дорівнює розміру в байтах поля [data]. У разі надсилання запиту на отримання певних даних [data length] = 0.
- 32 - 39 біт — [code]. Код команди. Деякі коди команд будуть зазначені нижче в цьому підрозділі.
- 40 - n біт — [data]. Кількість байтів цього поля не становить 1 октет, як у випадку інших полів команди MSP. Розмір поля даних залежить від того яка команда буде надіслана і які дані необхідно передати разом з нею.
- (n + 1) – (n + 8) біт — [checksum]. Контрольна сума формується за допомогою операції XOR для полів [data length], [code] та значення кожного байта поля [data].

Нижче перелічені команди та коди до них, які стануть в нагоді при розробці системи донавдення:

- MSP_RC — код 105. Отримати поточні значення каналів керування з польотного контролера дрону.
- MSP_SET_RAW_RC — код 200. Встановити значення каналів керування на польотному контролері дрону.

Отже, на основі матеріалу розглянутому в цьому розділі, був обраний протокол для взаємодії між одноплатним комп'ютером та польотним контролером дрону. Також було детально розглянуто структуру цього протоколу. Отримані знання дадуть нам можливість побудувати ефективну взаємодію між апаратним забезпеченням.

2.5 Алгоритми керування FPV дроном

Для розробки ефективного алгоритму керування для спрямування дрона на ціль засобами комп'ютерної системи варто розібратися, як відбувається керування дроном, тобто яким чином польотний контролер дрону перетворює команди керування на команди для двигунів, тим самим спрямовуючи дрон в бажаному напрямку.

З певною частотою кожну секунду пульт засобами радіозв'язку передає поточне положення стіків та тумблерів (каналів) на приймач керування на дроні. Частота передачі складає в середньому від 25Hz до 200Hz. Зібрані значення каналів оцифровуються — кожен канал зазвичай має значення в діапазоні 1000–2000 мкс (мікросекунд), яке відповідає ширині імпульсу в протоколах типу PWM. У сучасних цифрових протоколах, як SBUS, CRSF (Crossfire), ExpressLRS, DSM, дані інкапсулюються в пакет, який включає номер фрейму, значення каналів, контрольну суму для перевірки цілісності, та іноді службову інформацію — телеметрію, статус зв'язку тощо. Приймач

на дроні приймає сигнал, розпаковує пакет і передає значення каналів на польотний контролер, як правило засобами протоколу UART.

Після цього отримані команди керування, які є першими чотирма значеннями каналів в пакеті керування, оброблюються за допомогою алгоритму Rate Profile. Цей алгоритм встановлює взаємозалежності між значеннями команд керування (1000–2000 мкс) та кутовими швидкостями в градусах на секунду дрона по трьом осям. Ця взаємозалежність, як правило подається у вигляді графіку і її можна гнучко налаштовувати в програмному забезпечені дрона.

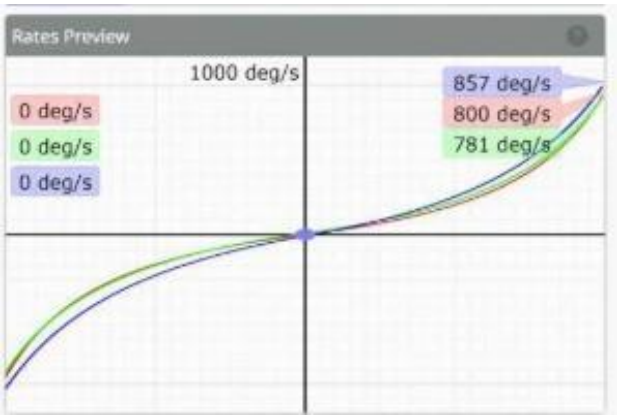


Рис 2.10 — Графічне подання Rate Profile

PID-регулятор — це класичний алгоритм керування зі зворотнім зв’язком, який забезпечує точне та стабільне досягнення бажаного значення системою[9]. Виходячи з назви PID регулятор має 3 складові, а саме: P пропорційна, I інтегральна та D диференційна. PID-регулятор в FPV дроні застосовується для стабілізації руху — він постійно намагається зменшити різницю між бажаною кутовою швидкістю обертання (значенням отриманим з пульта та пропущеним через алгоритм Rate Profile) і тією, яка є на даний момент. Його завдання обчислити керуючі команди на мотори таким чином,

щоб ця різниця (помилка) була мінімальною, і дрон точно слідував заданій траєкторії або команді.

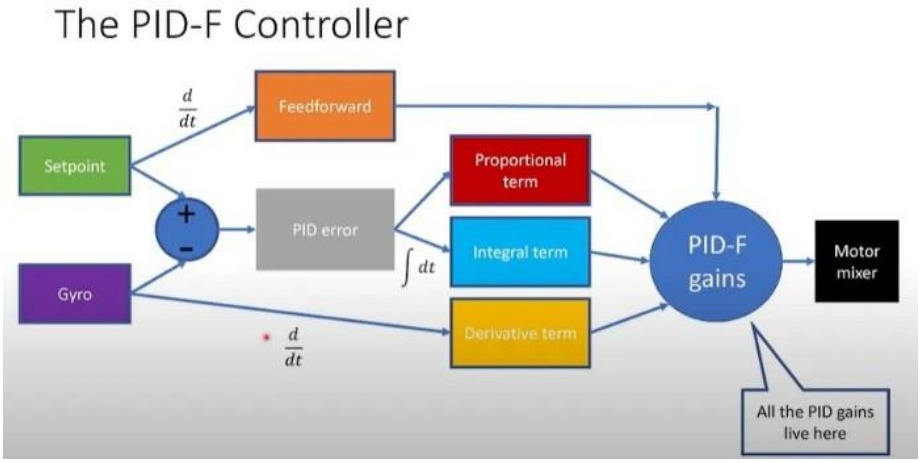


Рис 2.11 — Графічне представлення роботи PID-регулятора на дроні

Давайте розберемо вхідні дані в PID регулятор:

- Gyro - це вхідний сигнал, що надходить з гіроскопу та показує поточні кутові швидкості (deg / s) по 3 осям (Roll, Pitch, Yaw). Перед тим, як gyro сигнал надходить на вхід PID регулятора, він проходить процес фільтрації.
- Setpoint - це вхідний сигнал, який показує кутові швидкості, які ми хочемо отримати.

Після отримання вхідних даних PID-регулятор за допомогою трьох складових (proportional, intagral, derivative), в яких закладені певні математичні формули, обраховує значення для кожної із трьох осей дрона. Кожна складова регулятора має свій коефіцієнт ваги, на який вона множиться після математичних операцій. Згодом всі складові сумуються і відправляються на Motor Mixer, який подає команди на мотори в залежності від отриманої суми PID регулятора. Однією із важливих умов коректної роботи PID-регулятора є його налаштування, що полягає у визначенні

оптимальних коефіцієнтів для трьох складових, а також оптимального співвідношення між ними. Тобто вихідний сигнал обчислюється по такій формулі:

$$out = P*kP + I*kI + D*kD$$

Для підбору оптимальних коефіцієнтів PID-регулятора варто розібратися, яким чином кожна із тьох складових впливає на його роботу.

Proportional term (P term) - складова що є різницею між setpoint та gyro, або інакше PID Error. Тобто в основі пропорційної складової лежить проста математична операція віднімання setpoint від gyro. Звідси, чим більша різниця між бажаним та поточним положенням дрону по одній із трьох осей, то більше значення буде мати пропорційна складова по цій осі. P term можна порівняти з пружиною у автомобілі: чим більше стискаємо, тим більше віддача. Чим більше коефіцієнт P, тим скоріше система виходить на встановлене (бажане) значення, але при великому P коефіцієнті ця складова може породжувати осциляції, бо реагує на помилку в поточний момент часу. Якщо говорити на прикладі пружини, то для цієї складової потрібен амортизатор, який буде не давати P складовій створювати осциляції. На графіку нижче зображено, як веде себе система при великому P коефіцієнті (поганий варіант).

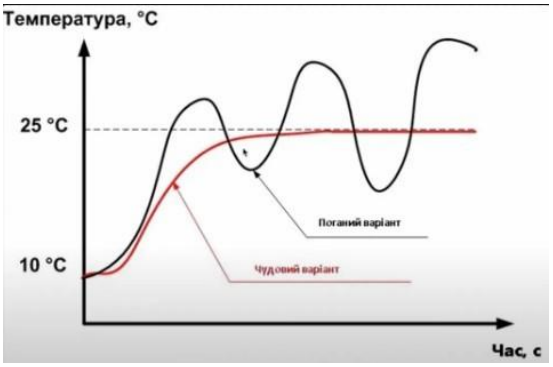


Рис. 2.12 — Осциляції сигналу викликані великим коефіцієнтом P складової

Derivative term (D term) - це складова, що є зворотною гальмуючою силою для P складової, таким собі амортизатором для пружини. Математично D term розраховується, як похідна від зміни сигналу Gyro. Звідси, чим швидше відбувається зміна значення gyro, тим більша ця складова, або чим швидше змінюється gyro, тим більше D складова гальмує зміну gyro. Чим більше коефіцієнт D, тим сильніше він гальмує вихід системи на встановлене (бажане) значення, також при високому D коефіцієнті система більше реагує на шуми з gyro, підсилюючи їх, це пов’язано з тим, що шуми схожі на часту та швидку зміну кутової швидкості, а отже D складова намагається їх виправити, що призводить до надмірно частих команд на двигуни. Чим більше коефіцієнт D, тим сильніше він гальмує вихід системи на встановлене (бажане) значення, також при високому D коефіцієнті система більше реагує на шуми з gyro, підсилюючи їх, це пов’язано з тим, що шуми схожі на часту та швидку зміну кутової швидкості, а отже D складова намагається їх виправити, що призводить до надмірно частих команд на двигуни.

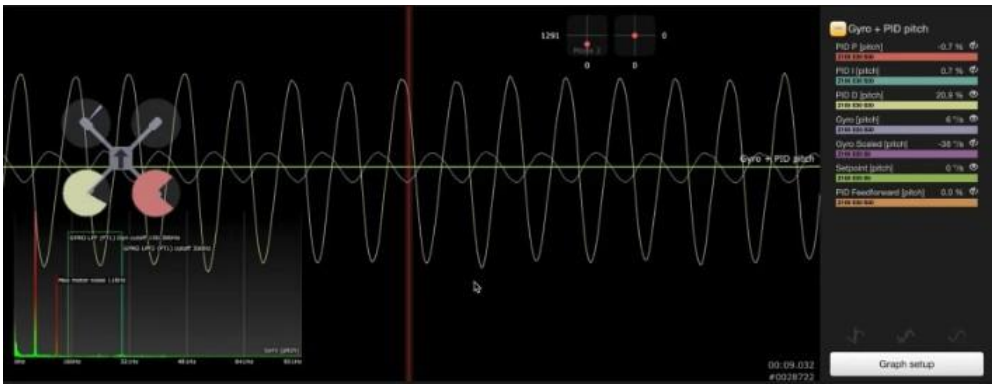


Рис 2.13 - Поведінка дрона при великому значенні коефіцієнта D складової

Отже, P складова є рушійною силою, а D складова є гальмуючою силою, звідси для ідеальної роботи PID регулятора нам слід знайти баланс між коефіцієнтами P та D складової таким чином, щоб графік максимально

швидко доходив до setpoint, але при цьому не перелітав це значення і не з’являлися осциляції.

Integral term (I term) - це складова, що призначена для виправлення помилки, що діє протягом значного періоду часу, наприклад: вітер, неправильно підвішений боєприпас, неправильно встановлена батарея, яка порушує центр ваги тощо. Математично I term розраховується, як інтеграл від PID Error за певний період часу. Тобто чим більше триває помилка, тим більше I складова намагається її виправити.

Для визначення команд керування на основі положення цілі доцільно використати описаний вище PID-регулятор. Вважається, даний типу регулятора є одним з найпростіших і найефективніших способів, як змусити FPV-дрон слідувати за виявленою ціллю на зображенні. Як було описано вище в цьому підрозділі, до PID-регулятора, що розміщений на польотному контролері, на вхід надходить два значення: setpoint від пульта керування та поточна кутова швидкість дрона по кожній із 3 осей. У випадку реалізації PID регулятора в системі донавдення ідея полягає у тому, що якщо ціль зміщена від центру кадру, то ми обраховуємо помилку (error) і за допомогою PID-регуляторів генеруємо відповідні команди керування, щоб дрон «виправився» і навівся на ціль. Тобто на вхід до регулятора ми подаємо два значення: координати центру кадру та нормалізовані координати цілі на кадрі. Також емпіричним методом згідно описаних закономірностей варто підібрати коефіцієнти для кожної складової PID-регулятора для точного спрямування дрона в ціль.

Отже, в цьому підрозділі був розібраний повний алгоритм керування FPV-дроном. На основі цього було сформоване бачення, яким чином варто реалізувати алгоритм визначення команд керування для донавдення FPV-дрона на ціль.

2.6 Вибір мови програмування та необхідних бібліотек

В якості мови програмування для реалізації системи було обрана Java, через низку переваг, а саме:

- По-перше, ця мова є платформонезалежною, програмне забезпечення написане на Java легко переноситься на різні пристрої, включаючи Raspberry Pi.
- По-друге, за даними багатьох тестувань Java є доволі швидкою мовою програмування, порівняно з основним конкурентом для таких задач, а саме Python.
- По-третє, Java має бібліотеку для роботи з Raspberry PI під назвою Pi4J. Також Java має реалізацію бібліотеки OpenCV.

Для створення системи на основі Java, як було описано вище, буде використовуватися дві основні бібліотеки: OpenCV та Pi4J.

OpenCV (Open Source Computer Vision Library) — це потужна та широко використовувана бібліотека з відкритим кодом, призначена для комп'ютерного зору, обробки зображень та машинного навчання. Вона забезпечує багатий набір інструментів для задач виявлення об’єктів, сегментації, розпізнавання облич, трекінгу, фільтрації, трансформацій зображень, роботи з відео та камерою, і багато іншого. У випадку реалізації системи на Raspberry Pi, OpenCV дозволяє підключати камеру, зчитувати відеопотік і в реальному часі обробляти кадри. Також ця бібліотека має в собі реалізацію трекінгу MOSSE, який був обраний в якості алгоритму трекінгу для реалізації системи.

Pi4J, це Java-бібліотека, створена спеціально для роботи з апаратними інтерфейсами Raspberry Pi. Вона дозволяє Java-програмам взаємодіяти з GPIO (входи/виходи загального призначення), UART, I2C, SPI та іншими низькорівневими портами Raspberry. Звідси, за допомогою цієї бібліотеки ми можемо встановити з’єднання між одноплатним комп’ютером та польотним

контролером дрону за допомогою UART та протоколу MSP для передачі команд керування на FC, а у зворотній бік ми можемо отримувати дані телеметрії від дрону для внесення коректівів в подальші команди керування.

Отже, в цьому підрозділі були розглянуті основні програмні засоби, які будуть використані для створення комп'ютерної системи донавдення на ціль.

3. ОПИС РЕАЛІЗАЦІЇ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1. Загальна структура системи

В минулому розділі були розглянуті теоретичні положення та обрані необхідні апаратні та програмні компоненти для розробки системи. Проте однією із найважливіших та найскладніших задач є об’єднати отримані теоретичні знання в працюючи систему та поєднати її всі компоненти, забезпечивши надійну та чітку роботу донаведення у реальному часі. Нижче на рисунку 3.1 наведена загальну структура розробленої системи, що відповідає описаним вище вимогам.

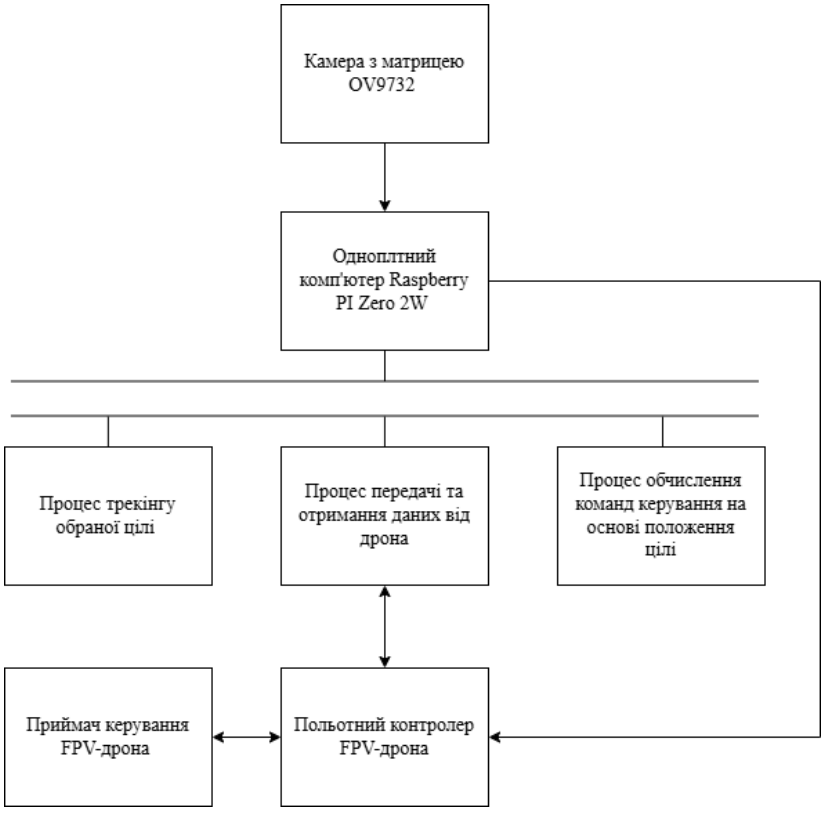


Рис 3.1 — Структурна схема роботи системи

Для отримання зображення використовується цифрова камера, яка під’єднана до одноплатного комп’ютера через USB порт і через кабель

передає відео на одноплатний комп’ютер Raspberry Pi Zero 2W. Як було описано в минулому розділі, використання USB камери пов’язано з деякими особливостями підтримки драйверів одноплатним комп’ютером та бібліотекою OpenCV, яка є основною для роботи системи трекінгу та передачі зображення на польотний контролер. Як правило, до одноплатних комп’ютерів камери під’єднуються за допомогою роз’єма CSI, який забезпечує вищу пропускну здатність, а відповідно якіснішу картинку з меншою затримкою. Проте CSI камери в Raspberry Pi функціонують на основі новітньої бібліотеки libcamera, який натомість не підтримується бібліотекою OpenCV, яка не здатна зчитати зображення з цього роз’єму для подальшого процесу трекінгу. USB камери в Raspberry Pi працюють на основі драйвера v42l, який підтримується OpenCV, проте є повільнішим за свого конкурента libcamera.

Далі на черзі, як зображено на рисунку 3.1, йде одноплатний комп’ютер, на якому запущені три паралельних процеси, а саме: трекінг, передача та отримання команд від польотного контролера, обчислення команд керування. Першочергово, планувалося, що ці процеси будуть виконуватися послідовно, проте при реалізації системи було встановлено, що процес передачі та отримання команд від польотного контролера гальмує послідовне виконання програми, через затримку та очікування зворотних команд від дрона. Для оптимізації системи був запроваджений показник dt, який визначає часові проміжки між кожною генерацію і відповідно надсиланням команд на контролер дрону. Цей показник є числовим патерном, за яким можна визначати на скільки система загалом швидко працює. Також при послідовному виконанні програми виникали постійні проблеми з трекінгом, які характеризувалися постійною втратою об’єкта алгоритмом трекінгу, через те що був великий часовий проміжок між пошуком об’єкта на зображені отриманого з камери дрона.

```
dt = 5.7958397
[COMMANDS] Yaw: 1497 Pitch: 1494
OK
dt = 3.9283541
[COMMANDS] Yaw: 1499 Pitch: 1499
```

Рис 3.2 — Значення показника dt при послідовному виконанні програми

Як можна побачити на рисунку 3.2, при послідовному виконанні програми часові проміжки між кожною ітерацією генерації керуючих команд складають від 4 до 6 секунд, що є не припустимою затримкою для стабільного керування FPV-дроном.

В результаті було прийняте рішення виконувати процеси трекінгу, надсилання команд та генерації керуючих команд паралельно, щоб зазначені вище процеси не гальмували один одного. Відповідно ці зміни призвели до досягнення мінімальної затримки.

```
dt = 0.0308922
[COMMANDS] Yaw: 1534 Pitch: 1669
1000
dt = 0.0309624
[COMMANDS] Yaw: 1558 Pitch: 1664
1000
dt = 0.030419
[COMMANDS] Yaw: 1503 Pitch: 1672
1000
```

Рис 3.3 — Значення показника dt при паралельному виконанні програми

Як можна побачити на рисунку 3.3 після запуску програми в паралельному режимі проміжки між кожною ітерацією генерації керуючих команд складають приблизно 0.03 секунди, що є покращенням за часовою

характеристикою порівняно з послідовним виконанням більше ніж в 100 раз. Такі часові показники дозволять досягти потрібної частоти команд для чіткого спрямування дрона до обраної цілі. Також при паралельному виконані програми процес трекінгу став працювати набагато стабільніше, втрати об’єкту під час трекінгу звелися до мінімуму.

В свою чергу, окрім з’єднання одноплатного комп’ютера з польотним контролером для передачі команд керування засобами протоколу UART, Raspberry Pi передає на польотний контролер картинку з певним інтерфейсом, який був накладений алгоритмом трекінгу. Як було зазначено в другому розділі, польотний контролер FPV-дрона підтримує лише на вхід аналогове відео, а отже з Raspberry Pi необхідно передавати саме аналоговий відеосигнал.

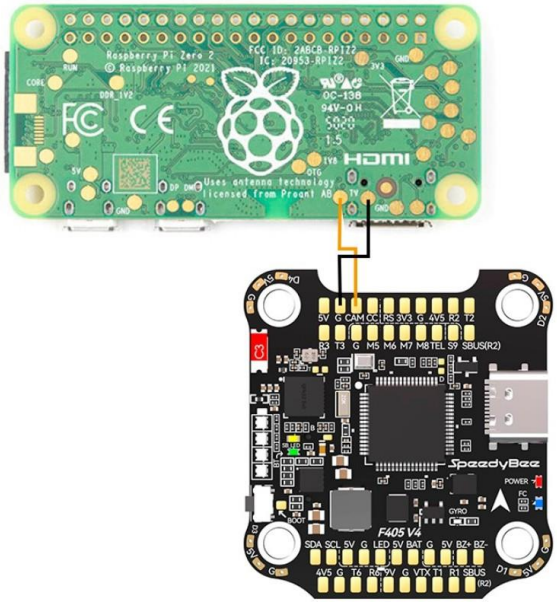


Рис 3.4 — З’єднання між Raspberry та FC з метою передачі відеосигналу

Як можна побачити на рисунку 3.4, Raspberry Pi Zero 2W має контакт TV, що є композитним відеовиходом, який можна використати для поставленої задачі. В програмному забезпечені одноплатного комп’ютера

було налаштовано цей відеовихід для передачі зображення у форматі PAL, що є популярним типом аналогового відеосигналу і що підтримується багатьма польотними контролерами. Згодом на цей відеосигнал польотний контролер накладає OSD та передає на відеопередавач, який транслює картинку з композитного роз'єму на окуляри чи екран пілота.

Ще одним компонентом в схемі наведеній на рисунку 3.1 є приймач керування. Як було згадано вище, зв'язок між польотним контролером та одноплатним комп'ютером є двостороннім, а отже ми можемо відправляти на польотний контролер команди керування, натомість з польотного контролера система може отримувати положення тумблерів та команди пілота з пульта керування. Як було зазначено в другому розділі, саме приймач керування отримує команди з пульта, а отже за допомогою нього ми можемо отримати сигнал про активацію системи донавдення з пульта пілота, згодом ця команда піде на польотний контролер дрона, де згодом потрапить на одноплатний комп'ютер та програмне забезпечення системи донавдення. Отримавши команду на активацію, одноплатний комп'ютер активує трекінг цілі.

Отже, в цьому підрозділі була розглянута загальна структура системи донавдення. Цей підрозділ дав нам уявлення про взаємодію різних програмних та апаратних компонентів системи між собою. Натомість в наступних підрозділах буде розглянуто детальніше реалізацію кожного із наведених компонентів системи.

3.2 Реалізація модуля захоплення та супроводження цілі

В минулому підрозділі була розглянута загальна структура системи і одним із ключових модулем є захоплення та супроводження цілі, детальна реалізація якого буде описана в цьому підрозділі. Під час розгляду різноманітних алгоритмів трекінгу об'єкту, було прийнято рішення взяти за основу алгоритм трекінгу MOSSE через його високу швидкість виконання,

що дозволяє коректно функціонувати системі в режимі реального часу. Реалізація даного алгоритму вже міститься в додатковому пакеті для бібліотеки OpenCV. Проте при практичній роботі з даною реалізацію алгоритму не вдалося отримати бажаної точності розпізнавання об’єкту, особливо гостро проблема проявлялася при наближені об’єкта до камери. Тому було прийнято рішення внести деякі покращення в цей алгоритм для успішної ітеграції його в загальний алгоритм роботи з відео в програмному забезпеченні системи та пристосувати його до використання в системі донаведення дрона на військові цілі.

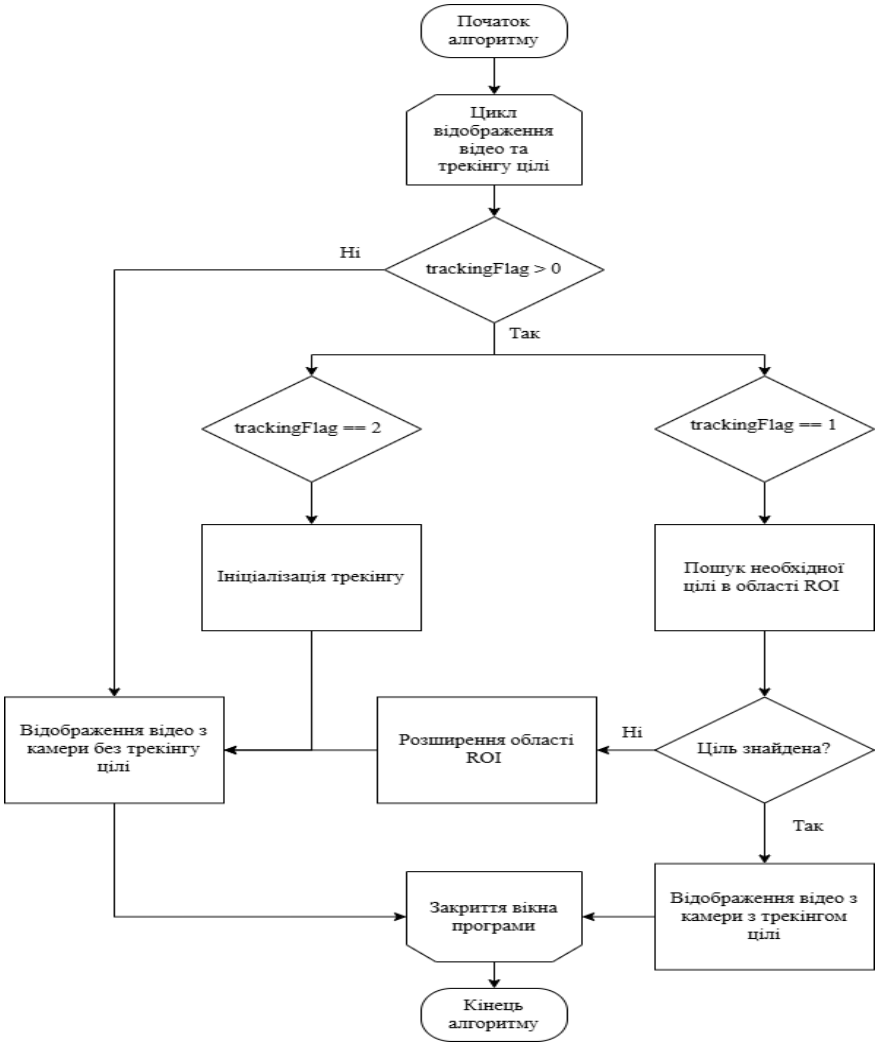


Рис 3.5 — Алгоритм роботи механізму захоплення та супроводження

На рисунку 3.5 можна помітити, що алгоритм захоплення та супроводження починається з циклу, умова закінчення цього циклу є закриття вікна програми. Проте у реальному застосуванні цієї системи на дроні, цей цикл стає умовно нескінченним, так як він закінчує роботу під час вимикання батареї дрона.

В кожну ітерацію циклу надходить двохвимірна матриця, що є представленням кадру отриманого з курсової цифрової камери дрону. Після цього йде перевірка значення змінної tracking flag. Ця змінна є керуючою в середині циклу, а саме дає можливість активувати трекінг, шукати об’єкт на кадрі чи «запам’ятовувати» його. Розглянемо детальніше, на що впливає кожне можливе значення змінної tracking flag.

- tracking flag = 0. При цьому значенні змінної трекінг об’єкта не активований
- tracking flag = 2. Це значення змінної встановлюється після того, як пілот активує донавдення тумблером на пульті або таке значення може бути встановлене в іншій частині програми. При цьому значенні іде ініціалізація, або «запам’ятовування» об’єкта який міститься в прицілі, який розташований на зображенні в окулярах пілота. Сам процес ініціалізації являє собою перетворення об’єкта в градації сірого, після цього робляться різні спотворення об’єкта і в кінці генерується фільтр, на основі якого буде проходити пошук об’єкта на подальших кадрах.
- tracking flag = 1. При цьому значенні трекінг на основі згенерованого фільтра шукає та виділяє потрібний об’єкт на зображенні.

Для оптимізації трекінгу алгоритм шукає потрібний об’єкт не по всьому зображенню, а лише в зоні ROI та поряд з нею. Приціл в інтерфейсі програми і є зоною ROI.

Згідно рисунка 3.5 перша перевірка tracking flag є на те чи більше ця змінна ніж нуль. Це зроблено для того, щоб у випадку неактивованого трекінгу час не витрачався на додаткові перевірки. У випадку якщо tracking flag менше або дорівнює нулю, то алгоритм просто відображає кадр з певним інтерфейсом. У випадку, якщо tracking flag = 2, то відбувається ініціалізація трекінгу, а згодом відображається кадр з інтерфейсом. Якщо tracking flag = 1, то відбувається пошук об’єкта, який був проініціалізований, якщо об’єкт знайдений то відображається кадр з інтерфейсом для користувача, а також рамка, яка обводить об’єкт. Якщо об’єкт не знайдений то відбувається розширення області ROI.

В стандартному алгоритмі MOSSE не відбувається розширення області ROI при втраті об’єкта. Проте алгоритм трекінгу в системі донавдення повинен передбачати те, що дрон буде наближатися до цілі аж до її ураження, відповідно сама ціль по мірі наближення буде мати все більший і більший розмір. Натомість стандартний алгоритм MOSSE працює гарно лише зі незначними змінами масштабу об’єкта, що контролюється. Тому для застосування даного алгоритму в системі донавдення був додана додаткова ланка алгоритма. При втраті об’єкту відбувається розширення зони пошуку в 1.5 рази та для змінної tracking flag встановлюється значення 2, що призводить до повторної ініціалізації об’єкту.

Рис. 3.6 — Тестування недопрацьованого алгоритму трекінгу

На рисунку 3.6 можна побачити, що стандартний алгоритм трекінгу втратив об’єкт при наближенні дрона до нього. У такому випадку система донавдення спрацює некоректно, що призведе до невиконання бойового завдання дроном.

Рис. 3.7 — Тестування допрацьованого алгоритму трекінгу

На рисунку 3.7 можна помітити, що об’єкт знаходиться навіть ближче, ніж на рисунку 3.6, але при цьому трекінг зберігається. Це пов’язано з тим що на певному етапі наближення БПЛА до цілі, вона була втрачена алгоритмом трекінгу, проте область пошуку ROI була розширена і трекінг був проініціалзований ще раз. Завдяки цьому алгоритм зміг знову знайти необхідний об’єкт і продовжити його супроводження.

Отже в цьому підрозділі була детальна розглянута реалізація алгоритму захоплення та супроводження цілі. За основу був взятий існуючий розповсюджений алгоритм трекінгу MOSSE, проте він був допрацьований під задачу донавдення і успішно інтегрований в загальну систему.

3.3 Реалізація модуля керування БПЛА на основі захопленої цілі

Ключовим модулем для реалізації системи донавдення FPV-дрона є модуль керування БПЛА, який складається:

- З набору алгоритмів для обчислення команд керування на основі положення цілі, яке було отримане з алгоритму трекінгу.
- З алгоритму для передачі команд та отримання телеметрії з польотного контролера FPV-дрона.

Саме ці компоненти модуля керування забезпечують політ і ураження дроном захопленої цілі. Від правильності реалізації цих підмодулів залежить швидкість реакції та точність влучання дрона у ворожий об’єкт.

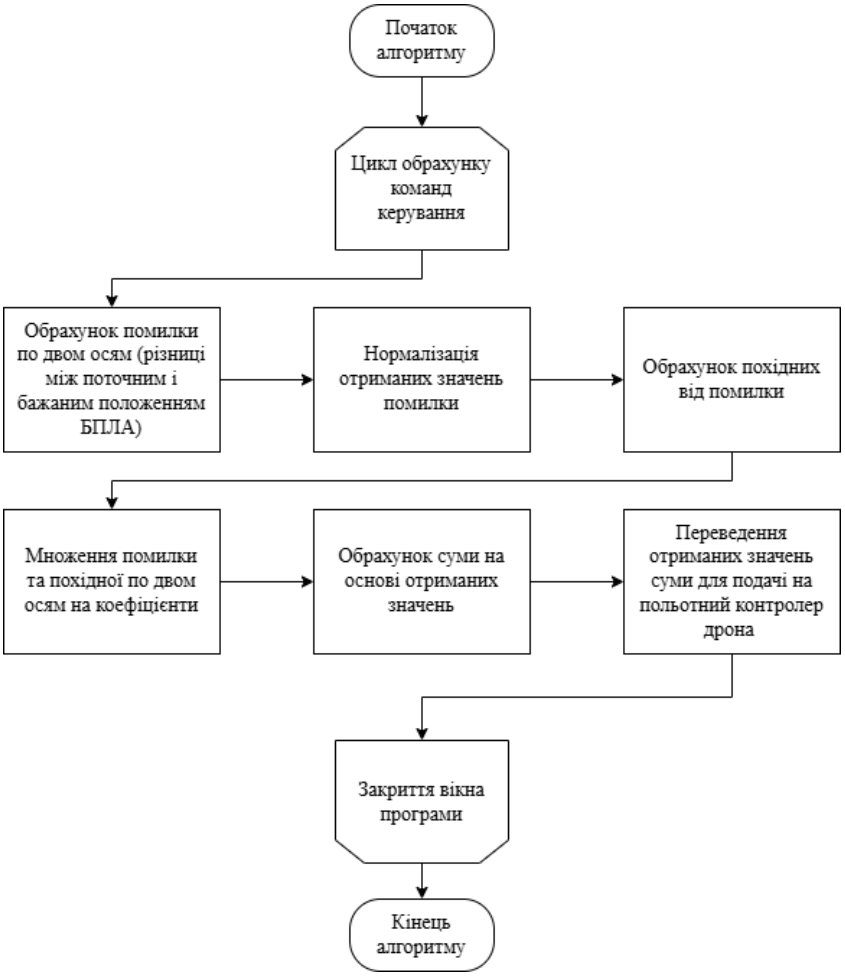


Рис. 3.8 — Алгоритм обчислення команд керування

На рисунку 3.8 можна побачити, що алгоритм обчислення команд керування починається з входу у цикл, умова закінчення цього циклу є закриття вікна програми. Проте у реальному застосуванні цієї системи на

дроні, цей цикл стає умовно нескінченним, так як він закінчує роботу під час вимикання батареї дрона, тобто умова виходу з головного циклу цього алгоритму є ідентичною до умови в циклі алгоритму трекінгу.

В другому розділі дипломної проекту був наведений опис, яким чином польотний контролер FPV-дрона визначає команди керування для моторів БПЛА на основі поточного положення літального апарату та бажаного. Як було зазначено вище, фундаментальним для цього процесу є PID регулятор. В основі розробленого алгоритму обчислення команд керування в рамках системи донаведення теж використовується PID регулятор, проте для реалізації бажаної поведінки необхідно лише дві складові: P та D складова. Проаналізувавши польоти бойових пілотів FPV-дронів можна помітити ключову особливість, що під час заходу на ціль, оператор БПЛА завжди намагається утримати її у центрі екрану, тому що саме цей підхід забезпечує точне попадання у ціль. Ця особливість стала однієї із фундаментальних в розробленому алгоритмі обчислення команд керування, тому що задача PID регулятора в цьому алгоритмі розраховувати команди таким чином, щоб дрон при наближенні до зафіксованої пілотом цілі, завжди тримав її у центрі екрану. Ціль на екрані має певні координати, які виражені через x та y, звідси для досягнення бажаного результату нам достатньо контролювати такі складові:

- вісь рискання та крену, яка залежить від положення цілі по вісі x
- вісь тангажу яка залежить від положення цілі по вісі y
- рівень тяги

Першою дією в головному циклі алгоритму є обрахунок різниці по двом осям між поточним положенням цілі на екрані пілота та бажаним. Поточне положення цілі ми отримуємо з алгоритму трекінгу, який надає координати положення об’єкта на екрані по осям x та y. Бажаним положенням

цілі є центр екрану пілота. Отримані різниці є за своєю суттю є P складовими для розрахунку необхідних команд керування по вісі рискання та тангажу.

Після отримання значень різниці по двом осям, ці значення нормалізуються для подальшої легшої обробки. Наступним етапом є розрахунок значень для складової D, що є за своєю суттю похідною і розраховується на за нижче наведеною формулою.

$$derivative = (error - previousError) / dt,$$

де *error* – різниця між поточним та бажаним положенням об’єкту на екрані;

previousError — значення змінної error в попередній ітерації циклу;

dt – час в секундах між ітераціями циклу розрахунку команд керування.

Наступним кроком в алгоритмі є розрахунок значення pidSum по двом осям на основі отриманих значень P та D. Обчислення значень відбувається по нижче наведеній формулі.

$$pidSum = kp * error + kd * derivative,$$

де *kp* – коефіцієнт P складової, який визначає «силу» цієї складової у вихідному значенні PID регулятора, дана складова визначається емпіричним методом;

error – різниця між поточним та бажаним положенням об’єкту на екрані;

kd – коефіцієнт D складової, який визначає «силу» цієї складової у вихідному значенні PID регулятора, дана складова визначається емпіричним методом;

derivative — значення похідної.

Завершальним кроком алгоритму є перетворення отриманих значень суми PID регулятора в діапазон від 1000 до 2000. Необхідність цієї операції

пов’язана з принципом, за яким одноплатний комп’ютер на програмному рівні здатен подавати команди керування на польотний контролер БПЛА. Основним принципом є те, що одноплатний комп’ютер після активації системи донавдення починає підміняти команди з пульта по трьом каналам, а саме: канал рискання, канал тангажу та канал тяги. Таким чином польотний контролер, продовжує коректно функціонувати, так як вважає, що команди надходять з пульта. Проте для виконання такої операції необхідно привести команди керування з PID регулятора нашої системи до діапазону значень, які при нормальній експлуатації надсилає передавач з пульта на приймач дрона, і цей діапазон складає від 1000 до 2000 умовних одиниць.

Після формування команд керування їх необхідно відправити на польотний контролер дрону, цим процесом в розробленій системі донавдення займається алгоритм побудови та передачі пакетів команд керування. Як було зазначено в другому розділі дипломного проєкту, для цієї задачі застосовується протокол MSP, який має всі необхідні команди для передачі цих команд. Проте цей алгоритм не тільки відправляє команди керування на польотний контролер, а також робить запити для отримання даних з FC, а саме положення деяких тумблерів на пульті пілота. Ці дані дають можливість системі зрозуміти, коли пілот активував на пульті тумблер включення донавдення, а також вони дозволяють реалізувати деякі програмні захисти для безпеки пілота.

Нижче на рисунку 3.9 можна зауважити, що алгоритм починається з циклу, умова закінчення цього циклу є закриття вікна програми. Проте у реальному застосуванні цієї системи на дроні, цей цикл стає умовно нескінченним, так як він закінчує роботу під час вимикання батареї дрона.

Наступною операцією в алгоритмі є перевірка значення змінної trackingFlag, яка, як було описано вище, відповідає за активацію, ініціалізацію та роботу трекінгу.

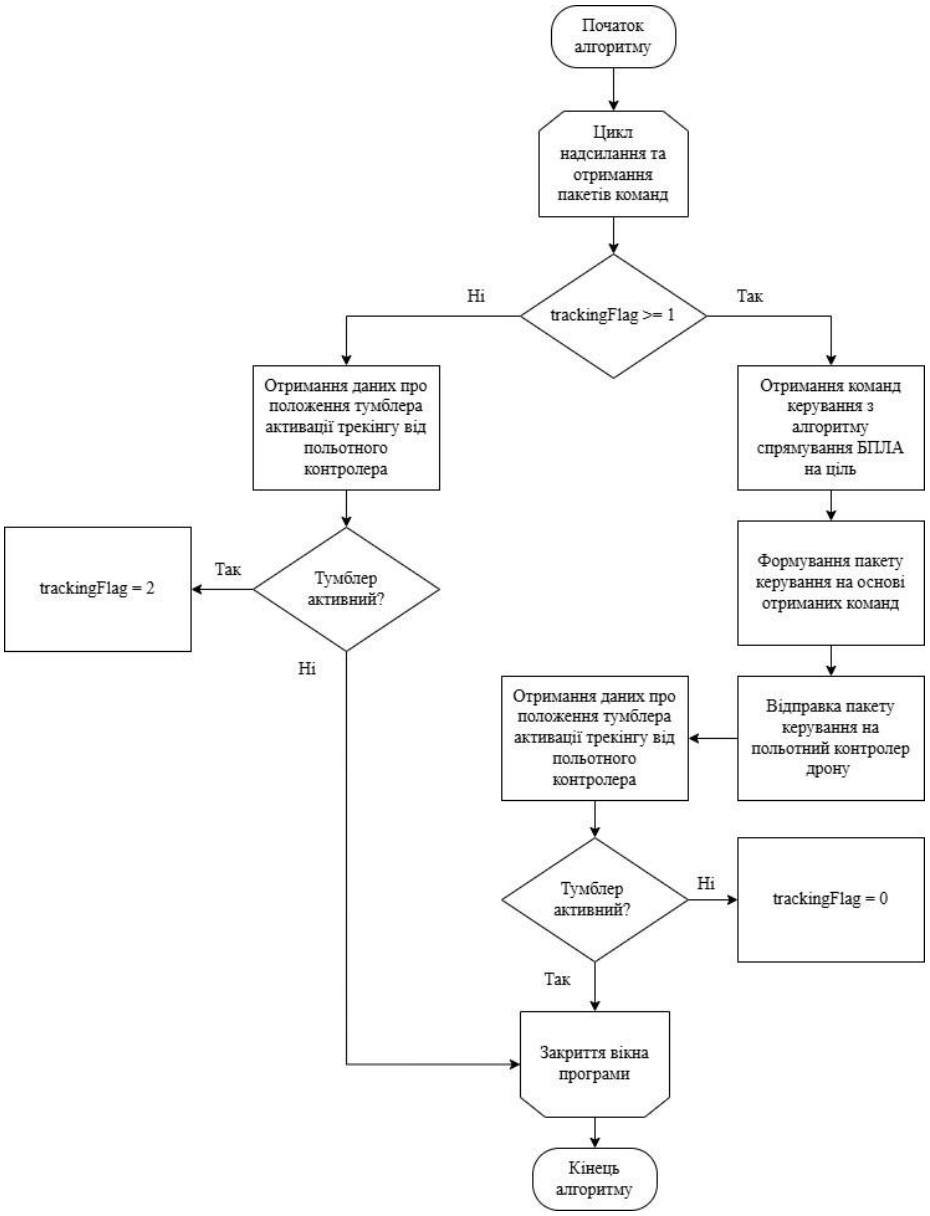


Рис 3.9 — Алгоритм побудови та передачі пакетів команд керування

В першій перевірці перевіряється за допомогою змінної trackingFlag чи активований трекінг, якщо ні, то алгоритм робить запит до польотного контролера за допомогою команди MSP_RC. У такому разі у відповідь програма отримує положення всіх стіків та тумблерів на даний момент на пульті пілота. У випадку, якщо в даний момент на пульті активований тумблер донавдення, то змінна trackingFlag отримує значення 2, що в паралельному процесі призводить до активації алгоритму трекінгу.

У разі, якщо trackingFlag більше або дорівнює одиниці, то це означає, що трекінг вже активований, а отже польотному контролеру необхідно надсилати команди керування. Першою операцією в цій гілці алгоритму є отримання масиву команд керування, який складається зі значення для вісі рискання та вісі тангажу.

Після цього відбувається формування MSP пакету для відправлення команд на польотний контролер дрону. Як було зазначено вище, для того щоб FC БПЛА міг виконати потрібні команди з одноплатного комп'ютера, йому необхідно підмінити дані з пульта керування, на ті, які було розраховані в алгоритмі обчислення команд керування. Для цього в протоколі MSP існує команда MSP_SET_RAW_RC, а отже пакет даних в шістнадцятковому представлені для польотного контролера виглядає таким чином [10]:

24 4D 3C 08 C8 rollValue pitchValue throttleValue yawValue checksum,

де *24,4D* – два байти преамболи $M протоколу MSP;

3C – байт символу «<», який позначає, що ми відправляємо команду на польотний контролер;

08 – байт, що позначає розмір даних, які будуть передані на FC БПЛА, так як одна команда керування займає 2 байти, то розмір становить 8 байт;

C8 – байт, що позначає ID команди в MSP протоколі, MSP_SET_RAW_RC має ID C8;

rollValue – два байти, в яких міститься бажане значення по вісі roll ;

pitchValue – два байти, які містять розраховане значення для вісі pitch на основі алгоритму обчислення команд керування;

throttleValue – два байти, в яких міститься значення бажане значення throttle

yawValue – два байти, які містять розраховане значення для вісі yaw на основі алгоритму обчислення команд керування;

Наступним етапом роботи алгоритму є відправлення сформованого пакету на польотний контролер дрону. Для цього застосовується функціонал бібліотеки Pi4J, яка відправляє пакет команд на FC БПЛА засобами UART з'єднання.

Опісля бібілотека Pi4J відправляє запит на отримання значень каналів пульта пілота, щоб пересвідчитися у тому, що пілот не деактивував донавдення. Проте цей запит відправляється не кожну ітерацію циклу, а лише кожну соту ітерацію. Це пов'язано з тим, що часті запити до польотного контролеру дрона перевантажують його та у відповідь він надсилає хибні дані. У випадку, якщо пілот деактивував донавдення, то змінна trackingFlag отримує значення 0, що призводить до того, що трекінг об'єкту та відправлення команд керування на FC припиняється.

Отже, в цьому підрозділі був детально розглянути важливий модуль системи донавдення, а саме модуль обчислення та відправлення команд керування на польотний контролер дрону.

3.4 Інтеграція системи з апаратним та програмним забезпеченням FPV дрона

Вище були детально розглянуті всі модулі системи донавдення. Проте ця система не здатна виконувати свої задачі без інтеграції її з програмним та апаратним забезпеченням FPV-дрона.

Для початку, варто забезпечити на апаратному рівні зв'язок між польотним контролером та одноплатним комп'ютером, на якому функціонує система донавдення. Зв'язок цих двох компонентів варто забезпечити з трьох причин:

- Одноплатний комп'ютер при активованій системі донавдення повинен відправляти на FC команди керування та отримувати дані про значення каналів пульта пілота

- Одноплатний комп’ютер після обробки зображення з камери повинен надсилати його на польотний контролер дрону, який накладе OSD та відправить це зображення на відеопередавач, таким чином пілот зможе побачити його на своєму екрані

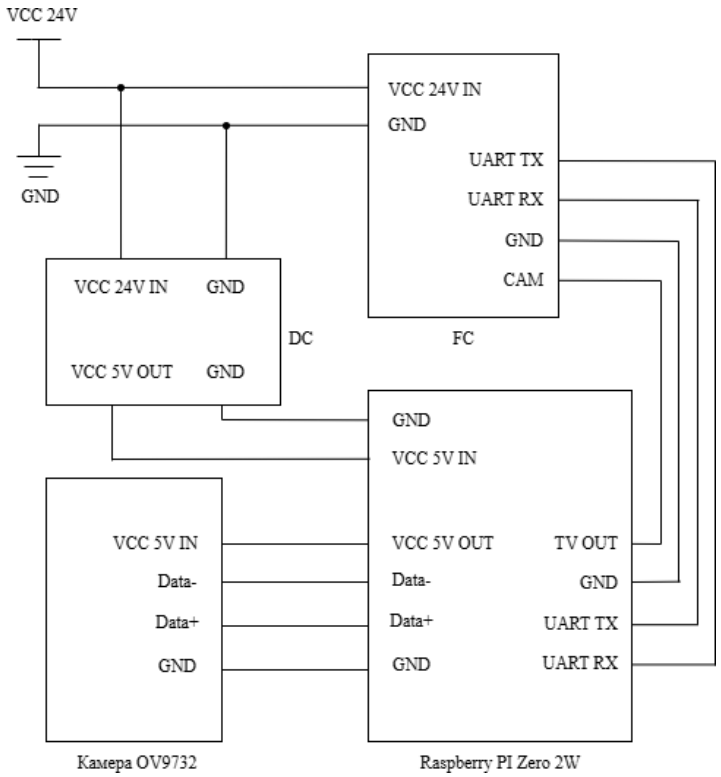


Рис 3.10 — Схема електрична принципова

На рисунку 3.10 можна побачити, що вхідним джерелом живлення для всіх компонентів системи є батарея FPV дрона, яка в середньому має напругу 24V. Польотний контролер дрону під’єднаний до батареї напряму, проте одноплатний комп’ютер потребує нижчої напруги для роботи. Згідно характеристик Raspberry Pi Zero 2W споживає напругу 5 вольт та 2.5 ампери струму. Відповідно для коректної роботи цього комп’ютера напруга з батареї понижується до 5 вольт завдяки схемі понижаючого DC перетворювача на

основі мікросхеми MP1584. Цифрова камера системи живеться по USB роз'єму, який іде від Raspberry.

Для передачі команд керування та телеметрії Zero 2W з'єднаний з польотним контролером за допомогою інтерфейсу UART. Як було описано в другому розділі дипломного проєкту, UART є дуплексним інтерфейсом, а отже для роботи потребує двох з'єднань між пристроями. Також для коректної передачі даних, варто з'єднувати контакт TX на першому пристрої з контактом RX на другому і навпаки.

Для передачі аналогового відеосигналу з одноплатного комп'ютера Zero 2W на ньому існує контакт TV OUT, що може видавати аналоговий сигнал формату PAL, саме цей формат аналогового відео може коректно оброблювати польотний контролер дрону. Зі сторони FC БПЛА сигнальний провід відео під'єднується до контакту CAM. У випадку звичайного підключення компонентів FPV-дрона цей контакт використовується для підключення аналогової курсової камери дрону, проте в системі донавдення замість камери використовується робочий стіл одноплатного комп'ютера, на якому запущене відповідне програмне забезпечення.

Також для коректної взаємодії системи необхідно налаштувати програмне забезпечення польотного контролера для роботи з донавденням. Як було описано в другому розділі дипломного проєкту, в якості програмного забезпечення для FC FPV-дрону було обрано прошивку Betaflight.

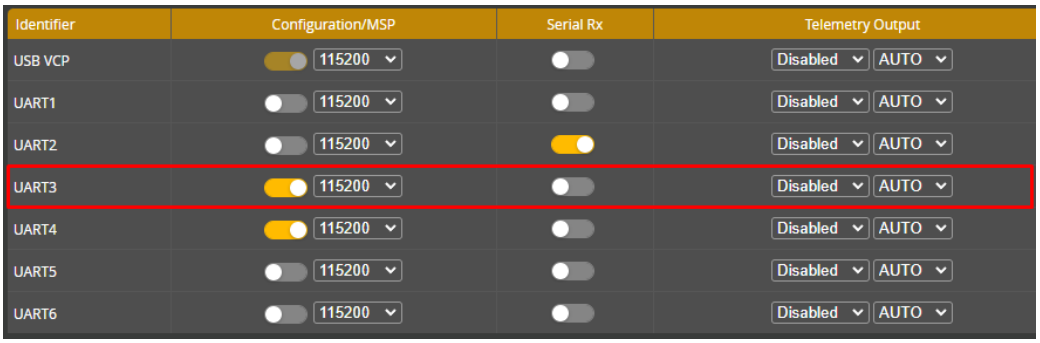


Рис 3.11 — Активація MSP

На рисунку 3.11 наведене перше необхідне налаштування в програмному забезпечені Betaflight, а саме активація обміну даними з одноплатним комп'ютером за допомогою протоколу MSP. Для цього зв'язку на польотному контролері був відведений UART, що має порядковий номер 3. Також для обміну даними була встановлена швидкість 115200 біт в секунду.

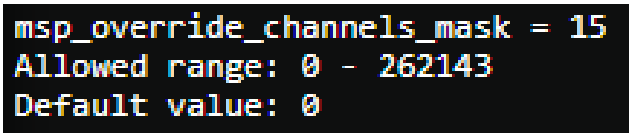


Рис 3.12 — Значення MSP маски

Наступним етапом налаштувань є встановлення значення для MSP маски. Це значення регулює, якими каналами для підміни даних з пульта система донаведення зможе керувати. На рисунку 3.12 цій масці було присвоєно значення 15. Нижче наведено формулу формування бітової маски:

aux4 aux3 aux2 aux1 yaw throttle pitch roll

0 0 0 0 1 1 1 1

Так як, системі необхідно керувати чотирма каналами, які задають положення літального апарату, то в перші чотири біти були встановлені одиниці, і отримано двійкове число 00001111, що в десятковому представлені є числом 15, яке і стало значенням для вище згаданої змінної.

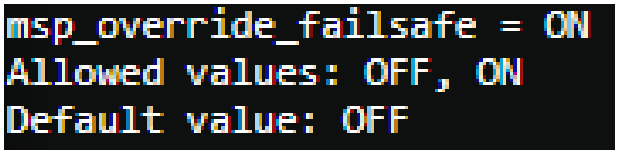


Рис 3.13 — Активація MSP при режимі failsafe

Як було зазначено в другому розділі дипломного проєкту, під час втрати зв’язку з пультом керування FPV-дрон починає виконувати процедуру failsafe. Проте одне із призначень системи донаведення при втраті зв’язку з пультом направляти дрон у раніше зафіксовану цілу. Тому на рисунку 3.13 показано встановлене значення змінної msp_override_failsafe = ON, що означає, що при втраті зв’язку з пультом, дрон замість стандартної процедури failsafe буде і далі виконувати команди керування від одноплатного комп’ютера системи донавдення.

Рис 3.14 — Визначення тумблера для активації MSP

Також для роботи системи необхідно в програмному забезпечені Betaflight задати номер та положення тумблера, при активації якого на пульті буде включатися система донаведення. Для цього в ПЗ польотного контролера був використаний режим MSP_OVERRIDE. Саме при активації цього режиму FC дрона починає отримувати команди керування від одноплатного комп’ютера та в системі запускається трекінг захопленої цілі.

Отже в цьому підрозділі, були розглянуті та реалізовані положення, які необхідні для успішної взаємодії програмного та апаратного забезпечення дрону з системою донавдення. Реалізована комп’ютерна система в дипломній роботі є гнучкою і дозволяє успішно інтегрувати її в переважну більшість FPV-дронів.

3.5 Опис інтерфейсу та взаємодії системи з користувачем

Одним із важливих елементів для пілота під час польоту є OSD. Як було зазначено в другому розділі, OSD формує польотний контролер дрону. Таким чином пілот бачить на екрані не тільки зображення з дрону, а ще до прикладу: заряд батареї, якість зв’язку, висоту, потужність відеопередавача, режим польоту тощо.

Проте для системи донавдення варто розробити власний інтерфейс (OSD), де буде розміщена вся необхідна інформація для пілота.

Рис 3.15 — Прицільна область системи донавдення

На рисунку 3.15 зображений перший елемент розробленого інтерфейсу, а саме квадрат прицільної області. Пілоту для активації системи донавдення варто помістити бажану ворожу ціль в цей квадрат та активувати тумблер донавдення, після цього система почне спрямовувати дрон на обраний об’єкт. Під час неактивного трекінгу цей прицільний квадрат знаходиться в центрі екрану та має зелений контур.

Рис 3.16 — Прицільна область під час активованого донавдення

Після активації донавдення та захоплення системою цілі, прицільний квадрат змінює колір контура на червоний (рисунок 3.16) та починає переміщатися відповідно до поточного положення цілі, що супроводжується.

Рис 3.17 — Збільшене зображення прицільної області

Для зручності захоплення цілі, для пілота у правому верхньому куті передбачене збільшене зображення прицільної області (рисунок 3.17). Завдяки цьому оператор FPV-дрона може точніше навести прицільний квадрат на бажаний ворожий об’єкт.

Рис 3.18 — Попередження в інтерфейсі системи

Також в інтерфейсі системи передбачено вивід деяких попереджувальних повідомлень. До прикладу на рисунку 3.18 зображене попередження про те, що перед запуском FPV-дрона варто деактивувати тумблер донавдення. До того ця заборона не дозволить взлетіти на FPV-дроні поки відповідний тумблер не буде вимкнений. Це безпекове обмеження зумовлене тим, що при старті БПЛА система може захопить небажану ціль для пілота і направити дрон на захоплений об’єкт.

Отже, в цьому підрозділі були розглянуті елементи інтерфейсу, які необхідні для того, щоб пілот міг зручно та безпечно використовувати розроблену систему донавдення.

4. ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Випробування системи донавдення

В минулому розділі була детально розглянута реалізація кожного модуля системи, а також описана взаємодія між ними. Проте кожна комп’ютерна система, особливого, яка буде застосовуватися в бойових діях, потребує ретельного тестування для виявлення можливостей системи, а також для знаходження прихованих недопрацювань. Для тестування системи донаведення необхідно встановити її на FPV-дрон, для цього був використаний 7 дюймовий БПЛА даного типу з розповсюдженою конфігурацією компонентів (рисунок 4.1).

Рис 4.1 — FPV-дрон для випробування розробленої системи

Тестування системи донавдення було розділено на декілька етапів, з метою протестувати кожен модуль розробленого ПЗ окремо, дані тести можна провести в лабораторних умовах. Після проведення описаних випробувань всю систему необхідно протестувати в наближеному до реальних умов середовищі, для такої задачі був використаний спеціально відведений полігон для випробування БПЛА.

Для оцінки коректності та ефективності роботи алгоритму трекінгу цілі були використані дві формули, одна із яких оцінює точність трекінгу за допомогою метрики перекриття IoU:

$$IoU = S\ перетину / S\ об'єднання$$

Вхідними даними у вище наведену формулу є площа перетину двох прямокутників: перший прямокутник це область де знаходиться об'єкт на даний момент; другий прямокутник це область, де трекінг знайшов об'єкт. Площа об'єднання є сумою площ обох прямокутників відняти площу перетину цих прямокутників.

Друга формула для оцінки ефективності трекінгу рахує кількість випадків у відсотках в яких алгоритм втратив ціль.

Ймовірність втрати цілі = Кадри, на яких ціль була втрачена / Всі кадри

Для проведення випробування алгоритму трекінгу в лабораторних умовах була використана велика кількість відео з польотами бойових FPV-дронів. Дані відеоматеріали були знайдені у відкритому доступі.

Рис 4.2 — Перше випробування алгоритму трекінгу. Фіксація цілі

Рис 4.3 — Перше випробування алгоритму трекінгу. Результат роботи алгоритму після наближення до цілі

На рисунку 4.2 та 4.3 наведений приклад одного із випробувань системи. Згідно отриманих результатів за час тестування алгоритм жодного разу не втратив ціль. Також варто розрахувати точність трекінгу на основі проведеного випробування. Для цього виділимо на зображені область, яку повинен був утримувати трекінг помаранчевим кольором.

Рис 4.4 — Порівняння фактичної та бажаної області трекінгу

На основі наведеного малюнку отримуємо необхідні вхідні дані для формули: *S перетину = 3021, S об'єднання = 3251*

Виходячи з цього, точність трекінгу для першого наведеного прикладу є

IoU (1) = 3054 / 3218 = 0.93

Рис 4.5 — Друге випробування алгоритму трекінгу. Порівняння фактичної та бажаної області трекінгу

Під час проведення цього випробування алгоритм трекінгу жодного разу не втратив ціль. На основі отриманих результатів розрахуємо точність трекінгу: *S перетину = 2915, S об'єднання = 3357*

IoU (2) = 2915 / 3357 = 0.87

Рис 4.6 — Третє випробування алгоритму трекінгу. Порівняння фактичної та бажаної області трекінгу

Під час проведення випробування, що зображено на рисунку 4.6 алгоритм трекінгу жодного разу не втратив ціль. На основі отриманих результатів розрахуємо точність трекінгу: *S перетину = 2970, S об'єднання = 3302*

$$IoU\ (3) = 2970 / 3302 = 0.899$$

Вище зображена лише частина проведених випробувань над системою фіксації та супроводу цілі, проте результати всіх тестів є дуже схожими між собою. А отже на основі вище проведених розрахунків, можна зробити такі висновки:

- Розроблений алгоритм трекінгу цілі не втрачає об'єкт на жодному кадрі, навіть в складних умовах.
- Середній показник точності IoU є на рівні 0.9, звідси точність супроводження цілі складає приблизно 90%, що є високим результатом і показує, що алгоритм точно ідентифікує і утримує необхідну ціль.

Наступним на черзі алгоритмом для випробування, є модуль для обміну даними між одноплатним комп'ютером та польотним контролером дрону. Для вимірювання ефективності цього алгоритму було обрано використовувати часову метрику, яка відображає з якою частотою надсилаються команди керування з Raspberry Pi на FC БПЛА.

```
dt = 0.0308922
[COMMANDS] Yaw: 1534 Pitch: 1669
1000
dt = 0.0309624
[COMMANDS] Yaw: 1558 Pitch: 1664
1000
dt = 0.030419
[COMMANDS] Yaw: 1503 Pitch: 1672
```

Рис 4.7 — Часові проміжки між обміном командами керування

На рисунку 4.7 видно, що команди керування відправляються на польотний контролер дрону приблизно кожні 0.03 секунди. Переведемо секунди в герци, щоб дізнатися частоту надсилання команд:

Частота надсилання команд = 1 / 0.03 = 33 Герци

Такого значення частоти оновлення команд керування є цілком достатньо для коректного функціонування системи донавдення з кількох причин.

По-перше, в системі донавдення частота кадрів становить приблизно 30 кадрів на секунду, а отже розроблене програмне забезпечення встигає обробити кожний кадр відео і в залежності від положення цілі на ньому, обчислити і передати на дрон пакет команд керування.

По-друге, частота передачі команд на рівні 30 герц забезпечує нам стабільність роботи з'єднання UART та роботи протоколу MSP. При вищих значення частоти з'єднання може бути перевантажене, що призведе до виникнення затримок.

Ще одним важливим параметром системи донавдення, який варто перевірити є затримка відео. Тобто це різниця в часі між тим коли, до прикладу, об'єкт з'явився перед камерою, та тим коли пілот побачив його на своєму екрані. Для проведення даного випробування був використаний секундомір та камера мобільного телефона, яка одразу знімала реальний час на секундомірі та екран з картинкою з дрона, при цьому камера дрону була направлена на той самий секундомір. В результаті такого випробування можна розрахувати затримку відео в системі порахувавши різницю між двома значенням секундоміру. Результати експерименту можна побачити на рисунку 4.8.

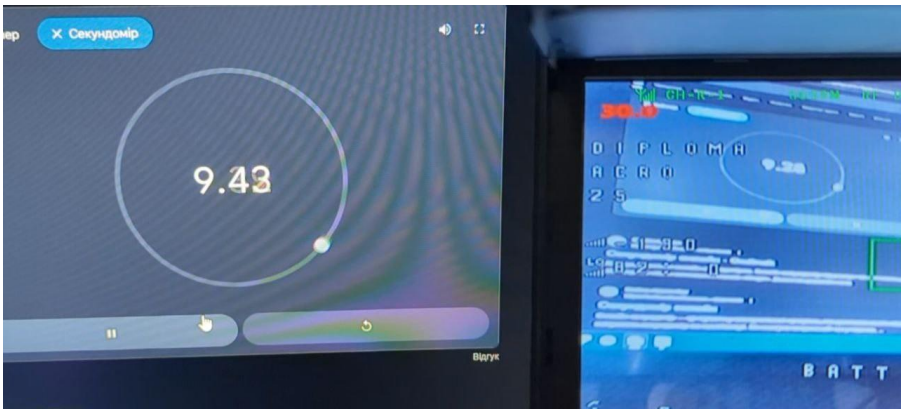


Рис 4.8 — Результати вимірювання затримки відео в розробленій системі

На рисунку 4.8 зліва показано значення секундоміра в реальному часі (9:43), з права можна побачити значення секундоміра на екрані пілота FPV-дрона зі встановленою системою донавдення (9:26). Звідси, порахувавши різницю між цими двома значеннями, можна отримати величину затримки відео в системі, а саме 170 мілісекунд.

Затримка передачі аналогового відеосигналу між дроном та екраном пілота в середньому становить від 20 до 40 мілісекунд. Отже, можна зробити висновок, що система донаведення створює затримку приблизно 130-150мс. Для порівняння затримка у цифрових відеосистемах для FPV-дронів, до прикладу SIYI HM30, становить приблизно від 150-170 мс. Звідси можна зробити висновок, що затримка відео в розробленій системі є допустимою, що дозволяє пілоту достатньо швидко реагувати на зміну ситуації у повітрі та точно керувати літальним апаратом.

Наступний параметр, який важливо оцінити це завантаженість системи при роботі трекінгу, обчислення команд керування та обміну даними між одноплатним комп’ютером та польотним контролером дрону. На рисунку 4.9, що наведений нижче, можна побачити, що при активній роботі системи донавдення завантаженість процесора складає 63%, а об’єм ОЗП, що використовується складає менше 50% від загального об’єму.

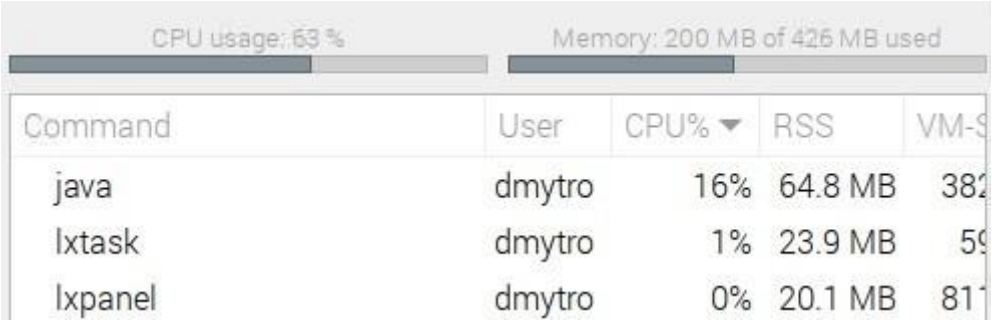


Рис 4.9 — Дані про завантаженість системи під час активної роботи розробленої системи

Отже, можна зробити висновок, що обчислювальних потужностей Raspberry Pi Zero 2W цілком вистачає для коректної роботи розробленою системи донавдення.

Ще одним важливим показником створеного програмного забезпечення є FPS вихідного відеопотоку з одноплатного комп'ютера системи.

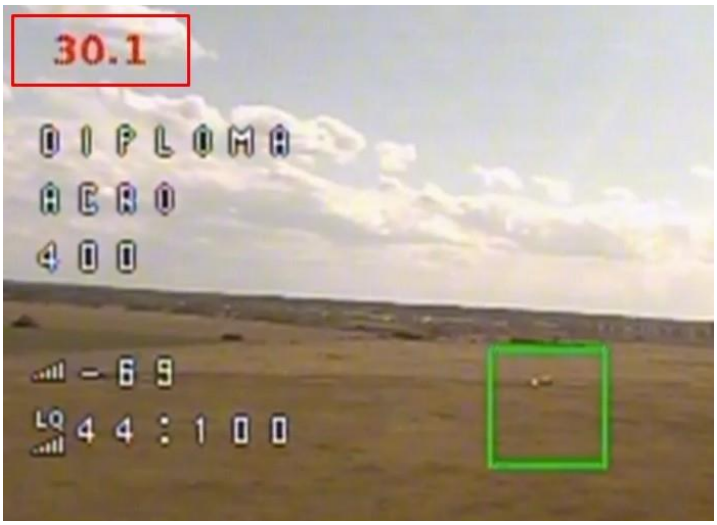


Рис 4.10 — FPS при польоті без активного трекінгу

Як видно з рисунку 4.10, при неактивованій системі донавдення пілот бачить картинку з частотою 30 кадрів на секунду. Проте при активації системи донавдення FPS опускається до рівня 27 кадрів на секунду (рисунок 4.11)

Рис 4.11 — FPS при польоті з активованим трекінгом

Частота кадрів стандартного аналогового PAL відеосигналу становить 25 Гц або 25 кадрів на секунду. Отже, значення FPS навіть при активному трекінгу є більш ніж достатнім для комфортного та звичного пілотування FPV-дроном.

Наступним етапом після лабораторних тестів, є комплексне випробування розробленої системи донавдення на полігоні, з метою отримати результати максимально наближені до тих, що будуть одержані в реальних умовах застосування системи. Для об’єктивної оцінки коректності роботи розробленого програмного забезпечення, необхідно розрахувати ефективність системи на основі різних випробувань. В якості метрики, для оцінки ефективності системи буде використовуватися точність донаведення дрона на бажану ціль. Ця велична буде розраховуватися за такими формулами:

$$Accuracy=(1-\frac{Error}{Emax})*100\,\%$$

де *Error* – помилка виражена в пікселях між центром зображення (напрямок польоту дрону) та між центром об’єкта, на який здійснюється донавдення;

Emax – максимально можлива похибка в пікселях.

$$Error=\sqrt{(x2-x1)^2+(y2-y1)^2}$$

де *(x2, y2)* – координати центра об’єкта, на який система донаводиться;

(x1, y1) – координати центра зображення.

$$Emax=\sqrt{(\frac{w}{2})^2+(\frac{h}{2})^2}$$

де *w* – ширина в пікселях відео на екрані пілота;

h – висота в пікселях відео на екрані пілота.

Звідси, знаючи роздільну здатність вихідного відеопотоку, а саме 720 на 576 пікселів, можна розрахувати значення змінної *Emax.*

$$Emax=\sqrt{(\frac{720}{2})^2+(\frac{576}{2})^2}=461$$

Тепер розглянемо із вибірки проведених випробувань на полігоні три стоп-кадри та розрахуємо точність донаведення. Для об’єктивності розрахунків ці три кадри взяті з трьох різних випробувань, в яких система супроводжувала різні за контрастом та дальністю цілі.

Нижче на рисунку 4.12 зображений кадр випробування донавдення при наведені на контрасту ціль з відстані 100м. Після прицілювання на бажаний об’єкт та активації системи, дрон перейшов на автоматичне керування. Під час польоту до цілі БПЛА вів себе передбачувано та стабільно, трекінг цілі не втрачався та алгоритм чітко утримував ціль.

На основі рисунку 4.12 розрахуємо необхідні значення для обчислення точності донавдення в цьому випробуванні.

$$Error=\sqrt{(379-360)^2+(277-288)^2}=21.96$$

Рис 4.12 — Випробування системи на контрастній ближній цілі

А отже, тепер можемо розрахувати точність донавдення в цьому випробуванні у відсотках:

$$Accuracy=(1-\frac{21.96}{461})*100\,\%=95\,\%$$

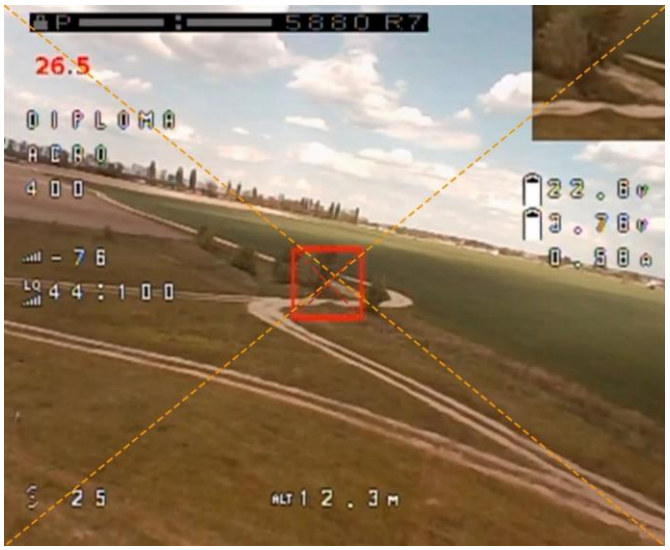


Рис 4.13 — Випробування системи на не контрастній ближній цілі

Під час випробування системи на ближній відстані на неконтрастній цілі дрон наближався до об’єкта стабільно та тримав необхідний курс, алгоритм трекінгу жодного разу не втратив об’єкт і чітко утримував бажану ціль, навіть по при те, що вона була не контрастною відносно навколишнього середовища. На основі рис 4.13 розрахуємо точність донаведення у цьому випробуванні:

$$Error=\sqrt{(351-360)^2+(296-288)^2}=12.04$$

$$Accuracy=(1-\frac{12.04}{461})*100\ \%=97\ \%$$

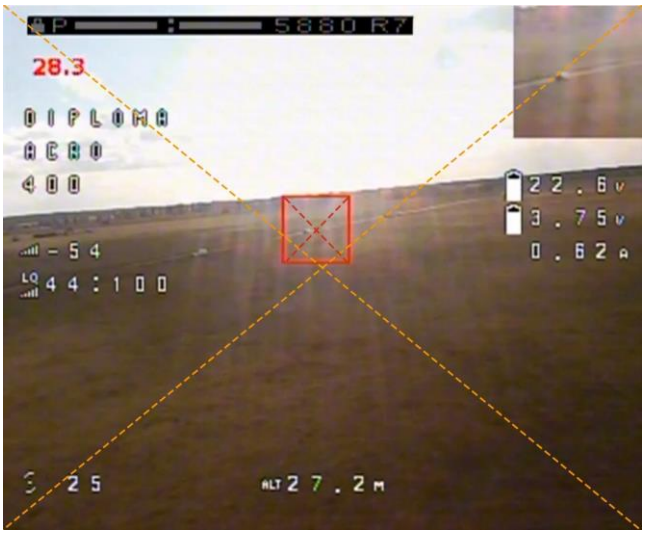


Рис 4.14 — Випробування системи на не контрастній дальній цілі

На рисунку 4.14 зображене випробування системи в складних умовах, а саме неконтрастна ціль була зафіксована на відстані 500м. Проте, алгоритм трекінгу захопив та утримував бажану ціль на протязі всієї роботи системи, сам БПЛА на основі отриманих команд рухався стабільно та тримав необхідний курс для попадання у бажаний об’єкт. На основі проведеного випробування, розрахуємо точність донавдення:

$$Error=\sqrt{(353-360)^2+(244-288)^2}=44.55$$

$$Accuracy = (1 - \frac{44.55}{461}) * 100\,\% = 90.34\,\%$$

Поза межами пояснювальної записки, розрахунки точності донаведення були зроблені для всіх проведених випробувань на полігоні. На основі цих розрахунків можна стверджувати, що точність донаведення становить від 90%, Для кожного випробування число точності варіювалося в залежності умов випробування. Під час усіх проведених тестів незалежно від дальності та контрастності цілі алгоритм трекінгу безперебійно утримував необхідну ціль.

Основний фактор, який впливав на точність донаведення у проведених випробуваннях стала відстань, на якій знаходиться ціль під час активації системи. Чим ближче ціль, тим точніше система летить на обраний об'єкт. Цю залежність можна описати тим, що з часом система накопичує незначну помилку в командах керування, тому чим далі ціль, тим менший відсоток точності.

Проте навіть попри описану вище залежність рівень точності системи залишається на рівні вище 90%, а отже і ефективність розробленої системи можна визначити на рівні такого ж значення, а саме 90%.

Таким чином в цьому підрозділі були проведені всебічні лабораторні та польові випробування розробленої системи донавдення. Згідно отриманих результатів система з точність приблизно 90% утримує ціль алгоритмом трекінгу; надсилає команди керування з частотою в середньому 33Гц; має затримку по відеосигналу 170 мс, що достатньо для комфортного пілотування; показує високу ефективність в реальних умовах застосування, а отже можна стверджувати, що система є придатною для використання. Розроблене програмне забезпечення є високоефективними (точність

донавдення 90%) та стабільним, а також відповідає всім поставленим перед ним вимогам.

4.2 Огляд можливих покращень системи

В минулому підрозділі була успішно протестована розроблена система донаведення і відповідно підтверджена її працездатність та ефективність. Проте кожний програмний продукт можна і необхідно удосконалювати та розширювати його функціонал особливо у випадку військової розробки, адже сучасна війна динамічно змінюється. Огляд сучасних тенденцій в цій області дозволяє визначити кілька перспективних напрямків для вдосконалення розробленого рішення.

Одним із найочевидніших шляхів покращення розробленого рішення є використання камери з вищою частотою кадрів, а саме зі значенням 60 FPS. Така апаратна зміна дозволить суттєво зменшити ефект змазування зображення під час руху дрона, а отже оператор БПЛА зможе краще контролювати політ. Також при збільшені вихідної частоти кадрів камери, алгоритм трекінгу буде краще утримувати необхідний об’єкт, особливо ефективність збільшиться при роботі з рухомими цілями. Це пов’язано з тим, що алгоритм трекінгу буде отримувати більше кадрів на секунду на вхід і відповідно оброблювати більше кадрів і відмічати на них необхідний об’єкт.

Окрім цього, важливим аспектом подальшого розвитку системи є зменшення загальної затримки відеопотоку. На мою думку, варто прагнути до затримки відеопотоку на рівні 50-70 мс. Для досягнення цієї мети, варто впровадити більш ефективні кодеки відео та реалізувати підтримку апаратного прискорення обробки зображень.

Проте наведені вище покращення неможливі без зміни апаратного забезпечення системи, а саме одноплатного комп’ютера. Це пов’язано з тим, що алгоритму доведеться оброблювати вдвічі більшу кількість кадрів і при

цьому мінімально затримувати потік, з метою зниження затримки відеопотоку. Виходячи з цього, згідно попередніх прорахунків, для цих задач необхідно використати одноплатний комп’ютер Raspberry Pi 4 на 4 GB ОЗП.

Також на потужностях оновленого одноплатного комп’ютера можна буде розгорнути алгоритми штучного інтелекту для автоматичного пошуку та розпізнавання цілі. В такому випадку оновлена система дозволить зробити БПЛА більш автономним, адже оператор зможе задавати квадрат, у якому FPV-дрону необхідно шукати ціль. Після цього БПЛА самостійно починає шукати цілі в зазначеній області, розпізнавати їх та розставляти пріоритети на їх ураження. Наступним етапом є самостійне ураження найпріоритетнішої цілі.

Також перспективним напрямком є застосування тепловізійної камерою з розробкою відповідного алгоритму супроводження цілі. У такому випадку БПЛА з цією системою зможе виконувати бойові завдання навіть вночі.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено комп'ютерну систему донаведення FPV-дрона на ціль, здатну захоплювати бажаний об'єкт та супроводжувати дрон до моменту влучання в нього. Робота охоплює як теоретичні, так і практичні аспекти створення подібної системи, зокрема застосування методів комп'ютерного зору, обробки відеопотоку в реальному часі, трекінгу об'єктів, обчислення керуючих команд та передавання їх на польотний контролер дрона.

В дипломній роботі було досягнуто низки важливих результатів. По-перше, забезпечено захоплення відеопотоку та передача його з прийнятною затримкою та частотою кадрів на екран пілота. По-друге, реалізовано ефективний механізм захоплення цілі, після чого система здатна стабільно супроводжувати об'єкт навіть під час його активного руху або зміні масштабу. По-третє, налагоджено надійне обчислення та передачу команд на автопілот дрона за допомогою MSP-протоколу, що дозволяє здійснювати безперервне донаведення на ціль з урахуванням її положення в кадрі.

Проведене тестування в лабораторних умовах та під час польотів на полігоні підтвердило працездатність і ефективність системи. Дрон успішно реагував на положення цілі, коригував напрям руху відповідно до змін її координат. Система продемонструвала високу стабільність трекінгу та точність у донаведенні, що є критично важливими показниками для практичного застосування.

Загалом, розроблена система повністю відповідає поставленій меті. Вона підтвердила свою ефективність у реальних умовах експлуатації, що є головним досягненням даної роботи. Отримана система може слугувати базою для подальшого розвитку систем донаведення, зокрема з використанням методів штучного інтелекту для більшої автономності БПЛА,

камер із вищою частотою кадрів та оптимізованих алгоритмів обробки зображень з метою зниження затримки відеопотоку.

СПИСОК ЛІТЕРАТУРИ

1. AeroVironment. Switchblade 600 [Електронний ресурс] / AeroVironment. Режим доступу: https://www.avinc.com.
2. Shahed-136: аналіз бойового застосування [Електронний ресурс] / Military Watch Magazine. Режим доступу: https://militarywatchmagazine.com.
3. Застосування баражуючих боєприпасів у сучасних конфліктах [Електронний ресурс] / The Drive – War Zone. Режим доступу: https://www.thedrive.com/the-war-zone.
4. DJI. Офіційний сайт [Електронний ресурс] / DJI. Режим доступу: https://www.dji.
5. Autel Robotics. EVO II Series [Електронний ресурс] / Autel Robotics. Режим доступу: https://auteldrones.com.
6. Skydio 2+: Автономний безпілотник з комп’ютерним зором [Електронний ресурс] / Skydio. Режим доступу: https://www.skydio.com. Betaflight. Betaflight Wiki [Електронний ресурс]. Режим доступу: https://betaflight.com/docs.
7. Matek Systems. Політні контролери та сенсори [Електронний ресурс]. Режим доступу: https://www.mateksys.com.
8. Klaus K. UAV Communication Protocols: SPI, I2C, UART and PWM // IEEE Transactions on Aerospace and Electronic Systems. – 2021. Burkov A. Modern Drone Navigation and Control. – Springer, 2020. – 320 с.
9. iNav Flight. PID Tuning [Електронний ресурс] / iNav Flight Wiki. Режим доступу: https://github.com/iNavFlight/inav/wiki/PID-Tuning.

10. MSP Protocol Specification [Електронний ресурс] / Betaflight GitHub. Режим доступу: https://github.com/betaflight/betaflight/wiki/MSP-Protocol.
11. LearnOpenCV. MOSSE Tracker: Fast Object Tracking using OpenCV [Електронний ресурс] / LearnOpenCV. Режим доступу: https://learnopencv.com/mosse-tracker-object-tracking-opencv-c-python/.
12. PX4 Autopilot Documentation. PX4 Open Source Flight Control [Електронний ресурс]. – Режим доступу: https://docs.px4.io.

ДОДАТОК 1

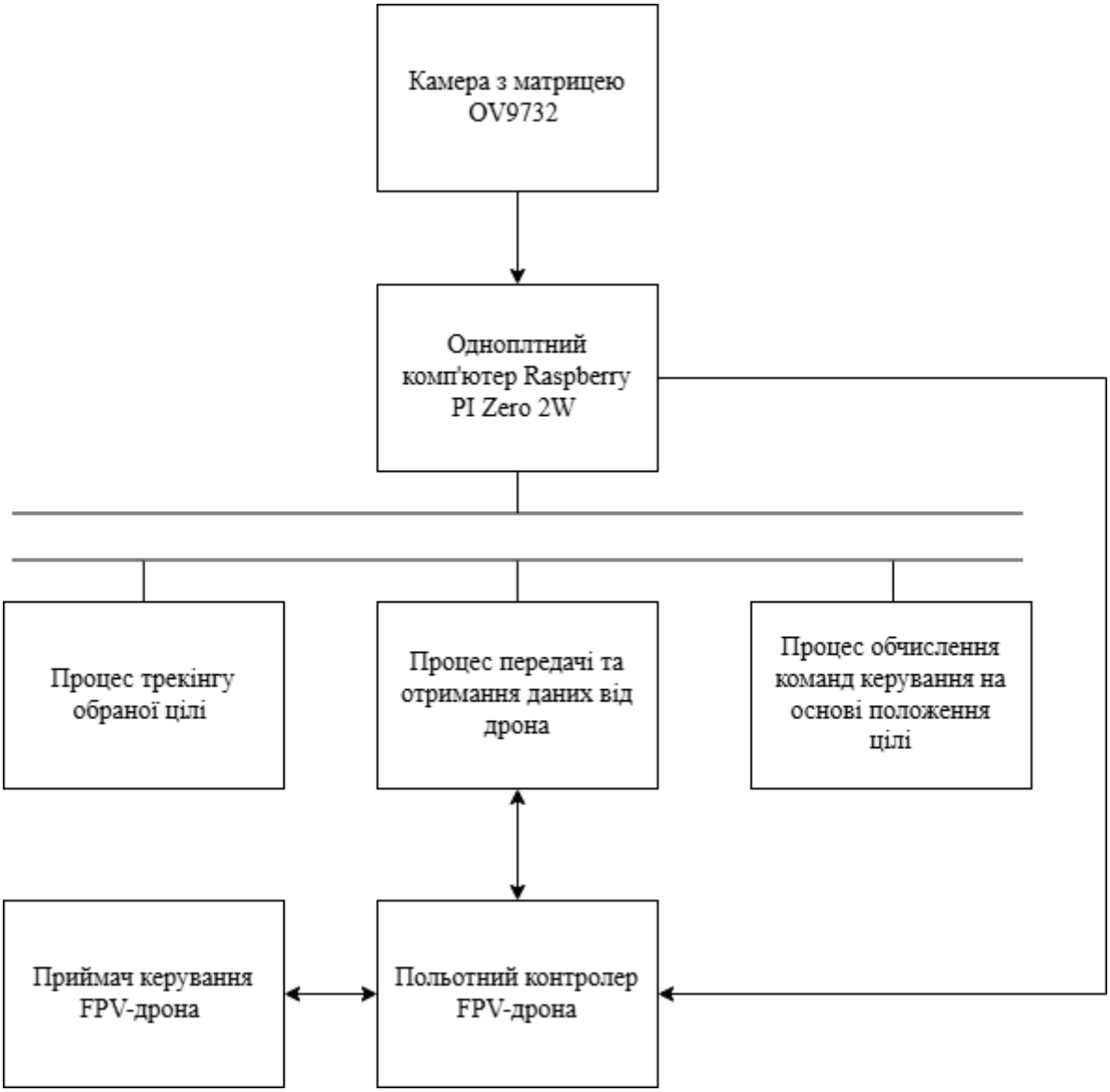
Комп’ютерна система донавдення FPV-дрона на ціль

Структурна схема роботи системи

ІАЛЦ.045490.005 Д1

Аркушів 1

Київ - 2025



					ІАЛЦ.045490.005 Д1		
Змін	Арк.	№ докум.	Підпис	Дата			
Ро·зробив		Гультяєв Д.А.			Комп’ютерна система донаведення FPV-дрона на ціль ***Структурна схема роботи ситеми***	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Петрашенко А.В					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Затвердив		Романкевич В.О.					

ДОДАТОК 2

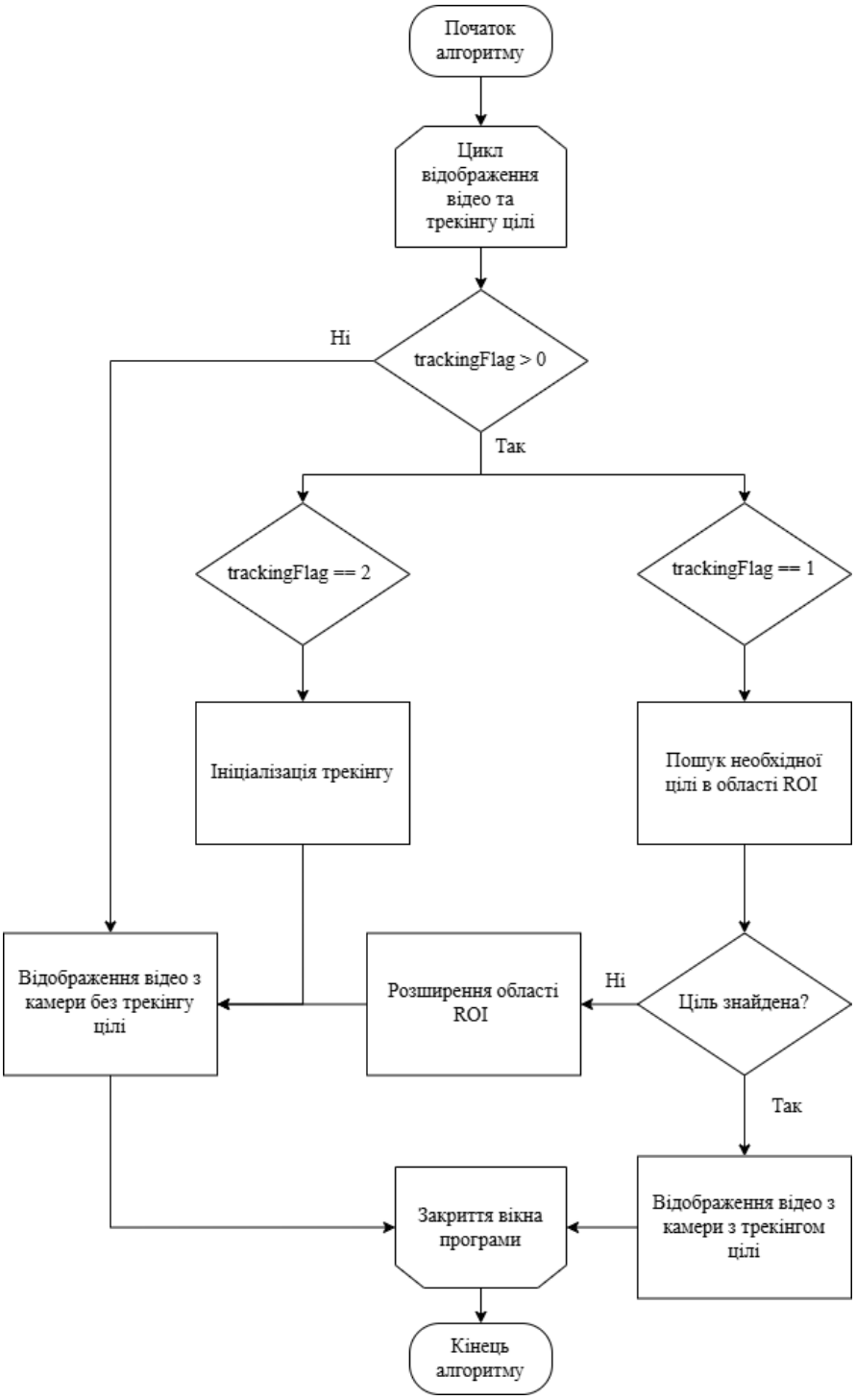
Комп’ютерна система донавдення FPV-дрона на ціль

Алгоритм роботи механізму захоплення та супроводження цілі

ІАЛЦ.045490.006 Д2

Аркушів 1

Київ - 2025



					ІАЛЦ.045490.006 Д2		
Змін	Арк.	№ докум.	Підпис	Дата			
Ро·зробив		Гультяєв Д.А.			Комп'ютерна система донаведення FPV-дрона на ціль ***Алгоритм роботи механізму захоплення та супроводження цілі***	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Петрашенко А.В					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Затвердив		Романкевич В.О.					

ДОДАТОК З

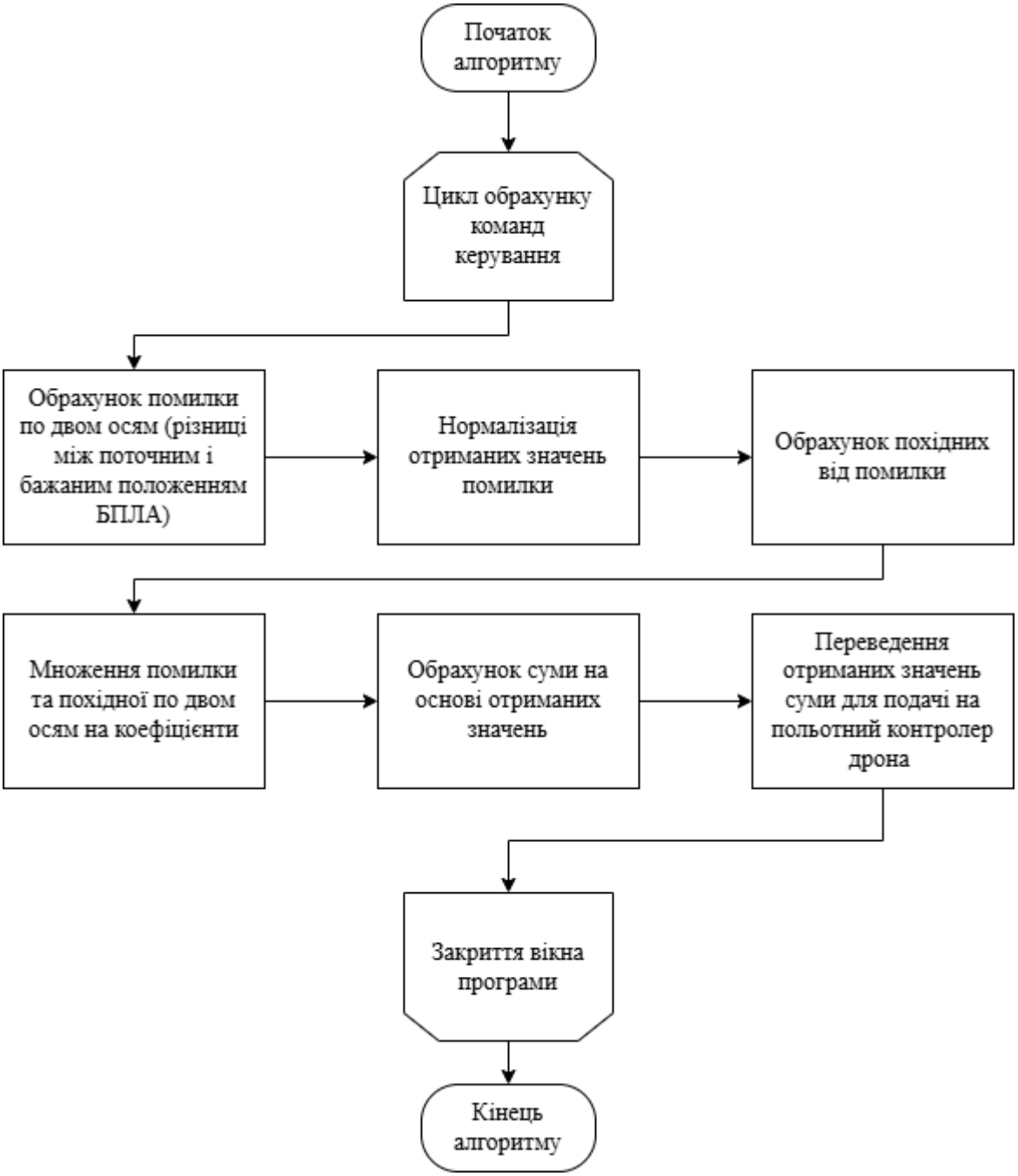
Комп’ютерна система донавдення FPV-дрона на ціль

Алгоритм спрямування БПЛА на захоплену ціль

ІАЛЦ.045490.007 Д3

Аркушів 1

Київ - 2025



					ІАЛЦ.045490.007 Д3		
Змін	Арк.	№ докум.	Підпис	Дата			
Ро·зробив		Гультяєв Д.А.			Комп'ютерна система донаведення FPV-дрона на ціль ***Алгоритм спрямування БПЛА на захоплену ціль***	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Петрашенко А.В					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Затвердив		Романкевич В.О.					

ДОДАТОК 4

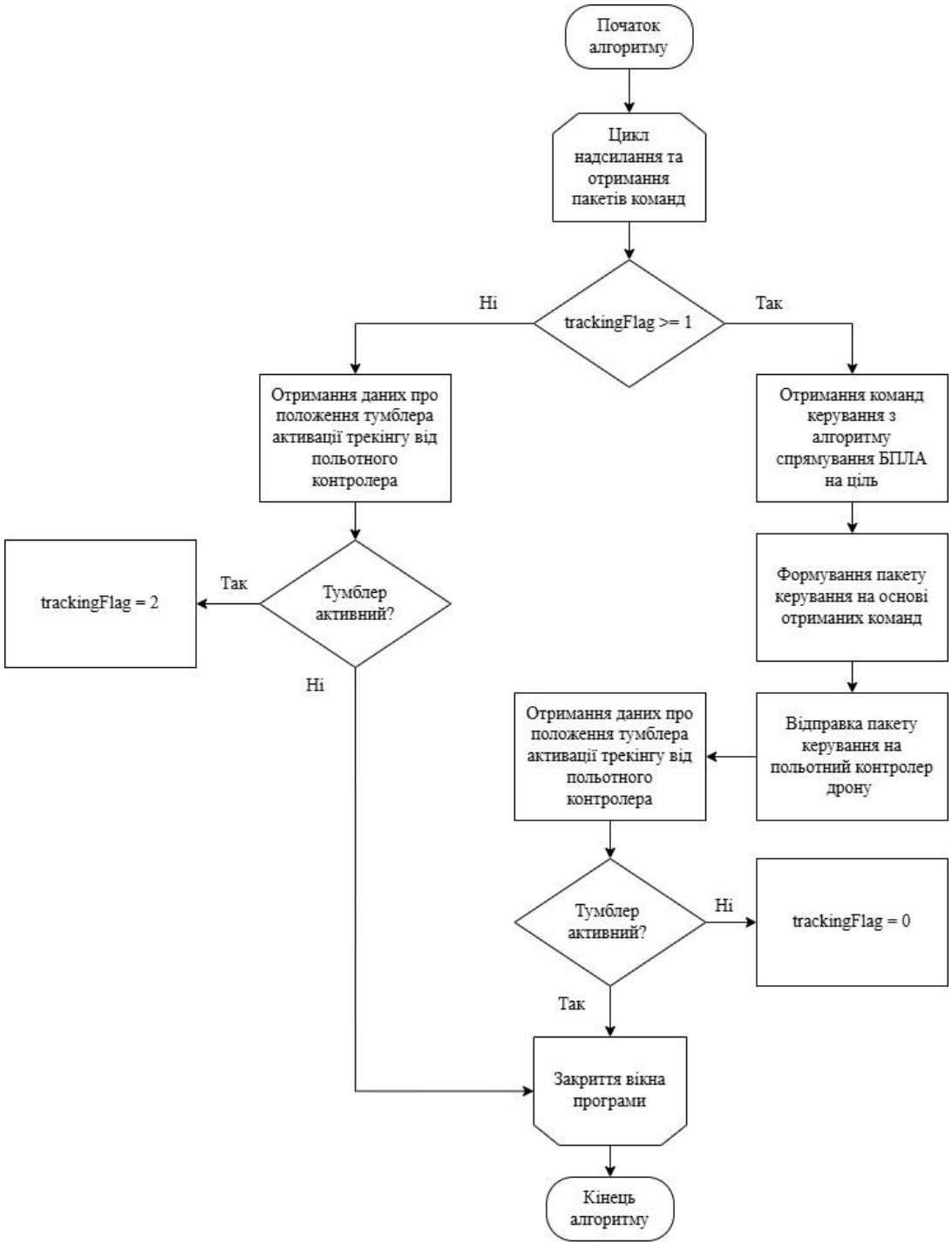
Комп’ютерна система донавдення FPV-дрона на ціль

Алгоритм побудови та передачі пакетів команд керування

ІАЛЦ.045490.008 Д4

Аркушів 1

Київ - 2025



					ІАЛЦ.045490.008 Д4		
Змін	Арк.	№ докум.	Підпис	Дата			
Ро·зробив		Гультяєв Д.А.			Комп'ютерна система донаведення FPV-дрона на ціль ***Алгоритм побудови та передачі пакетів команд керування***	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Петрашенко А.В					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Затвердив		Романкевич В.О.					

ДОДАТОК 5

Комп’ютерна система донавдення FPV-дрона на ціль

Текст програмного коду

ІАЛЦ.045490.009 Д5

Аркушів 1

Київ - 2025

Main.java

```
import org.opencv.core.*;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;
import org.opencv.tracking.legacy_Tracker;
import org.opencv.tracking.legacy_TrackerMOSSE;
import org.opencv.videoio.Videoio;

import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;

public class Main {
    public static final int WIDTH = 720;
    public static final int HEIGHT = 576;
    public static final int ROI_SCREEN_SIZE = WIDTH / 5;

    public static int roi_width = (int) (WIDTH * 0.1);
    public static int roi_height = (int) (WIDTH * 0.1);
    public static int roi_x = (WIDTH - roi_width)/2;
    public static int roi_y = (HEIGHT - roi_height)/2;

    public static Color color = Color.GREEN;
    private static int lostTrackingCounter = 0;
    public static volatile int trackingFlag = 0;
    public static volatile double xTarget = (double) WIDTH /2;
    public static volatile double yTarget = (double) HEIGHT /2;

    public static BufferedImage matToBufferedImage(Mat mat) {
        int type = BufferedImage.TYPE_3BYTE_BGR; // Для RGB
        BufferedImage image = new BufferedImage(mat.width(), mat.height(),
type);
        mat.get(0, 0, ((DataBufferByte)
image.getRaster().getDataBuffer()).getData());
        return image;
    }

    public static void main(String[] args) {
        JFrame window = new JFrame("Detection");
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setUndecorated(true);

        Toolkit toolkit = Toolkit.getDefaultToolkit();
        BufferedImage cursorImg = new BufferedImage(16, 16,
BufferedImage.TYPE_INT_ARGB);
        Cursor invisibleCursor = toolkit.createCustomCursor(cursorImg, new
java.awt.Point(0, 0), "Invisible");

        window.setCursor(invisibleCursor);
```

```
        try {
            Thread.sleep(500);
            Robot robot = new Robot();
            robot.mouseMove(0, 100);
        } catch (AWTException | InterruptedException e) {
            e.printStackTrace();
        }

        CameraPanel panel = new CameraPanel();
        panel.setBackground(Color.BLACK);
        panel.setPreferredSize(new Dimension(WIDTH, HEIGHT));

        window.setContentPane(panel);
        window.pack();
        panel.setFocusable(true);
        window.setVisible(true);

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        // Init camera
        VideoCapture videoFlow = new VideoCapture(0); //
/home/dmytro/diploma/code/media/6.mp4
        videoFlow.set(Videoio.CAP_PROP_FPS, 60);
        videoFlow.set(3, WIDTH);
        videoFlow.set(4, HEIGHT);
        Mat frame = new Mat();

        MSPController msp = new MSPController();
        msp.start();

        // Init tracker
        Rect2d roi = null;
        legacy_Tracker tracker = null;

        // Main loop
        if (videoFlow.isOpened()) {
            while (videoFlow.read(frame)) {
                if (trackingFlag > 0) {
                    if (trackingFlag == 2) {
                        tracker = legacy_TrackerMOSSE.create();
                        roi = new Rect2d(roi_x, roi_y, roi_width,
roi_height);
                        tracker.init(frame, roi);
                        trackingFlag = 1;
                        System.out.println("[TRACKING] Initialized tracker");
                    }

                    if (trackingFlag == 1) {
                        boolean trackingSuccess = tracker.update(frame, roi);

                        if (trackingSuccess) {
                            roi_width = (int) roi.width;
                            roi_height = (int) roi.height;
                            roi_x = (int) roi.x;
                            roi_y = (int) roi.y;
                            color = Color.RED;
                            lostTrackingCounter = 0;

                            synchronized (Main.class) {
```

```
                                xTarget = roi.x + roi.width / 2.0;
                                yTarget = roi.y + roi.height / 2.0;
                            }
                        } else {
                            lostTrackingCounter++;
                            if (lostTrackingCounter > 10) {
                                roi_width = Math.min((int) (roi_width * 1.5),
WIDTH);
                                roi_height = Math.min((int) (roi_height *
1.5), HEIGHT);
                                roi_x = Math.max(0, (int) (roi_x - roi_width
* 0.25));
                                roi_y = Math.max(0, (int) (roi_y - roi_height
* 0.25));

                                if (roi_x + roi_width > WIDTH) roi_x = WIDTH
- roi_width;
                                if (roi_y + roi_height > HEIGHT) roi_y =
HEIGHT - roi_height;

                                System.out.println("[TRACKING] Lost.
Expanding ROI.");
                                trackingFlag = 2;
                            }
                        }
                    }
                }

                BufferedImage image = matToBufferedImage(frame);
                panel.setImage(image);

                if (roi_x >= 0 && roi_y >= 0 && roi_x + roi_width <=
frame.cols() && roi_y + roi_height <= frame.rows()){
                    Mat roiFrame = frame.submat(new Rect(roi_x, roi_y,
roi_width, roi_height));
                    Mat magnifiedFrame = new Mat();
                    Imgproc.resize(roiFrame, magnifiedFrame, new
Size(ROI_SCREEN_SIZE, ROI_SCREEN_SIZE));
                    BufferedImage magnifiedImage =
matToBufferedImage(magnifiedFrame);
                    panel.setROIImage(magnifiedImage);
                }

                panel.repaint();

                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            }
        } else {
            System.out.println("Camera didn't found!");
        }

        msp.stopController();
        videoFlow.release();
    }
```

```
}
```

CameraPanel.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;

public class CameraPanel extends JPanel {
    private BufferedImage image;
    private BufferedImage roiImage;
    public static String textInforamtion = "";

    public void setImage(BufferedImage image){
        this.image = image;
    }

    public void setROIImage(BufferedImage img) {
        this.roiImage = img;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;

        if (image != null) {
            g2d.drawImage(image, 0, 0, this);
        }
        if (roiImage != null) {
            g.drawImage(roiImage, Main.WIDTH - Main.ROI_SCREEN_SIZE, 0,
null);
        }

        g2d.setColor(Main.color);
        g2d.setStroke(new BasicStroke(3));
        g2d.drawRect(Main.roi_x, Main.roi_y, Main.roi_width,
Main.roi_height);

        g2d.setFont(new Font("Arial", Font.BOLD, 24));
        FontMetrics fm = g2d.getFontMetrics();

        int textWidth = fm.stringWidth(textInforamtion);
        int x = (getWidth() - textWidth) / 2;

        g2d.setColor(Color.RED);
        g2d.drawString(textInforamtion, x, 380);
    }
}
```

MSPClient.java

```
import java.io.IOException;
import com.pi4j.io.serial.*;
import java.io.ByteArrayOutputStream;
import java.nio.ByteBuffer;
```

```
public class MSPClient {
    private final Serial serial;
    private static final String PREAMBLE = "$M";

    public MSPClient(String USBport) {
        serial = SerialFactory.createInstance();

        SerialConfig config = new SerialConfig();
        config.device("/dev/serial0")
                .baud(Baud._115200)
                .dataBits(DataBits._8)
                .parity(Parity.NONE)
                .stopBits(StopBits._1)
                .flowControl(FlowControl.NONE);

        try {
            serial.open(config);
        } catch (IOException e) {
            System.out.println("Serial wasn`t open");
            throw new RuntimeException(e);
        }

        if(serial.isOpen()) {
            System.out.println("Serial opened");
        }
    }

    public void sendRequestToGetData(MSPCommands command){
        byte[] message = createGetDataCommand(command);
        try {
            serial.write(message);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public void sendRCRequest(int[] RCCommands) {
        byte[] message = createMspSetRCCommand(RCCommands);
        try {
            serial.write(message);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public int[] getRCData(){
        int[] data = new int[30];

        byte[] response = new byte[1024];
        try {
            serial.read(ByteBuffer.wrap(response));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        int pos = 0;

        for(int i = 5; i <= 32; i+=2){
```

```
                int respDec = ((response[i+1] & 0xFF) << 8) | (response[i] &
0xFF);
                if((respDec >= 800 && respDec <= 2100) || respDec == 0){
                    data[pos] = respDec;
                    pos++;
                }
            }
            return data;
        }

        private byte[] createGetDataCommand(MSPCommands command){
            ByteArrayOutputStream commandStream = new ByteArrayOutputStream();
            int checksum = 0;

            commandStream.write(0x24);
            commandStream.write(0x4D);
            commandStream.write(0x3C);

            commandStream.write(0);

            commandStream.write(command.getID());
            checksum ^= command.getID();
            commandStream.write(checksum & 0xFF);

            return commandStream.toByteArray();
        }

        private byte[] createMspSetRCCommand(int[] RCCommands){
            ByteArrayOutputStream commandStream = new ByteArrayOutputStream();
            int checksum = 0;

            commandStream.write(0x24);
            commandStream.write(0x4D);
            commandStream.write(0x3C);

            int length = RCCommands.length * 2;
            commandStream.write(length & 0xFF);
            checksum ^= length & 0xFF;

            commandStream.write(0xC8);
            checksum ^= 0xC8;

            for (int i = 0; i < RCCommands.length; i++) {
                int RCCommand = RCCommands[i];
                commandStream.write(RCCommand & 0xFF);
                commandStream.write((RCCommand >> 8) & 0xFF);
                checksum ^= RCCommand & 0xFF;
                checksum ^= (RCCommand >> 8) & 0xFF;
            }

            commandStream.write(checksum & 0xFF);
            for(byte num : commandStream.toByteArray()){
                System.out.printf("%02X ", num);
            }
            //System.out.println("");
            return commandStream.toByteArray();
        }
    }
```

MSPCommands.java

```
public enum MSPCommands {
    MSP_RC(0x69),
    MSP_SET_RAW_RC(0xC8);

    private int id;

    MSPCommands(int id) {
        this.id = id;
    }
    public int getID() {
        return id;
    }
}
```

MSPController.java

```
import java.awt.*;
import java.util.Arrays;

public class MSPController extends Thread {
    private final MSPClient mspClient = new MSPClient("COM4");
    private final PIDController pid = new PIDController();
    private volatile boolean running = true;
    private int i = 0;
    private int j = 0;
    private boolean armWarning = false;
    private int throttleLevel = 1000;

    @Override
    public void run(){
        while (running){
            try {
                if(Main.trackingFlag >= 1){
                    double x, y;
                    synchronized (Main.class) {
                        x = Main.xTarget;
                        y = Main.yTarget;
                    }
                    int[] commands = pid.calculateCommands(new double[]
{x,y});
                    System.out.println("[COMMANDS] Yaw: " + commands[0] + "
Pitch: " + commands[1]);
                    int[] RCCommands =
{1500,commands[1],throttleLevel,commands[0]};
                    mspClient.sendRCRequest(RCCommands);
                    if(i == 10){
                        mspClient.sendRequestToGetData(MSPCommands.MSP_RC);
                        int[] data = mspClient.getRCData();
                        if(data[7] < 1800){
                            System.out.println("FINISH TRACKING");
                            Main.trackingFlag = 0;
```

```
                            Main.roi_width = (int) (Main.WIDTH * 0.1);
                            Main.roi_height = (int) (Main.WIDTH * 0.1);
                            Main.roi_x = (Main.WIDTH - Main.roi_width)/2;
                            Main.roi_y = (Main.HEIGHT - Main.roi_height)/2;
                            Main.color = Color.GREEN;
                        }
                        i = 0;
                    }
                    i++;
                } else {
                    mspClient.sendRequestToGetData(MSPCommands.MSP_RC);
                    int[] data = mspClient.getRCData();
                    System.out.println(Arrays.toString(data));
                    if(j == 100){
                        int[] tRCCommands = {1500,1500,throttleLevel,1500};
                        mspClient.sendRCRequest(tRCCommands);
                    }
                    if(data[7] < 1800){
                        throttleLevel = data[3];
                    }
                    if (data[7] > 1800) { //MSP Switch
                        if(data[4] > 1800 && !armWarning){ //ARM Switch
                            Main.trackingFlag = 2;
                        } else if(!armWarning) {
                            CameraPanel.textInforamtion = "DISABLE TRACKING
SWITCH BEFORE ARM";
                            armWarning = true;
                            int[] RCCommands = {1500,1500,1050,1500};
                            mspClient.sendRCRequest(RCCommands);
                        }
                    } else {
                        CameraPanel.textInforamtion = "";
                        armWarning = false;
                    }
                    j++;
                }

                Thread.sleep(40);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public void stopController() {
        running = false;
    }
}
```

PIDController.java

```
public class PIDController {
    private double kp, kd;
    private double xCenter, yCenter;
    private double[] previousErrors;
    private double dt;
```

```
    private long lastTime;

    public PIDController() {
        kp = 0.5;
        kd = 0.025;
        previousErrors = new double[]{0, 0};
        lastTime = System.nanoTime();
        xCenter = (double) Main.WIDTH / 2;
        yCenter = (double) Main.HEIGHT / 2;
    }

    public int[] calculateCommands(double[] targetCoordinates){
        long now = System.nanoTime();
        dt = (now - lastTime) / 1_000_000_000.0;
        System.out.println("dt = " + dt);
        lastTime = now;

        double errorX = targetCoordinates[0] - xCenter;
        double errorY = targetCoordinates[1] - yCenter;
        double normErrorX = errorX / xCenter;
        double normErrorY = errorY / yCenter;

        double[] pidResults = calculatePIDSum(new double[]{normErrorX,
normErrorY});

        return mapToRC(pidResults);
    }

    private double[] calculatePIDSum(double[] errors){
        double[] pidSum = new double[2];

        double derivativeYaw = (errors[0] - previousErrors[0]) / dt;
        double derivativePitch = (errors[1] - previousErrors[1]) / dt;
        pidSum[0] = kp * errors[0] + kd * derivativeYaw;
        pidSum[1] = kp * errors[1] + kd * derivativePitch;
        previousErrors[0] = errors[0];
        previousErrors[1] = errors[1];

        return pidSum;
    }

    private double clamp(double value) {
        return Math.max(-1.0, Math.min(1.0, value));
    }

    private int[] mapToRC(double[] pidResults){
        return new int[]{(int) (1500 + 500 * clamp(pidResults[0])), (int)
(1500 + 500 * clamp(pidResults[1]))};
    }
}
```