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

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Системи керування квадрокоптером універсального призначення
Назва теми

КвРКІ 210246.21.02.08 ПЗ
Шифр

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

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

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

Виконав: студент IV курсу, група КІ2-21-2 _____ Денис ПОПЛАВСЬКИЙ
Підпис Ініціали, прізвище

Керівник _____ Олег САВЕНКО
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата Ініціали, прізвище

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

«16» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Поплавському Денису Юрійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система керування квадрокоптера універсального призначення

Керівник проекту (роботи) Олег САВЕНКО, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 12.06.2025 р.

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

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

Система керування квадрокоптера універсального призначення

проєктування інтелектуальної підсистеми обробки візуальної інформації у кіберфізичній системі керування універсального квадрокоптера

програмно-апаратна реалізація керування квадрокоптером універсального призначення за допомогою жестів

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

Архітектура ПЗ проєкту

Архітектура ПЗ для кіберфізичної системи

Апаратне забезпечення проєкту

6. Консультанти розділів дипломного проекту (роботи)

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

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – Кіберфізична система керування універсальним квадрокоптером: постановка задач	01.03.2025	виконано
4	Робота над розділом 2 – обробка зорової інформації в кіберфізичній системі квадрокоптера	01.04.2025	виконано
5	Робота над розділом 3 – система жестового керування квадрокоптером	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	21.05.2025	виконано
7	Попередній захист ВКР	23.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Денис ПОПЛАВСЬКИЙ
Підпис Ініціали, прізвище

Керівник роботи _____ Олег САВЕНКО
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 210246.21.02.08	Пояснювальна записка	60		
			Графічні матеріали			
2		КвРКІ 210246.21.02.08	Архітектура ПЗ проєкту	1		
3		КвРКІ 210246.21.02.08	Принципова електрична	1		
			схема проєкту			
4		КвРКІ 210246.21.02.08	Апаратне забезпечення	1		
			проєкту			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 210246.21.02.08 ВП		
Розробив		Поплавський			Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Савенко				У	1	1
Н. контр.		Кисіль		16.06.25		ХНУ, КІ2-21-2		
Затв.		Павлова		16.06.25				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система керування квадрокоптера універсального призначення».

Автор роботи: Денис ПОПЛАВСЬКИЙ.

Керівник роботи: Савенко Олег Станіславович.

Пояснювальна записка: 64 с., 17 рис., 1 табл., 3 дод., 54 джерел.

Графічна частина: 3 креслення.

КВАДРОКОПТЕР, СИСТЕМА КЕРУВАННЯ, Архітектура, МОНІТОРИНГ.

Метою дипломної роботи є розробка та дослідження системи керування квадрокоптером універсального призначення, яка забезпечує стабільний політ, маневреність та можливість адаптації до різних сценаріїв застосування (моніторинг, доставка, картографування тощо), з використанням сучасних апаратних та програмних засобів.

Об'єктом дослідження є процеси автоматичного керування безпілотними літальними апаратами типу квадрокоптер.

Предметом дослідження є програмно-апаратні засоби та алгоритми, що забезпечують стабільне, ефективне та гнучке керування квадрокоптером універсального призначення в різних умовах експлуатації.

Під час проведення дослідження на цю тему було використано метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.

Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП 4

1 КІБЕРФІЗИЧНА СИСТЕМА КЕРУВАННЯ КВАДРОКОПТЕРОМ УНІВЕРСАЛЬНОГО ПРИЗНАЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇЇ РОЗРОБКИ І ВДОСКОНАЛЕННЯ 4

1.1 Аналіз структурних і функціональних особливостей системи керування квадрокоптером універсального призначення 5

1.2 Аналіз програмно-апаратних засобів реалізації системи керування квадрокоптером та формулювання технічних вимог та постановка задачі щодо вдосконалення системи 12

1.3 Висновки 16

2 МЕТОДИ ОБРОБКИ ІНФОРМАЦІЇ В АДАПТИВНІЙ КІБЕРФІЗИЧНІЙ СИСТЕМІ МОНІТОРИНГУ ДЛЯ БЕЗПІЛОТНОГО АПАРАТА 18

2.1 Визначення апаратних і програмних підсистем програмно-технічного засобу 18

2.2 Бібліотека та їх класи OpenCV 24

2.3 Розробка системи управління квадрокоптером на базі IMU та ультразвукового далекоміра HC-SR04 26

2.4 Проєктування ПЗ для розпізнавання жестів руки з відеосигналу за допомогою бібліотек OpenCV 36

2.5 Висновок 40

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ КЕРУВАННЯ КВАДРОКОПТЕРОМ УНІВЕРСАЛЬНОГО ПРИЗНАЧЕННЯ ЗА ДОПОМОГОЮ ЖЕСТІВ 42

3.1 Опис реалізації модулів апаратного та програмного забезпечення програмно-технічного засобу 42

3.2 Обробка зображення 44

3.3 Обробка результатів 53

3.4 Висновки 60
ВИСНОВКИ 62
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ..... 64
ДОДАТОК А 66
ДОДАТОК Б 67
ДОДАТОК В 68

ВСТУП

На сучасному етапі розвитку технологій безпілотні літальні апарати (БПЛА) стали однією з найдинамічніших сфер інженерії. Вони широко застосовуються як у військовій справі, так і в цивільному секторі. Особливо стрімко зростає популярність мультироторних БПЛА, зокрема мультикоптерів, які вирізняються простотою керування, відносною дешевизною та широким функціоналом. Сьогодні такі апарати активно використовуються для логістичних задач, аерофотозйомки, спостереження за навколишнім середовищем, участі в рятувальних операціях тощо.

Інженерні та наукові розробки в цій галузі ведуться не лише великими компаніями, а й окремими ентузіастами, що сприяє стрімкому технічному прогресу. У рамках цього проєкту було реалізовано систему керування чотириротором мультикоптером з акцентом на інтуїтивність та високу чутливість взаємодії. Метою дослідження стало спрощення керування дроном шляхом усунення необхідності в традиційних пультах дистанційного управління.

У результаті було спроєктовано прототип пристрою, здатного здійснювати передачу команд керування мультикоптером з наземного пункту, використовуючи жести однієї руки.

Процес створення пристрою охоплював усі етапи розробки – від початкового технічного задуму до програмної реалізації. У розробці активно застосовувалися технології з відкритим вихідним кодом, зокрема мікроконтролерна платформа Arduino для організації апаратного управління та бібліотека OpenCV для реалізації алгоритмів комп’ютерного зору.

Хоча наразі пристрій перебуває на стадії функціонального прототипу, він уже демонструє реальний потенціал до подальшого вдосконалення та практичного використання. Основні напрями майбутнього розвитку включають підвищення точності інтерпретації жестів, розширення можливостей управління та оптимізацію апаратної складової.

КІБЕРФІЗИЧНА СИСТЕМА КЕРУВАННЯ КВАДРОКОПТЕРОМ УНІВЕРСАЛЬНОГО ПРИЗНАЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇЇ РОЗРОБКИ І ВДОСКОНАЛЕННЯ

1.1 Аналіз структурних і функціональних особливостей системи керування квадрокоптером універсального призначення

Системи керування безпілотними літальними апаратами (БПЛА), зокрема квадрокоптерами, є одним із найбільш динамічно розвинених напрямів у галузі робототехніки та автоматизації. Універсальний квадрокоптер характеризується здатністю виконувати широкий спектр задач – від моніторингу територій до автономної доставки, що зумовлює високі вимоги до його системи керування.

Типовий квадрокоптер має хрестоподібну конструкцію, де на кінцях променів рами закріплені чотири двигуни з пропелерами. Ключовим елементом, який забезпечує функціональність усієї системи, є політний контролер (Flight Controller, FC). Це мікропроцесорний пристрій, який здійснює обробку інформації від сенсорів, виконує алгоритми стабілізації та формує сигнали управління.

До складу системи керування квадрокоптером входять такі основні компоненти:

1. Сенсори орієнтації та положення:

- Акселерометр – вимірює прискорення за трьома осями, що дозволяє оцінити нахил апарата.
- Гіроскоп – вимірює кутову швидкість обертання, використовується для стабілізації.
- Магнітометр – забезпечує орієнтацію відносно магнітного поля Землі.
- Барометр – вимірює атмосферний тиск, використовується для утримання висоти.
- GPS – модуль – визначає географічне положення і швидкість руху.

2. Блок живлення – включає літій–полімерний (LiPo) акумулятор, який подає напругу на всі електронні компоненти. Живлення двигунів проходить через

електронні регулятори швидкості (ESC), що приймають ШІМ–сигнал від контролера та формують напругу на обмотках двигунів.

3. Комунікаційні модулі – відповідають за передачу інформації між квадрокоптером та наземною станцією. Зазвичай використовуються радіоканали (2.4 ГГц або 5.8 ГГц), Wi-Fi, Bluetooth або телеметричні передавачі (наприклад, MAVLink-протокол).

4. Програмне забезпечення та прошивка контролера – визначає логіку роботи квадрокоптера. Відомі прошивки типу ArduPilot, Betaflight, PX4 містять реалізації стабілізаційних алгоритмів, фільтрації сенсорних даних, обробки команд користувача, та ін.

Функціональні особливості – система керування квадрокоптером повинна забезпечувати виконання таких основних функцій:

- Стабілізація положення у просторі – компенсація зовнішніх впливів (вітер, зміщення маси) та утримання заданого положення.
- Навігація – можливість пересування за заздалегідь заданим маршрутом або в режимі ручного управління.
- Контроль висоти – регулювання тяги всіх двигунів на основі даних барометра та/або GPS.
- Управління орієнтацією (Roll, Pitch, Yaw) – досягається шляхом змінювання швидкості обертання окремих двигунів.

Принципи дії – кожен з чотирьох пропелерів квадрокоптера створює підйомну силу та момент обертання. Для того щоб зберегти рівновагу:

- Два двигуни обертаються за годинниковою стрілкою, два – проти. Це дозволяє компенсувати момент, що виникає через обертання.
- Зміна швидкості обертання певних двигунів дозволяє нахиляти апарат у відповідному напрямку. Наприклад, зменшення швидкості обертання задніх двигунів і збільшення передніх спричинить нахил уперед (pitch).

– Зміна співвідношення швидкостей лівих і правих двигунів спричиняє нахил (roll), а зміна загального моменту всіх двигунів – обертання навколо вертикальної осі (yaw).

Управління реалізується через спеціальні алгоритми стабілізації, переважно на основі ПІД–регуляторів. Такий регулятор дозволяє формувати управляючий сигнал, який враховує поточну помилку, її інтегральну складову (накопичення похибки) і похідну (тенденцію зміни). Застосування ПІД–регуляторів забезпечує точну стабілізацію навіть у складних умовах.

Універсальність системи – особливістю універсального квадрокоптера є можливість зміни режимів роботи залежно від задачі:

– Ручне управління – оператор повністю керує польотом.

– Напівавтономний режим – утримання висоти або позиції із можливістю ручної корекції.

– Повністю автономний режим – політ за заданим маршрутом із автоматичним уникненням перешкод.

Також системи можуть доповнюватися камерами, лідаром, гіростабілізованими платформами, модулем комп’ютерного зору або нейронною мережею для розпізнавання об’єктів. Усе це значно розширює сферу застосування – від рятувальних операцій до моніторингу сільськогосподарських угідь.

Найпростіший набір систем керування включає:

1) процесор;
2) барометр для визначення висоти;
3) акселерометр;
4) гіроскоп;
5) навігаційний модуль;
6) оперативну пам’ять;
7) приймач сигналу.

Збір інформації для оператора і ПЗ самої машини надходить з датчиків різних типів. Використовуються лазерні, звукові, інфрачервоні і інші типи.

Зазвичай безпілотні літальні пристрої складаються з таких основних елементів:

– Акумуляторна батарея.
– Пристрій регулювання обертів двигуна.
– Пропелер.
– Двигун.
– Польотний контролер.
– Рама.

Основними функціями БПЛА слід вважати: моніторинг (спостереження); пошуково рятувальні операціїї; аерофотозйомку та картографію; технічну інспекцію, доставка вантажів; освітні процеси; наукові дослідження (табл.1.1).

Таблиця 1.1 – Функції та завдання квадрокоптеру

1.	Моніторинг та відео спостереження	Охорона територій (складів, об'єктів критичної інфраструктури, кордонів). Контроль сільськогосподарських угідь (стан посівів, зрошення, шкідники). Екологічний моніторинг (стан лісів, водойм, викиди у повітря).
2.	Пошуково рятувальні операції	Пошук людей у важкодоступних місцях (гори, ліси). Доставка аптечок, їжі, обладнання до зони НС. Передача відео з місця подій у режимі реального часу.

Кінець таблиці 1.1

3.	Аерофотозйомка та картографування	Створення тривимірних моделей рельєфу. Оцифрування місцевості для будівельних або інженерних проєктів. Фотограмметричне обстеження.
4.	Технічна інспекція	Перевірка стану ЛЕП, мостів, веж, дахів. Обстеження промислових об’єктів без необхідності зупинки виробництва.
5.	Наукові дослідження	Збір метеорологічних даних. Вивчення флори, фауни, кліматичних змін. Випробування нових алгоритмів автономного управління.
6.	Освітнє застосування	Використання у навчальному процесі з робототехніки, програмування та мехатроніки. Розробка студентами власних інноваційних рішень на базі відкритих платформ (Pixhawk, STM32 тощо).
7.	Доставка вантажів малої маси	Транспортування медикаментів, документів або невеликих посилок у межах міста або важкодоступних районів. Забезпечення оперативної доставки компонентів або інструментів на підприємствах чи під час аварійно–рятувальних робіт. Розвиток логістичних рішень для останньої милі (last–mile delivery), особливо в умовах урбанізації або нестачі інфраструктури.

Ефективне керування квадрокоптером неможливе без використання сучасних алгоритмів регулювання, здатних забезпечити стабільність, швидкодію та точність під час польоту.

Найбільш поширеним і перевіреним на практиці рішенням є використання ПІД-регуляторів (пропорційно-інтегрально-диференціальних регуляторів), які формують управляючі дії на основі поточної похибки між бажаним та фактичним станом системи.

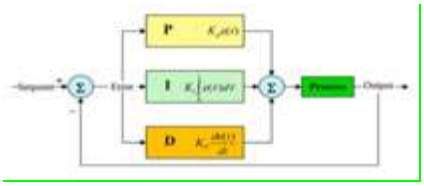


Рисунок 1.1 – Схема роботи ПІД-регулятора [1]

ПІД-регулятор виконує управління за трьома основними компонентами:

1. Пропорційна складова (P) – забезпечує управляючу дію, пропорційну поточній похибці:

$$P = K_p * e(t)$$

де $К_р$ - коефіцієнт пропорційності,

$e(t)$ - похибка між бажаним та фактичним значенням.

2. Інтегральна складова (I) - враховує накопичену похибку за певний час:

$$I = K_i * \int_0^t e(t)dT$$

Вона корисна для усунення систематичної похибки (наприклад, коли квадрокоптер постійно "сповзає").

3. Диференціальна складова (D) - реагує на швидкість зміни похибки:

$$D = K_d * \frac{de(t)}{dt}$$

Забезпечує демпфування та запобігає коливанням системи.

Загальна формула керування:

$$u(t) = K_p * e(t) + K_i * \int_0^t e(\tau)d\tau + K_d * \frac{de(t)}{dt}$$

Реалізація ПІД–регуляторів у квадрокоптері

У квадрокоптері ПІД-регулятори зазвичай реалізуються окремо для кожного ступеня свободи - Roll (нахил), Pitch (ківок), Yaw (поворот) та Altitude (висота). Таким чином, загальна система керування складається з декількох незалежних ПІД-контурів, кожен із яких керує відповідним каналом:

- Roll PID – регулює нахил вліво-вправо.
- Pitch PID – регулює нахил вперед-назад.
- Yaw PID – регулює поворот навколо вертикальної осі.
- Altitude PID – контролює тягу для стабілізації висоти.

Переваги використання ПІД–регулятора

- Простота реалізації – реалізація можлива навіть на мікроконтролерах з обмеженими ресурсами.
- Висока точність і стабільність – за правильного налаштування дозволяє точно утримувати квадрокоптер у повітрі.
- Гнучкість – легко адаптується до різних умов польоту шляхом корекції коефіцієнтів.

Налаштування ПІД–параметрів

Одним з найбільш важливих етапів є тюнінг (налаштування) коефіцієнтів K_p, K_i, K_d .

Існує кілька методів:

- Метод Зіглера-Ніколса – експериментальне визначення критичних параметрів.
- Ручне налаштування – поступова зміна одного параметра з оцінкою відповіді системи.
- Автоматизовані системи автотюнінгу – використовуються в просунутих прошивках (наприклад, Betaflight або ArduPilot).

При неправильному налаштуванні можливі:

- осциляції (занадто високий K_p),
- повільна реакція (занадто низький K_p),
- накопичення помилок (занадто великий K_i),
- надмірне демпфування (велике K_d).

Обмеження ПІД–регулювання

Незважаючи на переваги, ПІД–регулятори мають і певні обмеження:

- Нездатність до адаптації – не змінюють свою поведінку за зміни зовнішніх умов (наприклад, пориви вітру, зміна маси).
- Чутливість до шумів – диференціальна складова може підсилювати сенсорні перешкоди.
- Труднощі в нелінійних системах – при значних змінних режимах або нелінійній динаміці ПІД–регулятор може втрачати ефективність.

Тому останнім часом ПІД–регулятори часто поєднують із адаптивними або нечіткими методами, зокрема в автономних інтелектуальних системах.

1.2 Аналіз програмно-апаратних засобів реалізації системи керування квадрокоптером та формулювання технічних вимог та постановка задачі щодо вдосконалення системи

Фундаментом функціонування мультикоптера є польотний контролер – центральна плата управління. Його обчислювальним ядром виступає

мікроконтролер, який забезпечує виконання всіх необхідних операцій. Для цієї мети використовуються різні типи мікроконтролерів: від компактних Atmega328 до потужніших Atmega2560 або сучасних ARM-контролерів, зокрема (STM32)

До функцій польотного контролера належать:

– стабілізація апарата в повітрі;

– утримання висоти (за допомогою барометра) і позиції (за допомогою GPS);

– автоматичний політ за заданими точками (опціонально);

– передача на землю поточних параметрів польоту (через модем або Bluetooth);

– забезпечення безпеки польоту (повернення в точку зльоту при втраті сигналу, автопосадка);

– підключення додаткової периферії: OSD, світлодіодної індикації тощо.

Кількість доступних функцій залежить від наявності відповідної периферії на борту. У бюджетних контролерах деякі функції можуть бути відсутніми. У даному проєкті використовується контролер ArduPilot Mega – повноцінне рішення для БПЛА, що підтримує як ручне управління, так і автономний політ за заданим маршрутом із двосторонньою телеметрією та журналюванням даних.

Особливості ArduPilot Mega:

– 3-осьовий гіроскоп, акселерометр, магнітометр та високоточний барометр;

– система стабілізації з підтримкою повітряної акробатики;

– утримання позиції, політ за точками, повернення на старт;

– підтримка інфрачервоних та ультразвукових сенсорів;

– управління камерами, телеметрія, підтримка OSD, налаштування через Google Maps;

– мікроконтролери Atmel ATmega2560 та ATmega32U-2;

– підтримка оновлень прошивки, сумісність з PWM/PPM-приймачами тощо.

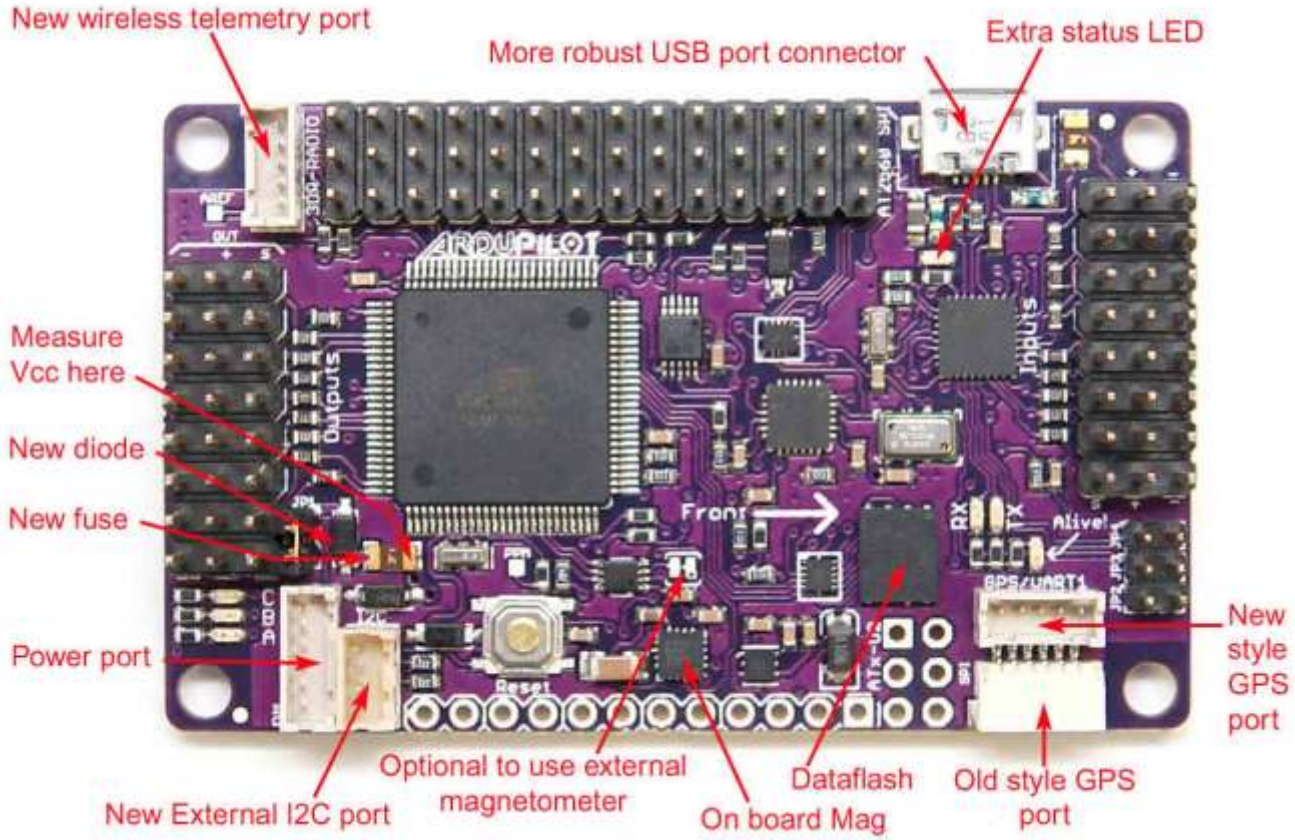


Рисунок 1.2 – Плата ARM 2.6[2]

Доступні режими польоту контролера:

– Stabilize – утримання горизонту;
– AltHold – утримання висоти;
– Loiter – зависання;
– RTL – повернення на старт;
– Auto – автономний політ;
– Acro – акробатика;
– Circle – політ по колу;
– Position – фіксація позиції;
– Land – посадка;
– Simple – спрощене управління.

Контролер підтримує програмування режимів польоту самостійно.

Фільтр Калмана – один з найбільш ефективних методів фільтрації даних, зокрема у робототехніці. Цей алгоритм використовує динамічну модель системи, відомі керуючі впливи та низку вимірювань для формування оптимальної оцінки стану. Він працює у два етапи: передбачення та корекція. На першому – розраховується прогнозований стан системи, а на другому – за новими даними він уточнюється.

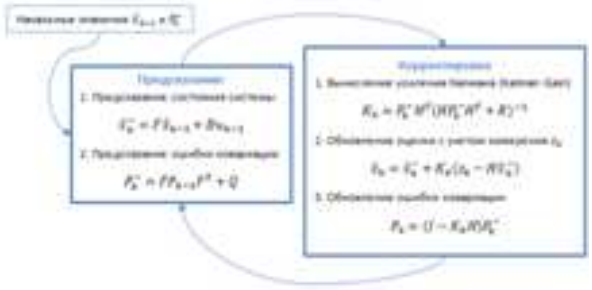


Рисунок 1.3 – Схема фільтра Калмана

Алгоритм застосовується для згладжування шумів, стабілізації траєкторії польоту та узгодження показів сенсорів. Його ефективність забезпечується математичним апаратом матричних рівнянь. У випадку однієї змінної матриці редукуються до скалярів.

Технології комп’ютерного зору – це методи та алгоритми, які дозволяють електронним пристроям виявляти, ідентифікувати та класифікувати об’єкти на зображеннях або відеопотоці. Це динамічно зростаюча галузь, яка базується на статистиці, обробці сигналів, оптимізації та машинному навчанні.

Головною складністю комп’ютерного зору є те, що машина не здатна інтуїтивно «бачити» – вона оперує лише цифровими значеннями. Саме тому комп’ютер потрібно «навчати» розпізнаванню форм, кольорів і текстур через програмні алгоритми. Завданням комп’ютерного зору є отримання описової інформації із зображення та формування висновків на її основі.

Таким чином, метою даної роботи є розробка, реалізація та тестування системи вдосконаленого керування квадрокоптером з використанням технології

комп’ютерного зору, що дозволить підвищити рівень автономності та надійності апарату.

Це передбачає:

- Вибір апаратної платформи для обробки відеопотоку в реальному часі.
- Інтеграцію камери з політним контролером через проміжне обчислювальне середовище.
- Розробку алгоритмів виявлення об’єктів, відстеження траєкторії, та реакції на зміну навколишнього середовища.
- Реалізацію експериментальної моделі та проведення тестових польотів.

Очікуваним результатом є створення прототипу квадрокоптера, який здатен орієнтуватися у просторі, виконувати завдання зі збирання даних, пошуку цілей або інспекції об’єктів без участі оператора, використовуючи виключно вбудовану камеру та алгоритми обробки зображення.

1.3 Висновки до першого розділу

У першому розділі було проведено ґрунтовний аналіз сучасного стану розробки систем керування квадрокоптерами, їх основних функцій, завдань, а також апаратно-програмних засобів, які використовуються в таких системах. Розглянуто структуру та ключові елементи мультикоптерів, зокрема центральну роль, яку відіграє політний контролер у забезпеченні стабільного, безпечного й автономного польоту. Встановлено, що сучасні контролери, зокрема ArduPilot Mega, забезпечують широку функціональність, включаючи підтримку автоматичного маршруту, передачу телеметрії, стабілізацію, обробку даних з численних сенсорів та підтримку різних режимів польоту.

Однак детальний аналіз існуючих рішень дозволив виявити певні обмеження традиційних систем керування, серед яких найбільш критичними є: надмірна залежність від GPS, обмежена здатність орієнтації в просторі без зовнішніх джерел навігації, відсутність активної взаємодії з динамічним середовищем (перешкодами,

рухомими об’єктами) та обмежена інтелектуальна адаптивність у нестандартних ситуаціях.

Ці недоліки обумовлюють актуальність і необхідність вдосконалення системи керування за рахунок впровадження новітніх технологій - насамперед, комп’ютерного зору, що відкриває нові можливості для автономної навігації, виявлення об’єктів, побудови карт місцевості та прийняття рішень на основі візуальної інформації. Аналіз апаратної складової також показав перспективність використання одноплатних комп’ютерів (таких як Raspberry Pi або Jetson Nano) у ролі окремого візуального процесора, що взаємодіє з основним контролером і розширює інтелектуальні можливості квадрокоптера.

У підсумку, сформульовано технічні вимоги до модернізованої системи керування, які включають можливість автономної навігації без GPS, підтримку камерного бачення, уникнення перешкод, взаємодію з наземними станціями, гнучкість налаштування та відповідність енергетичним і конструктивним обмеженням. Визначено також основні задачі для подальшої розробки системи, серед яких - вибір апаратної платформи, інтеграція сенсорів, розробка алгоритмів обробки зображення в реальному часі та реалізація прототипу.

Таким чином, проведений аналіз закладає наукове та технічне підґрунтя для практичної реалізації вдосконаленої системи керування квадроптером.

ПРОЄКТУВАННЯ СИСТЕМИ ОБРОБКИ ІНФОРМАЦІЇ У КІБЕРФІЗИЧНІЙ СИСТЕМІ АДАПТИВНОГО ЗАСТОСУВАННЯ МОНІТОРИНГОВИХ ЕЛЕМЕНТІВ РОЗВІДУВАЛЬНОГО БПЛА

2.1 Визначення апаратних і програмних підсистем програмно-технічного засобу.

При розробці та аналізі програмно-технічних комплексів, що базуються на комп'ютерному зорі**,** ключовим є чітке розмежування їхніх апаратних і програмних складових**.** Важливо розуміти, що дисципліна комп'ютерного зору не заглиблюється у тонкощі технологій створення цифрових зображень; її основним фокусом є виключно обробка вже існуючих візуальних даних**.**

Отже, без детального розгляду процесів формування зображень, ми зосередимося на сутності пристрою захоплення. Зазвичай це цифрова камера**,** яка за допомогою чутливих до світла елементів генерує окремі зображення (фотографії) або послідовності зображень (відео)**,** що відображають об'єкти реального світу.

З точки зору комп'ютера, будь-яке "зображення" є не що інше, як масив числових даних**,** організованих певним, заздалегідь визначеним чином. Саме цей числовий масив слугує вхідною інформацією, яка згодом піддається складному аналізу та інтерпретації за допомогою спеціалізованих алгоритмів і програмних підсистем комп'ютерного зору.

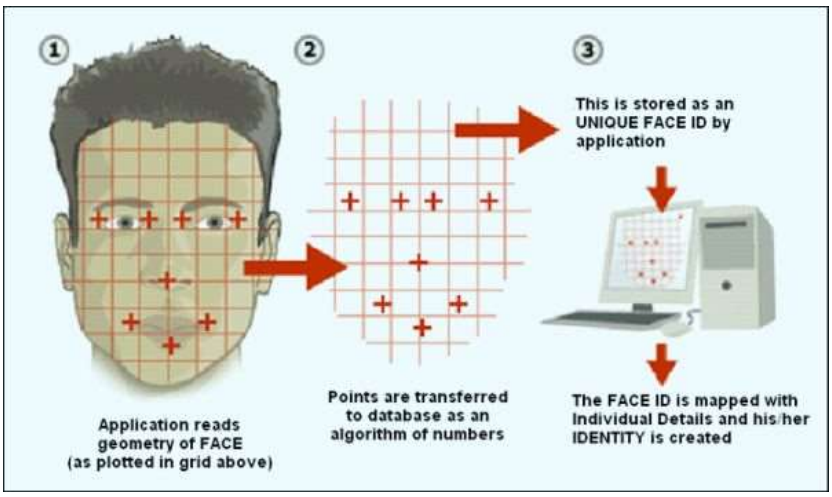


Рисунок 2.1 – Подання зорової інформації комп'ютером[3]

Кожне цифрове зображення можна уявити як двовимірний масив**,** де кожен елемент, або піксель**,** має своє унікальне положення та значення. Для півтонового зображення це значення визначає яскравість пікселя. У випадку кольорового зображення**,** кожен піксель кодується як мінімум трьома числовими значеннями, що зазвичай відповідають інтенсивності червоної, зеленої та синьої (RGB) складових.

Значна частина методів комп'ютерного зору оперує з бінарними зображеннями**,** які складаються виключно з чорних або білих пікселів. Зосередження на цих методах є цілком виправданим у даній роботі.

Алгоритми, що працюють з бінарними зображеннями, можуть виконувати широкий спектр завдань: від простого підрахунку певних ознак до складніших операцій, таких як розпізнавання об'єктів, визначення їхнього розташування та контроль**.**

Для перетворення напівтонового або кольорового зображення I у бінарне зображення B застосовується операція, що виділяє певну підмножину пікселів переднього плану (foreground pixels)**.** Ці пікселі є ключовими для аналізу

зображень, тоді як решта ігнорується як фонові (background pixels)**.** Процес відбору пікселів може бути як простим (наприклад, порогова операція**,** що вибирає пікселі за заданим діапазоном яскравості або колірного підпростору), так і складним алгоритмом класифікації**.**

У подальшому, ми будемо вважати, що бінарне зображення вже сформовано і розглядати його як вхідні дані для виконання операцій аналізу зображень. Концепція бінарного зображення наочно ілюструється на рис. 2.2, де представлені приклади чотирьох бінарних зображень рукописних символів.

Рисунок 2.2 – Бінарне зоображення рукописних символів.

У системах цифрової обробки зображень особливе місце займають бінарні (двоколірні) зображення. Кожен елемент такого зображення може набувати лише одного з двох станів:

Активний стан (1) - відповідає пікселям, що належать до об'єктів переднього плану. Неактивний стан (0) - позначає фонові області.Для роботи з такими зображеннями використовують матричне представлення розміром M×N, де:

– M - кількість рядків (нумерація від 0 до M-1)

– N - кількість стовпців (нумерація від 0 до N-1)

Доступ до конкретного пікселя здійснюється через індексацію B[r,c], де:

– B[0,0] - лівий верхній кут

– B[M-1,N-1] - правий нижній кут

При обробці зображень ключову роль відіграє аналіз оточення кожного пікселя. Розрізняють два основних типи околиць:

4-зв'язне оточення (N_4)

Включає безпосередніх сусідів по вертикалі та горизонталі:

– [r-1,c] - верхній

– [r+1,c] - нижній

– [r,c-1] - лівий

– [r,c+1] - правий

8-зв'зне оточення (N_8)

Крім основних сусідів, враховує діагональні пікселі:

– [r-1,c-1] - північно-західний

– [r-1,c+1] - північно-східний

– [r+1,c-1] - південно-західний

– [r+1,c+1] - південно-східний

Рисунок 2.3 – Аналіз сусідніх пікселів: методи 4-зв'язного та 8-зв'язного оточення

У сучасних алгоритмах обробки зображень поняття околиць пікселів (4-зв'язних чи 8-зв'язних) є фундаментальним інструментом для переходу від піксельного представлення до аналізу контурів. Цей підхід лежить в основі скелетних методів обробки, які дозволяють виділяти структурні особливості об'єктів.

Класифікація методів аналізу:

1. Скелетні методи:

– Використовують аналіз околиць пікселів

– Позбавляють об'єкти надлишкових деталей

– Зберігають топологічні властивості

2. Методи на основі 3D-моделювання:

– Враховують просторову структуру об'єкта

– Вимагають додаткових даних про глибину

– Застосовуються в комп'ютерній графіці та CAD-системах

3. Методи 2D-аналізу:

– Працюють із плоскими зображеннями

– Використовують інформацію про яскравість та колір

– Найбільш поширені у традиційній обробці зображень

Кожен з цих підходів має свої переваги та обмеження, а їх вибір залежить від конкретної задачі та наявних даних. Скелетні методи особливо ефективні для аналізу бінарних зображень, тоді як 3D-підходи краще підходять для роботи з об'ємними об'єктами.

Розглянемо детальніше скелетні (структурні) методи, які базуються на аналізі топологічних властивостей об'єктів. Основним елементом таких методів є контур - зовнішня границя об'єкта, яка містить ключову інформацію про його форму. Важливо відзначити, що при контурному аналізі враховуються лише крайові точки, тоді як внутрішня структура об'єкта ігнорується.

Обмеження контурного аналізу:

1. Проблеми виділення меж:

– Низький контраст між об'єктом і фоном

– Наявність шумів і перешкод

– Перекриття або нашарування об'єктів

2. Чутливість до артефактів:

– Часткова оклюзія об'єктів

– Нечіткі межі

– Складні форми з розривами

Незважаючи на ці обмеження, контурний підхід має істотні переваги:

– Значне зменшення обчислювальної складності

– Висока швидкість обробки

– Ефективність для чітко виражених об'єктів на контрастному тлі

Процес скелетизації включає:

1. Перетворення контуру до бінарного представлення
2. Послідовне утончення до отримання скелету
3. Кодування структури за допомогою:

– Опорних точок (ключових вузлів)

– Ланцюгових кодів (послідовності напрямків)
– Спеціальних маркерів:
– Кінцеві точки
– Точки розгалуження (трійникові вузли)
– Перехідні елементи

Ключові характеристики скелетного представлення:

– Зберігає топологічні властивості оригіналу
– Дозволяє компактно кодувати форму
– Чутливе до шумів, але ефективне для структурного аналізу
– Широко застосовується в системах розпізнавання образів

Цей метод особливо ефективний у поєднанні з попередньо описаними підходами до аналізу околиць пікселів, що дозволяє створювати гібридні алгоритми обробки зображень з урахуванням як локальних, так і глобальних характеристик об'єктів. На рис. 2.4 зображено образ, що володіє двома внутрішніми контурами, однією кінцевою точкою і трьома тріодами.

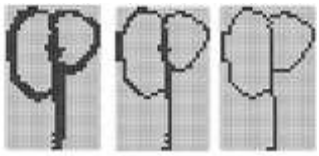


Рисунок 2.4 – Образ, який пройшов стерелізацію.

На етапі попередньої обробки здійснюється спрощення контурного опису шляхом: вилучення незначних фрагментів (коротких ліній); об'єднання сусідніх точок розгалуження; усунення дрібних внутрішніх контурів.

Кожен зовнішній контур отримує унікальний ідентифікатор у вигляді топологічного коду. Процес кодування передбачає: фіксацію послідовності ключових точок при русі за годинниковою стрілкою; стандартизацію нумерації цих точок; порівняння отриманої структури з базовими геометричними шаблонами.

Такий підхід дозволяє ефективно систематизувати контури за їх структурними особливостями, що значно спрощує подальший аналіз форми об'єктів.

2.2 Бібліотека та їх класи OpenCV

У даному розділі розглядається програмний інструментарій, що був використаний для реалізації проекту – бібліотека OpenCV (Open Source Computer Vision). Це провідне програмне рішення для завдань комп'ютерного зору, яке представляє собою набір спеціалізованих алгоритмів та функцій для обробки зображень. З технічні особливості OpenCV можна виділити первинну реалізацію на C/C++ з підтримкою кросплатформності та офіційні інтерфейси для популярних мов програмування (Python, Java, MATLAB тощо). Також має відкритий код з ліцензією BSD, що дозволяє використання як в наукових, так і комерційних цілях Вона є лідером серед бібліотек комп'ютерного зору (понад 5 млн інсталяцій) та використовується як референтна реалізація в індустрії. Включена в стандарт Khronos для комп'ютерного зору.

Бібліотека надає широкий спектр функціоналу – від базових операцій з обробки зображень до складних алгоритмів машинного навчання, що робить її ідеальним вибором для сучасних проектів у галузі комп'ютерного зору.

Модуль core становить фундамент OpenCV, надаючи базові структури даних та обчислювальні можливості. Він включає роботу з матрицями – багатовимірними масивами, що є фундаментальним поняттям в OpenCV. На відміну від суворої лінійної алгебри, матриці тут є більш абстрактними; їхні елементи можуть бути не лише числовими значеннями, але й об'єктами, зокрема з інших бібліотек. Також core забезпечує широкий спектр обчислювальних операцій, як-от математичні функції, генератори випадкових чисел, функції для дискретного перетворення Фур'є (DFT) та дискретного косинусного перетворення (DCT), а також можливості для введення/виведення даних у форматах XML та YAML.

Модуль imgproc є основним для всіх операцій з обробки зображень. Він містить функції для фільтрації зображень, застосування різноманітних фільтрів для згладжування, виділення контурів та шумозаглушення. Також він надає інструменти для геометричних перетворень (зміна розміру, обертання, масштабування) та перетворення колірних просторів, дозволяючи конвертувати зображення між різними моделями кольору, наприклад, RGB в HSV або сірі відтінки. Ключовою структурою цього модуля є Mat, що є базовим представленням зображень. Ці зображення можуть бути як чорно-білими, так і кольоровими (наприклад, 3-канальними), або навіть 4-канальними (RGB + альфа-канал для прозорості). Кожен канал може містити як цілі, так і дійсні значення, що робить цей тип даних універсальнішим за стандартні 8-бітові 3-канальні зображення. OpenCV надає великий арсенал операторів для маніпуляцій з цими зображеннями, дозволяючи змінювати їхні розміри, витягувати окремі канали та комбінувати зображення. Важливою відмінністю imgproc від Mat є поведінка покажчика imageData: тоді як у Mat дані є об'єднанням, що дозволяє гнучко задавати тип покажчика, для imgproc покажчик imageData жорстко визначений як uchar*. Це спрощує доступ до піксельних даних, оскільки зміщення можна використовувати "як є", тоді як при роботі з іншими матрицями може знадобитися коригування зміщення, якщо тип покажчика даних не byte.

Модуль HighGUI надає простий, але ефективний графічний інтерфейс користувача (UI), а також функції для введення та виведення зображень і відео. Ця бібліотека об'єднує функції OpenCV, що забезпечують взаємодію з операційною системою, файловою системою та апаратними засобами, такими як камери. HighGUI дозволяє відображати зображення, відкриваючи вікна для виводу візуальної інформації, працювати з файлами (читати та записувати графічні файли), обробляти прості події від миші, сенсорних пристроїв та клавіатури, а також створювати такі корисні елементи керування, як повзунки. Значною перевагою HighGUI є її кросплатформенність, що дозволяє розробляти додатки, сумісні з різними операційними системами. Структурно HighGUI поділяється на апаратну,

файлову та GUI частини. Апаратна частина передусім стосується роботи з камерами, надаючи прості механізми для підключення та отримання відеопотоку. Файлова частина забезпечує функції для завантаження та збереження зображень і відео, причому з уніфікованим підходом до обробки відеопотоку – однакові методи можуть використовуватися як для файлів, так і для потоку з камери. Аналогічна логіка реалізована для обробки зображень: функції автоматично визначають розширення файлу та обробляють операції кодування/декодування. GUI частина надає базові функції для створення вікон, відображення в них зображень та обробки подій, що надходять від миші та клавіатури.

Окрім перерахованих вище, OpenCV містить низку інших важливих модулів, що розширюють її функціонал. Модуль ML (Machine Learning) включає реалізації різних моделей машинного навчання, таких як опорні векторні машини (SVM), дерева рішень та навчання зі стимулюванням. Features2d призначений для розпізнавання та опису плоских примітивів (наприклад, за допомогою алгоритмів SURF, FAST) та надає спеціалізований фреймворк для роботи з ключовими точками. Модуль Video забезпечує функціонал для аналізу руху та відстеження об'єктів, включаючи розрахунок оптичного потоку, аналіз шаблонів руху та усунення фону. Objdetect (Object Detection) фокусується на виявленні об'єктів на зображенні, зокрема для пошуку облич (за алгоритмом Віоли-Джонса) та розпізнавання людей (з використанням HOG-дескрипторів). Нарешті, Calib3d (Camera Calibration and 3D Reconstruction) містить інструменти для калібрування камери, пошуку стерео-відповідності та інших елементів обробки тривимірних даних.

2.3 Розробка системи управління квадрокоптером на базі IMU та ультразвукового далекоміра HC-SR04.

Ультразвуковий далекомір (рис. 2.5) є важливою складовою нашої роботи, за допомогою нього – ми можемо визначити відстань до об’єкту.

Рисунок 2.5 – Ультразвуковий далекомір HC-SR04 [4]

Цей проект народився з бажання переосмислити традиційний підхід до керування безпілотниками. Сучасні пульти з їхніми джойстиками та безліччю кнопок, хоч і є стандартом, вимагають від оператора значних навичок та тривалого навчання. Навіть враховуючи вдосконалені системи стабілізації сучасних квадрокоптерів, процес керування залишається досить складним для освоєння.

Основна ідея нашої розробки - замінити класичний пульт управління на природні жести руки, зробивши взаємодію з дроном більш інтуїтивною та доступною. Замість того, щоб запам'ятовувати призначення численних кнопок та ручок, користувач зможе керувати апаратом за допомогою звичних рухів, схожих на ті, що ми використовуємо в повсякденному житті.

У фокусі дослідження - шість основних каналів керування, які включають як базові параметри польоту (газ, нишпорення, тангаж і крен), так і додаткові функції зміни режимів, реалізовані в популярній платформі Ardupilot. Особлива увага приділяється адаптації жестового інтерфейсу до існуючих апаратних рішень, що дозволить легко інтегрувати нову систему керування з сучасними безпілотниками без необхідності їх серйозної модифікації.

Такий підхід відкриває нові можливості для пілотування БПЛА, роблячи його доступнішим для широкого кола користувачів, а також знаходить перспективне

застосування в спеціалізованих сферах, де важлива швидкість та інтуїтивність управління.

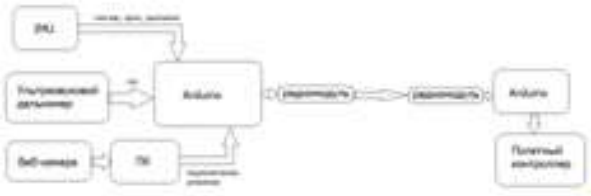


Рисунок 2.5 – Схема пристрою

Робоча схема пристрою базується на мікроконтролерній платформі Arduino з чипом AVR, яка виконує функції центрального обробника сигналів. Вхідні дані від різних сенсорів надходять на мікроконтролер, де проходять аналіз та після обробки передаються безпровідним зв'язком до польотової системи безпілотника.

Регулювання потужності двигунів здійснюється за допомогою ультразвукового датчика відстані – збільшення дистанції між рукою оператора та сенсором пропорційно підвищує швидкість обертання гвинтів. Для контролю основних параметрів польоту (нишпорення, тангажу та крену) використовується інерційний блок IMU, що інтегрує показання гіроскопа, акселерометра, магнітометра та барометричного датчика. Система відстежує орієнтацію пристрою в просторі за кутами Ейлера та передає ці дані для синхронізації з положенням коптера.

Перемикання режимів польоту реалізовано через візуальне розпізнавання жестів за допомогою комп'ютерного зору. Відеопотік з камери аналізується спеціальним програмним забезпеченням на Java, що забезпечує крос-платформну сумісність та можливість подальшого використання на мобільних пристроях.

На етапі прототипування як базовий контролер обрано плату Arduino з мікроконтролером ATmega2560, що значно прискорює процес розробки та тестування. Такий підхід дозволяє на початковій стадії використовувати готове апаратне рішення з можливістю подальшого переходу на спеціалізовану мікроконтролерну платформу для серійного виробництва.

У рамках розробки системи керування безпілотного літального апарата реалізовано обробку даних з чотирьох основних каналів управління: каналу тяги (газ), каналу повороту (нишпорення або yaw), каналу крену (roll) і каналу нахилу (pitch). Ці канали забезпечують основні можливості маневрування квадрокоптера в повітрі. Польотний контролер сприймає значення від кожного каналу у вигляді 8-бітних цілих чисел: для газу – в діапазоні 0–255, а для інших каналів – у межах від –127 до 127.

Щоб реалізувати обчислення та обробку цих значень на мікроконтролері, у програмному коді необхідно створити відповідні змінні для зберігання вхідних даних та значень, підготовлених до передачі або використання в алгоритмах керування. Для цього визначено змінні throttle, yaw, pitch, roll, що зберігають "сирі" дані з сенсорів, а також змінні sendThrottle, sendYaw, sendPitch, sendRoll, у яких містяться оброблені дані, придатні для керування моторами.

Одним із найважливіших параметрів для забезпечення стабільного польоту є значення тяги. Для його визначення застосовується ультразвуковий датчик HC-SR04. Це простий, недорогий пристрій, здатний точно вимірювати відстань до об’єкта за допомогою звукового імпульсу. Датчик має два сигнальні виводи: TRIG (тригер) і ECHO (відгук), які необхідно підключити до відповідних цифрових пінів Arduino. У даному випадку обрано пін №8 для TRIG та пін №9 для ECHO. Крім того, необхідно забезпечити живлення: 5В (Vcc) і GND.

Для зручності роботи з датчиком у середовищі розробки Arduino використовується бібліотека NewPing.h, яка дозволяє спростити процес вимірювання відстані, приховуючи реалізацію на низькому рівні. Після імпорту бібліотеки створюється об’єкт sonar типу NewPing, який ініціалізується з параметрами: пін тригера, пін відгуку та максимальна допустима дистанція, наприклад, 200 см. Метод sonar.ping() або sonar.ping_cm() дозволяє отримувати актуальне значення відстані до найближчої перешкоди.

Оскільки ультразвукові датчики схильні до значного рівня шуму – як через особливості середовища, так і через апаратні обмеження – доцільно впровадити

алгоритм фільтрації. У цьому проєкті реалізовано спрощену версію фільтра Калмана, що дає змогу стабілізувати показники, усуваючи випадкові флуктуації, але зберігаючи здатність реагувати на реальні зміни ситуації.

Функція спрощеного фільтра Калмана реалізована як окрема процедура, яка приймає поточне вимірювання та попереднє значення. Один із ключових параметрів – коефіцієнт адаптації (gain), підібраний емпіричним шляхом. Після серії експериментів визначено, що оптимальним є значення gain = 0.02, що забезпечує компроміс між чутливістю та стійкістю сигналу.

Також для налаштування фільтрації необхідно знати стандартне відхилення вхідного сигналу. Для цього було зібрано вибірку з понад 700 елементів за незмінних умов, після чого обчислено стандартне відхилення (σ) за допомогою табличного процесора Microsoft Excel, використовуючи відповідну статистичну функцію. Отримане значення зберігається в окремій константі в коді. Це дозволяє забезпечити стабільність роботи фільтра за умови повторних запусків або зміни зовнішніх параметрів.

Фільтрація впроваджується безпосередньо у функції loop, яка є головним циклом виконання програми на Arduino. Значення, отримане з датчика, передається до фільтра, результат записується назад у змінну throttle, після чого значення можна виводити на монітор порту для перевірки. З метою уникнення надмірного навантаження на мікроконтролер і перешарування імпульсів введено затримку у 5 мс після кожного циклу.

Аналогічний підхід застосовується до обробки даних з трьох інших каналів керування – нишпорення, крену та тангажу. Відповідні параметри отримуються з інерційного вимірювального модуля (IMU), до складу якого входять гіроскоп, акселерометр, барометр і компас. Пристрій підключається через інтерфейс I2C, що дозволяє передавати велику кількість даних лише по двох проводах – SDA та SCL. У Arduino Mega ці лінії відповідають піновим контактам №20 та №21.

Для забезпечення зв’язку з IMU необхідно імпортувати дві бібліотеки: Wire.h, яка надає базову реалізацію роботи з I2C, та TroykaIMU.h, яка містить

об’єкти та методи для роботи з конкретними сенсорами. У програмі створюються об’єкти класів Accelerometer, Gyroscope та Compass, відповідно accel, gyro і compass. Оскільки для фільтрації використовується алгоритм Madgwick, створюється ще один об’єкт типу MadgwickFilter, названий filter.

Інформація від кожного модуля вимірюється у трьох просторових осях, тому створюються змінні ax, ay, az (для акселерометра), gx, gy, gz (для гіроскопа), mx, my, mz (для компаса). Частота вибірок фільтра зберігається у змінній fps, яка ініціалізується зі значенням 100 та оновлюється під час виконання циклу.

Щоб забезпечити правильність роботи компаса, передбачено етап калібрування. Для цього у коді ініціалізуються дві структури: compassCalibrationBias – вектор з трьох елементів, і compassCalibrationMatrix – матриця 3×3. Значення цих структур заповнюються згідно з результатами попередньої калібрувальної процедури, які можуть бути отримані з технічної документації виробника або в процесі експериментального налаштування.

У функції setup відбувається ініціалізація всіх модулів. Кожен із об’єктів сенсорів викликає метод .begin(). Для компаса додатково викликається метод .calibrateMatrix(), що приймає відповідні матриці. Якщо всі пристрої були ініціалізовані без помилок, на монітор серійного порту виводиться повідомлення про успішне налаштування.

У функції loop, окрім обробки газу, реалізовано зчитування поточних значень акселерометра, гіроскопа та компаса, подальше передавання їх у фільтр Madgwick, та оновлення даних орієнтації. Для вимірювання fps зчитується час на початку та в кінці кожного циклу за допомогою функції millis(), після чого розраховується різниця часу у мілісекундах і на її основі – частота вибірки у герцах.

Таким чином, описана система дозволяє в реальному часі обробляти дані з сенсорів, фільтрувати шумові сигнали та формувати адекватні значення для каналів керування квадрокоптером. Реалізовані методи забезпечують необхідну чутливість до змін середовища, зберігаючи при цьому стійкість системи до випадкових коливань або помилкових зчитувань. У подальших етапах система може бути

розширена завдяки впровадженню більш складних моделей фільтрації, інтеграції з алгоритмами стабілізації або використанню машинного навчання для прогнозування руху та адаптивного керування.

Алгоритм роботи системи передбачає наступну послідовність дій. Спочатку відбувається збір даних з підключених сенсорів: акселерометр через метод readGXYZ записує показники прискорення у змінні ax, ay, az, гіроскоп за допомогою readRadPerSecXYZ фіксує кутові швидкості у gx, gy, gz, а компас із застосуванням readCalibrateGaussXYZ вимірює магнітне поле, зберігаючи результати у mx, my, mz.

Наступний етап передбачає обробку отриманої інформації за допомогою фільтруючого алгоритму. Спочатку ініціалізується процес оновлення коефіцієнтів фільтра через метод setKoeff, який приймає два параметри - частоту дискретизації fps та стабільний коефіцієнт BETA (стандартне значення 0.22, що може бути скориговане при необхідності). Ця константа визначається директивою #define на початку програми.

Після налаштування параметрів виконується оновлення вхідних даних фільтра методом update, що приймає значення з усіх трьох сенсорів. Завершальним етапом є отримання кінцевих результатів - метод getYawDeg повертає кут нишпорення, getPitchDeg визначає тангаж, а getRollDeg обчислює крен. Отримані значення записуються у відповідні змінні та можуть бути виведені через серійний порт для візуального контролю та аналізу роботи системи. Цей процес забезпечує точне визначення орієнтації пристрою в просторі за рахунок комплексної обробки даних від різних датчиків.

Використання фільтрації дозволяє мінімізувати похибки вимірювань та забезпечити стабільну роботу системи управління

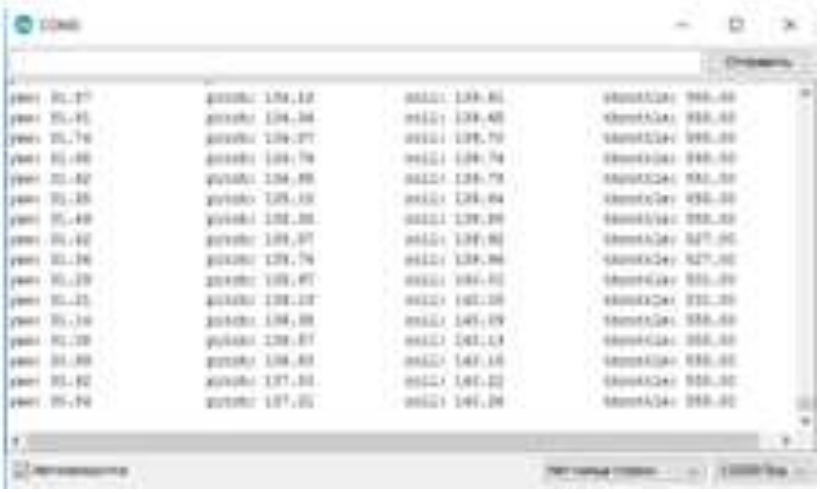


Рисунок 2.6 – данні з порту

Перед перетворенням даних у необхідний для польотного контролера вигляд, необхідно врахувати ще два аспекти. Перший - те, що протягом певного проміжку часу після запуску контролера датчики не можуть видавати правильні показання і відправляти такі дані на польотний контролер неприпустимо. Другий полягає в тому, що нишпорення має приймати значення від -127 до 127, де 0 - це положення рівноваги, при якому коригування нишпорення не здійснюється. Проблема полягає в тому, що нульове значення нишпорення з інерційного вимірювального пристрою виставляється за компасом і відповідає напрямку на північ. Природно, що в реальному пристрої не повинно бути прив'язки до сторін світла, а нульове значення має бути будь-яким зручним для користувача.

Для виправлення обох вищеописаних проблем додамо до схеми кнопку запуску. Вона повинна замикати виведення живлення 5В та один із логічних висновків. Оголосимо номер її виведення як константу з ім'ям BUTTON_PIN і функцією pinMode задамо виведення значення INPUT_PULLUP, тобто вхід з включенням резистора, що підтягує, вбудованого в плату Arduino. Створимо функцію, до якої включено різні типи відпрацювання натискань кнопок. У цьому коді будуть використовуватися тільки одинарні натискання із застосуванням захисту проти брязкоту контактів, але функція підтримує також подвійні натискання, натискання з утриманням і т.д. Всі ці режими можуть стати в нагоді

при подальшій розробці програми. Робота програми починається зі зчитування стану кнопки за допомогою функції digitalRead, результат якої зберігається у змінній button1S. Після виклику функції обробки кнопкових подій перевіряється факт одинарного натискання. Якщо такий зафіксовано, система виконує наступні дії: скидає прапорець натискання, змінює стан змінної check на протилежний та фіксує поточне значення кута нишпорення як нову нульову точку yawCenter. Для візуалізації режиму очікування перед виведенням даних додається спеціальний маркер у вигляді знаку оклику.

Дані з датчиків підлягають спеціальному форматуванню перед передачею. Значення газу нормалізуються до діапазону 0-255, а параметри тангажу та крену приводяться до інтервалу -127...+127 з урахуванням інверсії для каналу тангажу. Обробка нишпорення враховує зсув нульової точки yawCenter, при цьому кутові відхилення обмежені діапазоном +-40° через фізіологічні особливості руху руки оператора. Експериментально підтверджено, що позитивні значення відповідають повороту вліво, а негативні - повороту вправо, що забезпечує природність керування. Вся ця логіка реалізована з урахуванням оптимального балансу між точністю вимірювань та зручністю управління безпілотним апаратом.

ля перетворення вхідних даних сенсорів у керуючі сигнали для дрона необхідно визначити відповідні функції для кожного з чотирьох основних каналів: нишпорення (yaw), тангажу (pitch), крену (roll) та газу (throttle).

Для каналу нишпорення нам потрібна функція sendYaw, яка залежить від різниці поточного кута yaw і встановленої нульової точки yawCenter. Ця функція повинна відповідати наступним умовам:

- Якщо yaw−yawCenter=0, то sendYaw=0.
- Якщо yaw−yawCenter=40 (градусів), то sendYaw=-127.
- Якщо yaw−yawCenter=-40 (градусів), то sendYaw=127.

Шляхом розв'язання системи лінійних рівнянь, ми отримуємо функцію: sendYaw=-3.175*(yaw-yawCenter).

Після обчислення sendYaw, необхідно переконатися, що отримане значення знаходиться в допустимому діапазоні від -127 до 127. Для цього застосовуємо обмежувальні умови:

– Якщо sendYaw>127, то sendYaw прирівнюється до 127.

– Якщо sendYaw<-127, то sendYaw прирівнюється до -127.

Для каналу тангажу функція sendPitch повинна відображати наступні відповідності:

– Якщо pitch=180 або pitch=−180, то sendPitch=0.

– Якщо pitch=140, то sendPitch=−127.

– Якщо pitch=-140, то sendPitch=127.

Отримана функція sendPitch визначається кусково:

– sendPitch=3.175*pitch-571.5, якщо -180<=pitch<-140.

– sendPitch=3.175*pitch+571.5, якщо 140<pitch<=180.

– sendPitch=127, якщо 0>pitch≥−140.

– sendPitch=-127, якщо 0<=pitch<=140.

Для каналу крену функція sendRoll повинна задовольняти аналогічні умови:

– Якщо roll=180 або roll=−180, то sendRoll=0.

– Якщо roll= -140, то sendRoll=-127.

– Якщо roll=140, то sendRoll=127.

Отримана функція sendRoll також є кусковою:

– sendRoll=-3.175*roll-571.5, якщо -180<roll<-140.

– sendRoll=−3.175*roll+571.5, якщо 140<roll<=180.

– sendRoll=127, якщо 0<roll<140.

– sendRoll=-127, якщо 0>roll>=140.

Для каналу газу необхідно визначити діапазон вхідних значень від датчика. Оскільки датчик не видає нульового значення, мінімальна точка throttle встановлюється на 400, а максимальна на 2000. Тоді функція sendThrottle має такі відповідності:

– Якщо throttle=400, то sendThrottle=0.

– Якщо throttle=2000, то sendThrottle=255.

Отримана функція sendThrottle є кусковою:

– sendThrottle=0.159375*throttle−63.75, якщо 400<throttle<2000.

– sendThrottle=0, якщо throttle<=400.

– sendThrottle=255, якщо throttle>=2000.

Після виконання цих перетворень дані готові до передачі на бортовий контролер. Наступним кроком буде реалізація програми для управління п'ятим і шостим каналами, які відповідають за режими польоту.

2.4 Проєктування ПЗ для розпізнавання жестів руки з відеосигналу за допомогою бібліотек OpenCV.

У цьому розділі сформульовано мету – розробити програмне забезпечення для розпізнавання жестів руки. Такий додаток може функціонувати як складова частина системи керування безпілотним літальним апаратом, виконуючи роль модуля взаємодії з користувачем. Виведені з програми результати можуть ініціювати запуск окремих функцій або активувати певні режими роботи дрону.

У системах керування дронами, заснованих на мікроконтролерах, команди для перемикання режимів передаються через окремі канали зв’язку. Ці канали, у свою чергу, об’єднуються за допомогою методів часової мультиплексії. Таким чином, загальний керуючий сигнал формується з кількох імпульсів, які передаються у часовій послідовності та мають широтно-імпульсну модуляцію. Першими зазвичай ідуть імпульси, що визначають рівень напруги для роботи двигунів, а за ними – сигнали, що позначають активність окремих функціональних режимів польоту.

Створене програмне забезпечення повинно мати можливість ідентифікувати жести, які демонструє користувач, повертати відповідні значення у вигляді кодів, що відповідають визначеним жестам, та передавати їх до основного програмного модуля, який аналізує дані з сенсорів і генерує сигнали для керування

квадрокоптером. Виконання цього завдання вимагає суттєвих обчислювальних ресурсів, оскільки аналіз відеозаписів з камери базується на складних алгоритмах комп’ютерного зору. Через це мікроконтролери сімейства AVR не здатні впоратися з обробкою подібного рівня складності, тоді як персональні комп’ютери через свою громіздкість не підходять для мобільного використання.

Оптимальним рішенням у цьому контексті є використання сучасного смартфона на платформі Android. Це дозволяє створити додаток, що здійснює відеозахоплення з камери мобільного пристрою, проводить попередню обробку зображення, розпізнає жести та передає отримані результати через Bluetooth-з’єднання до основного керуючого блоку квадрокоптера.

На початковому етапі були розглянуті такі варіанти:

– Python з OpenCV - найпопулярніший варіант для комп'ютерного зору, має широкі можливості, простий у використанні. Проте має низьку продуктивність на мобільних пристроях та ускладнену інтеграцію з Android. Крім того, Python інтерпретується, а не компілюється, що ускладнює створення ефективного мобільного застосунку.

– C++ з OpenCV - дає високу продуктивність, однак потребує значного часу на розробку і складне середовище для мобільної компіляції. Також має високу вартість підтримки.

– JavaFX (Java) - сучасна кросплатформна бібліотека для створення графічних інтерфейсів на Java. Має зрозумілу архітектуру MVC (Model-View-Controller), підтримує роботу з відео та зображенням, легко масштабується, а також дозволяє безболісно перенести код на Android (з деякими адаптаціями).

– Android (Java/Kotlin) - оптимальний варіант для кінцевої версії застосунку, однак незручний для швидкого тестування під час розробки.

Для реалізації програмного забезпечення було обрано мову програмування Java, яка є стандартом у сфері розробки застосунків для платформи Android. На початковому етапі створено десктопну версію програми з графічним інтерфейсом, що працює на базі JavaFX під операційною системою Windows. Такий підхід

дозволяє зручно відлагоджувати функціонал та проводити тестування. У подальшому перенесення розробки на мобільну платформу Android не становитиме суттєвих труднощів, оскільки Java підтримує кросплатформенність.

Перша реалізація має базовий функціонал – визначення кількості зігнутих пальців та оцінку конфігурації руки (розчепірені або зведені пальці). Побудова програмної логіки здійснюється поетапно. Першим кроком є захоплення відеосигналу з камери та виділення окремого кадру з потоку. Усі подальші обчислення виконуються на основі цього кадру, після чого процес повторюється.

Наступним етапом є попередня обробка зображення. Вона спрямована на те, щоб спростити процес аналізу: зменшити роздільну здатність, знизити кількість кольорів, а за потреби – здійснити перетворення зображення в бінарний формат. Це обґрунтовано тим, що в подальшому застосовуються алгоритми, орієнтовані саме на двоколірні зображення.

На наступному кроці програма за допомогою методів комп’ютерного зору виявляє контури об'єкта та виконує аналіз їхніх особливостей – в тому числі, знаходить так звані дефекти контурів. Після цього потрібно виявити ключові точки (екстремальні координати), які несуть основну інформацію про форму руки. На базі положення цих точок програма виконує математичні обчислення та робить висновок щодо того, який саме жест демонструється. Усі етапи повинні супроводжуватись графічним відображенням інформації на екрані для зручності користувача. Після завершення циклу розпочинається обробка наступного кадру.

Для розробки було створено проєкт у середовищі IntelliJ IDEA з використанням JavaFX – технології, призначеної для створення графічних інтерфейсів на Java. Оформлення інтерфейсу реалізується за допомогою FXML-файлу – це спеціалізований XML-формат, який описує структуру GUI, положення елементів, їх стиль та інші властивості. Взаємодія між FXML та основним кодом програми здійснюється через спеціальний клас-контролер controller.java, який забезпечує логічне зв’язування графічних компонентів з функціональністю.

У процесі практичного використання такого ПЗ, що в реальному часі працює з відеопотоком з камери та аналізує зображення руки, можлива поява різного роду помилок і збоїв. Це обумовлено складністю алгоритмів та високими вимогами до обчислювальних ресурсів.

Рисунок 2.8 – блок схема процесу обробки відео

Надійність системи визначається не лише її здатністю правильно інтерпретувати жести, але й тим, як вона реагує на помилки, зовнішні збурення, нестабільні умови зйомки або некоректне введення.

Типові помилки під час роботи системи

– Відсутність камери або неможливість її ініціалізації

Система повинна перевіряти наявність підключеної камери та забезпечувати відповідне повідомлення користувачу у випадку помилки. Типова ситуація: веб-камера відключена або використовується іншою програмою.

Рішення: Виведення повідомлення: "Камера не виявлена. Перевірте підключення". Повторна спроба підключення кожні 5 секунд. Перехід у "режим очікування" з мінімальним навантаженням на систему.

– Надто темне або засвічене зображення

Результати бінаризації можуть бути некоректними, якщо кадр дуже темний або яскравий. Це призводить до втрати контурів або шуму.

Рішення: Використання адаптивної порогової фільтрації (adaptiveThreshold), додатковий аналіз яскравості сцени (розрахунок середньої яскравості), автоматичне коригування яскравості, підказка користувачу змінити умови освітлення.

– Відсутність руки в кадрі або сторонні об’єкти

Система може обробити шум або сторонній об’єкт як "руку", що призведе до хибного жесту.

Рішення: Введення фільтра на розмір об’єкта (мінімальна та максимальна площа),обов’язкова перевірка кількості пальців – якщо їх надто багато або надто мало, виводиться повідомлення "Жест не розпізнано".

– Нестабільна камера або рух під час зйомки

Якщо користувач трясе камерою або рука швидко рухається, алгоритм не встигає проаналізувати кадр.

Рішення: Введення фіксації кадру – жест повинен бути стабільним протягом 1–2 кадрів поспіль для підтвердження, можливість ручного "захоплення" кадру (натиснення кнопки).

– Неправильне розпізнавання жесту

Може бути спричинено неуніфікованим положенням руки або помилкою аналізу.

Рішення: Виведення на екран візуалізації ключових точок, щоб користувач міг сам оцінити розпізнавання, реалізація механізму "перепідтвердження" – жест вважається дійсним, якщо розпізнано 2 рази поспіль.

Загальний підхід до обробки винятків

Усі модулі програми обгорнуті в конструкції try-catch, що дозволяє обробляти виняткові ситуації без аварійного завершення. Крім того, реалізовано логування подій, що допомагає розробникам відстежувати, в який момент виникла помилка і за яких умов.

У майбутньому можливе додавання модуля автоматичної адаптації параметрів в залежності від поточних умов кадру (освітлення, шум), що значно підвищить стійкість програми до збурень зовнішнього середовища.

2.5 Висновок

У другому розділі було здійснено детальне проєктування інтелектуальної підсистеми обробки візуальної інформації, яка є невід’ємною частиною кіберфізичної системи керування універсального квадрокоптера. Розглядалося як

апаратне, так і програмне забезпечення, необхідне для реалізації системи, орієнтованої на інтерпретацію візуальних жестів оператора.

Було проаналізовано основні компоненти програмно-апаратного комплексу, обґрунтовано вибір програмного забезпечення, зокрема використання мови Java та бібліотеки OpenCV, що забезпечує потужні засоби комп'ютерного зору для попередньої обробки зображення, виділення контурів, пошуку екстремальних точок та аналізу жестів руки.

Значну увагу приділено можливості подальшої інтеграції розробленої підсистеми з модулями керування квадрокоптером, що функціонують на базі датчиків IMU та ультразвукових далекомірів. Це дозволяє підвищити автономність, гнучкість і варіативність способів керування дроном, зокрема завдяки інтерпретації жестових команд користувача.

На завершальному етапі було здійснено проєктування окремого програмного модуля для розпізнавання жестів руки за відеосигналом, що реалізується з використанням алгоритмів OpenCV. Розробка такого модуля дозволяє спростити передачу команд та підвищити зручність взаємодії з безпілотним літальним апаратом.

Отже, результати розділу демонструють, що запропонований підхід є перспективним для побудови інтелектуальних систем візуального управління квадрокоптером, забезпечуючи ефективну інтеграцію комп'ютерного зору в структуру кіберфізичної системи керування.

ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ КІБЕРФІЗИЧНОЇ СИСТЕМИ АДАПТИВНОГО ЗАСТОСУВАННЯ МОНІТОРИНГОВИХ ЕЛЕМЕНТІВ РОЗВІДУВАЛЬНОГО БПЛА

3.1 Опис реалізації модулів апаратного та програмного забезпечення програмно-технічного засобу

На початковому етапі розробки програмного забезпечення ключовим завданням стало забезпечення трансляції відео з камери безпосередньо на екран користувача. Хоча це можна реалізувати й без сторонніх інструментів, для розширення функціональності було прийнято рішення відразу застосувати бібліотеку OpenCV, яка значно спрощує роботу з відеопотоком.

Початкове налаштування програми виконується у методі start класу Main. Тут ініціалізується компонент FXMLLoader, що відповідає за завантаження графічного інтерфейсу, визначеного у відповідному .fxml файлі. Завантажений шаблон інтерфейсу пов’язується з контролером, через який здійснюється логіка взаємодії між компонентами.

Далі створюється головне вікно за допомогою об’єктів Stage і Scene. У процесі налаштовуються його основні параметри – назва, розміри, можливість масштабування. Коли всі елементи підготовлені, метод show() відкриває вікно для взаємодії з користувачем.

Основна логіка керування розміщується у класі Controller, де ініціалізуються графічні елементи, зокрема кнопка для запуску або зупинки камери (Button), поле для відображення відео (ImageView), а також таймер (ScheduledExecutorService), що забезпечує регулярне оновлення зображення. Відеозахоплення реалізується через об'єкт VideoCapture з бібліотеки OpenCV, який забезпечує зчитування даних з фізичної камери.

Для обробки одного кадру з потоку передбачено метод grabFrame, який повертає об’єкт типу Mat – це спеціальний клас OpenCV, що містить зображення. Спочатку відбувається перевірка активності підключення (this.capture.isOpened()),

і лише за наявності сигналу метод read() записує зображення в об'єкт frame. У разі, якщо кадр справді містить дані, можна застосовувати алгоритми обробки. Інакше кадр повертається без змін.

Взаємодія користувача з камерою реалізується через подієвий обробник – метод startCamera, що активується при натисканні відповідної кнопки. Після запуску відбувається відкриття потоку (capture.open(cameraId)), а подальше оновлення кадрів здійснюється із фіксованим інтервалом у 33 мілісекунди, що відповідає 30 кадрам на секунду. Отримане зображення конвертується у формат, придатний для виводу на екран (Image), після чого з'являється у вікні. Метод також відповідає за повідомлення про помилки, зміну тексту кнопки та правильне завершення сесії.

Завершальним кроком є створення графічного інтерфейсу користувача. Це можна зробити вручну у .fxml файлі або за допомогою SceneBuilder. В макеті використовуються елементи GridPane і BorderPane для гнучкого розташування елементів. Основними компонентами є поле зображення (ImageView) та кнопка керування (Button), які забезпечують взаємодію з користувачем і запуск відеопотоку.

Рисунок 3.1 – Інтерфейс програми SceneBuilder

Представлена на Рис. 3.2 програма - результат на даний момент. Поки що вона вміє лише захоплювати відео з камери і виводити його на екран. Так само є кнопка Start/Stop Camera, яка відповідно вмикає і вимикає камеру.

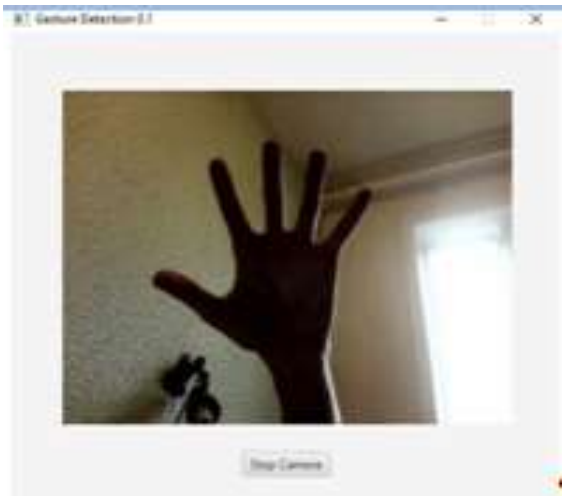


Рисунок 3.2 – Програма

3.2 Обробка зображення

Перед тим як розпочати реалізацію алгоритму аналізу візуальної інформації, одним із ключових етапів попередньої обробки зображення є його редагування. Це обумовлено тим, що сирове зображення, отримане з камери, у більшості випадків не є придатним для безпосередньої обробки алгоритмами комп'ютерного зору через наявність зайвих даних, шумів, складної колірної палітри або недостатньої контрастності.

Одним із найпоширеніших і найбільш ефективних підходів у таких випадках є спрощення зображення до його базових складових - контрасту між об'єктом та фоном. Це дає змогу не лише зменшити навантаження на систему обробки, а й підвищити точність розпізнавання. Для цього зображення переважно переводиться з повноколірного (RGB) формату у двоколірний або бінарний формат. У такому форматі кожен піксель зображення набуває одного з двох можливих значень - чорного або білого, що спрощує логіку подальшої обробки.

Колірна інформація, попри свою візуальну привабливість, часто не несе корисного навантаження для задач, пов'язаних із виявленням контурів, обрисів або положення об'єкта в кадрі. Більше того, в деяких випадках кольори можуть дезорієнтувати алгоритм або створити хибні зони розпізнавання через відблиски,

тіні або зміну освітлення. У зв’язку з цим, на першому етапі попередньої обробки доцільно позбутися кольору, сконцентрувавшись виключно на світлості та темності окремих ділянок зображення.

У бібліотеці OpenCV, яка використовується для комп’ютерного зору, багато методів передбачають саме двокольорове зображення як вхідний параметр. Тобто робота з такими функціями вимагає, щоб пікселі були представлені лише у вигляді "чорне-біле", без градацій сірого. Це не тільки технічна вимога, а й оптимізаційний прийом, який дозволяє суттєво пришвидшити обробку.

Одним із традиційних підходів до бінаризації є семплювання кольору, тобто виділення певного відтінку як ключового (наприклад, колір шкіри чи одягу людини) та перетворення всіх схожих відтінків на білий, а всіх інших - на чорний. Однак для даного проєкту, де передбачається розпізнавання темного об’єкта на світлому фоні (наприклад, руки користувача на тлі стіни чи неба), ця процедура є надмірною. У такому випадку можна використати прості методи порогової обробки, які ґрунтуються на яскравості пікселів: пікселі нижче певного порогу стають чорними, а ті, що перевищують його - білими.

У процесі реалізації алгоритму обробки зображення за основу береться кадр відео - об’єкт типу Mat, що є основною структурою в OpenCV для зберігання зображень. Цей об’єкт зберігає матрицю пікселів, які складають кадр, що надходить із відеопотоку. У межах створеної системи обробки візуальної інформації потрібно сформувати окремий метод, який буде виконувати всі необхідні перетворення над цим об’єктом.

У згаданому методі - умовно назвемо його imageEdit - виконуються всі необхідні кроки для перетворення повноколірного зображення у придатну до подальшої обробки форму. Цей процес включає в себе кілька етапів, таких як зниження колірного простору, фільтрація шумів, застосування порогових фільтрів і переведення до бінарної маски. Результат цих дій - це зображення, яке чітко позначає контури об’єкта, що розпізнається, і значно спрощує наступні кроки, пов’язані з аналізом його положення, руху або форми.

Головна мета цих перетворень полягає в тому, щоб забезпечити максимально точне відображення геометрії об’єкта в кадрі при мінімальному впливі зовнішніх факторів. Наприклад, тло має бути ідеально однорідним - повністю білим або чорним, - щоб не заважати і не створювати хибних контурів. У свою чергу, сам об’єкт має бути чітко виділений контрастним кольором, завдяки чому алгоритм зможе точно визначити його координати та форму.

Таким чином, попередня обробка зображення є важливим проміжним етапом, що значно впливає на ефективність роботи всієї системи. Вона дає змогу полегшити подальший аналіз, зменшити обсяг необхідних обчислень і підвищити загальну точність розпізнавання. Виконання цих кроків на початку обробки є запорукою стабільної та ефективної роботи системи, особливо у реальному часі, де наявність затримок або помилок є критичною.

Першим етапом обробки кольорового зображення є його конвертація у відтінки сірого, що дозволяє зменшити складність подальшої обробки, позбавляючи зайвих кольорових каналів. Для цього застосовуємо функцію Imgproc.cvtColor з бібліотеки OpenCV. Цей метод вимагає три параметри: вхідний об’єкт зображення, об’єкт для збереження результату та тип перетворення. В нашій реалізації використовуємо один і той самий об’єкт frame і конвертуємо зображення за схемою COLOR_BGR2GRAY.

Після отримання монохромного зображення потрібно здійснити його перетворення у бінарний формат. Для цього звертаємося до методу Imgproc.threshold, який виконує порогове відсікання яскравості пікселів. Метод приймає п’ять аргументів: джерело, об’єкт для результату, порогове значення, максимальне значення пікселя та тип порогової операції. В нашому випадку вибираємо тип, що відповідає інверсному порогу, де пікселі із яскравістю нижчою за поріг набувають білого кольору, а решта – чорного.

Оскільки фіксоване порогове значення не завжди відповідає змінним умовам освітлення, для динамічного налаштування параметра порогу реалізуємо графічний елемент – слайдер, що дозволяє користувачу регулювати межу безпосередньо під

час виконання програми. Значення слайдера отримуємо за допомогою методу getValue() і передаємо у відповідний параметр функції threshold, що робить процес більш гнучким і адаптивним.

З метою зменшення шумів та небажаних артефактів на бінарному зображенні застосовуємо медіанний фільтр, реалізований через метод Imgproc.medianBlur. Цей алгоритм замінює кожен піксель на медіану значень у його околі, що ефективно усуває дрібні перешкоди. Для тонкого налаштування ступеня розмиття також створюємо окремий слайдер, який дозволяє експериментально підібрати оптимальні параметри для конкретного зображення та підвищити якість подальшої обробки.

Таким чином, послідовність операцій – від конвертації у відтінки сірого, через порогове бінаризацію до медіанного розмиття – формує базову схему підготовки зображення для подальшого аналізу.

Рисунок 3.3 – Вигляд програми після редагування зображення

Після досягнення якісного бінарного зображення руки можна перейти до його подальшого аналізу. На даному етапі комп'ютер сприймає це зображення як двовимірний булевий масив, де чорні пікселі відповідають значенню true, а білі — false. Основним завданням є витяг корисної інформації з цього масиву і прийняття рішення на її основі. Алгоритм розділений на дві ключові складові: перша — це структуризація даних, що містяться у зображенні, шляхом формування спеціалізованої структури даних; друга — зіставлення цієї структури з наборами фіксованих параметрів, заданих розробником. Якщо відбувається співпадіння між

обробленими даними і однією з констант, система повертає відповідний результат, інакше — виводить повідомлення про помилку або відсутність необхідного жесту.

Найпростіша реалізація ігнорує перший етап і безпосередньо порівнює вхідний двійковий масив із заздалегідь визначеними шаблонами жестів. Цей підхід фактично базується на кореляції — кожен піксель нового зображення множиться на відповідний піксель шаблону, після чого всі добутки сумуються. За отриманим значенням можна оцінити ступінь подібності. Такий метод є простим у реалізації, але надзвичайно чутливий до змін масштабу, положення руки та індивідуальних особливостей користувача, що призводить до значних похибок. Тому в даному проекті було прийнято рішення перейти до аналізу контурів.

Важливо визначити, які саме характеристики зображення слід витягти для подальшої обробки. Для початку слід зосередитись на простих жестах, коли долоня орієнтована перпендикулярно камері, і для аналізу використовувати довжини пальців (визначення їх згину) та кути між ними. Для цього потрібно виявити кілька ключових точок — екстремумів, серед яких кінчики пальців і перетинки між ними. На основі цих точок створюються два контури: внутрішній, який точно повторює форму руки, і зовнішній, що є багатокутником, який з’єднує ці ключові точки. Внутрішній контур відповідає контуру руки, а зовнішній — його обвідному контуру (convexHull). Екстремальні точки пальців — це вершини обвідного контуру, а точки між пальцями визначаються як максимальні відхилення внутрішнього контуру від зовнішнього.

Для реалізації обробки необхідно знайти і намалювати обидва типи контурів. В програмному коді створюється список контурів у вигляді LinkedList<MatOfPoint>, де MatOfPoint — це структура, що зберігає множину точок на двовимірній площині зображення. Кожен контур являє собою масив точок, що описують замкнену криву. Спочатку очищуємо цей список методом clear(), після чого викликаємо функцію findContours з бібліотеки Imgproc, яка виділяє контури на основі кольорових переходів і записує їх у список.

Алгоритм пошуку контурів налаштовується двома параметрами: перший (RETR_EXTERNAL) інструктує програму ігнорувати внутрішні контури, виділяючи лише зовнішні контури об’єктів, другий (CHAIN_APPROX_TC89_KCOS) визначає спосіб апроксимації контуру, обраний експериментально як оптимальний.

Далі серед усіх знайдених контурів вибираємо той, що має найбільшу площу — вважаємо його основним об'єктом інтересу. Для цього створюємо змінну finalContours типу MatOfPoint і записуємо в неї перший контур зі списку, а потім ітеруємо по всіх контурах, порівнюючи їхні площі за допомогою функції contourArea. Якщо площа поточного контуру більша за площу finalContours, оновлюємо finalContours. Таким чином, отримуємо контур найбільшого об’єкта для подальшого опрацювання.

Для візуалізації контурів на зображенні використовується метод drawContours із відповідними параметрами: зображення, масив контурів, колір ліній і товщина.

Визначення обвідного контуру (convex hull) є складнішою операцією. OpenCV працює з різними типами даних: контури зберігаються у MatOfPoint, що містить координати точок, а convexHull повертає індекси крайніх точок у вигляді MatOfInt. Для перетворення індексів у координати створюється масив точок hullPoints. За допомогою циклу для кожного індексу з convexHull вибирається відповідна точка з внутрішнього контуру і записується у hullPoints. Після цього масив hullPoints конвертується у MatOfPoint hullMOP, який використовується для подальшого відображення обвідного контуру.

Для знаходження обвідного контуру викликається метод convexHull, який приймає у параметрах внутрішній контур і порожній масив для заповнення. Після трансформації у MatOfPoint цей контур можна вивести на екран за допомогою drawContours, застосувавши відмінний колір і товщину ліній для кращої візуалізації.

Таким чином, у результаті роботи програми ми отримуємо чітке відображення основного контуру руки і його обвідного контуру, які служать базою для подальшого розпізнавання жестів.

Рисунок 3.4 – Програма після виділення контурів

Далі здійснимо ідентифікацію так званих «дефектів» у межах обвідного контуру. Для цього ініціалізуємо нову структуру даних – масив об’єктів типу MatOfInt4, який позначимо як convDef. Вказаний тип даних має особливість групувати інформацію по чотири цілі значення (Integer) у кожному елементі. Така організація пов’язана з тим, що кожен дефект, визначений алгоритмом, кодується чотирма параметрами: точкою початку (start), точкою завершення (end), власне дефектом (defect) та глибиною (depth) – величиною, що характеризує відстань від контуру до дефектної ділянки.

Рисунок 3.5 – Схематичне зображення значень, методу Imgproc.convexityDefects

Для повного опису в рамках цього проєкту достатньо врахувати лише три типи координат: початкову точку (start), кінцеву точку (end) та точку дефекту (defect). Для зручності створимо окремий одновимірний список цілих чисел (Integer), який назвемо cdList. За допомогою функції Imgproc.convexityDefects отримаємо набір дефектів обвідного контуру. Цей метод вимагає три параметри: внутрішній контур, контур-обвідник і об'єкт для збереження інформації – convDef. Формат MatOfInt4, у якому повертаються дані, ускладнює їхнє подальше використання, тому для зручності перетворимо цю інформацію у звичайний список через convDef.get(0).toList.

Наступним кроком створимо масив cdList, заповнивши його послідовністю цілих чисел від 0 до кількості дефектів, помноженої на 4 (оскільки кожен дефект описується чотирма числами). Організація даних у цьому масиві така: індекс 0 містить початок першого дефекту, 1 – його кінець, 2 – координату дефекту, 3 – глибину дефекту, 4 – початок другого дефекту і так далі.

Далі формуємо три окремі масиви типу Point, у яких по черзі зберігаються всі точки початку, кінця та самі дефектні точки. Залишилось лише реалізувати цикл, який за допомогою функції Imgproc.circle візуалізує кожну точку на екрані, що дозволить переконатися в коректності обробки.

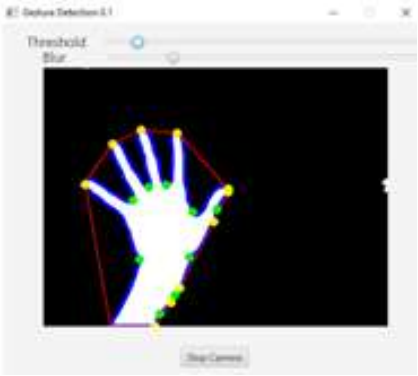


Рисунок 3.6 – Вигляд програми після пошуку екстремальних точок

Перед початком аналізу знайдених точок необхідно провести їхню попередню очистку, адже не всі виявлені дані є релевантними. По-перше, слід

відфільтрувати точки defect, які утворюють надто широкий кут разом із сусідніми точками start та end. Адже, наприклад, максимально можливий кут між вказівним і великим пальцем не може перевищувати π/2, тому більші кути не несуть корисної інформації щодо положення пальців. По-друге, варто виключити точки start, розташовані занадто близько одна до одної, оскільки алгоритм іноді неправильно розпізнає кілька сусідніх точок як окремі, що призводить до помилкового подвоєння пальців. По-третє, потрібно виключити точки, які знаходяться надто близько до країв кадру, адже на цих ділянках часто трапляються помилки.

Для зручності обробки трьох наборів точок створимо додаткові масиви для координат x та y. Назвемо їх startX, startY, endX, endY, defectX та defectY, кожен з типом double. Дані про координати кожної точки занесемо у відповідні масиви через ітерацію по індексах усіх точок.

Далі починаємо процес фільтрації з використанням умовних операторів. Всередині циклу буде розміщено оператор if, який перевірятиме три умови одночасно (логічне «і»). Якщо всі умови виконані, відповідні точки записуються у нові масиви – startFiltered та defectFiltered.

Перша умова базується на розмірі кута, утвореного точками start, defect і end. В OpenCV та Java немає вбудованої функції для обчислення кута за трьома точками, тому доведеться реалізувати її самостійно. Використаємо теорему косинусів: сформуємо уявний трикутник із цих трьох точок, обчислимо довжини його сторін за координатами, а потім знайдемо кут, який позначимо як angle. Перша умова вимагає, щоб цей кут був меншим за π/2.

Друга умова перевіряє відстань між точками start і end. Якщо ця відстань є недостатньо великою (тобто точки дуже близько одна до одної), то точка start має бути виключена. Визначимо це, порівнявши суму квадратів різниць координат із заданим порогом.

Третя умова стосується розташування точок у межах кадру: кожна координата повинна знаходитися у певному діапазоні, не надто близько до країв зображення. Зазначимо, що відступ від країв для фільтрації виберемо рівним 30

пікселів. Ця перевірка також здійснюється за допомогою логічного оператора «і», що гарантує, що значення координати одночасно перевищує нижню і не перевищує верхню межі.

Рисунок 3.7 – Вигляд программи після фільтрації

3.3 Обробка результатів

На цей момент у нас вже накопичено достатньо даних про розташування пальців руки: кожен з п’яти пальців відображений у вигляді точки в одному масиві, де індекс елемента збігається з порядковим номером пальця зліва направо. Крім того, проміжки між пальцями позначені точками з іншого масиву. З використанням цієї інформації можна реалізувати різні функції. Для прикладу, розширимо програму так, щоб вона відображала кількість пальців, що не стиснуті, і змінювала колір внутрішнього контуру на червоний у випадку, коли пальці зімкнуті. Кількість активних пальців визначається розміром масиву startFiltered. Перетворимо це число в рядок і виведемо його на екран за допомогою функції Imgproc.putText. Функція обчислення кутів, яка була створена раніше під час фільтрації, тепер буде використана для підсумовування всіх кутів у циклі з подальшим обчисленням середнього значення. Отримане середнє арифметичне можна порівняти з певним порогом, і якщо воно буде нижче цього порогу, параметри об’єкта класу Scalar, який відповідає за колір внутрішнього контуру, зміняться відповідно.

Кінцевий результат можна спостерігати на рис. 3.8.

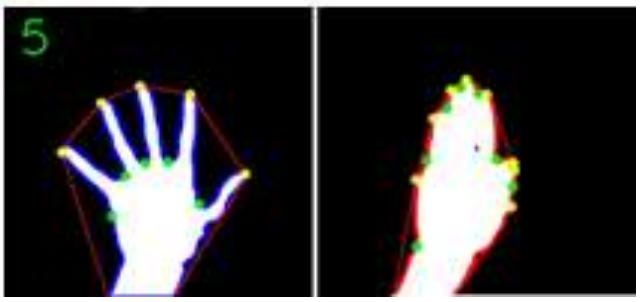


Рисунок 3.8 – Кінцевий вигляд програми

Після отримання даних постає питання їх передачі на польотний контролер. Дані з програми на Java мають бути передані на наземну установку на базі Arduino і, надалі, відправлені на польотний контролер. У майбутньому, під час реалізації проєкту як застосунку для смартфонів, можна буде скористатися бездротовим зв'язком за протоколом Bluetooth, але наразі зручнішим для налагодження варіантом буде надіслати дані через серійний порт.

Для роботи із серійним портом будемо користуватися бібліотекою RXTX. Додамо новий клас і назвемо його Serial.java. Після імпортування всіх необхідних елементів бібліотеки в тілі класу необхідно ініціалізувати об'єкти класів serialPort, input, output, а також оголосити константи: TIME_OUT = 2000 (час очікування інформації), DATA_RATE = 9600 (швидкість передавання серійного порту, має співпадати із заданою в коді для Arduino), PORT_NAMES (масив імен порту для різних операційних систем). Також у класі має міститися кілька методів.

Перший метод має назву initialize і використовується для підготовки порту та ініціалізації змінних і об'єктів. Спочатку в ньому обирається відповідне ім'я порту: у циклі за всіма елементами PORT_NAME порівнюється значення елемента масиву та імені порту, отриманого із системи. Якщо не потрібна мультиплатформеність, то ця частина необов'язкова і для Windows достатньо задати ім'я порту як, наприклад, «COM4».

Потім у цьому ж методі потрібно методомportId.open відкрити серійний порт та встановити параметри DATA_RATE, DATABITS_8, STOPBITS_1 і PARITY_NONE. МетодамигetInputStreamigetOutputStream відкриваємо вхідні та

вихідні потоки даних. Вхідний потік не стане в нагоді, оскільки інформація передаватиметься тільки в бік Arduino, але можна додати цей метод для подальших удосконалень. Також увімкнемо так зване прослуховування подій методом addEventListener - тоді програма реагуватиме на вхідні сигнали.

У процесі реалізації цієї взаємодії важливо не тільки правильно ініціалізувати серійний порт, а й передбачити логіку його завершення роботи. Для цього створюється окремий метод, призначений для коректного закриття серійного порту після завершення сесії обміну. Такий підхід є необхідним, оскільки незакритий порт може спричинити помилки при повторному запуску програми або конфлікти з іншими пристроями.

Закриття порту здійснюється з урахуванням того, що він був раніше відкритий. У разі підтвердження його активності потрібно деактивувати прослуховування подій, що надходять із порту. Це досягається за допомогою виключення обробника подій, який слід видалити, аби уникнути зайвих обчислень у фоновому режимі. Після цього сам порт закривається, що гарантує безпечне завершення сесії зв’язку та звільнення ресурсу.

Для організації передачі даних передбачено створення методу, який дозволяє надсилати одиничний байт інформації. Така структура передачі є зручною, зокрема, коли мова йде про передачу керуючих сигналів у вигляді числових кодів або команд. Метод передбачає прийом на вхід одного параметра – байта, що є одиницею інформації, і який безпосередньо надсилається через серійний порт. Сам процес передачі забезпечується через об’єкт виводу (output), до якого байт надходить як елемент потоку даних.

Зчитування вхідних даних відбувається автоматично в момент їхнього надходження через серійний порт. Для цього застосовується спеціальний механізм, заснований на обробці подій. Кожного разу, коли через порт надходить новий сигнал, спрацьовує метод, який обробляє цю подію. Його основним аргументом є об’єкт, який містить інформацію про саму подію – це дає змогу точно ідентифікувати момент отримання даних.

У середині цього методу виконується зчитування вхідного байта. Для цього використовується вхідний потік (input), з якого витягується значення. Отримане значення, як правило, представлено у вигляді байта, тобто числового типу з обмеженим діапазоном. У подальшому його можна перевести у цілочисельний тип для подальшої обробки або виводу.

З міркувань безпеки та контролю за коректністю даних доцільно перевіряти, чи входить отримане значення у дозволений інтервал – від 0 до 255. Такий діапазон є стандартним для байтових даних і дозволяє уникнути помилок під час їхньої інтерпретації. У разі позитивного результату перевірки, значення може бути виведене у консоль або використане для подальших обчислень.

Виведення отриманих даних у консоль або графічний інтерфейс має важливе значення на етапі налагодження програми, оскільки дозволяє розробнику або оператору в реальному часі відстежувати хід передавання, швидкість обміну, коректність отриманих значень тощо. Це є надзвичайно зручним інструментом діагностики у разі виникнення збоїв або втрати даних.

Уся описана логіка – від ініціалізації порту до його закриття, передавання байтів і обробки вхідних сигналів – забезпечує простий, але водночас надійний механізм взаємодії між комп’ютером і мікроконтролером. Завдяки цьому досягається плавний обмін інформацією, що є ключовим для коректної роботи інтелектуальної системи управління квадрокоптером.

Особливістю такого підходу є його універсальність – він може бути адаптований під різні типи пристроїв, плат або навіть операційних систем. За умови правильної реалізації, подібна система забезпечує високий рівень стабільності та дозволяє інтегрувати складні алгоритми штучного інтелекту в системи реального часу, які керують фізичними об’єктами в просторі.

Тепер для класу можна створювати об'єкти і викликати їхні методи.Створимо об'єкт segialкласу Segiali запустимо метод його ініціалізації у функції start. У методі setClosed, що виконується при закритті програми, пропишемо serial.close.

Тепер закодуємо інформацію з двох змінних, одержуваних із камери, в один байт. У разі розтиснутих пальців у цьому байті міститиметься просто кількість відфільтрованих контурів. При стислих байт міститиме значення {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1} (127). Наприкінці функції, що аналізує зображення, варто додати метод, який здійснює надсилання даних, - serial.sendSingleByte, і в аргументах йому задати отриманий байт.

Наступний етап - модернізувати код для Arduino таким чином, щоб він міг приймати дані, які надсилаються в серійний порт. Додамо змінну типу Byte і назвемо її getByte. У кожному повторенні циклу Іоортеперь можна додати команду Serial.read, за умови, якщо Serial.avaliable.

Відомо, що контролер AirduPilot сприймає сигнал зміни режиму польоту так само, як і сигнали газу, рискання, тангажу і крену, - у вигляді сигналу з широтно-імпульсною модуляцією. Конкретні режими, які перемикатимуться цим сигналом, можна налаштувати в програмі MissionPlanner. Одному каналу можна присвоїти функцію зміни до 256 режимів, але найчастіше один канал на пульті змінює два або три режими. У нашому випадку такий варіант є оптимальним. Зробимо так, щоб при стиснутих пальцях перший канал передавав байт 11111111, а при розтиснутих - 0000000. Другий канал буде управлятися кількістю пальців - нехай за стиснутих пальців значення дорівнює 0, за трьох розтиснутих - 86, за чотирьох 172, за п'яти - 255.

Прийнятий байт потрібно «розкласти» назад на два сигнали, аналогічними умовами: якщо прийнятий байт дорівнює 127, то змінна mode1 типу іптравна 255, а mode2 - нулю, інакше mode 1 дорівнює нулю, а mode2 визначається за вищеописаним методом.

Тепер передамо дані на інший мікроконтролер, який перебуватиме на борту квадрокоптера. Для цієї мети будуть використовуватися два модулі nRF24L01. Це модулі радіозв'язку, що працюють на частоті 2,4 ГГц і мають вбудовану бібліотеку для платформ Arduino і RaspberryPI. Модулі можуть передавати сигнал на відстані

до 100 м зі швидкістю до 2 Мб/с. Модуль взаємодіє з контролером за допомогою інтерфейсу SPI.

Модуль має наступні виходи:

- SCK (Serial ClocK) - тактування (синхронізація).
- MOSI / MI (Master Out Slave In) вхід данних.
- MISO / MO (Master In Slave Out) вихід данних.
- CE/SS - Вибір основного на шині SP з декількох пристроїв.
- SCN - Вибір режиму приймання/передача.
- IRQ - вихід переривання,часто не використовується.

Необхідний для моментальної реакції мікроконтролера під час приймання нового пакета даних.

У процесі створення інтелектуальної підсистеми керування квадрокоптером особливу увагу необхідно приділити забезпеченню стабільного і надійного радіозв'язку між передавальною та приймальною частинами. Для реалізації такого зв'язку були використані бездротові модулі nRF24L01+, які працюють у діапазоні частот 2,4 ГГц і забезпечують ефективну передачу даних між елементами системи.

На обох сторонах взаємодії - як на передавальній, так і на приймальній - використовуються мікроконтролери Arduino. Для організації зв'язку між ними передбачено підключення та використання спеціальних бібліотек, які дозволяють керувати радіомодулями. Основні бібліотеки забезпечують взаємодію з апаратним SPI-інтерфейсом, реалізують протоколи радіозв'язку, а також містять методи налаштування параметрів роботи модуля.

Після підключення бібліотек, першим етапом є створення об'єкта, який відповідає за обмін даними між пристроями. Під час створення цього об'єкта визначаються номери цифрових виводів плати, до яких під'єднано модуль. Це важливо, оскільки модуль nRF24L01+ працює за допомогою SPI-з'єднання, і правильне визначення виводів є критичним для забезпечення зв'язку.

Далі необхідно оголосити унікальні адреси каналів зв'язку, які в бібліотеці радіозв'язку називаються трубами. Кожна труба відповідає певному напрямку

передавання або приймання даних. Адреса передається у вигляді байтового масиву і є ідентифікатором пристрою в мережі.

У функції ініціалізації відбувається запуск модуля, перевірка наявності зв’язку та налаштування базових параметрів. Одним із важливих кроків є активація модуля та встановлення режиму підтвердження отримання пакета, що дає змогу підвищити надійність зв’язку за рахунок повторного надсилання інформації у разі її втрати. Для цього вказується кількість повторів передачі та часовий інтервал між ними.

Окрім цього, встановлюється фіксований розмір пакета даних. Такий підхід дозволяє уникнути помилок при обміні інформацією між пристроями та забезпечує однакову інтерпретацію отриманих даних як на передавальній, так і на приймальній стороні. У конкретному випадку використовується об’єм 6 байтів, який є оптимальним для передачі керуючих сигналів з інтелектуальної підсистеми на квадрокоптер.

Після цього налаштовується канал, на якому здійснюється радіозв’язок. Діапазон каналів охоплює частоти від 2.400 до 2.527 ГГц, кожна з яких представлена числовим значенням у шістнадцятковій системі. Правильний вибір каналу дозволяє уникнути перешкод, спричинених іншими пристроями, які працюють у тому ж частотному діапазоні, зокрема Wi-Fi роутерами.

Не менш важливим параметром є рівень потужності передавача. Залежно від умов експлуатації (наприклад, в приміщенні, на відкритому просторі чи за наявності перешкод), можна вибрати мінімальний або максимальний рівень потужності. Використання максимального рівня дозволяє досягти більшої дальності зв’язку, проте при цьому зростає споживання енергії.

Ще одним параметром, який впливає на стабільність та якість з’єднання, є швидкість передачі даних. Модуль nRF24L01+ дозволяє працювати з кількома стандартними швидкостями, однак для стабільної передачі на великі відстані доцільно використовувати найменшу доступну швидкість, що становить 250 кбіт/с. Хоча така швидкість є відносно низькою, вона забезпечує найбільшу чутливість і,

як наслідок, максимальну відстань зв’язку.

Після завершення налаштувань радіомодуль переходить у робочий стан. Для пристрою, який виконує роль приймача, активується режим прослуховування, що дозволяє йому постійно перевіряти наявність вхідного сигналу. Передавальний модуль, навпаки, припиняє прослуховування і готується до передачі інформації.

Процес обміну даними здійснюється у двох напрямках. Передавальна сторона формує масив керуючих даних (наприклад, координати, значення жестів, сигнали перемикачів тощо) і відправляє їх у пакеті на приймач. Приймальна сторона, у свою чергу, зчитує дані, після чого вони можуть бути інтерпретовані та передані безпосередньо на керувальні виходи польотного контролера. Таким чином, команди, сформовані на основі візуального розпізнавання жестів, транслюються у конкретні електричні сигнали, які впливають на поведінку квадрокоптера.

Завдяки використанню модуля nRF24L01+ забезпечується ефективна та енергоощадна бездротова передача даних у реальному часі. Такий підхід дозволяє інтегрувати інтелектуальні алгоритми керування в мобільні роботизовані системи, уникаючи складних дротових з'єднань та зберігаючи високу гнучкість і надійність у процесі експлуатації.

3.4 Висновок до третього розділу

У цьому підрозділі були розглянуті всі ключові етапи обробки відеосигналу: захоплення кадру, зменшення роздільної здатності зображення, приведення його до монохромного або бінарного вигляду, застосування фільтрів згладжування або порогової обробки, виявлення контурів, пошук екстремальних точок та інтерпретація отриманих даних. Продемонстровано, що за допомогою простих алгоритмів можна реалізувати розпізнавання кількості пальців на руці або положення руки (відкрита/закрита), що є базовим способом передачі керуючих команд.

Було обґрунтовано вибір середовища розробки - IntelliJ IDEA - як ефективного інструменту для створення Java-додатків із використанням JavaFX. Досліджено структуру проєкту, зокрема роль файлу FXML у побудові графічного інтерфейсу та взаємодії з основним програмним кодом через клас controller.java. Такий підхід дозволяє створювати зручний і гнучкий інтерфейс користувача, в якому відображаються не лише результати аналізу зображення, але й проміжні етапи обробки, що значно полегшує налагодження системи.

Запропонований підхід до побудови інтелектуальної підсистеми дозволяє значно розширити функціональні можливості квадрокоптера. Використання візуального керування на основі жестів відкриває перспективи створення безконтактного, інтуїтивно зрозумілого інтерфейсу, що не вимагає додаткових контролерів або джойстиків. Такий тип управління є особливо цінним у випадках, коли оператор не може або не хоче використовувати стандартні засоби керування - наприклад, під час виконання завдань у польових умовах, при виконанні рятувальних операцій, моніторингу територій, тощо.

ВИСНОВОК

У другому розділі було детально проаналізовано та реалізовано ключові етапи проєктування інтелектуальної підсистеми обробки візуальної інформації, яка є важливою складовою кіберфізичної системи керування універсального квадрокоптера. Основна мета цього розділу полягала у створенні програмного модуля, здатного здійснювати аналіз зображень, отриманих з відеокамери, для подальшого виявлення та розпізнавання жестів руки користувача, що слугують командами для безпілотного літального апарата.

На початковому етапі розглянуто особливості апаратного та програмного забезпечення, які доцільно використовувати для реалізації інтелектуальної підсистеми. Було обґрунтовано вибір архітектури побудови системи, яка має враховувати як можливості обробки зображення в реальному часі, так і вимоги до мобільності, енергоефективності та взаємодії з іншими модулями керування квадрокоптером. З цієї точки зору особливу увагу приділено вибору платформи розробки програмного забезпечення - Java, з подальшою можливістю переносу на мобільну ОС Android. Такий підхід дозволяє реалізувати максимально компактну та мобільну систему, яка не потребує окремого ПК, що суттєво підвищує практичну цінність запропонованого рішення.

Досліджено основні функціональні можливості бібліотеки OpenCV, яка є провідною у галузі комп’ютерного зору та обробки зображень. Детально розглянуто її структуру, принципи роботи, а також конкретні класи, які реалізують функції попередньої обробки зображення, фільтрації шуму, бінаризації, пошуку контурів, виділення екстремальних точок тощо. Аналіз показав, що OpenCV має усі необхідні засоби для реалізації функціоналу підсистеми обробки візуальної інформації та підтримує роботу в режимі реального часу на пристроях із обмеженими обчислювальними ресурсами.

У межах дослідження також було розглянуто принципи роботи сенсорних систем для керування квадрокоптером. Зокрема, вивчено особливості взаємодії з модулями IMU (інерціальні вимірювальні одиниці), які дозволяють відстежувати просторове положення квадрокоптера, а також із ультразвуковим датчиком відстані HC-SR04, який забезпечує додаткову інформацію про простір перед літальним апаратом. Було продемонстровано, що ці дані можуть бути ефективно інтегровані з жестовим керуванням, забезпечуючи більш безпечне, надійне та інтуїтивно зрозуміле управління дроном навіть у складних умовах середовища.

Окрему увагу приділено створенню програмного модуля для розпізнавання жестів руки.

У підсумку можна стверджувати, що результати проєктування, розглянуті у цьому розділі, є вагомим кроком на шляху до реалізації повноцінної кіберфізичної системи управління квадрокоптером з використанням сучасних технологій комп’ютерного зору. Запропоноване рішення має високий потенціал до масштабування, подальшого вдосконалення та адаптації до різних сценаріїв використання. У подальших дослідженнях можливе впровадження більш складних алгоритмів глибокого навчання для підвищення точності розпізнавання жестів, розширення набору доступних команд та покращення адаптивності системи до зовнішніх умов (освітлення, фон, розмір руки тощо).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Схема роботи ПІД-регулятора URL: https://www.tanklessology.com/new-technology.html (дата звернення: 14.05.2025).

2. Плата ARM 2.6 URL: https://ardupilot.org/plane/docs/common-apm25-and-26-overview.html (дата звернення: 14.05.2025).

3. Подання зорової інформації комп'ютером URL: https://atmega32-avr.com/how-facial-recognition-systems-work/ (дата звернення: 14.05.2025).

4. Ультразвуковий далекомір HC-SR04 URL: https://prom.ua/p67925784-ultrazvukovoj-datchik-rasstoyaniya.html (дата звернення: 14.05.2025).

5. Sabatini R. Avionics and control systems for UAVs. *Progress in Aerospace Sciences*. 2020. Vol. 109. P. 100543. URL: https://www.sciencedirect.com/science/article/abs/pii/S0376042119300919 (дата звернення: 11.05.2025).

6. Faessler M. та ін. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *Autonomous Robots*. 2016. Vol. 40. P. 1031–1050. URL: https://link.springer.com/article/10.1007/s10514-016-9546-0 (дата звернення: 12.05.2025).

7. Mehta A. M., Eguren J. Designing Flight Controllers for Multirotor Drones. Artech House, 2020. 296 p. URL: https://www.artechhouse.com/Designing-Flight-Controllers-for-Multirotor-Drones-P2004.aspx (дата звернення: 12.05.2025).

8. DJI SDK Documentation. URL: https://developer.dji.com/documentation/ (дата звернення: 12.05.2025).

9. PX4 Autopilot User Guide. 2024. URL: https://docs.px4.io (дата звернення: 10.06.2025).

10. Ardupilot Documentation. 2024. URL: https://ardupilot.org/ (дата звернення: 14.05.2025).

11. Corke P. Robotics, Vision and Control: Fundamental Algorithms In MATLAB. Springer, 2022. 812 p. URL: https://link.springer.com/book/10.1007/978-3-030-43089-4 (дата звернення: 14.05.2025).

12. Beard R. W., McLain T. W. Small Unmanned Aircraft: Theory and Practice. Princeton University Press, 2017. 448 p. URL: https://press.princeton.edu/books/hardcover/9780691149219/small-unmanned-aircraft (дата звернення: 14.05.2025).

13. MATLAB UAV Toolbox Documentation. 2023. URL: https://www.mathworks.com/products/uav.html (дата звернення: 14.05.2025).

14. Papachristos C. та ін. Autonomous navigation for aerial robots in underground environments. *Springer Tracts in Advanced Robotics*. 2017. Vol. 120. P. 745–758. URL: https://link.springer.com/chapter/10.1007/978-3-319-50115-4_35 (дата звернення: 17.05.2025).

15. Barrientos A. та ін. UAV Navigation in GPS Denied Environments. Springer, 2018. 385 p. URL: https://link.springer.com/book/10.1007/978-3-319-76492-4 (дата звернення: 17.05.2025).

16. Hoffmann G. M. та ін. Quadrotor Helicopter Flight Dynamics and Control. Stanford University, 2016. URL: https://web.stanford.edu/~boyd/papers/quadrotor.html (дата звернення: 17.05.2025).

17. Saska M. Autonomous deployment of micro aerial vehicles for building inspection. *Journal of Field Robotics*. 2020. Vol. 37. P. 817–841. URL: https://onlinelibrary.wiley.com/doi/10.1002/rob.21938 (дата звернення: 17.05.2025).

18. Zhu H. та ін. Path Planning for Quadcopters in Unknown Environments. *Sensors (MDPI)*. 2021. Vol. 21(9). P. 3160. URL: https://www.mdpi.com/1424-8220/21/9/3160 (дата звернення: 19.05.2025).

19. Kendoul F. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*. 2016. Vol. 33. P. 133–161. URL: https://onlinelibrary.wiley.com/doi/10.1002/rob.21513 (дата звернення: 19.05.2025).

20. CUAV Autopilot Systems Technical Docs. 2023. URL: https://doc.cuav.net (дата звернення: 19.05.2025).

21. Kumar V. та ін. Cooperative Autonomous UAVs. Springer, 2022. 350 p. URL: https://link.springer.com/book/10.1007/978-3-030-47483-6 (дата звернення: 23.05.2025).

22. Loianno G. та ін. Visual inertial odometry for quadrotors. *Autonomous Robots*. 2017. Vol. 41. P. 1–21. URL: https://link.springer.com/article/10.1007/s10514-017-9613-2 (дата звернення: 23.05.2025).

23. Quigley M. та ін. ROS: The Complete Reference (Volume 1). Springer, 2018. 700 p. URL: https://link.springer.com/book/10.1007/978-3-319-26054-9 (дата звернення: 24.05.2025).

24. Gupte S. та ін. A survey of quadrotor UAV flight control techniques. *Journal of Intelligent & Robotic Systems*. 2016. Vol. 74. P. 485–502. URL: https://link.springer.com/article/10.1007/s10846-015-0239-2 (дата звернення: 25.05.2025).

25. Garcia J. C. та ін. Model Predictive Control for UAVs. *Drones (MDPI)*. 2018. Vol. 2(3). P. 19. URL: https://www.mdpi.com/2504-446X/2/3/19 (дата звернення: 25.05.2025).

26. Wang H. та ін. Reinforcement Learning for UAV Control. *IEEE Transactions on Neural Networks and Learning Systems*. 2019. Vol. 30. P. 289–302. URL: https://ieeexplore.ieee.org/document/8699423 (дата звернення: 26.05.2025).

27. OpenCV Documentation. 2024. URL: https://docs.opencv.org (дата звернення: 26.05.2025).

28. Robot Operating System (ROS) Wiki. 2024. URL: https://wiki.ros.org (дата звернення: 26.05.2025).

29. UAVCAN Protocol Specification. 2023. URL: https://uavcan.org (дата звернення: 26.05.2025).

30. Velivela A. Embedded Systems for UAVs. CRC Press, 2019. 312 p. URL: https://www.routledge.com/Embedded-Systems-for-UAVs/Velivela/p/book/9780367135060 (дата звернення: 26.05.2025).

31. Nguyen T. та ін. Control of quadrotors using nonlinear methods. *Nonlinear Dynamics*. 2021. Vol. 105. P. 879–896. URL: https://link.springer.com/article/10.1007/s11071-021-06680-7 (дата звернення: 26.05.2025).

32. D'Andrea R. The future of flying robots. *Nature*. 2016. Vol. 531. P. 438–442. URL: https://www.nature.com/articles/531438a (дата звернення: 29.05.2025).

33. Google Scholar. URL: https://scholar.google.com (дата звернення: 29.05.2025).

34. SpringerLink. URL: https://link.springer.com (дата звернення: 29.05.2025).

35. ScienceDirect (Elsevier). URL: https://www.sciencedirect.com (дата звернення: 29.05.2025).

36. IEEE Xplore (Digital Library). URL: https://ieeexplore.ieee.org (дата звернення: 01.06.2025).

37. MDPI - Drones Journal. URL: https://www.mdpi.com/journal/drones (дата звернення: 01.06.2025).

38. HAL - Open Science Archive. URL: https://hal.archives-ouvertes.fr (дата звернення: 02.06.2025).

39. ACM Digital Library. URL: https://dl.acm.org (дата звернення: 02.06.2025).

40. TechRxiv Preprints. URL: https://www.techrxiv.org (дата звернення: 02.06.2025).

41. ResearchGate. URL: https://www.researchgate.net (дата звернення: 02.06.2025).

42. arXiv (preprints). URL: https://arxiv.org (дата звернення: 02.06.2025).

43. UAV Systems International Resources. URL: https://uavsystemsinternational.com (дата звернення: 05.06.2025).

44. Dronecode.org - PX4 community. URL: https://www.dronecode.org (дата звернення: 05.06.2025).

45. DroneDeploy Blog (технічні розбори). URL: https://www.dronedeploy.com/blog (дата звернення: 05.06.2025).

46. Auterion Software Docs. URL: https://docs.auterion.com (дата звернення: 07.06.2025).

47. Unmanned Systems Technology. URL: https://www.unmannedsystemstechnology.com (дата звернення: 07.06.2025).

48. Pixhawk Hardware Specs. 2024. URL: https://pixhawk.org (дата звернення: 08.06.2025).

49. MicroPilot UAV Autopilots. URL: https://www.micropilot.com (дата звернення: 10.06.2025).

50. Jenkins M. Autonomous Drone Navigation. Wiley, 2020. 256 p. URL: https://www.wiley.com/en-us/ Autonomous+Drone+Navigation-p-9781119552076 (дата звернення: 10.06.2025).

51. Mahony R., Kumar V., Corke P. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. IEEE Robotics & Automation Magazine. 2016. Vol. 23. P. 20–32. URL: https://ieeexplore.ieee.org/document/7442157 (дата звернення: 02.06.2025).

52. Bouabdallah S. Design and control of quadrotors with application to autonomous flying. EPFL Press, 2017. 218 p. URL: https://infoscience.epfl.ch/record/97574 (дата звернення: 10.05.2025).

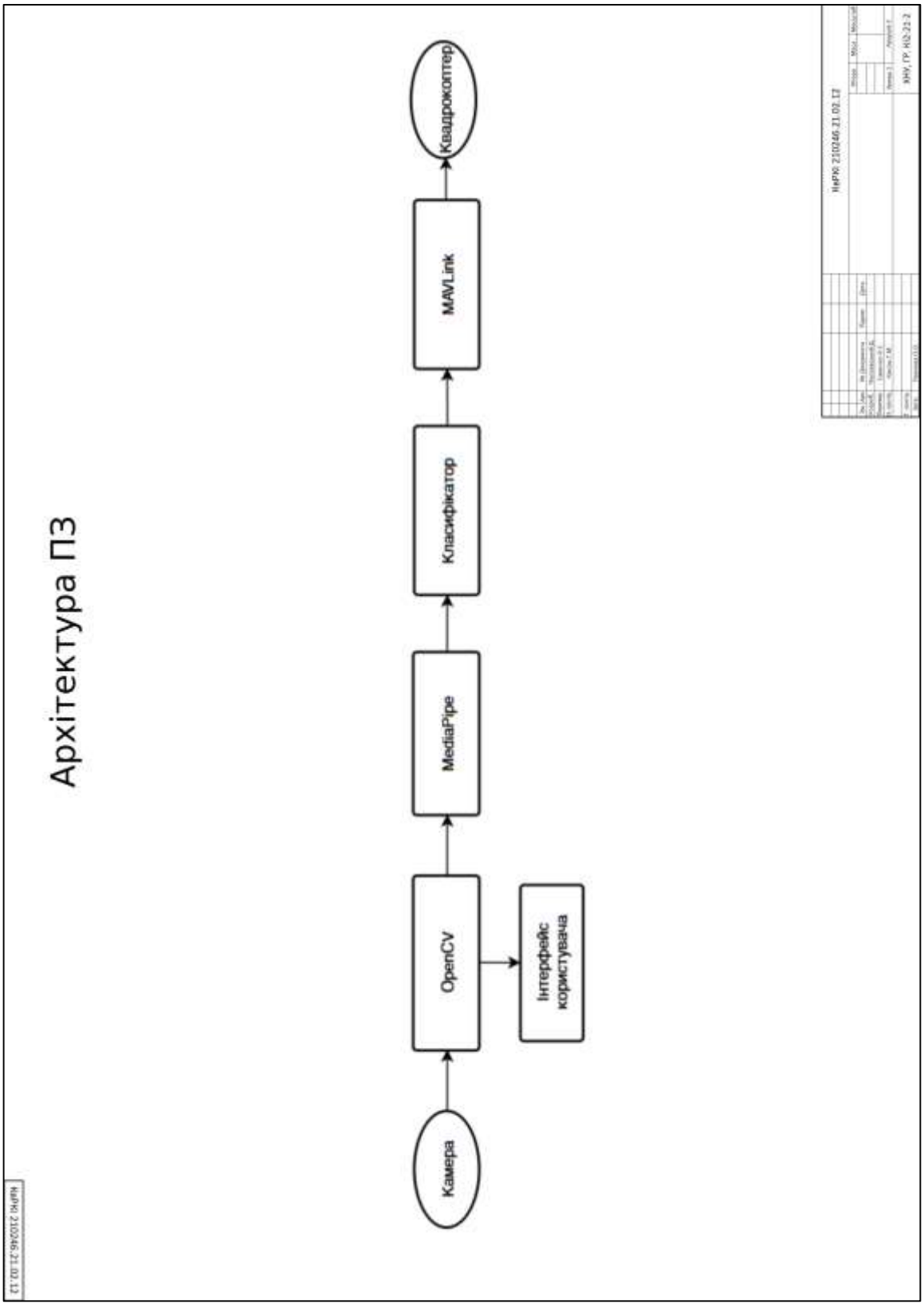
53. Mellinger D., Kumar V. Minimum Snap Trajectory Generation and Control for Quadrotors. IEEE ICRA. 2018. P. 2520–2525. URL: https://ieeexplore.ieee.org/document/5980409 (дата звернення: 11.05.2025).

54. Shakhatreh H. та ін. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. IEEE Access. 2019. Vol. 7. P. 48572–48634. URL: https://ieeexplore.ieee.org/document/8612750 (дата звернення: 11.05.2025).

Додаток А

(обов’язковий)

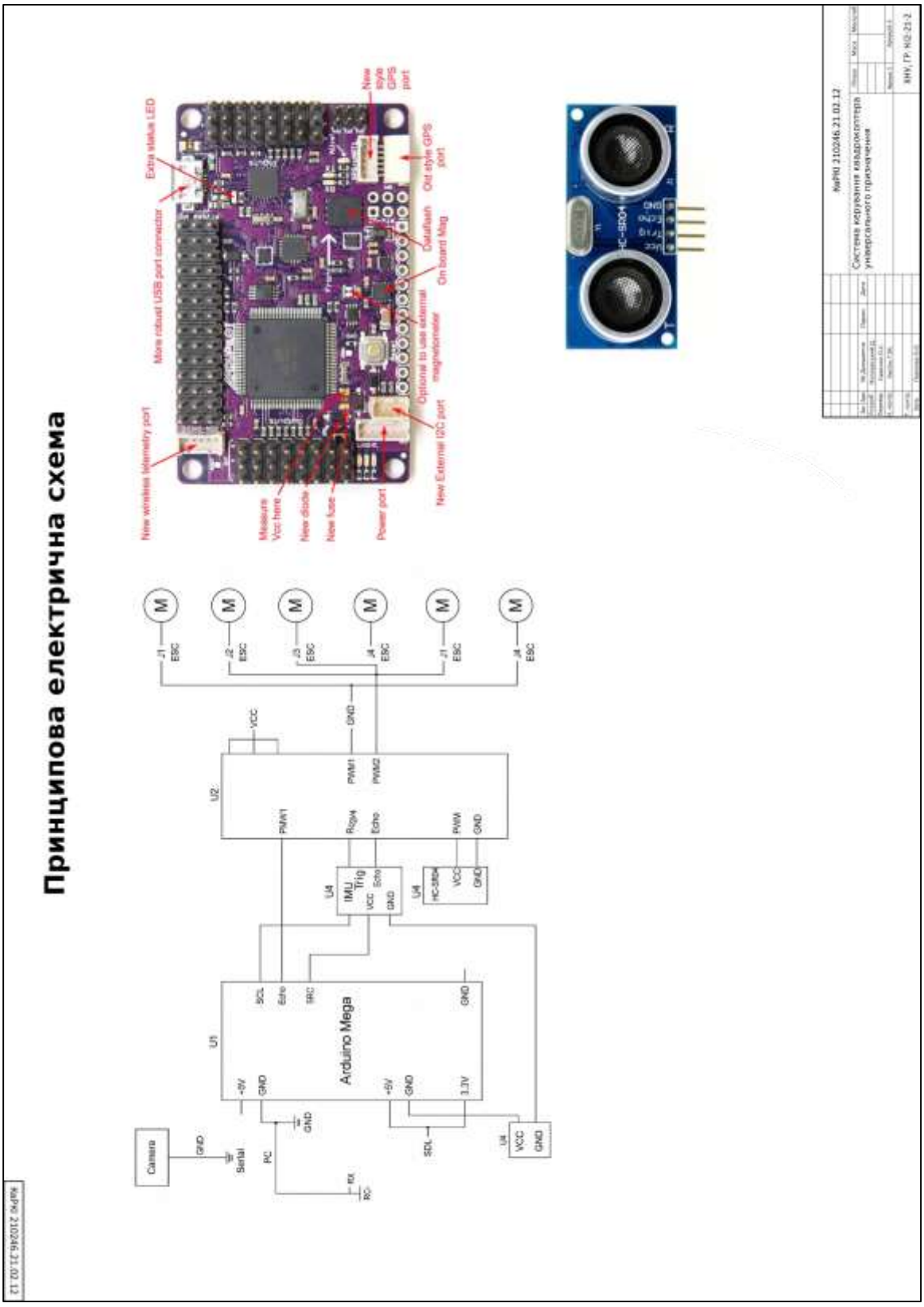
КОПІЯ КРЕСЛЕННЯ «АРХІТЕКТУРА ПЗ ПРОЄКТУ»



Додаток Б

(обов'язковий)

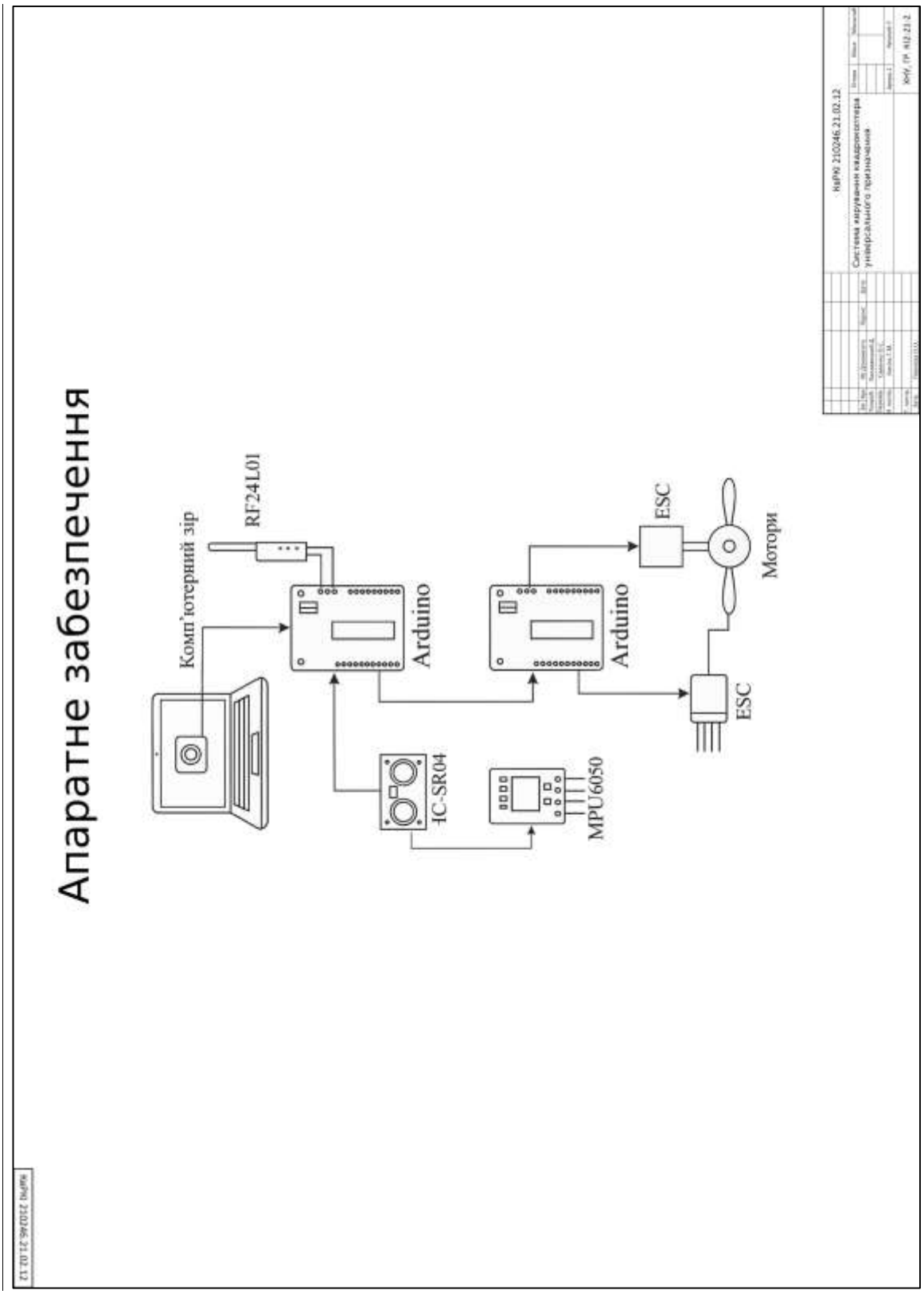
КОПІЯ КРЕСЛЕННЯ «ПРИНЦИПОВА ЕЛЕКТРИЧНА СХЕМА»



Додаток В

(обов’язковий)

КОПІЯ КРЕСЛЕННЯ «АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ПРОЄКТУ»



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Поплавський Денис Юрійович

Тема: Система керування квадрокоптером універсального призначення

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___64___

1. Короткий зміст роботи та прийнятих рішень: розробка та дослідження системи керування квадрокоптером універсального призначення, яка забезпечує стабільний політ, маневреність та можливість адаптації до різних сценаріїв застосування (моніторинг, доставка, картографування тощо), з використанням сучасних апаратних та програмних засобів.

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

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки та передових методів роботи:

У першому розділі кваліфікаційної роботи здійснено грунтовний аналіз предметної області, а саме: розглянуто сучасні методи керування безпілотними літальними апаратами, досліджено архітектуру кіберфізичних систем, а також виконано огляд передових технологій, що застосовуються у сфері комп'ютерного зору, стабілізації польоту та сенсорного керування. Проведено постановку задачі розробки системи інтуїтивного жестового управління квадрокоптером. Визначено структуру системи, обрано відповідні апаратні компоненти (Arduino Mega, ArduPilot, IMU, ультразвуковий далекомір), обгрунтовано їх вибір.

У другому розділі роботи реалізовано проєктування інтелектуальної підсистеми обробки візуальної інформації. Виконано підключення камери, реалізовано захоплення та виведення відеопотоку в реальному часі. Проведено попередню обробку зображення за допомогою бібліотеки OpenCV: фільтрація шумів, виділення контурів, сегментація об'єктів. Розроблено алгоритм розпізнавання жестів руки, який

дозволяє визначати положення долоні та її орієнтацію. Запропоновано метод передачі результатів аналізу до керуючого мікроконтролера для подальшого формування керуючих сигналів.

У третьому розділі здійснено апаратну реалізацію системи: реалізовано взаємодію між ПК (на якому обробляється відеопотік) та мікроконтролером Arduino Mega через UART. Розроблено програмну логіку на Arduino для обробки команд керування, перетворення їх у PWM-сигнали та передачі до контролера ArduPilot. Проведено узгодження роботи з сенсорами (IMU, HC-SR04), реалізовано обчислення yaw, pitch, roll. Побудовано принципову електричну схему системи, проведено її моделювання та перевірку працездатності. Розроблено структурну схему системи керування квадрокоптером, яка відображає взаємодію всіх апаратних та програмних компонентів.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: – .

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

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Кисоч Юрій Павлович, к.т.н., зав.кафедри

кібернетики, ХНУ

"10" 06 2025 р. _____ (підпис)

Протокол аналізу звіту подібності експертом

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

Автор: Денис ПОПЛАВСЬКИЙ

Співавтор:

Назва: Поплавський_2_Системи керування квадрокоптером універсального призначення

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:6.2%

Коефіцієнт подібності 2:2%

Мікропробіли: 6

Заміна букв: 7

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-12 22:22:42.0

Після аналізу Звіту подібності констатую наступне:

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

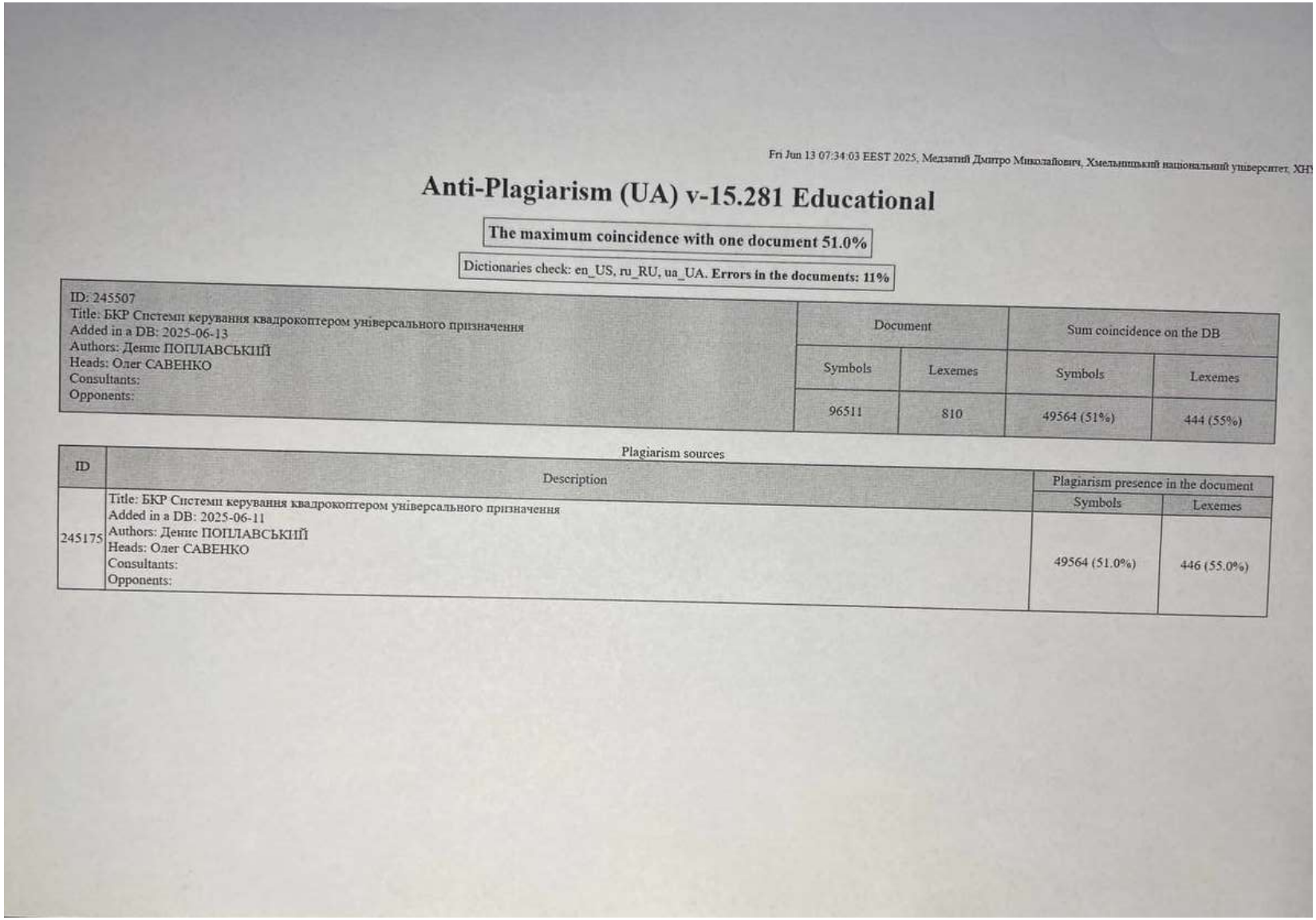
☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

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

Обгрунтування:

2025-06-13 Доцент Андрій Нічепорук

Дата експерт



Fri Jun 13 07:34:03 EEST 2025, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 51.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 11%**

	Document		Sum coincidence on the DB	
ID: 245507 Title: БКР Системи керування квадрокоптером універсального призначення Added in a DB: 2025-06-13 Authors: Денис ПОПЛАВСЬКИЙ Heads: Олег САВЕНКО Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	96511	810	49564 (51%)	444 (55%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes
245175	Title: БКР Системи керування квадрокоптером універсального призначення Added in a DB: 2025-06-11 Authors: Денис ПОПЛАВСЬКИЙ Heads: Олег САВЕНКО Consultants: Opponents:	49564 (51.0%)	446 (55.0%)

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

Дениса ПОПЛАВСЬКОГО
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-21-2

ЗАЯВА

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

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

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

11.06 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

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

Назва: Система керування квадрокоптером універсального призначення

Автор: Денис ПОПЛАВСЬКИЙ

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Олег САВЕНКО, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

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

Підтвердження:

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

1) Запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, що є загальним оглядом літератури та не стосуються безпосередньо авторського дослідження чи оригінальних результатів роботи.
2) Усі запозичення є фрагментарними та належним чином оформлені відповідно до академічних вимог, з наданням точних посилань на джерела, з яких вони були отримані.
3) Окремі збіги, зафіксовані системою, є загальновживаними фразами або термінами, що часто використовуються в галузі, про що свідчить численні збіги з 10-40 джерелами на один фрагмент тексту. Такі фрази не є результатом авторського плагіату, оскільки належать до загальновизнаних термінів.
4) В окремих випадках система зафіксувала послідовності чотиризначних двійкових кодів, що є типовими вхідними даними для множини задач, і не можуть розглядатися як об'єкт авторських прав, оскільки ці послідовності є стандартними елементами для математичних чи технічних розрахунків.
5) Всі зафіксовані системою ознаки модифікації тексту пов'язані з комбінуванням латинських символів зі скороченнями українських індексів у формулах. Це не є модифікацією змісту, оскільки такі скорочення є стандартною практикою в математичній та технічній літературі і не порушують авторських прав.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 6.2% і адресується до 401 першоджерела; та системою Anti-Plagiarism складає 51%, що базуються на попередній спробі,що , з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Олег САВЕНКО

Гарант ОП _____ Андрій НІЧЕПОРУК

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА