НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету (Директор ННІ)	**Завідувач кафедри**
Інформаційних технологій (назва факультету(ННІ))	Комп'ютерних систем, мереж та кібербезпеки (назва кафедри)
_____ Болбот І.М., д.т.н, проф. (підпис) (ПІБ, вчене звання і ступінь)	_____ Касаткін Д.Ю., к. пед.н., доц. (підпис) (ПІБ, вчене звання і ступінь)
«___»_____2025 р.	«___»_____2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Дослідження та поліпшення підсистем керування та телеметрії безпілотного літального апарату»

Спеціальність 123 «Комп’ютерна інженерія»
(код і найменування)

Освітня програма Комп’ютерні системи та мережі
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

д.т.н., доцент	_____	Шкарупило В.В.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Керівник магістерської кваліфікаційної роботи

д.т.н., доцент	_____	Шкарупило В.В.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Виконав _____ Вернигора В.Ю.
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
к.пед.н., доц. _____ Касаткін Д.Ю.
(вчене звання і ступінь) (підпис) (ПІБ)
«__» _____ 20__ р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Вернигорі Валентину Юрійовичу
(прізвище, ім'я, по батькові)

Спеціальність 123 «Комп'ютерна інженерія»
(код і найменування)

Освітня програма Комп'ютерні системи та мережі
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи: «Дослідження та поліпшення підсистем керування та телеметрії безпілотного літального апарату»

затверджена наказом ректора НУБіП України від "29" жовтня 2024р. № 1941 «С»

Термін подання завершеної роботи на кафедру 14 листопада 2025 р.

Вихідні дані до магістерської кваліфікаційної роботи

Перелік питань, що підлягають дослідженню:
1.
2.
3.

Перелік графічного матеріалу (за потреби)

Дата видачі завдання " 29 **"** жовтня 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Шкарупило В.В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Вернигора В.Ю.
(підпис) (прізвище та ініціали)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз предметної області	29.10.2024 – 15.01.2025	Виконано
2	Теоретичне моделювання підсистем	16.01.2025 – 31.03.2025	Виконано
3	Проведення порівняльного аналізу	01.04.2025 – 31.05.2025	Виконано
4	Оформлення пояснювальної записки	01.06.2025 – 15.10.2025	Виконано
5	Оформлення графічного матеріалу	16.10.2025 – 31.10.2025	Виконано
6	Фіналізація роботи та підготовка до захисту	01.11.2025 – 14.11.2025	Виконано

Студент _____ **В.Ю. Вернигора**

(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **В.В. Шкарупило**

(підпис) (ініціали та прізвище

РЕФЕРАТ

Пояснювальна записка: 91 сторінка, 31 рисунки, 7 таблиць, 4 лістинги, 2 додатки, 43 джерела.

БПЛА, ПОЛЬОТНИЙ КОНТРОЛЕР, ПРЕДМЕТНО-ОРІЄНТОВАНА СИСТЕМА, КОМЕРЦІЙНЕ РІШЕННЯ, INAV, STM32F401, STM32F405, OSD, EKF, MAVLINK, РЕБ

Об’єкт дослідження – підсистеми керування польотом та телеметрії безпілотного літального апарату.

Мета роботи – теоретичне дослідження та обґрунтування шляхів поліпшення підсистем керування та телеметрії БПЛА шляхом порівняльного аналізу предметно-орієнтованих та комерційних рішень.

Робота складається з чотирьох розділів.

Перший розділ присвячено аналізу сучасних апаратних архітектур, програмних екосистем , протоколів та основних загроз каналам зв’язку в умовах РЕБ.

Другий розділ присвячено теоретичному моделюванню підсистем: базової предметно-орієнтованої, типової комерційної та поліпшеної предметно-орієнтованої, з інтеграцією EKF, OSD та захищених протоколів.

У третьому розділі проведено теоретичний порівняльний аналіз розроблених моделей за критеріями функціональності, гнучкості, вимог до ресурсів, потенційної стійкості до загроз та економічної доцільності.

У результаті було розроблено методику порівняльного аналізу, теоретично обґрунтовано шляхи інтеграції комерційних прошивок у предметно-орієнтовані контролери та надано рекомендації щодо вибору архітектури БПЛА.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

БПЛА – Безпілотний літальний апарат

ГНСС – Глобальна навігаційна супутникова система

ПІД – Пропорційно-інтегрально-диференціальний (регулятор)

ПК – Польотний контролер

ПЗ – Програмне забезпечення

РЕБ – Радіоелектронна боротьба

ШІМ – Широтно-імпульсна модуляція

AES – Advanced Encryption Standard (вдосконалений стандарт шифрування)

BEC – Battery Elimination Circuit (схема виключення батареї)

CLI – Command Line Interface (інтерфейс командного рядка)

COTS – Commercial Off-The-Shelf (комерційне готове рішення)

CRSF – Crossfire (протокол керування)

EKF – Extended Kalman Filter (розширений фільтр Калмана)

ELRS – ExpressLRS (протокол керування)

ESC – Electronic Speed Controller (електронний регулятор швидкості)

FHSS – Frequency-Hopping Spread Spectrum (псевдовипадкова перебудова частоти)

FPV – First Person View (вид від першої особи)

GCS – Ground Control Station (наземна станція керування)

GPS – Global Positioning System (глобальна система позиціювання)

I2C – Inter-Integrated Circuit (внутрішньосхемний інтерфейс)

IMU – Inertial Measurement Unit (інерційний вимірювальний блок)

MAVLink – Micro Air Vehicle Link (протокол телеметрії)

MCU – Microcontroller Unit (мікроконтролер)

MSP – MultiWii Serial Protocol (серійний протокол MultiWii)

OSD – On-Screen Display (екранне меню)

RTH – Return To Home (повернення додому)

Rx – Receive (Приймач)

Tx – Transmit (Передавач)

SPI – Serial Peripheral Interface (послідовний периферійний інтерфейс)

UART – Universal Asynchronous Receiver-Transmitter (інтерфейс)

VTX – Video Transmitter (відеопередавач)

ВСТУП

Сучасні безпілотні літальні апарати (БПЛА) вимагають надійних та гнучких систем керування. Це ставить розробників перед вибором: використовувати комерційні готові рішення (COTS) чи розробляти власні, предметно-орієнтовані системи.

Актуальність дослідження полягає у необхідності науково обґрунтованого порівняння цих двох підходів, оскільки відсутній чіткий аналіз їх переваг та недоліків за технічними, програмними та економічними критеріями.

Темою даної роботи є дослідження та поліпшення підсистем керування та телеметрії БПЛА шляхом порівняльного аналізу комерційних і предметно-орієнтованих рішень. Основні завдання включають аналіз існуючих архітектур та загроз, порівняння предметно-орієнтованих та комерційних підсистем та формування реомендацій щодо вдосконалення предметно-орієнтованих рішень.

Об’єктом дослідження є підсистеми керування польотом та телеметрії БПЛА. Предметом дослідження є методи, моделі та алгоритми стабілізації, обробки сенсорів та забезпечення гнучкості ПЗ в обох типах контролерів.

Наукова новизна полягає у теоретичному обґрунтуванні шляхів поліпшення предметно-орієнтованих контролерів, зокрема їх модифікації для інтеграції OSD та сумісності з поширеним відкритим ПЗ.

Практичне значення одержаних результатів полягає у наданні розробникам обґрунтованої бази для прийняття рішень при виборі між предметно-орієнтованою архітектурою та стандартними комерційними рішеннями. Результати аналізу технічних, програмних та економічних параметрів дозволить оцінити доцільність застосування кожного підходу для конкретних умов експлуатації БПЛА.

1 АНАЛІЗ ПІДСИСТЕМ КЕРУВАННЯ ТА ТЕЛЕМЕТРІЇ БПЛА

1.1 Класифікація та застосування БПЛА, роль та складові підсистем керування і телеметрії

Безпілотні літальні апарати (БПЛА), які пройшли шлях від вузькоспеціалізованих військових розробок до широкодоступних комерційних інструментів, сьогодні є невіддільною частиною багатьох галузей. Їх ефективність та функціональність визначаються не лише аеродинамікою, але й досконалістю бортових комп'ютерних систем.

1.1.1 Класифікація та сфери застосування БПЛА

Безпілотні літальні апарати (БПЛА) класифікуються за різними ознаками, що відображають їхні технічні характеристики та функціональне призначення. Основні класифікаційні критерії включають масштаб завдань та клас NATO (STANAG 4670) [1]:

1) Клас I – системи до 150 кг для тактичного рівня. Це найширший клас, який охоплює все: від кишенькових дронів до невеликих літаків. Діляться на додаткові підгрупи: мікро/міні (до 25 кг) та малі (25-150 кг). Їхніми основними завданнями є розвідка, коригування вогню, доставка невеликих посилок чи вибухових снарядів. Прикладами є дрони типу FPV, Mavic, «Лелека-100», «Фурія» і тд. Приклад зображено на рис 1.1.

Рисунок 1.1 Дрони класу I – «Лелека-100» та «Фурія»

2) Клас II – оперативно-тактичний рівень, вага 150-600 кг. Мають змогу нести потужніші сенсори і перебувати в повітрі 10-20 годин. Забезпечують розвідку на глибину десятків і сотень кілометрів. Прикладами є RQ-7 Shadow та «Горлиця» (АН-БК-1), що зображені на рис. 1.29 [2].

Рисунок 1.2 – Дрони класу II – RQ-7 Shadow та «Горлиця»

3) Клас III – стратегічний клас, MALE/HALE (середньовисотні та висотні) системи понад 600кг. Вони є «важкою артилерією» у світі БПЛА. Це дорогі, складні машини, які виконують завдання на рівні командування цілого театру бойових дій або навіть на національному стратегічному рівні. MALE системи працюють на «середніх» висотах (зазвичай 3 000 – 14 000 метрів) і можуть залишатися в повітрі дуже довго (20-40+ годин), що робить їх

ідеальними для розвідки, спостереження та нанесення ударів. HALE системи літають на величезних висотах (понад 18 000 метрів), вище комерційного авіа трафіку, і можуть вести спостереження 24-48 годин і довше. Їхня головна задача – стратегічна розвідка над величезними територіями. Прикладами таких БПЛА є MQ-9B SkyGuardian, RQ-4 Global Hawk, Bayraktar Akıncı, MQ-1C Gray Eagle, Hermes 900, "Сокіл-300". Приклади зображено на рис 1.3 [3][4].

Рисунок 1.3 – Дрони класу III – MQ-9B SkyGuardian та «Сокіл-300»

Дрони класифікують за різними критеріями: кількістю та розташуванням гвинтів або крил, мети використання, дальності польоту, масі корисного навантаження, базового механізму та ін. за технічними характеристиками виділяють дрони таких видів: вертолітного/коптерного, літакового та гібридного VTOL (з гвинтами та крилами), а також Nano та Micro UAV [5][6][7].

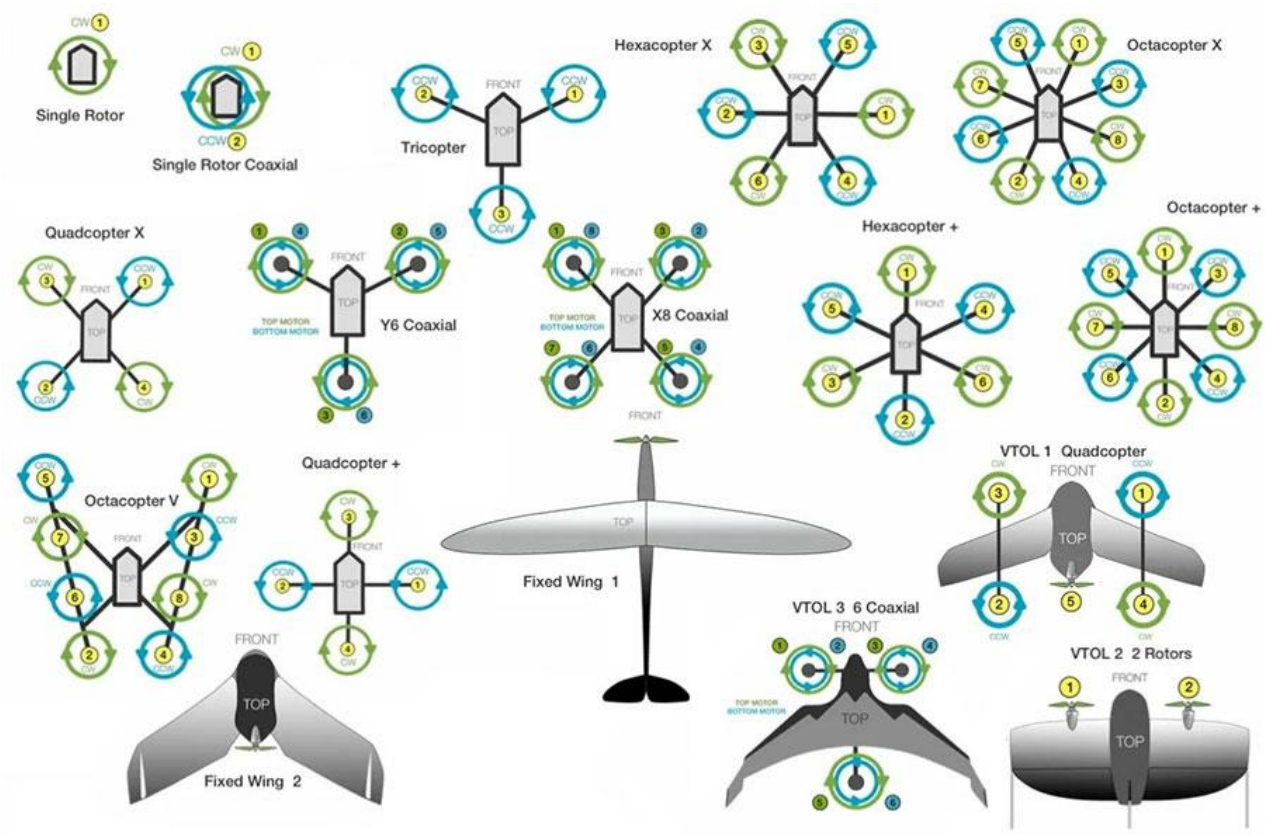


Рисунок 1.4 – БПЛА різних типів

Квадрокоптери та мультикоптери — це компактні БПЛА з кількома роторами, які забезпечують вертикальний зліт і високу маневреність. Вони зручні для польотів у складних умовах, легко запускаються й мають низьку вартість. Проте мають обмежену дальність і тривалість польоту через швидке споживання енергії.

Літакоподібні БПЛА мають крила, завдяки чому можуть долати великі відстані й довго залишатися в повітрі. Вони енергоефективніші та можуть

нести більший вантаж. Недоліки — потреба у злітно-посадковій смузі та менша маневреність.

Гібридні БПЛА поєднують можливості вертикального зльоту мультикоптерів і дальності літаків. Вони універсальні та ефективні на середні дистанції без потреби в інфраструктурі. Мінусами є складність конструкції, вища ціна та складніше обслуговування. Приклад гібридних БПЛА на рис. 1.5.

Рисунок 1.5 – БПЛА гібридного типу MegaDrone Hawk

Цивільні дрони застосовуються аерофотозйомці, моніторингу, сільському господарстві, доставці товарів, інспекції інфраструктури зйомці відео тощо, що зображено на рис. 1.6. Рятувальні та гуманітарні дрони забезпечують пошук і порятунок людей, доставку медикаментів або води у важкодоступні райони.

Рисунок 1.6 – Галузі застосування цивільних дронів

БПЛА широко використовуються для повітряної розвідки (тактичної та стратегічної), коригування артилерійського вогню, спостереження за полем бою, перехоплення повітряних цілей, мінування та розмінування, радіоелектронної боротьби, ретрансляції зв'язку та як ударні платформи. Сучасні військові операції демонструють критичну важливість БПЛА для ведення бойових дій з мінімальним ризиком для персоналу [8][9]. Приклад на рис. 1.7 та 1.8.

Рисунок 1.7 – Приклад військового дрона

Рисунок 1.8 – Дрон бомбардувальник Vampire

Військові безпілотні літальні апарати (БПЛА) відіграють ключову роль у сучасних бойових діях, забезпечуючи широке коло функцій: від розвідки до точкового ураження противника. Їх класифікація здійснюється за низкою ознак, що враховують конструктивні, функціональні та тактичні особливості.

Серед окремих типів військових БПЛА вирізняють літак-носій, що слугує платформою для транспортування та запуску інших безпілотників. Іншим типом є літак-мішень – апарат, призначений для тренування операторів систем ППО та випробування зенітних ракетних комплексів. Значну роль відіграють БПЛА радіоелектронної боротьби (РЕБ), здатні створювати радіоперешкоди, глушити GPS-сигнали, перехоплювати керування іншими дронами або виконувати функції ретрансляції радіосигналу.

Особливу категорію становлять баражуючі боєприпаси, або дрони-камікадзе (рис. 1.9), що містять бойову частину та здатні тривалий час перебувати у повітрі в очікуванні цілі. Після її виявлення та отримання команди або відповідно до закладеного алгоритму, апарат здійснює самознищення, уражаючи ціль. Такі дрони часто мають відносно низьку вартість, що робить їх особливо ефективними проти високовартісної техніки. Також активно використовуються ударні БПЛА, призначені для доставки вибухівки або керованих боєприпасів.

Рисунок 1.9 – Дрон камікадзе Taras 8

Класифікація військових БПЛА за типом управління поділяє їх на автономні, які не потребують постійного контролю людиною; дистанційно керовані оператором; та комбіновані системи, здатні діяти автономно у разі тимчасової втрати зв’язку. За дальністю дії БПЛА можуть охоплювати від надмалих (десятки метрів) до великих (кілька тисяч кілометрів) радіусів дії, що залежить від їхніх цілей та джерел живлення.

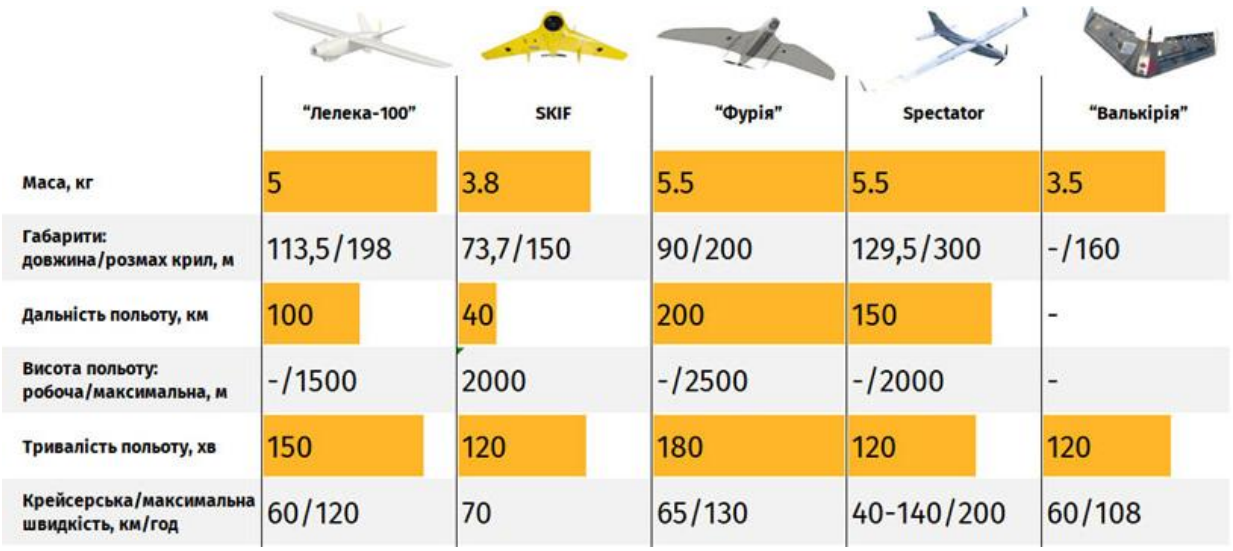
Також апарати поділяються за висотами польоту — від надмалих (десятки метрів) до великих (понад 10 км), що визначає їхню здатність до ураження цілей на різних рівнях або до прихованого спостереження. За тривалістю польоту існують апарати з надмалим ресурсом (кілька хвилин) до надтривалих систем, здатних перебувати в повітрі десятки діб без посадки.

Тип запуску також є важливою ознакою класифікації: БПЛА можуть стартувати з землі, злітати за допомогою злітно-посадкової смуги, катапульти, трампліна, вертикально, з руки, ракетою або з іншого літального апарата (повітряний старт). У випадку повітряного старту можливе повернення дрона на материнське повітряне судно або його одноразове використання.

За функціональним призначенням БПЛА бувають розвідувальними, ударними (як з можливістю використання озброєння, так і самі виступають як зброя), транспортними або універсальними з комбінованими функціями. Крім

того, апарати розрізняють за здатністю до групових дій – від одиночного використання до координації у складі десятків апаратів.

В Україні виготовлення військових дронів розпочалося з 2016 року на авіазаводі «Антонов». Зараз активно використовуються військові тактичні безпілотники "Лелека-100", Spectator-M1, "Фурія", PD-2, "Валькірія", призначені для розвідки. Таблицю з їх характеристиками зображено на рис.1.10.



	"Лелека-100"	SKIF	"Фурія"	Spectator	"Валькірія"
Маса, кг	5	3.8	5.5	5.5	3.5
Габарити: довжина/розмах крил, м	113,5/198	73,7/150	90/200	129,5/300	-/160
Дальність польоту, км	100	40	200	150	-
Висота польоту: робоча/максимальна, м	-/1500	2000	-/2500	-/2000	-
Тривалість польоту, хв	150	120	180	120	120
Крейсерська/максимальна швидкість, км/год	60/120	70	65/130	40-140/200	60/108

Рисунок 1.10 – Порівняльна таблиця розвідувальних дронів

Український ударно-розвідувальний БПЛА Punisher застосовують і в рятувальних операціях, дрон допомагає транспортувати медичні засоби, щоб надавати допомогу пораненим.

Такий широкий спектр типів і характеристик військових безпілотників зумовлює необхідність глибокого розуміння принципів їх дії, можливостей і вразливостей, що є особливо важливим у контексті розробки ефективних засобів захисту та протидії.

Різноманіття типів і можливостей БПЛА дозволяє ефективно використовувати їх як у мирному житті, так і у військових діях. З огляду на активне розширення їх функціоналу, актуальним є питання забезпечення надійного управління та захисту каналів зв’язку між оператором і апаратом.

Більш детальну класифікацію БПЛА продемонстровано в таблиці 1.1.

Таблиця 1.1 – Класифікація БПЛА

Новий критерій	Варіанти
Основне призначення	військові: (для розвідки, ударні, винищувальні, багатоцільові), цивільні: (комерційні, приватні, державні)
Рівень завдань	тактичні, оперативно-тактичні, оперативно-стратегічні
Життєвий цикл	одноразові (камікадзе), багаторазові
Вагова категорія	легкі (малорозмірні), середні, великі, надважкі
Конструкція (Аеродинаміка)	літакового типу, гелікоптерного/коптерного типу, гібридні (VTOL), легші за повітря
Кількість двигунів	без двигуна (планер), один двигун, два двигуни, багато двигунів
Швидкісні характеристики	дуже повільні, малошвидкісні, середньошвидкісні, надзвукові
Час у повітрі	мала тривалість, середня тривалість, велика тривалість
Висота польоту (стеля)	маловисотні, середньовисотні, висотні, стратосферні
Дальність/Радіус дії	ближній, малий, середній, дальній, дуже дальній
Спосіб зльоту	горизонтальний: (з аеродрому, з палуби, з води), вертикальний: (мультипідйомні), інший: (запуск з катапульти, з рук, нетиповий)
Спосіб посадки	горизонтальний: (на аеродром, на палубу, на воду), вертикальний: (мультиспускові), інший: (на парашуті, на трос/мачту, точковий, нетиповий), без посадки
Спосіб керування	дистанційне пілотування, дистанційне керування, повністю автоматичне (за програмою)
Режим польоту	візуальний (в полі зору), приладовий (поза полем зору), комбінований
Місце базування	наземні, морські (корабельні), космічні
Отримання даних (для розвідки)	в реальному часі, сеансами зв'язку, після посадки (з носія)
Система живлення/палива	монозаправні, полізаправні, наземне живлення, бортове, платформне
Тип паливних баків	тільки основні, основні та резервні
Тип крила	фіксовані, плаваючі

1.1.2 Архітектура та компоненти підсистеми керування

Архітектура системи керування БПЛА складається з кількох ієрархічних рівнів: високого (когнітивного) для планування, середнього для взаємодії та низького (реактивного) для безпосереднього керування польотом [10]. Центральним елементом є автопілот, що інтегрує дані з інерціальної навігаційної системи (IMU з гіроскопами та акселерометрами), GPS/GNSS приймачів, барометрів, магнітометрів, часто використовуючи PID-контролери для стабілізації [11][12][13]. Польотний контролер (ПК) виконує функції автоматичної стабілізації, керування моторами через електронні регулятори швидкості (ESC) та моніторинг сенсорів.

1.1.3 Підсистеми телеметрії, зв'язку та наземні станції

Телеметричні системи забезпечують двосторонній обмін даними між БПЛА та наземною станцією керування (GCS), передаючи положення, швидкість, стан батареї та відео. MAVLink (Micro Air Vehicle Link) є поширеним протоколом телеметрії, що забезпечує легку та ефективну комунікацію і сумісність з системами ArduPilot та PX4. Архітектура каналів зв'язку включає Uplink для команд керування (кілька кГц) та Downlink для телеметрії і відео (1-10 Мбіт/с), використовуючи діапазони UHF, C-band та Ku-band. Система зв'язку включає бортовий термінал (ADT), GCS, антени та, для операцій BVLOS, супутникові канали або 5G/LTE. Наземна станція керування (GCS) складається з апаратного та програмного забезпечення (напр., Mission Planner, QGroundControl) для планування місій та моніторингу.

У сучасних БПЛА, особливо в FPV-системах, зв'язок доцільно розглядати як три окремі потоки:

1) Канал керування (RC Link) – "Uplink" від пульта до апарата, що передає команди пілота через швидкісні протоколи на кшталт ELRS або CRSF. Це критично для миттєвої реакції дрона.

2) Канал телеметрії (Telemetry Link) – "Downlink. Дані про стан апарата можуть повертатися назад на пульт, оскільки ті ж ELRS/CRSF є двосторонніми. Для серйозних автономних місій вони можуть передаватися окремим потоком за протоколом MAVLink на ноутбук із наземною станцією керування (GCS). Втім, у FPV найчастіше використовується OSD (On-Screen Display) – технологія, що накладає дані телеметрії поверх відеосигналу.

3) Канал відео – найширший "Downlink", що транслює картинку з камери на окуляри пілота.

При цьому, повноцінну GCS на кшталт Mission Planner не слід плутати зі звичайним пультом та окулярами. GCS – це комплексне ПЗ для планування автономних місій "за точками" та глибокого аналізу, тоді як FPV-пілот, що покладається на OSD, може взагалі не використовувати GCS у польоті.

Відтак, цілісність системи якраз і визначається тим, наскільки глибоко ці три канали (керування, телеметрія та відео) інтегровані між собою. Саме тут комерційні рішення та предметно-орієнтовані збірки демонструють кардинально різні підходи.

1.2 Порівняльний аналіз програмних екосистем з відкритим кодом (INAV, ArduPilot, Betaflight) та середовищ розробки (Arduino IDE)

Програмне забезпечення є "свідомістю" польотного контролера, що визначає його функціональність. Вибір програмної платформи є фундаментальним компромісом між готовою, потужною функціональністю та повною гнучкістю предметно-орієнтованої розробки.

У контексті даного дослідження, програмні рішення доцільно розділити на два фундаментально різні підходи: повноцінні програмні екосистеми (готові прошивки) та середовища для предметно-орієнтованої розробки (створення прошивки з нуля).

Сучасна розробка програмного забезпечення для БПЛА базується на трьох основних екосистемах з відкритим кодом: ArduPilot, PX4, Betaflight та INAV. Кожна з цих платформ має унікальні характеристики, переваги та цільову аудиторію [14].

1.2.1 ArduPilot: Універсальна автопілотна платформа

ArduPilot є найбільш зрілою та всеосяжною екосистемою з відкритим кодом для автономного керування БПЛА. Система підтримує понад 700,000 рядків перевіреного коду та забезпечує керування різноманітними типами транспортних засобів: мультикоптерами, літаками, вертольотами, наземними роверами, підводними апаратами та VTOL-платформами [15][16].

Платформа використовує монолітну архітектуру з інтегрованою комунікацією через MAVLink протокол. Система забезпечує високу надійність завдяки подвійній інерційній системі навігації (dual-EKF3), яка голосує між показаннями датчиків на рівні оцінювача [17].

ArduPilot пропонує 25+ режимів польоту для мультикоптерів, включаючи повністю автономні режими: AUTO (виконання місій за waypoint), LOITER (утримання позиції), RTL (повернення додому), GUIDED (керування з наземної станції), CIRCLE (автоматичний політ по колу). Для літаків доступні режими FBWA/FBWB (fly-by-wire), CRUISE (відстеження курсу), AUTOTUNE (автоматичне налаштування) та спеціалізовані режими як THERMAL для планерів [18][19].

Програмне забезпечення розповсюджується під ліцензією GPLv3 [20]. Це означає, що будь-які похідні роботи повинні також розповсюджуватися під GPL, а вихідний код змін має бути доступний тим, кому надається бінарний код. Проте GPL не вимагає публічного розміщення коду — достатньо надати його кінцевим користувачам продукту. Функції можна додавати через Lua-скрипти або супутні комп'ютери, які знаходяться поза межами GPL.

Наземні станції керування включають Mission Planner (Windows), QGroundControl (кросплатформенна), APM Planner 2. Система підтримує симуляцію SITL (Software-In-The-Loop), яка домінує в академічних публікаціях завдяки простоті використання.

1.2.2 PX4: Модульна платформа для комерційного використання

PX4 розроблена з мікроядерною архітектурою, де центральною є система публікації/підписки uORB (micro Object Request Broker). Кожен драйвер датчика публікує дані як топіки з мітками часу, а модулі оцінювання та керування підписуються на ці топіки. Така архітектура забезпечує низьку латентність (<10 мс на референсній платі FMUv6X) та високу модульність.

Модульна структура PX4 дозволяє легко замінювати компоненти системи, додавати нові датчики та інтегрувати з ROS 2 через fastDDS міст. Голосування між датчиками відбувається на рівні драйверів, що ізолює збої окремих компонентів.

PX4 використовує ліцензію BSD 3-clause [21]. На відміну від GPL, BSD дозволяє створювати похідні продукти з комерційними обмеженнями без необхідності розкриття вихідного коду. Це робить PX4 привабливою для бізнесу та оборонних застосувань, де захист інтелектуальної власності є критичним.

PX4 домінує в сегменті доставки дронів (Amazon MK30, Zipline Sparrow) завдяки своїй ліцензії та сучасній архітектурі. Станом на 2024 рік, вартість кодової бази PX4 оцінюється понад 1 мільярд доларів США при обліку 59,7 млн рядків коду.

1.2.3 Betaflight: Лідер у FPV-сегменті

Betaflight є найпопулярнішою прошивкою для FPV-дронів, зосередженою на максимальній продуктивності польоту, відгуку та точності

керування. Створена у 2015 році як форк Cleanflight, Betaflight став стандартом індустрії для рейсингових та фрістайл-квадрокоптерів [22].

Платформа підтримує найширший спектр польотних контролерів на базі STM32 F4, G4, F7 та H7 процесорів. Майже кожен FC на ринку розроблений передусім для Betaflight. Прошивка підтримує DShot (150/300/600), Multishot, Oneshot протоколи моторів, двонаправлену телеметрію ESC, RPM-фільтрацію для усунення резонансів [23][24].

Betaflight пропонує найскладнішу систему налаштування PID з використанням слайдерів для спрощеного тюнінгу. Основні параметри включають: Damping (D-коефіцієнти), Tracking (P і I), Stick Response (Feedforward), Dynamic Damping (D Max), та окремі налаштування для осей Pitch/Roll. Система підтримує автоматичне налаштування через режим AUTOTUNE та інструменти як PIDtoolbox для аналізу польотних логів [25].

Платформа не призначена для повноцінної GPS-навігації або автономного польоту. Доступний лише базовий режим GPS Rescue для повернення додому в екстрених ситуаціях. Відсутня підтримка fixed-wing літаків та складних автономних місій.

1.2.4 INAV: GPS-навігація та автономія для любителів

INAV є форком Betaflight, оптимізованим для GPS-навігації, waypoint-місій та автономного польоту. Інтерфейс INAV дуже схожий на Betaflight, що полегшує перехід для користувачів.

Платформа підтримує автоматичне виконання waypoint-місій з можливістю планування маршруту в Mission Control. Система забезпечує режими Position Hold (утримання позиції), Return to Home (повернення додому з автоматичною посадкою), Cruise Mode, ALTITUDE HOLD. Для fixed-wing INAV пропонує найкращу підтримку серед легких прошивок, включаючи автоматичний зліт та посадку.

INAV версії 2.6 використовує Alpha-Beta фільтр (комплементарний фільтр) для оцінки позиції на основі акселерометра та GPS. Це простіше ніж розширений фільтр Калмана в ArduPilot, але забезпечує достатню точність для більшості застосувань. INAV вимагає компаса, барометра та GPS для режимів RTH, що додає складності налаштування порівняно з Betaflight [17].

Налаштування місій включає параметри: waypoint_radius (розмір waypoint), nav_wp_safe_distance (максимальна відстань першого waypoint від точки злету, за замовчуванням 100м), enforce_altitude_at_waypoint (точність висоти), waypoint_tracking_accuracy (точність відстеження шляху). Система підтримує круїзну швидкість для fixed-wing та автоматичне орбітування [26].

INAV поступається Betaflight у якості manual-польоту та тюнінгу. Відсутня підтримка bidirectional DShot, обмежене управління ресурсами, менша спільнота розробників. Для складних автономних місій ArduPilot залишається кращим вибором.

Саме INAV обрано в дослідженні як зразок для типового комерційного рішення, оскільки він поєднує масовість і доступність FPV-платформ з потужними навігаційними функціями.

1.2.5 Arduino IDE: Середовище для предметно-орієнтованої розробки

На противагу готовим екосистемам стоїть підхід, обраний для моделювання предметно-орієнтованої системи в даній роботі. Він полягає у використанні базового середовища розробки для написання власного коду з нуля.

Arduino IDE широко використовується для освітніх та прототипових проектів квадрокоптерів. Популярні платформи включають Arduino Uno (ATmega328P), Arduino Mega (ATmega2560) та Arduino Nano для мініатюрних дронів.

Простота програмування через зрозумілу мову на базі C/C++; Величезна бібліотека готових модулів для датчиків (MPU6050, BMP180, HMC5883, GPS); Доступність та низька вартість (Arduino Uno ~$5-10); Готові проекти з відкритим кодом (MultiWii firmware для Arduino); Можливість швидкого прототипування без глибоких знань електроніки [27].

Обмеження Arduino для польотних контролерів: Arduino платформи мають значні обмеження для серйозних БПЛА-застосувань: Обмежена обчислювальна потужність (16 МГц для Uno, 16 МГц для Mega) порівняно з 168-480 МГц для Pixhawk/STM32; Відсутність апаратного FPU (floating point unit) для складних обчислень; Обмежена оперативна пам'ять (2-8 KB) недостатня для складних алгоритмів фільтрації та навігації; Відсутність вбудованих IMU та барометрів високої якості; Складність реалізації 250+ Гц контрольних циклів для стабільної стабілізації.

Використання Arduino як допоміжного контролера: Arduino може ефективно використовуватись як допоміжний контролер для специфічних завдань: GSM/GPRS зв'язок для керування БПЛА поза прямою видимістю, обробка додаткових сенсорів, керування корисним навантаженням. У такій конфігурації Arduino спілкується з основним автопілотом (Pixhawk/APM) через MAVLink протокол.

Предметно-орієнтоване рішення хоч і використовує середовище розробки Arduino IDE, але базується на мікроконтролері STM32F401. Це 32-бітний ARM Cortex-M4 процесор з тактовою частотою 84 МГц та апаратним блоком FPU, що нівелює більшість зазначених обмежень 8-бітних платформ (продуктивність, пам'ять, FPU) і дозволяє створити значно більш потужне та конкурентоздатне рішення.

1.2.6 Критерії вибору програмної платформи

Таким чином, проведений аналіз програмних екосистем дозволяє чітко розмежувати сфери їх застосування, базуючись на балансі між надійністю, функціональністю, продуктивністю та ліцензійними умовами.

Так, ArduPilot залишається вибором для завдань, де критична надійність, перевірена стабільність та потрібні складні автономні місії. Його здатність підтримувати різноманітні типи апаратів (літаки, ровери, підводні човни) та філософія відкритого коду (GPLv3) роблять його стандартом для наукових та комплексних проєктів.

Натомість, PX4 займає нішу комерційної розробки. Його ключові переваги — модульна архітектура, інтеграція з ROS 2 та гнучка BSD-ліцензія, яка дозволяє компаніям створювати закриті комерційні продукти, захищаючи свою інтелектуальну власність.

Сегмент FPV-перегонів та фрістайлу безроздільно належить Betaflight. Весь проєкт оптимізовано під максимальну продуктивність, найнижчу затримку ручного керування та найширшу підтримку FPV-обладнання, свідомо жертвуючи при цьому складними навігаційними функціями (окрім базового GPS Rescue).

Цю прогалину закриває INAV – вибір для ентузіастів, яким потрібні GPS-можливості (політ за точками, повернення додому) та якісна підтримка літаків (fixed-wing), але на доступній і простій FPV-апаратній базі, знайомій по Betaflight.

Arduino IDE не розглядається як прямий конкурент, а скоріше як інструмент для освітніх проєктів, швидкого прототипування або розробки допоміжних функцій чи систем. Простота коду та низька вартість є важливішими за високу продуктивність чи складну навігацію.

Компроміс між гнучкістю (Arduino IDE з STM32) та функціональністю (INAV) і є предметом подальшого теоретичного аналізу в даній роботі.

1.3. Аналіз протоколів керування та телеметрії (ELRS, MAVLink, CRSF)

Програмні екосистеми та апаратні компоненти повинні спілкуватися між собою стандартизованою мовою. Протокол визначає, як саме пакуються, надсилаються та приймаються команди керування, пакети телеметрії та параметри місії. Вибір протоколу безпосередньо впливає на дальність, надійність, затримку та функціональність системи.

Ці протоколи можна умовно розділити на дві групи: "важковаговики" для автономних місій (MAVLink) та "спринтери" для FPV-польотів (CRSF, ELRS).

1.3.1. MAVLink: Стандарт для автономних місій

Для складних систем, орієнтованих на автономність (ArduPilot, PX4), MAVLink є галузевим стандартом.

"MAVLink (Micro Air Vehicle Link) є стандартним протоколом телеметрії для ArduPilot та PX4. Протокол версії 2.0 має лише 14 байт накладних витрат, підтримує до 255 одночасних систем, двосторонню комунікацію, автентифікацію повідомлень та backward compatibility з версією 1.0. MAVLink дозволяє передачу команд, телеметрії, параметрів місії та статусу через UDP, TCP, серійний порт або WiFi."

Ця архітектура оптимізована для повноцінної взаємодії з наземними станціями (GCS) та передачі великих обсягів даних, необхідних для планування місій (waypoints) та аналізу стану апарата.

1.3.2. Протоколи FPV-сегменту: CRSF та ExpressLRS (ELRS)

Для систем, де пріоритетом є миттєва реакція пілота (Betaflight, INAV), використовуються інші протоколи, де головне — низька затримка.

"Betaflight та INAV також підтримують FrSky SmartPort, CRSF (Crossfire), HoTT та інші протоколи телеметрії. CRSF забезпечує найнижчу латентність для FPV застосувань."

ExpressLRS (ELRS) — це проєкт з відкритим кодом, який взяв за основу принципи CRSF, але вивів їх на новий рівень завдяки використанню технології LoRa та досягненню надзвичайно низьких затримок при великій дальності.

Важливою тенденцією є зближення цих двох світів. Наприклад, "ExpressLRS з 2024 року підтримує нативний MAVLink для Betaflight (версія 2025.12.0-beta+), дозволяючи використовувати один радіоканал для RC-керування та телеметрії замість окремого телеметричного радіо." Це показує, що навіть FPV-системи починають інтегрувати потужні телеметричні можливості, які раніше були притаманні лише ArduPilot/PX4.

Логічний висновок: Таким чином, ми бачимо два різні підходи: багатий на дані та команди MAVLink для автономних польотів і надшвидкі CRSF/ELRS для ручного пілотування.

Але всі ці протоколи, незалежно від їхньої складності, мають одну спільну ахіллесову п'яту: вони працюють в радіоефірі. Це робить їх вразливими.

1.4 Аналіз основних загроз каналам керування та телеметрії (на основі курсової роботи)

Радіоканали, що поєднують оператора з БПЛА, є найбільш критичною та водночас найбільш вразливою ланкою всієї системи. На відміну від апаратних чи програмних збоїв, які можна мінімізувати тестуванням, канали

зв'язку функціонують у відкритому ефірі, що робить їх головною мішенню для зовнішнього впливу. Аналіз, проведений у попередній курсовій роботі, дозволяє виділити три основні класи загроз, кожна з яких має на меті або перервати, або перехопити керування місією.

Зв’язок між пілотом і БПЛА є критичною частиною системи управління безпілотником, і кожен метод порушення цього зв’язку несе серйозні ризики – від втрати контролю до повного захоплення пристрою.

1.4.1 Перехоплення даних (Sniffing)

Це пасивна атака, яка не втручається в роботу системи, а лише "слухає" радіоефір. Якщо канал не захищений або передає дані у відкритому вигляді, перехоплення стає дуже простим. Використовуючи спеціалізовані приймачі SDR (Software-Defined Radio) або доступні засоби типу RTL-SDR, зловмисник може зібрати телеметричну інформацію (координати, швидкість, висоту, навігаційні дані), сигнали управління (повороти, газ, режими), а також відеопотік у разі аналогової або відкритої цифрової трансляції. Це дозволяє сторонній особі непомітно моніторити політ, аналізувати поведінку дрона, а також підготувати активні атаки. Приклад пристрою прослуховуваннння на рис. 1.11.

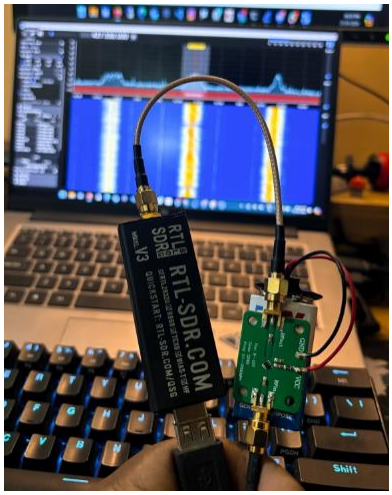


Рисунок 1.11 – Пристрій для прослуховування радіоефіру RTL-SDR Blog V4 Dongle

Небезпека полягає в тому, що жодних явних ознак атаки пілот може не помітити – все працює, як і раніше, але інформація вже перебуває в руках третьої сторони.

1.4.2 Спуфінг (Spoofing)

Цей тип атаки активний і набагато небезпечніший. Спуфінг GPS означає створення фальшивого сигналу GPS, який потужніший за справжній. Приймач дрона приймає ці фальшиві дані за реальні та починає орієнтуватися за ними, змінюючи свій курс, висоту або координати. Зловмисник може змусити дрон подумати, що він летить у правильному напрямку, хоча насправді той може сісти в зовсім іншому місці або взагалі вилетіти за межі дозволеної зони. Приклад пристрою для спуфінгу на рис. 1.12.

Рисунок 1.12 – Пристрій для GPS спуфінгу

Ще небезпечніший спуфінг команд управління. Якщо зловмисник може видати себе за пілота або наземну станцію – особливо за відсутності шифрування чи автентифікації – він може посилати власні команди: змінювати режим польоту, вимикати двигуни, відключати повернення додому (RTH), або навіть спрямувати дрон на ворожу ціль. Схема процесу спуфінгу на рис.1.13.

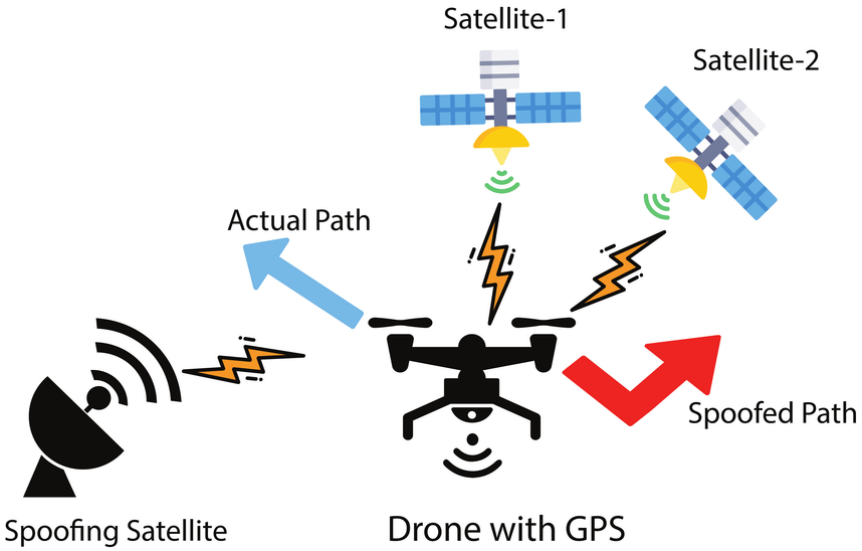


Рисунок 1.13 – Візуальна схема GPS спуфінгу

1.4.3 Глушіння (Jamming)

Це створення сильного шуму в діапазоні, на якому працює зв’язок дрона. Зазвичай застосовується в 2.4 ГГц (Wi-Fi, RC) або 5.8 ГГц (FPV відео) діапазонах, але може впливати і на GPS (1.5 ГГц). Високопотужний передавач, який створює широкий спектр шуму, перекриває сигнал керування або навігації. В результаті дрон "втрачає" пілота: не отримує команд або не бачить супутників.

Якщо у БПЛА є функції автоматичного повернення додому, він може врятуватися. Але без цього або при цілеспрямованому глушінні таких функцій апарат може впасти, зависнути в повітрі, або стати легкою здобиччю. Приклад пристрою глушіння на рис. 1.14.

Рисунок 1.14 – Глушилка FPV дронів
Wartex GETMAN FPV+MAVIC 280W

1.4.4 Атаки «людина посередині» (MITM)

MITM — це складна атака, яка потребує втручання у трафік між пілотом і дроном. Зловмисник створює фіктивний шлюз зв’язку, через який проходить весь трафік (рис. 1.15). Наприклад, він може розмістити підроблену наземну станцію, що приймає команди від пілота, змінює їх, а потім передає дрону.

Аналогічно, телеметрія від дрона може бути змінена — щоб показувати фальшиві координати, швидкість або помилкову інформацію про системи дрона. Це не лише дезорієнтує оператора, але й дозволяє приховано взяти контроль над апаратом, відхилити його від маршруту чи вивести з ладу.

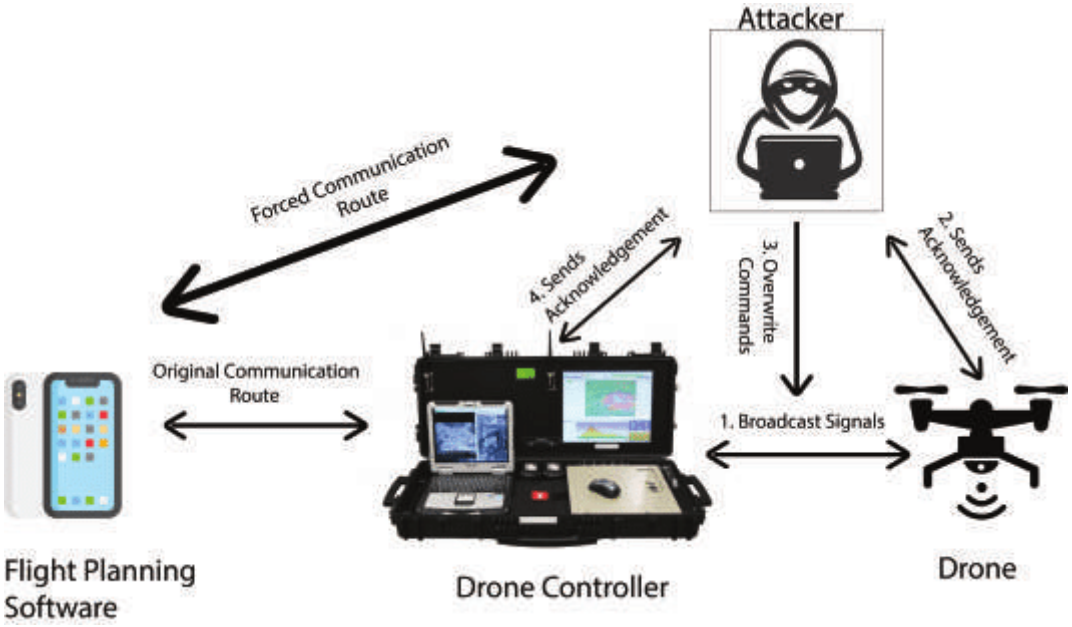


Рисунок 1.15 – Візуальна схема MITM

1.4.5 Інші види атак

Відмова в обслуговуванні (DoS/DDoS). Ця атака частіше трапляється у системах, що використовують TCP/IP протоколи, тобто передавання по Wi-Fi або LTE. Суть полягає в перевантаженні каналу фальшивими запитами, великою кількістю інформації або пакетами даних, які не дають змоги справжнім командам досягти цілі.

В результаті пілот не може підключитися до дрона, або зв’язок постійно переривається. У випадку DDoS — одночасна атака з кількох джерел — можна паралізувати роботу навіть складної системи керування або серверів, що обслуговують зв’язок. Це особливо небезпечно для розподілених мереж керування або хмарних платформ.

Фізичний доступ до модулів зв’язку. Якщо дрон втрачається або захоплюється, фізичний доступ до нього дає змогу підключитися через порти UART, I2C, USB або інші інтерфейси. В багатьох випадках інформація, включно з ключами шифрування, параметрами зв’язку, MAC-адресами чи навіть журналами польоту, не захищена і зберігається у відкритому вигляді.

Зловмисник може зчитати конфігурацію, витягти ключі, змінити прошивку, встановити бекдор і повернути дрон — який ззовні виглядає

справним, але вже повністю контрольований третім суб’єктом. Такі дії також дозволяють відтворити чи клонувати дрон для подальших атак.

Ін’єкція шкідливого ПЗ (Firmware Injection). Цей метод схожий на атаку через оновлення. Зловмисник, маючи доступ до каналу оновлень або перепрошивки дрона, може вмонтувати модифікований мікрокод. Наприклад, через Wi-Fi або Bluetooth, особливо якщо не використовується підпис цифровим сертифікатом.

Таке ПЗ може змінювати поведінку пристрою, створити прихований зв’язок із зовнішнім сервером, передавати координати, вмикати камери, або просто вимикати двигуни на команду. Виявити подібні загрози важко без детального аудиту коду та систем захисту на рівні заліза.

1.4.6 Протидія основним загрозам

Протидія загрозам каналам БПЛА вимагає комплексного, багаторівневого підходу, що охоплює як програмні, так і апаратні рішення.

Захист від перехоплення та MITM-атак. Ключовим є шифрування в реальному часі. Для цього використовують симетричне шифрування AES (для відеопотоку), приклад якого на рис. 1.16 та протоколи TLS/DTLS (для IP-мереж), які забезпечують і шифрування, і автентифікацію. Для запобігання довготривалого дешифрування застосовують автоматичне оновлення сесійних ключів (напр., Діффі-Геллман). Найнадійнішим є зберігання ключів у апаратних криптомодулях (TPM), що унеможливлює їх вилучення.

Захист від спуфінгу. Для протидії підміні GPS-сигналу використовується INS (інерційна навігаційна система), яка дозволяє апарату тимчасово орієнтуватися за гіроскопами та акселерометрами. Для захисту від підміни команд застосовують цифрові підписи (напр., ECDSA), що автентифікують оператора, та "nonce" (одноразові токени) для запобігання replay-атакам. Приклад дрона з INS системою на рис. 1.17.

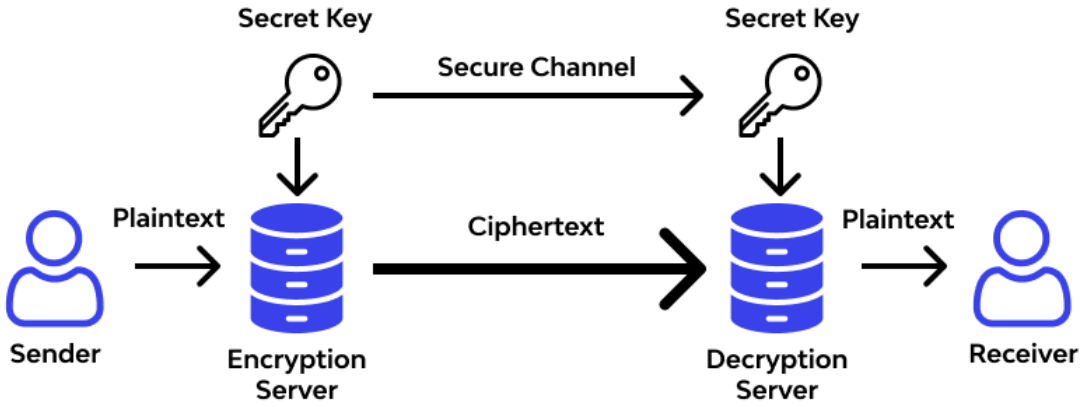


Рисунок 1.16 – Візуальна схема AES алгоритму

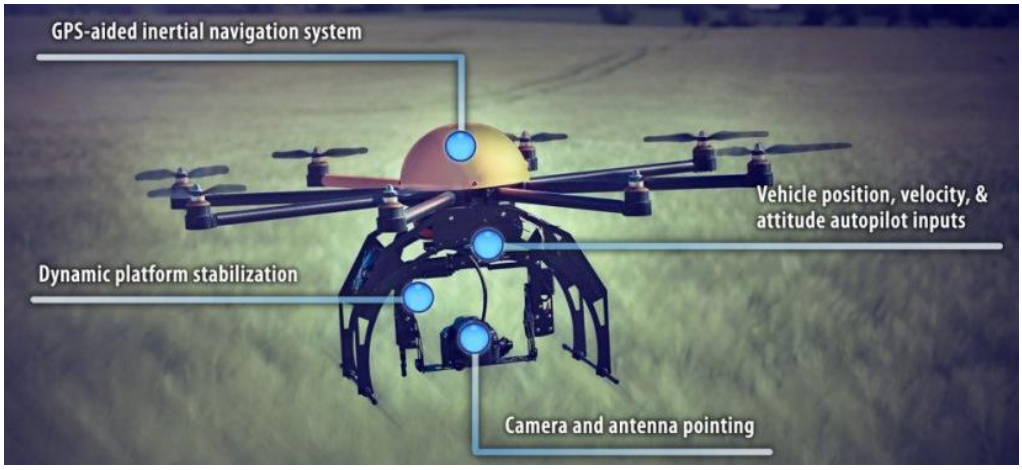


Рисунок 1.17 – Приклад дрона з INS системою

Захист від глушіння (Jamming): Оскільки глушіння є "білим шумом" на робочій частоті, головна протидія — ускладнити її визначення. Це досягається технологією FHSS (Frequency-Hopping Spread Spectrum), де приймач і передавач синхронно змінюють частоти. Додатково використовуються спрямовані антени (наприклад, Yagi, параболічні) (рис. 1.18) та резервні канали зв'язку (LTE, Wi-Fi). Як останній засіб, активуються програмні механізми (RTH, автопосадка).

Захист від DoS/DDoS-атак: Для запобігання перевантаженню процесора дрона зайвими запитами, на рівні прошивки впроваджують rate limiting (обмеження частоти запитів), фільтрацію за "білими списками"

(дозволені IP/MAC-адреси) та пріоритизацію каналу керування над каналом даних.

Рисунок 1.18 – Yagi та параболічна антени

Захист від фізичного доступу та ін'єкцій ПЗ: Навіть при захопленні апарата, дані та прошивка мають бути захищені. Це забезпечується шифруванням Flash-пам’яті, механізмом Secure Boot (перевірка цифрового підпису прошивки при завантаженні) та підписаними оновленнями. Усі ключі верифікації при цьому мають зберігатися в апаратному TPM/HSM.

1.4.7 Новітні методи управління та передачі даних БПЛА

Тенденція розвитку БПЛА рухається в бік цифрових та неконвенційних методів зв'язку для роботи в екстремальних умовах.

Перший перспективний напрямок — ройові системи, що використовують оптичний зв'язок Li-Fi (через світлодіоди). Перевагами є надвисока швидкість передачі даних, відсутність радіоінтерференцій та висока безпека, бо світловий сигнал важко перехопити. Однак технологія обмежена необхідністю прямої видимості та чутливістю до погодних умов (дим, туман).

Приклади на рис 1.19.

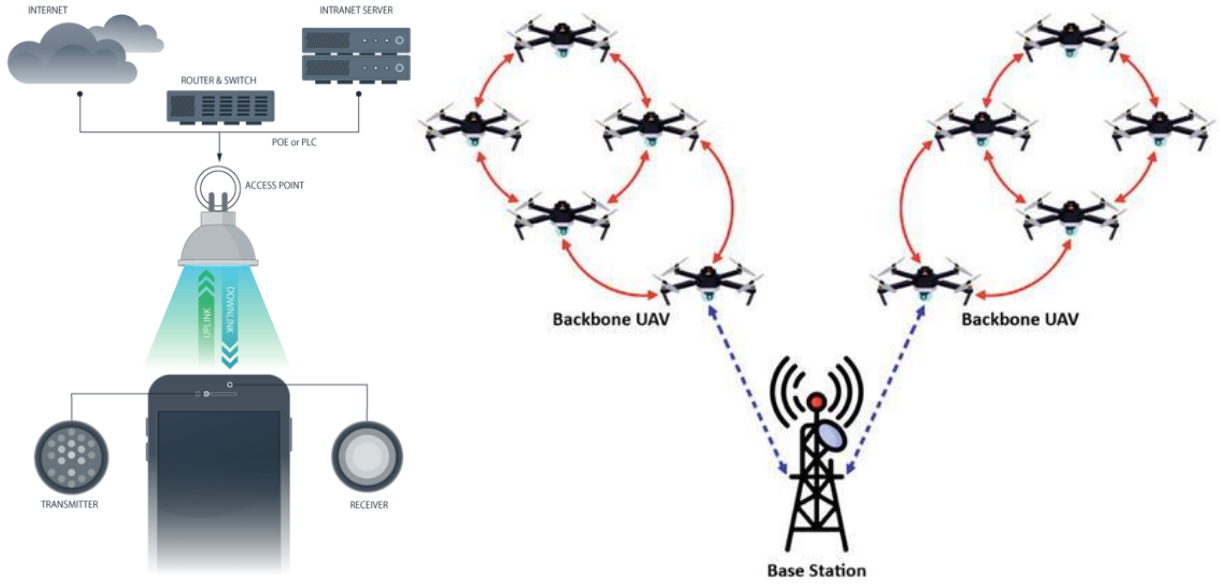


Рисунок 1.19 – Візуальна схема Li-Fi та схема роботи рою

Іншим підходом для забезпечення стабільного зв'язку є використання оптоволоконних кабелів. Ця технологія дозволяє створити надзвичайно швидкий і стабільний канал для передачі даних, що особливо важливо, коли потрібно передавати великі обсяги інформації, такі як відео високої чіткості або складні телеметричні дані. Оптоволокно забезпечує високу швидкість і захист від електромагнітних перешкод, оскільки сигнал передається через волокно, а не через радіохвилі.

Оптоволоконні дрони (рис. 1.20) є інноваційною технологією, яка має як суттєві переваги, так і значні недоліки. Однією з головних переваг є можливість забезпечення стабільного зв’язку між оператором і дронами. На відміну від традиційних радіокерованих дронів, оптоволоконний зв'язок не піддається впливу радіоелектронних перешкод, що дозволяє значно покращити точність управління. Проте, як і в будь-якої нової технології, оптоволоконні дрони мають низку обмежень, які необхідно враховувати при їх використанні.

Експерти виділяють кілька основних аспектів, які визначають ефективність використання оптоволоконних дронів у бойових умовах [28].

Рисунок 1.20 – Український БПЛА на оптоволокні «Громила»

Перший аспект — це маневровість. Через прив’язку до тонкого кабелю конструкція таких дронів здатна виглядати вразливою, адже будь-яка перешкода, наприклад, дерево чи стовп, може призвести до зупинки дрону. Це обмежує їхню дальність польоту та можливості, змушуючи використовувати дрони здебільшого в районах, де немає фізичних перешкод, наприклад, вздовж доріг. Однак інженер Максим Шеремет, який має досвід керування такими дронами, заперечує ці твердження. Він пояснює, що кабель під час польоту автоматично розмотується і спокійно лягає поверх дерев чи інших перешкод, що ускладнює його обрив. Шеремет також зазначає, що наявність оптоволоконного зв’язку значно полегшує керування дронами: оператор завжди має стабільну картинку, і не потрібно враховувати зони РЕБ або радіогоризонт.

Другим аспектом є висока вартість оптоволокна. Це стало однією з основних причин скепсису, особливо з боку українського Генштабу, який спочатку назвав ці дрони «неактуальними». Минулого року котушка з 10 км оптоволоконного кабелю коштувала близько 3 тисяч доларів, що майже в десять разів дорожче, ніж звичайний FPV-квадрокоптер. Однак ситуація змінилася завдяки великій кількості китайської продукції на ринку та початку

виробництва в Україні, що дозволило знизити ціну до 100 доларів за котушку. Це призвело до зниження вартості самого дрону до приблизно 1000 доларів.

Третім аспектом є обмежене корисне навантаження оптоволоконного дрона. Дрон, крім боєприпасу, повинен нести важку котушку з кабелем. Наприклад, котушка на 10 км важить приблизно 1,2 кг, а на 15 км — близько 2 кг. Це означає, що 10-дюймовий FPV-дрон із корисним навантаженням у 3-4 кг зможе нести лише боєприпас вагою до 1,5 кг. Проте чим більший дрон, тим вища його вантажопідйомність. Такі дрони можуть здійснювати точні удари по техніці, оскільки завдяки стабільній відеотрансляції через оптоволокно оператор завжди має чітке зображення і може більш точно прицілюватися.

Також одним із основних недоліків є обмежена маневровість. Оскільки дрон підключений до оператора через оптоволоконний кабель, його рухи обмежені довжиною кабелю, що значно знижує його гнучкість. Дрон може застрягнути на фізичних перешкодах, таких як дерева або будівлі, що обмежує його дальність польоту та можливості використання на різних територіях. Таким чином, оптоволоконні дрони найкраще підходять для використання на відкритих просторах або уздовж доріг, де немає великих фізичних бар’єрів.

Ще однією проблемою є обмежене корисне навантаження. Дрон, який використовує оптоволоконний кабель, змушений нести важку котушку, що обмежує його здатність транспортувати більші боєприпаси. Наприклад, дрон із котушкою на 10 км може нести лише невеликий боєприпас вагою до 1,5 кг. Це знижує ефективність таких дронів порівняно з радіокерованими, які можуть нести більші навантаження.

Незважаючи на це, такі дрони мають і значні переваги, зокрема, стабільність зв’язку та можливість здійснювати більш точні удари завдяки постійному відео-контролю, що забезпечується через оптоволоконний кабель. Також, оптоволоконні дрони здатні досягати високої точності при управлінні в умовах радіоелектронних перешкод, оскільки вони не залежать від радіосигналів.

З точки зору протидії оптоволоконним дронам, виявлення та знищення таких апаратів є складним завданням. Зазвичай, для виявлення оптоволоконних дронів використовують оптичні сенсори або акустичні засоби, оскільки традиційні радіолокаційні системи не здатні ефективно виявляти такі дрони через відсутність радіосигналу. Фізичне знищення таких дронів може здійснюватися за допомогою сіткометів або навіть дробовиків.

Хоча оптоволоконні дрони можуть стати важливим інструментом у військових операціях, вони мають низку суттєвих обмежень, які потребують подальшого вдосконалення технології. Це ускладнює їхнє масове використання, але, з іншого боку, вони можуть стати ефективним засобом для виконання специфічних завдань, таких як точні удари в умовах радіоелектронних перешкод.

Таким чином, кожен з цих підходів має свої переваги та обмеження. Li-Fi підходить для ситуацій, де потрібна висока швидкість передачі даних і є можливість підтримувати прямий зв'язок між дронами, а оптоволокно забезпечує стабільний, високошвидкісний зв'язок з високим рівнем захисту від перешкод, але обмежує мобільність дронів. Вибір між цими технологіями залежить від конкретних вимог місії, таких як дальність польоту, необхідність у стабільному зв'язку та рівень автономії.

1.5 Постановка задачі дослідження та обґрунтування критеріїв теоретичного порівняння

Проведений у першому розділі аналіз апаратних платформ (п. 1.1), програмних екосистем (п. 1.2), протоколів (п. 1.3) та, що особливо важливо, основних загроз каналам зв’язку (п. 1.4) дозволяє чітко окреслити фундаментальну проблему сучасного проєктування БПЛА.

1.5.1 Постановка задачі дослідження

Аналіз виявив ключовий компроміс, що стоїть перед розробником:

З одного боку, існують комерційні (COTS) рішення (напр., польотні контролери SpeedyBee F405 з прошивкою INAV). Вони пропонують надзвичайно високу функціональність "з коробки": надійні GPS-режими, інтегроване OSD, оптимізовані фільтри (як EKF) та підтримку новітніх протоколів. Однак їхні недоліки — це "чорна скринька" та стандартизація. Вони є відомою мішенню для загроз (п. 1.4), а їхня закрита або складна архітектура суттєво обмежує можливості глибокої модифікації для імплементації нестандартних методів захисту.

З іншого боку, існують предметно-орієнтовані рішення, наприклад, змодельовані на базі STM32F401 та Arduino IDE. Їхня головна перевага – повна прозорість коду та апаратна гнучкість. Це дає теоретичну можливість імплементувати будь-які специфічні алгоритми, включно з посиленими методами шифрування та протидії спуфінгу. Однак їхній недолік — низька початкова функціональність. Базові реалізації поступаються комерційним у точності фільтрації, не мають OSD та вимагають більше часу на розробку.

Таким чином, виникає науково-технічна задача: теоретично дослідити, чи можливо поліпшити предметно-орієнтовану систему, щоб вона могла конкурувати з комерційною за функціональністю (додавши OSD, покращивши фільтри), не втративши при цьому своєї головної переваги – гнучкості для модифікацій, зокрема у сфері безпеки.

Задачею даного дослідження є проведення теоретичного порівняльного аналізу. Для цього в наступних розділах будуть досліджені: базова предметно-орієнтована система, типова комерційна система та поліпшена предметно-орієнтована система.

Порівняння цих трьох теоретичних моделей дозволить обґрунтувати доцільність та шляхи поліпшення предметно-орієнтованих систем.

1.5.2 Обґрунтування критеріїв теоретичного порівняння

Для об’єктивного та всебічного порівняння розроблених моделей, необхідно спиратися на чіткі, обґрунтовані критерії.

Виходячи з поставленої задачі, ключовим критерієм є функціональність та гнучкість. Він оцінює як саму функціональність (наявність та якість реалізації можливостей на кшталт навігаційних режимів, OSD, фільтрації та підтримки протоколів), так і гнучкість (теоретичну легкість внесення змін у базову логіку чи додавання власних алгоритмів).

З цим безпосередньо пов'язані вимоги до апаратних ресурсів та обчислювальна складність. Цей критерій має дати відповідь на питання, які поліпшення може отримати предметно-орієнтований ПК (STM32F401). Оцінка проводитиметься через аналіз теоретичного навантаження на MCU (від алгоритмів фільтрації, OSD, шифрування), а також вимог до пам’яті (Flash, RAM) та периферії (кількість UART, I2C, SPI портів) для кожної моделі.

Третім критерієм є потенційна стійкість до загроз, що безпосередньо витікає з аналізу у п. 1.4. Тут порівнюватиметься оцінка вразливості стандартних компонентів із теоретичною легкістю імплементації захисних механізмів у предметно-орієнтовану архітектуру.

Четвертим критерієм, виступає економічна доцільність (TCO - Total Cost of Ownership). Вона враховує не лише пряму вартість компонентів , але й вартість розробки – теоретичну оцінку трудовитрат на розробку, налаштування та підтримку кожного рішення.

Саме ці чотири критерії формують методику порівняльного і дозволять надати обґрунтовані рекомендації щодо вибору архітектури.

2 ТЕОРЕТИЧНЕ МОДЕЛЮВАННЯ ТА ОБҐРУНТУВАННЯ ПОЛІПШЕНЬ ПІДСИСТЕМ

2.1 Моделювання предметно-орієнтованої підсистеми керування

2.1.1 Опис архітектури та обґрунтування компонентної бази

Апаратна архітектура предметно-орієнтованої системи базується на принципі модульності, де кожен ключовий компонент є окремою платою, що поєднані між собою. Основу такої системи, складають наступні компоненти:

1) Центральний мікроконтролер (MCU). Як обчислювальне ядро було обрано плату на базі STM32F401CCU6. Використання саме цієї плати обґрунтовано її технічними характеристиками, які були потрібні для використання бібліотеки CRSFforArduino [29]. Контролер використовує ARM Cortex-M4, працює на частоті до 84 МГц, має 34 доступних GPIO та флеш-пам'ять розміром 128 КБ. Зображення плати та її діаграми пінів на рис. 2.1.

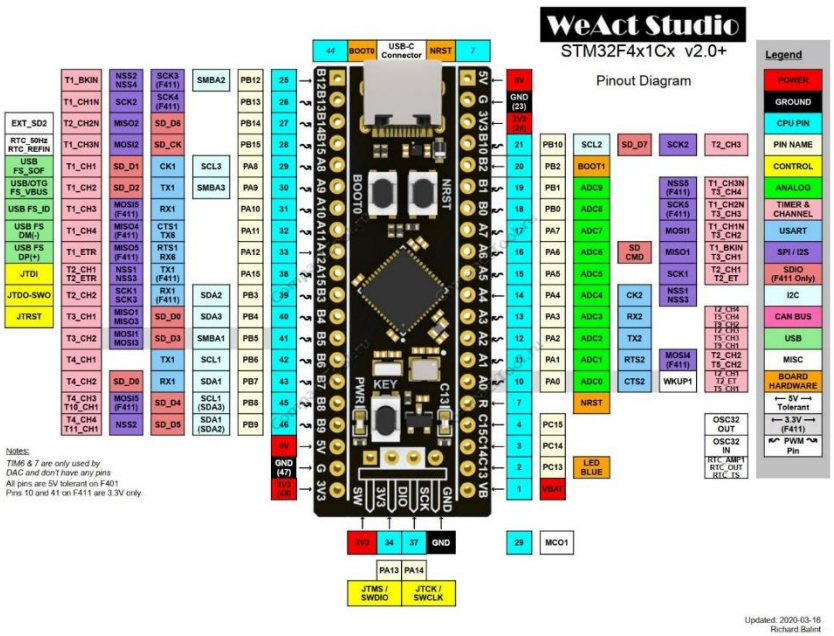


Рисунок 2.1 – Діаграма пінів STM32F411CCU6

2) Навігаційний модуль: Для отримання географічних координат та реалізації GPS-функцій було обрано приймач Ulox NEO-7M. Він забезпечує точне позиціювання та навігацію для FPV дронів. Відомий своєю високою чутливістю та швидким часом отримання сигналу від супутників. Модуль підтримує багато супутникових систем, включаючи GPS, GLONASS.

Перевагами NEO-7M є його надійність та точність, що є критично важливим для навігації та повернення додому. Також модуль відзначається невеликою вагою та компактними розмірами.

Недоліком є відносно висока вартість порівняно з деякими іншими модулями, а також потреба в налаштуванні для оптимальної роботи (рис. 2.2 та рис. 2.3).

Технічні характеристика модуля в таблиці 2.1.

Таблиця 2.1 – Технічні характеристики Ublox NEO-7M

Назва	Характеристика
Напруга живлення	1.65 - 3.6 В
Роздільна здатність	800 · 480 пікселів
Споживання енергії	17 мА
Робоча температура	–40 - 85 °C
Інтерфейс	UART
Кількість каналів GPS	56
Час холодного старту	30 - 32 с
Горизонтальна точність	2.5 м
Частота навігаційних даних	10 Гц
Точність вимірювання швидкості	0.1 м/с
Точність вимірювання напрямку	0.5 °

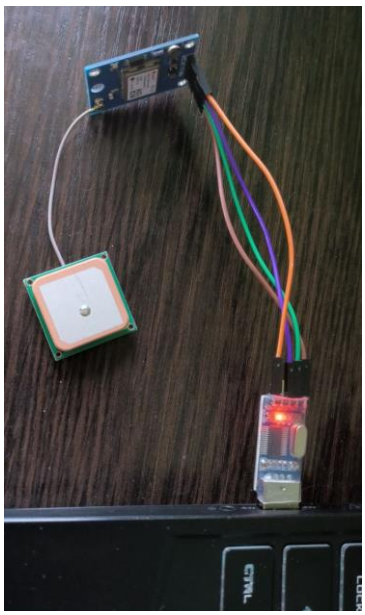


Рисунок 2.2 – Підключення модуля за допомогою конвертора

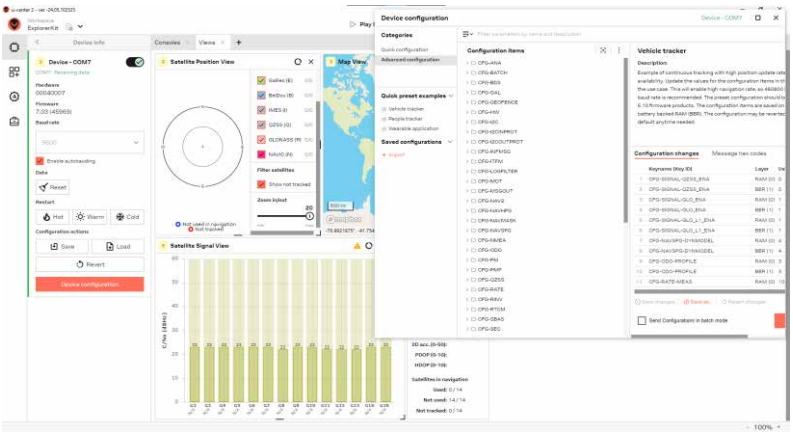


Рисунок 2.3 – Програма для налаштування U-Center 2

3) Інерційний вимірювальний блок (IMU). На рис. 2.4 зображено MPU6050 GY521, він є одним з найбільш доступних та часто використовуваних сенсорів для вимірювання інерційних параметрів руху. Він включає в себе тривісний акселерометр і тривісний гіроскоп, що дозволяє вимірювати як лінійні прискорення, так і кутові швидкості, а також має датчик температури.

Перевагами MPU6050 є висока інтеграція компонентів, що дозволяє зменшити розміри та вагу кінцевого пристрою, а також низьке енергоспоживання. Він також підтримує протокол I2C, що полегшує

інтеграцію з мікроконтролерами та іншими цифровими пристроями. Недоліком є необхідність калібрування для досягнення максимальної точності вимірювань, а також можливість виникнення шумів у даних при роботі в умовах високих вібрацій. Характеристики для ознайомлення в таблиці 2.2.

Рисунок 2.4 – Модуль MPU-6050

Таблиця 2.2 – Технічні характеристики MPU6050 GY-521

Назва	Характеристика
Інтерфейс	I2C
Напруга живлення	3.3 - 5 В
Споживання енергії	17 мА
Діапазон вимірювань гіроскопа	± 250°/с, ± 500°/с, ± 1000°/с, ± 2000°/с
Діапазон вимірювань акселерометра	± 2g, ± 4g, ± 8g, ± 16g
Розмір	21 · 16 мм
Вага	0.5 г

Приклад підключення Ublox NEO-7M та MPU6050 на рис. 2.5 та схема підключення на рис. 2.6.

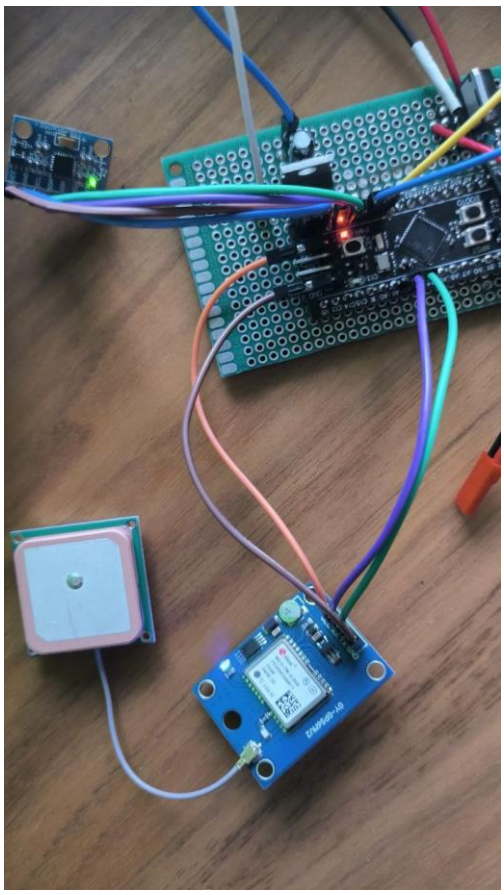


Рисунок 2.5 – Приклад підключення сенсорів

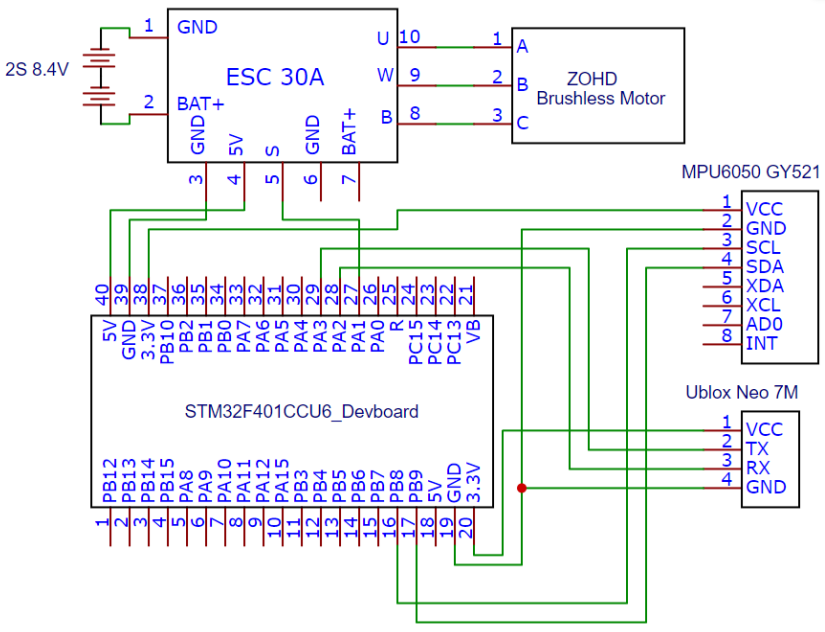


Рисунок 2.6 – Схема підключення сенсорів

Загальна робота системи безпілотного літального апарату на рис 2.6 та принципова схема підключення цих компонентів, що слугує основою даної теоретичної моделі, представлена у Додатку А.

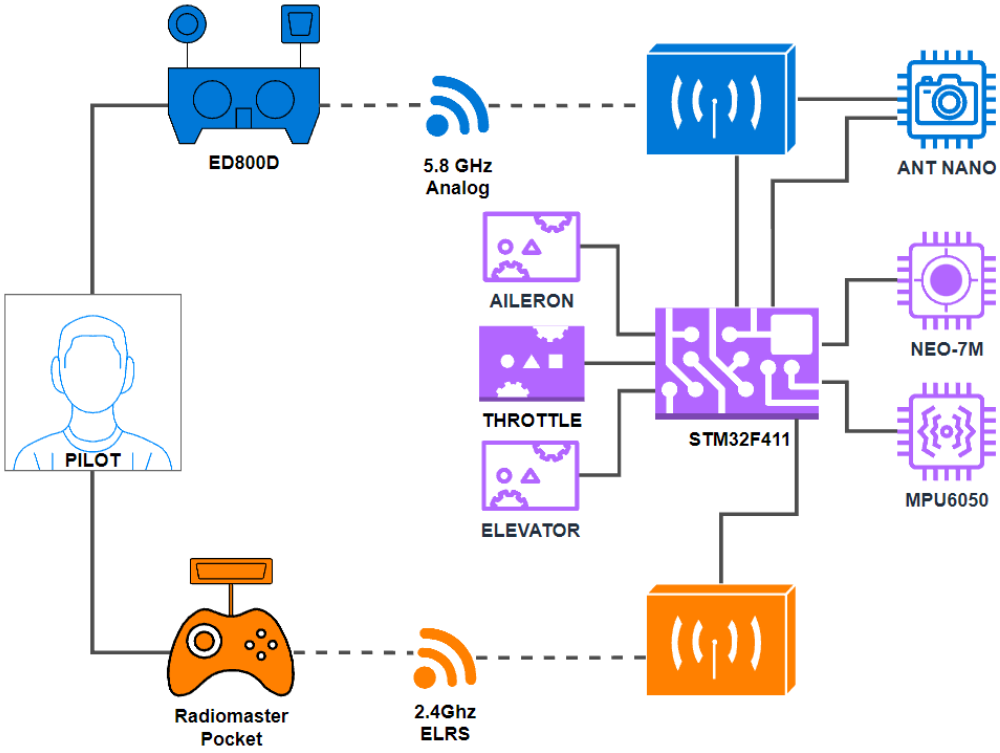


Рисунок 2.6 – Концептуальна схема роботи системи

Основна апаратна складова системи БПЛА на базі Arduino IDE:

– безколекторний мотор ZOHD 1406-2600KV;

– два сервоприводи;

– регулятор ZOHD 30A ESC Lite series з 5V 2A BEC;

– відеопередавач Boscam TS832;

– камера CADDX ANT Nano з OSD пультом;

– GPS модуль Ublox NEO-7M;

– Radiomaster RP3 Nano V2 ELRS (2.4GHz);

– модуль MPU6050 GY521;

– P-канальний мосфет IRF9640L;

– стабілізатор напруги VAST 7805;

– BAK N18650CNP 2500мАг 30A;

– плата розробника STM32F411CCU6;

Додаткові елементи та периферія:

– USB-UART(TTL) конвертор PL230HX;

– Модуль захисту BMS HX-2S-H20;

– Контролер заряду 2S li-ion 3-6v 2A;

– пульт радіоуправління Radiomaster Pocket ELRS 2.4GHz;

– аналоговий відеошлем EV800D;

– FPV крило ZOHD Dart250G (KIT).

2.1.2 Моделювання програмних алгоритмів обробки даних та стабілізації

Для програмування та роботи з платою STM32F411CCU6 в середовищі Arduino IDE були використані програма STM32CubeProgrammer, яка відповідає за запуск скомпільованого коду, а також включає в себе всі необхідні драйвери для справної роботи контролеру, та менеджер Arduino_Core_STM32.

Програмна частина (див. Додаток А) базується на Ardui №h, використовуючи HardwareSerial.h (UART), Wire.h (I2C) та HardwareTimer.h для роботи з апаратною частиною. Для отримання даних з сенсорів використовуються TinyGPS++.h (GPS) та Adafruit_MPU6050.h / MPU6050_light.h (IMU) разом з Adafruit_Sensor.h. Керування та телеметрія реалізовані через AlfredoCRSF.h (для ExpressLRS), а вихідні сигнали на виконавчі механізми (сервоприводи) формуються бібліотекою Servo.h.

Програмна логіка починається з функції setup(), яка відповідає за повну ініціалізацію системи. Тут реалізовано всі початкові налаштування апаратури: по-перше, налаштовуються інтерфейс I2C на нестандартних пінах PB8 та PB9, а також ініціалізуються два окремі апаратні порти HardwareSerial – один для GPS-модуля (NEO-7M), інший для приймача CRSF (RP3).

Далі конфігурується сам інерційний модуль MPU6050, для якого встановлюються робочі діапазони (8G для акселерометра, 500 град/с для гіроскопа) та внутрішня смуга пропускання фільтра на 21 Гц. Критичним кроком в setup() є одноразовий виклик функції mpuLight.calcOffsets(), яка обчислює та зберігає початкові зміщення (офсети) сенсорів перед стартом основного циклу.

Завершується ініціалізація налаштуванням виконавчих механізмів: проводиться автоматична послідовність калібрування газу для ESC, підключаються сервоприводи до їхніх пінів, а також налаштовується апаратний таймер TIM2 для генерації специфічного ШІМ-сигналу частотою 6.6 кГц, який в даній моделі використовувався для імітації сигналів OSD.

Після завершення setup(), система входить у головний цикл loop(), який представляє базовий алгоритм керування. Цей цикл у даній моделі є процедурним та синхронним. На кожній ітерації він послідовно викликає функції оновлення з бібліотек (crsf.update() для телеметрії та mpuLight.update() для IMU), після чого виконує набір кастомних функцій: gpsStatus(), mpuSatus(), BatteryData(), ControlPanel() та updateLinkStatusLed(). Ця проста лінійна структура і є базовою моделлю обробки даних.

Функція gpsStatus() (лістинг 2.1) працює в режимі простого опитування. Вона зчитує дані з серійного порту GPS, і як тільки бібліотека TinyGPS++ повідомляє про наявність оновлених координат, програма негайно пакує ці "сирі" дані та відправляє їх через функцію sendGps(), без будь-якої додаткової фільтрації.

Лістинг 2.1 – Функція gpsStatus()

```
void gpsStatus() {
  while (gpsSerial.available() > 0) {
    gps.encode(gpsSerial.read());
    if (gps.location.isUpdated()) {
      //gpsTextInfo();
      sendGps(gps.location.lat(),       gps.location.lng(),
gps.speed.kmph(),    gps.course.deg(),    gps.altitude.meters(),
gps.satellites.value());
    }}}
```

Схожим чином працює mpuSatus() (лістинг 2.2), що відповідає за обробку кутів. Важливо зазначити, що дана модель не реалізує власний фільтр синтезу даних (fusion filter). Натомість, робота повністю покладається на можливості бібліотеки MPU6050_light. Викликаються її функції getAngleX(), getAngleY() тощо, щоб отримати вже обчислені кути, які, ймовірно, є результатом роботи внутрішнього комплементарного фільтра самої бібліотеки. Програмний код лише бере ці готові значення, переводить їх у радіани та негайно відправляє телеметрією. Саме ця спрощена реалізація є головним недоліком моделі та кандидатом на заміну Розширеним фільтром Калмана (EKF).

Лістинг 2.2 – Функція mpuSatus()

```
void mpuSatus() {
  mpu.getEvent(&a, &g, &temp);

  // Отримання кутів у градусах
  float angleX_deg = mpuLight.getAngleX();
  float angleY_deg = mpuLight.getAngleY();
  float angleZ_deg = mpuLight.getAngleZ();

  // Перетворення кутів з градусів у радіани
  angleX_rad = angleX_deg * deg_to_rad;
  angleY_rad = angleY_deg * deg_to_rad;
  angleZ_rad = angleZ_deg * deg_to_rad;

  //mpuTextInfo();

  // Виклик функцій sendAttitude та sendMySensor для
відправки даних
  sendAttitude(angleY_rad, angleX_rad, angleZ_rad);
  sendMySensor(temp.temperature);

  Serial.println("");
}
```

Функція ControlPanel() (лістинги 2.3 та 2.4) реалізує логіку керування апаратом. У цій базовій моделі повноцінний PID-регулятор (тобто автопілот) відсутній. Система працює в режимі "passthrough", де алгоритми виконують роль мікшера для літаючого крила. Функція FLWings() (Лістинг 3.11) бере вхідні дані з джойстиків (aileronInput та elevatorInput) і за математичною

формулою обчислює необхідні відхилення для двох сервоприводів, реалізуючи таким чином керування елевонами. Модель також включає прості режими, що перемикаються тумблером, як-от обмеження потужності мотора до 70% у FLMotor() або перехід у статичний режим "набору висоти", де сервоприводи фіксуються у заданих положеннях, минаючи логіку мікшування.

Лістинг 2.3 – Виклик функцій управління

```
  if (CH6 == 1503) {
  } else {
    FLMotor();
  }
  if (CH7 == 1503) {
    rightServo.writeMicroseconds(1350);
    leftServo.writeMicroseconds(1600);
  } else {
    FLWings();
  }
```

Лістинг 2.4 – Функції FLMotor() та FLWings()

```
void FLMotor() {
  //Обмеження потужності до 70%
  if (CH6 == 2000) {
    CH3_value = constrain(CH3, 994, 1704);
  } else {
    CH3_value = constrain(CH3, 994, 2008); //потужність 100%
  }
  //Перетворення значень джойстика
  motor = map(CH3_value, 994, 2008, 800, 2300);
  constrain(motor, 800, 2300);
  //Подання даних драйверу
  flmotor.writeMicroseconds(motor);
}
void FLWings() {
  CH1_value = constrain(CH1, 1000, 2000);
  CH2_value = constrain(CH2, 1000, 2000);
  //Зчитування даних джойстика
  aileronInput = map(CH1_value, 1000, 2000, 500, -500);
  elevatorInput = map(CH2_value, 1000, 2000, 500, -500);
  //Формула Синхронізації закрилок
   rightServoOutput = constrain(1500 + aileronInput -
elevatorInput + 40, 600, 2100);
    leftServoOutput = constrain(1500 + aileronInput +
elevatorInput - 80, 600, 2100);
  //Подання даних на севомотори
  rightServo.writeMicroseconds(rightServoOutput);
  leftServo.writeMicroseconds(leftServoOutput);}
```

Проаналізувавши структуру коду виявлено низку системних недоліків, що суттєво обмежують її практичне застосування.

Архітектура головного циклу loop() є синхронною. Це створює критичну вразливість: будь-яка затримка в одній функції, наприклад, при тривалому опитуванні gpsSerial, призведе до повної зупинки всього циклу. Це, у свою чергу, заморозить виконання життєво важливої функції ControlPanel(), що для системи керування в реальному часі є неприпустимим.

У моделі повністю відсутня комплексна обробка сенсорів. Система не містить власних алгоритмів синтезу даних (sensor fusion), а сліпо покладається на готові кути з бібліотеки MPU6050_light (що, ймовірно, є простим комплементарним фільтром) та необроблені дані GPS. Такий підхід гарантує низьку точність оцінки положення, високу чутливість до вібрацій та неконтрольований дрейф, особливо по осі yaw.

Система де-факто не виконує функцій автопілота. Функція ControlPanel() не містить PID-регуляторів і не здатна самостійно стабілізувати апарат у просторі, а лише транслює команди пілота.

Таким чином, змодельована предметно-орієнтована підсистема є базовим прототипом, що суттєво поступається у надійності, точності та функціональності комерційним рішенням. Ці ідентифіковані недоліки формують чітку мету для теоретичних поліпшень, що розглядаються у наступних підрозділах.

Для усунення виявлених системних недоліків необхідна глибока теоретична модернізація базової програмної моделі, що стосується трьох ключових аспектів: архітектури виконання, обробки сенсорів та логіки керування.

Для вирішення проблеми блокуючої архітектури loop() пропонується перехід від синхронного процедурного виконання до асинхронної моделі, керованої подіями. Оптимальним рішенням є впровадження планувальника, що базується на апаратних перериваннях одного з таймерів STM32 (наприклад, TIM). Життєво важливий контур стабілізації (читання IMU та виконання

логіки керування) має бути винесений у обробник цього переривання. Такий підхід гарантує виконання стабілізації з фіксованою, високою частотою (напр., 500 Гц), незалежно від затримок менш пріоритетних завдань, як-от опитування GPS чи оновлення телеметрії, які залишаться в основному циклі.

Для усунення низької точності та високої чутливості до шумів, що є наслідком відсутності sensor fusion, пропонується впровадження розширеного фільтра Калмана (EKF) . На відміну від простого комплементарного фільтра, EKF є математичною моделлю, що здатна інтегрувати дані з усіх наявних джерел: акселерометр, гіроскоп, GPS [32][33][34]. Фільтр обчислює єдину, надійну оцінку повного стану апарату (його просторове положення, кутові швидкості, координати та швидкість), ефективно компенсуючи дрейф гіроскопів та фільтруючи шуми акселерометра і GPS.

Для перетворення системи з простого мікшера на повноцінний автопілот необхідно замінити логіку "passthrough" у функції ControlPanel() на каскадний PID-регулятор. Новий алгоритм має на вхід отримувати два значення: бажаний стан, наприклад, кути крену та тангажу, що надходять з пульта керування та реальний стан, отриманий з EKF. PID-регулятор обчислюватиме помилку між цими станами та генеруватиме керуючий сигнал для функції-мікшера (FLWings()), яка, у свою чергу, відхилятиме сервоприводи для мінімізації цієї помилки.

Сукупність цих трьох теоретичних поліпшень перетворює базовий прототип на функціональний польотний контролер, здатний до автономної стабілізації польоту.

2.2 Моделювання типової підсистеми на базі комерційного польотного контролера

Для теоретичного моделювання типової комерційної підсистеми обрано одну з найпопулярніших платформ у FPV-сегменті, що поєднує високу продуктивність та багату функціональність для GPS-польотів – польотний контролер SpeedyBee F405 V3/V4 під керуванням програмної екосистеми INAV.

2.2.1 Аналіз апаратної архітектури

SpeedyBee F405 V3/V4 є компактним 30×30 мм польотним контролером на базі мікроконтролера STM32F405RGT6 з тактовою частотою 168 МГц та 1 МБ Flash пам'яті. Це забезпечує достатню обчислювальну потужність для складних алгоритмів фільтрації та навігації [35].

Сенсорна підсистема включає IMU (BMI270 у V3 або ICM42688P у V4) та барометр (SPL06 у V3 або DPS310 у V4). Контролер також оснащений чипом OSD AT7456E для виведення телеметрії на відео та слотом MicroSD для логів Blackbox. Для підключення периферії він має багатий набір інтерфейсів: 6 портів UART, 1 шину I2C, 9 виходів PWM, USB Type-C та 6-контактний роз'єм для цифрових FPV систем DJI. Система живлення розрахована на вхід 3S-6S LiPo (9-25V) з TVS діодом для захисту та надає три BEC виходи: 5V/2.5A, 9V/2.5A та 3.3V/500mA [36].

Детальна схема SpeedyBee F405 V3, на рис. 2.7 показує розташування компонентів: FPV камера, 4-in-1 ESC роз'єм, приймач, Bluetooth чіп, orange LED індикатор режиму, батарейний індикатор, boot кнопка, OSD чіп AT7456E, GPS & компас, VTX, buzzer, гіроскоп BMI270, USB Type-C, MCU F405, PWM виходи. Схеми підключення показують типову архітектуру системи: analog/digital VTX, камери, датчик швидкості повітря, GPS модуль, різні типи

приймачів (ELRS, Crossfire, SBUS), LED стрічки, buzzer, servo. Кожне підключення має специфічні UART порти та напругу живлення.

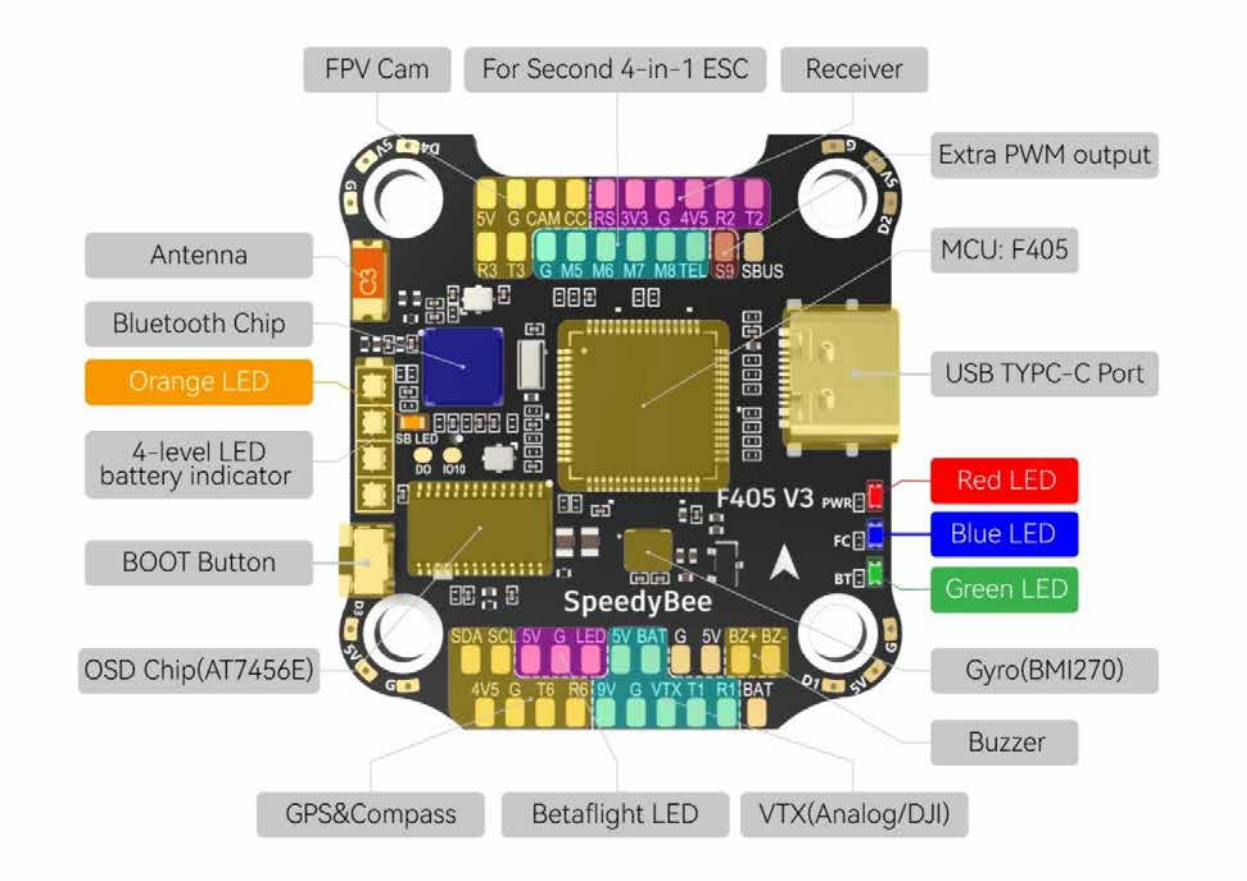


Рисунок 2.7 – Розпіновка контролера польоту SpeedyBee F405 V3

Детальна схема SpeedyBee F405 V3 показує розташування компонентів: FPV камера, 4-in-1 ESC роз'єм, приймач, Bluetooth чіп, orange LED індикатор режиму, батарейний індикатор, boot кнопка, OSD чіп AT7456E, GPS & компас, VTX, buzzer, гіроскоп BMI270, USB Type-C, MCU F405, PWM виходи.

Схеми підключення (рис. 2.8) показують типову архітектуру системи: analog/digital VTX, камери, датчик швидкості повітря, GPS модуль, різні типи приймачів (ELRS, Crossfire, SBUS), LED стрічки, buzzer, servo. Кожне підключення має специфічні UART порти та напругу живлення [37].

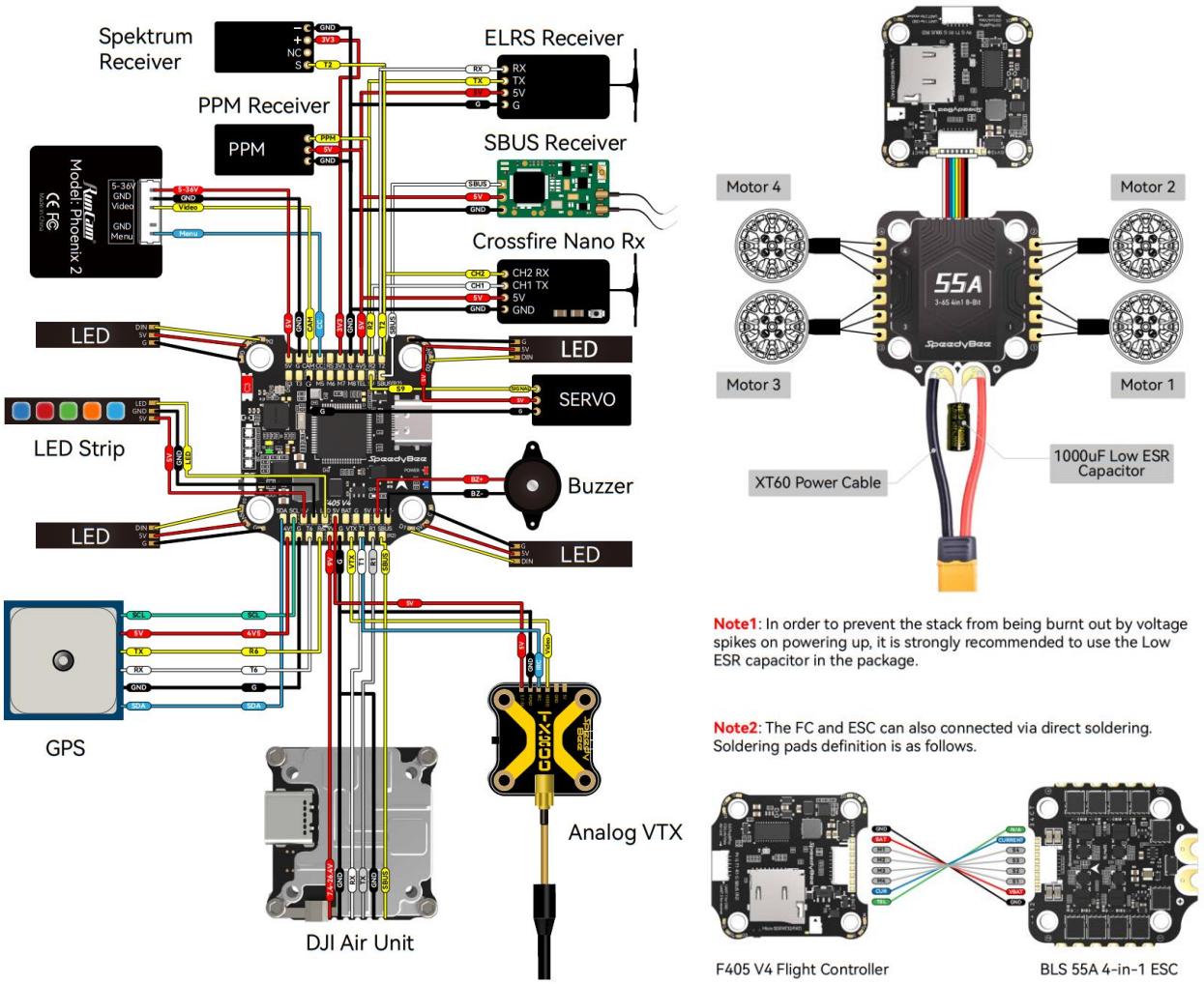


Рисунок 2.8 – Приклад схеми БПЛА на базі SpeedyBee F405

2.2.2 Аналіз програмної архітектури та стандартних алгоритмів (INAV)

Структура програмного забезпечення INAV (iNav Flight) є форком Cleanflight/Betaflight, оптимізованим для GPS-навігації та автономного польоту. Версії INAV 2.5-7.1 підтримують STM32F4/F7 мікроконтролери та широкий спектр сенсорів через SPI та I2C [39]. Ієрархія обробки даних чітко структурована: вона починається з рівня сенсорів (гіроскоп, акселерометр, GPS, барометр, магнітометр), з якого дані надходять на рівень оцінювання (Estimators). Цей рівень включає Attitude Estimator (використовує комплементарний фільтр або EKF для орієнтації), Position Estimator (злиття GPS та акселерометра) та Heading Estimator (визначення курсу). Далі в ієрархії

йде навігаційний двигун (NAV Engine), що є набором додаткових PID контролерів, які перетворюють дані з estimators у корекції setpoint. Ці корекції надходять на основні PID контролери для стабілізації, і, нарешті, міксер моторів перетворює виходи PID у сигнали для ESC. Примітно, що INAV 2.6 використовує Alpha-Beta фільтр (різновид комплементарного фільтру) замість розширеного фільтра Калмана для економії обчислень, що дає задовільні результати, хоча новіші версії можуть використовувати EKF для покращення точності.

Алгоритми керування базуються на PID-контролері, який є основою стабілізації БПЛА. Для кожної осі (roll, pitch, yaw) INAV обчислює вихідний сигнал за формулою:

$$\text{Output} = K_p \times \text{Error} + K_i \times \int \text{Error}\, dt + K_d \times \frac{d(\text{Error})}{dt} + \text{Feedforward}, \quad (2.1)$$

У цій структурі P (Proportional) забезпечує пропорційний відгук на поточну помилку, I (Integral) усуває сталу помилку, що критично для утримання позиції проти вітру, а D (Derivative) демпфує коливання. Додатково Feedforward (FF) передбачає необхідну силу на основі команди стіка, зменшуючи латентність. Процес налаштування PID в INAV може бути автоматичним (AutoTune) або ручним. Ручне налаштування, рекомендоване для фіксованих крил, включає кроки по ізоляції P та D термів, їх послідовне збільшення до досягнення критично демпфованого відгуку (найшвидший відгук без перерегулювання) та подальше додавання I [40].

Навігаційні режими INAV включають як базові, так і автономні. Доступні режими польоту: ANGLE (автоматичне вирівнювання з обмеженням кута), NAV ALTHOLD (утримання висоти), NAV POSHOLD (утримання GPS позиції), NAV RTH (повернення додому з посадкою) та NAV WP (виконання waypoint-місій). Автономні місії Waypoint дозволяють планувати до 60 точок на платах F4/F7 через INAV Configurator або мобільні додатки. Гнучкість місій

забезпечується параметрами waypoint, такими як nav_wp_radius (радіус досягнення точки, 100 см за замовчуванням), nav_wp_safe_distance (безпечна відстань до першої точки, 100 м) та waypoint_tracking_accuracy. Доступні різні типи waypoints, зокрема стандартний Waypoint, Infinite Position Hold, RTH та Timed Position Hold. Виконання місії починається при активації режиму NAV WP, і апарат послідовно рухається до точок, доки не досягне кінця списку або команди RTH [41].

Архітектура обробки сенсорів у SpeedyBee F405 оптимізована для швидкодії. Використовується протокол SPI для критичних датчиків (IMU, барометр, Flash, OSD), що дозволяє частоту вибірки гіроскопа до 8 кГц, на відміну від I2C, який обмежений ~1 кГц. I2C зарезервовано для некритичних периферійних пристроїв, як-от зовнішній компас. Обробка даних IMU полягає в опитуванні сенсорів з високою частотою (до 8000 Гц), проходженні даних через фільтри шуму та подальшому злитті через комплементарний фільтр або EKF для передачі чистої орієнтації в PID контролер (250-500 Гц). Обробка GPS відбувається через UART, де дані з частотою ~1-5 Гц зливаються з даними акселерометра у Position Estimator, а зовнішній магнітометр (I2C) забезпечує абсолютний курс.

Для керування моторами SpeedyBee F405 підтримує сучасні протоколи ESC, такі як цифровий DShot (150/300/600), що забезпечує високу швидкість та надійність, а також старіші аналогові Multishot/Oneshot125 та стандартний PWM.

У сфері телеметрії та комунікації INAV використовує MSP (MultiWii Serial Protocol) для зв'язку з конфігураторами, причому вбудований Bluetooth-модуль дозволяє бездротове налаштування. Хоча для телеметрії можуть використовуватися 3DR радіомодеми, INAV віддає перевагу MSP замість MAVLink. Забезпечується підтримка всіх сучасних приймачів, включаючи ELRS (найнижча латентність), TBS Crossfire (дальній зв'язок) та SBUS.

Практична реалізація системи вимагає низки кроків. Необхідно провести калібрування сенсорів: гіроскопа/акселерометра на рівній поверхні

та магнітометра (методом "фігури вісімки"), а також калібрування ESC. Для подальшого аналізу польоту використовується Blackbox логування на MicroSD карту, що дозволяє детально вивчити вібрації та точність GPS. Усі детальні параметри системи налаштовуються через CLI (Command Line Interface) в INAV Configurator.

Моделювання підсистеми на базі SpeedyBee F405 та INAV демонструє інтеграцію апаратної платформи STM32F405 з програмними алгоритмами фільтрації, PID керування та GPS-навігації для створення повнофункціонального автономного БПЛА. Ця модель являє собою високоінтегроване, потужне та функціональне комерційне рішення, яке слугуватиме еталоном для порівняння з предметно-орієнтованими системами.

2.3 Розробка теоретичних поліпшень для предметно-орієнтованої підсистеми

Аналіз базової предметно-орієнтованої моделі (п. 2.1), змодельованої на основі бакалаврської роботи, виявив її ключові переваги (прозорість коду, гнучкість) та суттєві недоліки порівняно з комерційною моделлю (п. 2.2). До недоліків належать: використання спрощених алгоритмів фільтрації, повна відсутність OSD та несумісність зі стандартними наземними станціями (GCS) та конфігураторами.

Метою даного підрозділу є розробка теоретичної моделі поліпшень, яка нівелює ці недоліки, зберігаючи при цьому переваги предметно-орієнтованого підходу, зокрема у сфері безпеки.

2.3.1 Обґрунтування вдосконалення алгоритмів фільтрації, перехід від простих фільтрів до EKF

Аналіз базової моделі виявив обмеженість простих комплементарних фільтрів при складних маневрах. Тому ключовим напрямком поліпшення є впровадження розширеного фільтра Калмана.

Перехід від комплементарних фільтрів до Extended Kalman Filter (EKF) дозволяє значно підвищити точність оцінки орієнтації та позиції БПЛА за рахунок багато-сенсорного злиття. EKF здатен комплексно обробляти дані не лише з базового IMU (гіроскоп, акселерометр), але й інтегрувати потоки з GPS, барометра, магнітометра та навіть датчиків оптичного потоку [42].

Критичною перевагою EKF, на відміну від простих фільтрів, є його здатність ефективно ідентифікувати та ігнорувати несправні або надмірно шумні вимірювання. Це дозволяє системі залишатися стабільною навіть при тимчасових збоях окремих датчиків. Крім того, архітектура EKF дозволяє масштабувати систему, комбінуючи дані з додаткових джерел, таких як лазерні далекоміри або ультразвукові висотоміри, для підвищення стійкості в складних умовах (наприклад, при посадці або польоті на низькій висоті).

Сучасні дослідження також вказують на ефективність гібридних підходів. Алгоритми-гібриди, що поєднують швидкість комплементарного фільтра та точність EKF, забезпечують кращу швидкодію системи керування та ефективніше пригнічують дрейф гіроскопів у порівнянні з ізольованими рішеннями. Це особливо актуально при наявності різких маневрів або сильних магнітних завад, де прості лінійні фільтри часто дають хибну оцінку горизонту [43].

Хоча EKF є значно більш вимогливим до ресурсів, ніж комплементарний фільтр, мікроконтролер STM32F401 (84 МГц, з апаратним FPU) є теоретично здатним виконувати обчислення EKF з необхідною частотою, на відміну від 8-бітних платформ.

2.3.2 Пропозиції щодо підвищення стійкості каналу телеметрії

Базова предметно-орієнтовна система, як і більшість простих Arduino-проєктів, передає дані телеметрії через UART у відкритому, незашифрованому вигляді. Як було показано в п. 1.4, це створює критичну вразливість: зловмисник може не лише перехопити відео, але й отримати GPS-координати точки "Дім", тобто місцезнаходження оператора.

Потенційним рішенням є використання переваг предметно-орієнтованої архітектури для імплементації методів захисту, проаналізованих у курсовій роботі. Оскільки ми контролюємо кожний байт прошивки, ми можемо додати шар шифрування.

Впровадження в код на STM32F401 легковагової симетричної криптографії. Оптимальним кандидатом є AES-128. Програмна реалізація AES на Cortex-M4 є достатньо швидкою для шифрування низькошвидкісного потоку телеметрії (GPS-координати, напруга) в реальному часі.

Для захисту від спуфінгу команд керування, в протокол "земля-борт" додається механізм автентифікації, наприклад HMAC-SHA256, для кожного пакета.

Це поліпшення є унікальною перевагою поліпшеної програмно-орієнтованої системи, ми отримуємо рівень безпеки, недосяжний для комерційних прошивок без їх глибокої та складної модифікації.

2.3.3 Розробка теоретичної моделі інтеграції функціоналу OSD

Відсутність візуального зворотного зв'язку є головним експлуатаційним недоліком базового ПК на базі Ardui № Без системи OSD (On-Screen Display) пілот позбавлений критично важливої інформації – напруги акумулятора, часу польоту, висоти та напрямку до точки зльоту, що унеможливлює безпечне виконання FPV-польотів на середні та великі дистанції. На відміну від комерційної Моделі 2, де чіп OSD розпаяний на платі, кастомна система потребує зовнішнього рішення.

Для розв'язання цієї проблеми у теоретичну модель вводиться окремий апаратний модуль MinimOSD (рис. 2.9).

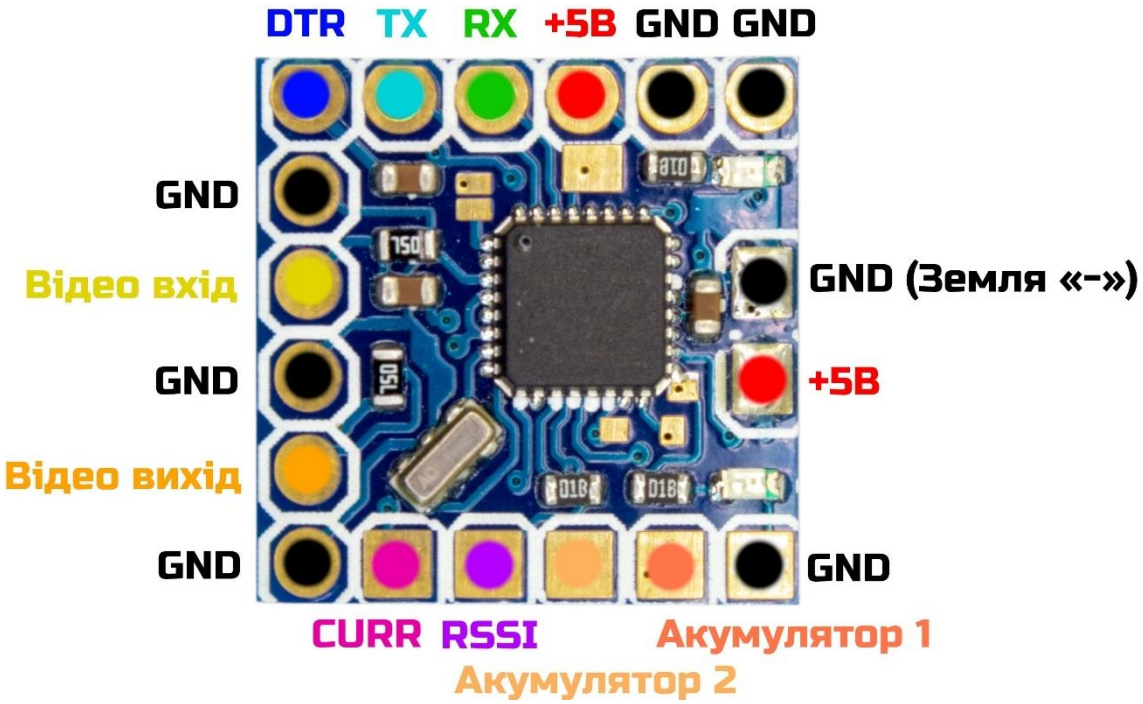


Рисунок 2.9 – Minim OSD

Архітектура системи змінюється з монолітної на розподілену. Модуль MinimOSD базується на власному мікроконтролері ATmega328P та спеціалізованій мікросхемі генерації символів MAX7456. Підключення до центрального польотного контролера (STM32F401) здійснюється не через швидкісну шину SPI, а через стандартний послідовний інтерфейс UART (Tx/Rx).

Схема проходження відеосигналу організовується транзитно: аналоговий сигнал від камери надходить на вхід Video In модуля MinimOSD, де мікросхема MAX7456 накладає на нього текстовий шар із телеметрією, після чого змішаний сигнал виходить з порту Video Out і подається на відеопередавач (VTX). Така архітектура забезпечує гальванічну розв'язку (при правильному живленні) та модульність системи. Приклад схеми підключення на рис. 2.10.

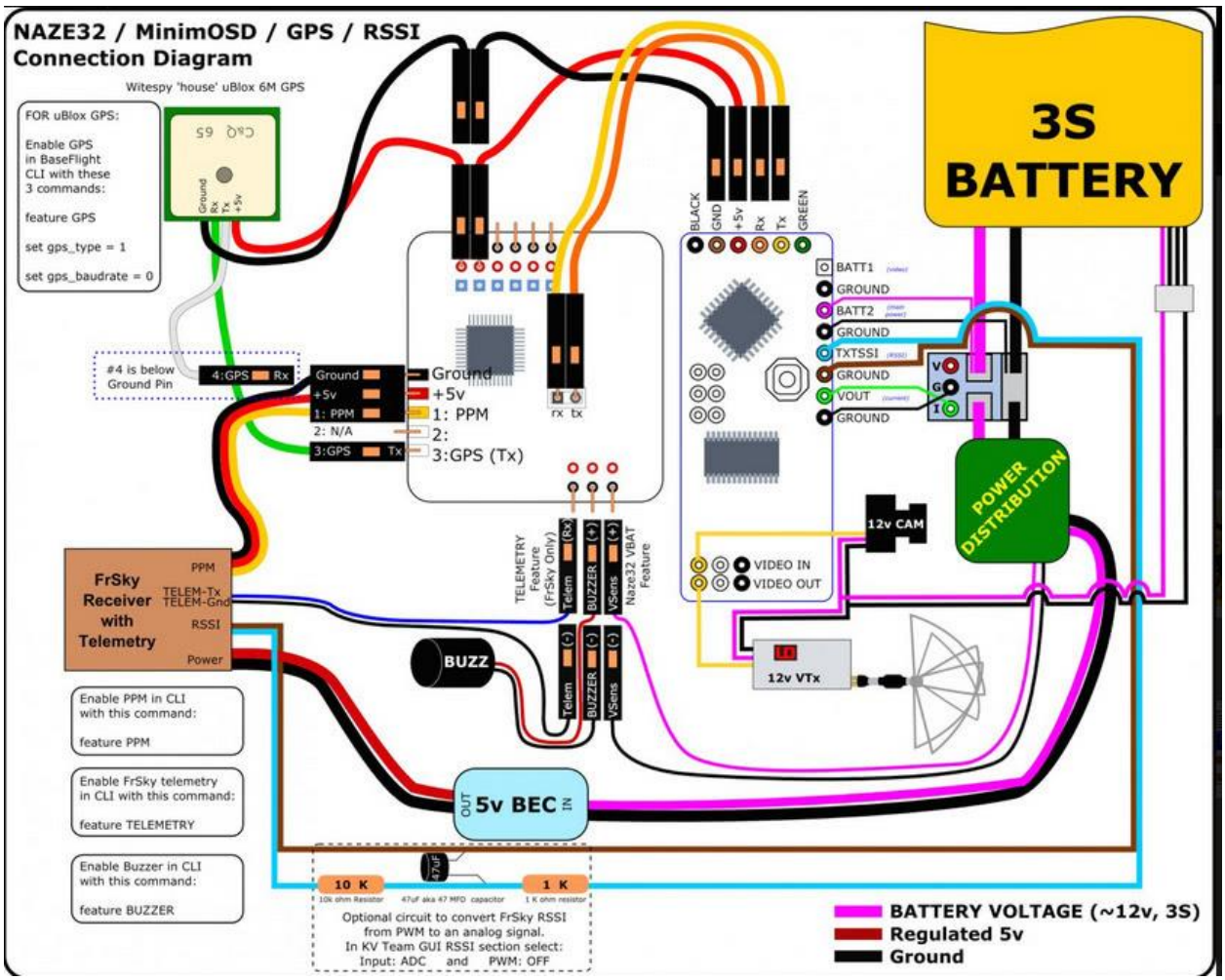


Рисунок 2.10 – Схема підключення Minim OSD

Інтеграція MinimOSD вимагає змін у програмному забезпеченні STM32F401, але, на відміну від SPI-варіанту, значно економить обчислювальні ресурси основного контролера.

Замість того, щоб самостійно формувати відео-буфер, основний контролер лише транслює пакет даних телеметрії (стан кутів, GPS-координати, напругу) через UART-порт із частотою 10-50 Гц.

Для сумісності з MinimOSD, прошивка STM32 повинна формувати дані у стандартизованому протоколі, найчастіше MAVLink або MSP (MultiWii Serial Protocol). Це ідеально узгоджується з задачами пункту 2.3.4 щодо сумісності.

На самому модулі MinimOSD працює власна прошивка (наприклад, MWOSD або KV_Team OSD), яка приймає MSP-пакети, декодує їх і самостійно керує виводом тексту на екран.

Таке рішення робить поліпшену систему функціонально еквівалентною комерційним аналогам, при цьому знижуючи навантаження на центральний процесор STM32F401, оскільки задача рендерингу тексту повністю делегується окремому модулю MinimOSD.

2.3.4. Аналіз методів забезпечення сумісності з відкритими програмними екосистемами

Головним недоліком Arduino системи та поліпшеного варіанту (фільтр EKF, шифрування AES, модуль OSD) залишається її програмна "закритість". Система, написана на Arduino IDE, вимагає перекомпіляції та перепрошивки для будь-яких змін PID-коефіцієнтів, налаштувань OSD чи калібрування. Вона абсолютно несумісна зі стандартними індустрії графічними конфігураторами (Betaflight Configurator, INAV Configurator) та наземними станціями (QGroundControl).

Одним з рішень є повна відмова від пропрієтарного коду на Arduino IDE на користь портування повноцінної відкритої прошивки, по типу INAV або Betaflight, безпосередньо на апаратну платформу предметно-орієнтованого контролера (STM32F401 та MPU6050).

Як показує аналіз існуючої інженерної практики зі збірки кастомних контролерів (зокрема, на поширених платах, як-от STM32F411 Blackpill, що є архітектурно близькою до F401), цей процес є нетривіальним, але можливим. Він складається з наступних теоретичних етапів:

1) Адаптація прошивки. Прошивки INAV/Betaflight не можуть бути просто завантажені на STM32. Вони потребують "target" – файлу конфігурації, який описує апаратну частину. Оскільки контролер STM32F401 з MPU6050 є унікальною збіркою, стандартний target відсутній.

Для Betaflight: Існують готові "targets" для схожих плат, наприклад, STM32F411, які можна використати. Для підтримки старих сенсорів, як-от MPU6050, може знадобитися використання старіших версій прошивки.

Для INAV ситуація складніша. Найчастіше потрібна кастомна збірка під сенсор, тобто ручна компіляція прошивки INAV з репозиторію, де вказано, який саме IMU використовується.

2) Створення "Ресурсної Карти" (Resource Map). Це ядро сумісності. Після прошивки контролера (зазвичай у DFU-режимі через CLI-команду dfu або bl), необхідно повідомити прошивці, де знаходяться наші компоненти. Це робиться через CLI (Command Line Interface) конфігуратора. Для нашої поліпшеної системи це виглядало б так:

resource I2C_SDA 1 PB7 (призначення піна SDA для MPU6050 на шині I2C1).

resource SERIAL_RX 2 PA3 (призначення GPS NEO-7M на UART2).

resource SERIAL_TX 1 PA9 (призначення MinimOSD з п. 2.3.3 на UART1).

resource MOTOR 1 PA0 ... resource MOTOR 4 PA3 (призначення виходів ШІМ (PWM) для ESC).

resource SPI_SCK 1 PA5 (резервування SPI для майбутнього Blackbox-модуля).

3) Врахування апаратних та обчислювальних нюансів. Практичний досвід спільноти показує низку нюансів, які необхідно врахувати в теоретичній моделі.

Навантаження на ЦПУ. Старий сенсор MPU6050 відомий тим, що створює значне навантаження на шину I2C та центральний процесор. Це обмежує кількість складних функцій. Щоб уникнути зависань плати, доведеться оптимізувати конфігурацію: знизити looptime, використовувати простіші протоколи ESC (напр., PWM/Multishot замість ресурсомісткого DShot) та ретельно налаштувати фільтри (gyroscope_lpf) для боротьби з вібраціями, що викликають дрейф горизонту.

Для MPU6050 критично важливо забезпечити швидкість шини I2C не менше 800kHz, інакше виникають зависання при опитуванні.

На відміну від комерційних плат, предметно-орієнтована збірка потребує зовнішнього захисту по живленню (напр., діод між платою та стабілізатором).

В результаті поліпшення отримано систему, яка апаратно залишається предметно-орієнтованою, модульною, ремонтопридатною та потенційно захищеною (з можливістю апаратного шифрування, як у п. 2.3.2), програмно стає повноцінним INAV-контролером, хоч і зі специфічною конфігурацією.

Така модель отримує всі переваги комерційної системи:

- підтримку EKF (п. 2.3.1), оскільки він є частиною прошивки INAV;

- нативну підтримку MinimOSD через MSP (п. 2.3.3);

- нативну підтримку INAV Configurator для налаштувань та QGroundControl/Mission Planner (через MAVLink) для GPS-місій;

- типові алгоритми навігації (RTH, Waypoint), які вже відтестовані спільнотою.

Таким чином отримана система – це гібрид, що поєднує повну гнучкість та модульність заліза предметно-орієнтованої контролера з усією потужністю програмного забезпечення комерційного контролера.

РОЗДІЛ 3. ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ МОДЕЛЬОВАНИХ ПІДСИСТЕМ

3.1 Розробка методики теоретичного порівняльного аналізу

Для всебічного та об'єктивного зіставлення трьох розроблених теоретичних моделей: базова предметно-орієнтована, типова комерційна та поліпшена предметно-орієнтована – необхідна єдина методика.

Методика даного теоретичного аналізу ґрунтується на наборі ключових критеріїв (таблиця 3.1), що були обґрунтовані у п. 1.5. Ці критерії дозволяють оцінити не лише номінальну продуктивність, але й довгострокову доцільність використання кожної архітектури. Порівняння проводитиметься шляхом теоретичної оцінки кожної моделі за наступними чотирма основними напрямками.

Таблиця 3.1 – Основні критерії порівняння

Критерій	Опис	Метрика / Пояснення
Гнучкість	Можливість адаптуватися під різні схеми, протоколи, ПЗ	Відкритість апаратної карти (hardware map), можливість кастомізації прошивки, легкість інтеграції нестандартних модулів.
Обчислювальна складність	Відповідність ресурсів MCU потребам алгоритмів керування	Тактова частота, об’єм Flash/RAM, підтримка складних фільтрів (EKF), пропускна здатність I/O.
Потенційна надійність	Здатність працювати безвідмовно, захищеність від збоїв	Стабілізація живлення, якість пайки/збірки, можливість резервування сенсорів, наявність апаратного захисту.
Функціональність	Набір підтримуваного обладнання, прошивок, режимів	Режими польоту, інтеграція з GPS/OSD/барометрами, кількість портів

Продовження таблиці 3.1

Критерій	Опис	Метрика / Пояснення
		(PWM/UART/I2C/SPI), Blackbox, телеметрія.
Вартість (TCO)	Сума витрат на побудову, запуск та підтримку	Вартість компонентів, час/складність складання та налаштування, вартість потенційних ремонтів (відсутність гарантії).

Методика порівняння:

1. Визначення параметрів для кожної підсистеми. Проводиться збір технічних параметрів з технічних даташитів для комерційної моделі (SpeedyBee F405) та для компонентів предметно-орієнтованої моделі (STM32F401, MPU6050, NEO-7M, MinimOSD). Для предметно-орієнтованих контролерів враховується якість монтажу та необхідність ручного створення пінмапу (resource map).

2. Опис алгоритмічних можливостей. Аналізується обсяг підтримуваних фільтрів (EKF, комплементарні) та режимів (PID, Feedforward). Оцінюється, наскільки ресурсів MCU (Flash/RAM) вистачає для виконання навігаційних/стабілізаційних задач. Оцінюються обмеження по частоті вибірки IMU та кількість портів.

3. Оцінка гнучкості та адаптивності. SpeedyBee F405 пропонує повну сумісність з сучасними прошивками (Betaflight, ArduPilot, INAV), просту інтеграцію периферії та оновлення. Предметно-орієнтована плата є відкритою до кастомних функцій (наприклад, шифрування з п. 2.3.2), але вимагає ручного мапінгу всіх портів, поглибленої компіляції прошивки (як у п. 2.3.4) та частіше стикається з несумісністю драйверів.

4. Надійність. SpeedyBee F405 – промислова плата, перевірена в масовому виробництві, має захист по живленню (TVS-діоди) та ESD. Предметно-орієнтована збірка – якість залежить від вручну проведених монтажу, паяння та розв’язки живлення. Схильна до помилок через людський фактор.

5. Функціональність: Проводиться підрахунок підтримуваних режимів польоту (manual, стабілізація, GPS hold, RTH, waypoints), підтримка телеметрії (MSP/MAVLink), Blackbox, протоколів ESC (DShot, PWM). Оцінюється інтеграція з OSD, Bluetooth, SD-card, кількість моторів/серво.

6. Фінансова оцінка і витрати часу: Порівняння роздрібної вартості компонентів.

Узагальнення для теоритичної оцінки представлено таблицею 3.2.

Таблиця 3.2 – Узагальнення для теоритичної оцінки

Критерій	Комерційна	Предметно-орієнтована
Гнучкість	Висока (для стандартних застосувань)	Максимальна (для експериментів, кастомного ПЗ, безпеки)
Обчислювальна складність	Висока (168 МГц, 1Мб Flash, багато портів)	Середня (84 МГц, 256Кб Flash, обмежені порти)
Надійність	Промислова, масове тестування	Залежить від якості збірки/пайки, наявності захисту
Функціональність	Повний стек підтримки (OSD, Blackbox, DShot)	Обмежена (вимагає ручної інтеграції OSD, Blackbox)
Вартість	$40–60 (готова, гарантована)	$10–25 (час на збірку, тести, можливі помилки)

Загальні рекомендації, що випливають з методики. Для швидкого, стабільного запуску з розширеною функціональністю – кращою є комерційна модель. Для досліджень, глибокого експериментування з алгоритмами безпеки, коли важлива низька вартість компонентів та повна прозорість коду – підходить предметно-орієнтована модель.

3.2. Порівняльний аналіз функціональності та гнучкості налаштування

Відповідно до розробленої методики (п. 3.1), цей аналіз є ключовим, оскільки він оцінює фундаментальний компроміс між готовими можливостями комерційних систем та потенціалом модифікації предметно-орієнтованих.

3.2.1. Гнучкість та інтерфейс налаштування

Базова предметно-орієнтована модель на Arduino IDE демонструє абсолютну гнучкість. Абсолютно все програмується вручну. Можна оптимізувати алгоритми, писати власні логіку/фільтри, змінювати порядок опитування сенсорів". Однак "інтерфейсом" налаштування є сам код на C++ в Arduino IDE. Всі налаштування параметрів та алгоритмів відбуваються через зміну коду, що вимагає компіляції та перепрошивки при кожній зміні PID.

Комерційна модель на INAV демонструє високу гнучкість налаштувань, але в межах платформи. Всі параметри (PID, фільтри, режими) зручно налаштовуються через GUI: INAV Configurator або CLI для розширених параметрів. Можлива конфігурація через Bluetooth, телеметрію. Однак гнучкість модифікації (додавання нестандартних алгоритмів) вимагає глибших знань та рекомпіляції прошивки.

Поліпшена предметно-орієнтована модель на INAV представляє гібридний підхід. Завдяки рішенню з п. 2.3.4, ця модель використовує INAV Configurator для налаштування. Але, на відміну від комерційного рішення, розробник володіє процесом збірки прошивки, зберігаючи абсолютну гнучкість для додавання власних модулів (наприклад, шифрування AES з п. 2.3.2) до вихідного коду INAV перед компіляцією.

3.2.2. Підтримка сенсорів та алгоритмів

Базова предметно-орієнтована модель на Arduino IDE теоретично, підтримує будь-які модулі, але лише якщо для них написана або портована бібліотека. Розробник вручну вирішує питання сумісності та багів. З точки

зору алгоритмів, гнучкість максимальна: будь-які PID, Kalman, власні фільтри, але все доводиться розробляти з нуля.

Комерційна модель на INAV надає величезну базу підтримуваних серійних сенсорів, авто-виявлення в GUI. Користувач отримує фіксовані відлагоджені алгоритми: PID з експертними налаштуваннями, комплементарний фільтр (або опціональний EKF), модулі для автоматизованих місій.

Поліпшена предметно-орієнтована модель, використовуючи прошивку INAV, ця модель успадковує всю базу драйверів та алгоритмів комерційного рішення (EKF, PID, OSD). Однак апаратна гнучкість на базі плати розробника дозволяє інтегрувати нестандартні сенсори, хоча це й вимагає ручного мапінгу ресурсів (як описано в п. 2.3.4).

3.2.3. Автоматизація, відмовостійкість та вартість

Базова предметно-орієнтована модель на Arduino IDE. Автоматизація місій реалізується вручну в коді, що є трудомістким. Відмовостійкість (FailSafe) також потрібно реалізовувати самостійно. Перевагою є дуже низька вартість компонентів, але високий поріг по часу та знанням.

Комерційна модель на INAV надає готові режими: Angle, Horizon, Acro, NAV POSHOLD, ALTHOLD, RTH, NAV WP, Guided, Mission Control та вбудовані обробники втрати сигналу, резервування, warning messages, самотестування компонентів.. Це готова, гнучка і потужна платформа, що мінімізує людські помилки, але має вищу ціну компонентів.

Поліпшена предметно-орієнтована модель отримує всі готові режими автоматизації та FailSafe. При цьому вона зберігає низьку вартість компонентів, але додає до неї високий поріг по часу та знанням, не для розробки та написання коду з нуля, а для складної інтеграції (збірки, паяння, компіляції прошивки та мапінгу ресурсів).

3.2.4. Апаратні обмеження (STM32F401 vs STM32F405)

Наданий аналіз порівнює INAV з базовими платами Arduino ("16–20 МГц"). Наше ж дослідження порівнює 32-бітні контролери, і тут різниця також суттєва:

STM32F401 Має 84 МГц тактової частоти та 256 КБ Flash. Це значно потужніше за Arduino, але є обмеженням.

STM32F405: Має 168 МГц тактової частоти та 1 МБ Flash.

Це означає, що поліпшений ПК може запускати прошивку INAV, але робить це на значно слабшому залізі, ніж комерційний ПК. Це призводить до теоретичних обмежень, виявлених у п. 2.3.4: вищого навантаження на ЦП, необхідності відмови від ресурсомістких протоколів (DShot) та використання старих сенсорів (MPU6050), що вимагають ретельної фільтрації.

3.2.5. Висновки та рекомендації

Аналіз функціональності та гнучкості показує:

Базова предметно-орієнтована модель на Arduino IDE оптимальна "для навчальних експериментів, дослідження нових протоколів", але є неконкурентоспроможною для реальних завдань через обмеження функціоналу, стабільності та високі трудовитрати на створення базової логіки.

Комерційна модель на INAV є готовою, гнучкою і потужною платформою, що дає всі сучасні функції. Це ідеальний вибір для користувача, якому важлива стійкість, швидка інтеграція, підтримка сучасних сенсорів і автоматизація місій.

Поліпшена предметно-орієнтована модель є вибором дослідника. Вона поєднує низьку вартість компонентів з майже повним функціоналом комерційного рішення. Її головна перевага – абсолютна гнучкість модифікації, яка дозволяє впроваджувати власні алгоритми безпеки на рівні вихідного коду, зберігаючи при цьому зручність графічного конфігуратора.

Таблиця 3.3 – Узагальнення порівняльного аналізу моделей

Критерій	Базова предметно-орієнтована	Комерційна, INAV	Поліпшена предметно-орієнтована
Апаратна база (MCU)	STM32F401 (84МГц, 256КБ Flash)	STM32F405 (168МГц, 1МБ Flash)	STM32F401 (84МГц, 256КБ Flash)
Програмна база	Arduino IDE (простий loop())	INAV (RTOS)	Повний INAV (RTOS)
Функціональність (GPS/RTH)	Відсутня	Повна (з коробки)	Повна (успадкована від INAV)
Функціональність (OSD)	Відсутня	Вбудоване (AT7456E)	Є (через зовнішній MinimOSD)
Інтерфейс налаштування	Тільки код (Arduino IDE)	INAV Configurator	INAV Configurator
Гнучкість (Апаратна)	Максимальна (модульна)	Низька (AIO-плата)	Максимальна (модульна)
Гнучкість (Програмна)	Максимальна (простий код)	Низька ("чорна скринька")	Висока (свій код + "форк" INAV)
Потенціал безпеки (додавання AES-шифрування)	Легко	Дуже складно	Можливо (через кастомну збірку)
Надійність (збірки)	Низька (залежить від пайки)	Висока (промислова)	Низька (залежить від пайки)
Вартість компонентів	Дуже низька	Висока	Дуже низька
Вартість розробки (час/складність)	Середня	Мінімальна	Дуже висока (пайка + компіляція)

3.3 Теоретичний аналіз продуктивності та вимог до апаратних ресурсів

Цей аналіз є критичним для визначення теоретичної життєздатності поліпшеної програмно-орієнтованої моделі. Він порівнює обчислювальні вимоги трьох моделей з апаратними можливостями їхніх платформ, відповідно до другого критерію методики (п. 3.1).

3.3.1 Вимоги до мікроконтролера (ЦП та Пам'ять)

Комерційне рішення слугує еталоном високої продуктивності. Платформа (STM32F405, 168 МГц) використовується на повну. Багатозадачність (RTOS), складні обчислення EKF, опитування кількох сенсорів по SPI та 6 UART-портах вимагають високої тактової частоти. Прошивка INAV (файл .hex) займає значний обсяг, вимагаючи 1 МБ Flash-пам'яті, наявної в F405. Високий обсяг RAM також необхідний для буферів RTOS, EKF та навігаційних розрахунків.

Базове предметно-орієнтоване рішення демонструє надлишковість ресурсів. Платформа (STM32F401, 84 МГц) майже не навантажена. Простий лінійний цикл loop(), комплементарний фільтр та базовий PID-регулятор (як у бакалаврській роботі) споживають мізерну частку доступної обчислювальної потужності. Базовий скетч Arduino займає лише малу частину з 256 КБ Flash-пам'яті, доступної в F401.

Поліпшене предметно-орієнтоване рішення стикається з критичним дефіцитом ресурсів. Тут ми теоретично намагаємося запустити вимогливе ПЗ (INAV) на слабкому "залізі" (STM32F401). ЦП є головним вузьким місцем. 84 МГц недостатньо для одночасного виконання EKF, логіки INAV та високошвидкісного опитування сенсорів. Як було зазначено в аналізі (п. 2.3.4), це змушує йти на компроміси: знижувати looptime, відмовлятися від

ресурсомістких протоколів (DShot), а також створює додаткове навантаження на ЦП через використання повільнішої шини I2C для сенсора MPU6050. Пам'ять – це другий критичний фактор. Повна прошивка INAV фізично не поміщається у 256 КБ Flash-пам'яті STM32F401. Як випливає з моделі 2.3.4, єдиним рішенням є кастомна збірка прошивки — ручне видалення з вихідного коду INAV невикористовуваних драйверів, протоколів та функцій (напр., підтримки літаків) для зменшення розміру бінарного файлу.

3.3.2. Вимоги до периферії (I/O)

Комерційне рішення повністю використовує свою багату периферію (6+ UART, SPI, I2C), що дозволяє одночасно підключити приймач, GPS, OSD (внутрішньо по SPI), VTX-контроль, телеметрію та Blackbox.

Базова предметно-орієнтоване рішення використовує мінімум: 1x I2C (MPU6050), 1x UART (NEO-7M), 4x PWM-виходи.

Поліпшена предметно-орієнтована також стикається з обмеженнями. Для реалізації поліпшень необхідно задіяти:

- 1x I2C (MPU6050);
- 1x UART (NEO-7M GPS);
- 1x UART (MinimOSD, як у 2.3.3);
- 1x UART (Приймач ELRS/CRSF).

Мікроконтролер STM32F401 має лише 3 повноцінних UART-порти, що змушує використовувати всі доступні порти впритул, не залишаючи резерву для додаткової телеметрії (напр., 4G-модему з 2.3.2) або Blackbox.

Комерційне рішення демонструє апаратну збалансованість. Предметно-орієнтоване є теоретично життєздатною, але працює на абсолютній межі своїх апаратних можливостей, вимагаючи глибокої оптимізації коду та ретельного планування ресурсів. Структурована інформація подана таблицею 3.4.

Таблиця 3.4 – Продуктивність та вимоги до апаратних ресурсів

Критерій	Базова предметно-орієнтована)	Комерційна, INAV)	Поліпшена предметно-орієнтована
MCU	STM32F401	STM32F405	STM32F401
Тактова частота	84 МГц	168 МГц	84 МГц
Flash-пам'ять	256 КБ	1 МБ	256 КБ
Теоретичне навантаження на ЦП	Дуже низьке (простий loop())	Високе (RTOS + EKF)	Критичне (RTOS + EKF на слабкому ЦП)
Теоретичне навантаження на Flash	Дуже низьке	Високе (повна прошивка INAV)	Критичне (прошивка INAV ледь вміщується)
Задіяні I/O (UART)	1 (для GPS)	3+ (GPS, Rx, VTX, Telemetry)	3 (всі доступні) (GPS, Rx, OSD)

3.4 Порівняльний аналіз стійкості підсистем телеметрії до зовнішніх загроз

Цей аналіз базується на третьому критерії методики (п. 3.1) та оцінює здатність трьох моделей протистояти загрозам, ідентифікованим у п. 1.4, з урахуванням поліпшень, запропонованих у п. 2.3.2 (на основі матеріалів курсової роботи).

3.4.1 Стійкість до перехоплення (Interception)

Базова предметно-орієнтована система (змодельована на Arduino IDE) є критично вразливою. Дані телеметрії (включно з GPS-координатами точки "Дім") передаються у відкритому, незашифрованому вигляді через UART.

Типова комерційна система (на базі INAV) також є вразливою. Хоча сам протокол керування (ELRS) може мати шифрування, стандартна реалізація телеметрії (MSP або MAVLink), що йде на OSD або наземну станцію, за

замовчуванням не шифрується. Це створює той самий ризик витоку GPS-координат оператора, що й у базовій предметно-орієнтованій системі.

Поліпшена предметно-орієнтована система є теоретично стійкою. Це її ключова перевага. Завдяки гнучкості (п. 3.2) та наявності повного контролю над кодом (п. 2.3.4), розробник може імплементувати запропоновані в п. 2.3.2 поліпшення:

Шифрування AES-128: Вбудувати у кастомну збірку INAV модуль, який шифрує весь потік телеметрії перед відправкою.

Автентифікація HMAC: Захистити канал керування від ін'єкцій фальшивих команд.

Комерційна система (INAV) не надає такої можливості на рівні користувача.

3.4.2 Стійкість до спуфінгу (Spoofing)

Базова предметно-орієнтована система має нульову стійкість. Вона повністю довіряє даним з модуля NEO-7M. Фальсифікація GPS-координат призвела б до повної дезорієнтації (якби в ній були реалізовані GPS-режими).

Типова комерційна система (на базі INAV) демонструє базову стійкість. Наявність EKF дає системі певний імунітет. EKF порівнює дані GPS з даними IMU та барометра. Якщо GPS-координати раптово "стрибнуть" на 100 метрів (що неможливо фізично за даними акселерометра), EKF розпізнає це як викид і тимчасово ігноруватиме дані GPS.

Поліпшена предметно-орієнтована система має високу теоретичну стійкість. Завдяки своїй гнучкості, вона може реалізувати значно складніші методи захисту (з п. 2.3.2):

INS (Інерційна навігація): Покращена логіка EKF, що дозволяє короткочасний політ у режимі GPS-denied.

ML-аналіз аномалій: Впровадження алгоритмів (Local Outlier Factor, Isolation Forest), здатних виявити *плавний* спуфінг (поступове "відведення" координат), який може обдурити стандартний EKF.

3.4.3 Стійкість до глушіння (Jamming)

Базова предметно-орієнтована система має низьку стійкість. Реакція на втрату зв'язку (FailSafe) є примітивною (напр., вимкнення моторів) і має бути реалізована розробником вручну.

Типова комерційна система (на базі INAV) демонструє високу стійкість. Вона має надійний, відтестований спільнотою механізм FailSafe (RTH, автопосадка), а її протоколи (ELRS) використовують FHSS.

Поліпшена предметно-орієнтована система має найвищу теоретичну стійкість. Вона успадковує всі переваги комерційної системи (надійний FailSafe від INAV, FHSS від ELRS), але додає до них гнучкість базової предметно-орієнтованої системи. Це дозволяє реалізувати запропоновані в п. 2.3.2 поліпшення:

Резервні канали: Імплементація логіки перемикання на 4G/5G-модем у разі втрати основного радіоканалу.

Адаптивна потужність: Динамічне керування потужністю передавача телеметрії.

Типова комерційна система (INAV) пропонує хороший стандартний рівень захисту від базових загроз (FailSafe, EKF). Однак поліпшена предметно-орієнтована система, хоч і є складною в реалізації, є єдиною системою, яка завдяки своїй програмній та апаратній гнучкості може бути *теоретично* оснащена багаторівневим, нешаблонним захистом (AES, ML-аналіз, 4G-бекап), необхідним для протидії цілеспрямованим атакам.

3.5. Оцінка теоретичної ефективності та доцільності запропонованих поліпшень

Оцінка ефективності запропонованих поліпшень проводиться шляхом порівняння базової предметно-орієнтованої системи (змодельованої в п. 2.1) з поліпшеною предметно-орієнтованою системою (змодельованою в п. 2.3). Аналіз показує не просто приріст, а якісний стрибок у трьох ключових областях:

Функціональність (OSD + Сумісність з INAV): Базова система мала нульову (0%) функціональність у сферах, критичних для FPV та автономних польотів (відсутність OSD, відсутність GPS-режимів, відсутність GUI). Впровадження модуля MinimOSD (п. 2.3.3) та портування прошивки INAV (п. 2.3.4) теоретично підвищує функціональність до 100% паритету з комерційними системами. Апарат отримує повний навігаційний стек (RTH, Waypoint) та зручний графічний інтерфейс для налаштування.

Точність та надійність (EKF): Перехід від простого комплементарного фільтра (п. 2.1.2) до Розширеного Фільтра Калмана (EKF) (п. 2.3.1) кардинально підвищує точність оцінки стану апарата. EKF, що зливає дані з IMU, GPS та барометра, забезпечує значно надійніше утримання позиції та висоти, що є обов'язковою умовою для виконання автономних GPS-місій. Це поліпшення є необхідною передумовою для ефективної роботи навігаційного стеку INAV.

Безпека та стійкість до загроз (AES, ML, 4G): Це ключове поліпшення, яке виправдовує весь предметно-орієнтований підхід. Як базова, так і комерційна (INAV) системи є критично вразливими до перехоплення телеметрії (п. 3.4). Поліпшена система, завдяки своїй гнучкості, дозволяє реалізувати запропоновані в п. 2.3.2 методи:

Шифрування AES підвищує захист каналу телеметрії від перехоплення GPS-координат оператора з нульового до високого рівня.

ML-аналіз аномалій та резервування каналу (4G) підвищують стійкість до спуфінгу та глушіння до рівня, теоретично недосяжного для стандартних комерційних рішень.

Запропоновані поліпшення роблять предметно-орієнтовані системи високоефективними для військового застосування. Комерційні системи є "відомою мішенню" (known target) зі стандартними, добре вивченими протоколами. Натомість поліпшена предметно-орієнтована система є "чорним лебедем" для засобів РЕБ: вона використовує нестандартні, кастомні методи шифрування та логіку захисту (AES, ML-аналіз), що робить її перехоплення та аналіз значно складнішим завданням.

3.6. Порівняльний економічний аналіз

Економічний аналіз (критерій 4 з методики 3.1) порівнює типову комерційну систему (напр., SpeedyBee F405) з поліпшеною предметно-орієнтованою системою (набір модулів на STM32F401).

1. Вартість компонентів (Bill of Materials - BOM). Це єдина категорія, де предметно-орієнтований підхід беззаперечно виграє.

Комерційна система (AIO): Вартість інтегрованої плати становить приблизно $40–$60.

Предметно-орієнтована система (модулі): Сукупна вартість компонентів (плата STM32F401 + MPU6050 + NEO-7M + MinimOSD) становить приблизно $10–$25.

Вартість апаратних компонентів предметно-орієнтованої системи є на 50-70% нижчою, ніж у комерційного аналога.

2. Вартість розробки (Non-Recurring Engineering - NRE). Тут ситуація діаметрально протилежна.

Комерційна система: Має майже нульову вартість розробки. Час інженера витрачається лише на налаштування в графічному інтерфейсі.

Предметно-орієнтована система: Вимагає колосальних трудовитрат. Як було проаналізовано в п. 2.3.4 та 3.3, інженер повинен витратити значний час на:

a. Фізичне паяння та збірку модулів.

b. Кастомну компіляцію прошивки INAV під обмежені ресурси (256КБ Flash).

c. Складне ручне налаштування "resource map" (пінмапу) в CLI.

d. Налагодження та пошук проблем (напр., високе навантаження ЦП від MPU6050).

3. Вартість життєвого циклу – ремонт та масштабування).

Комерційна система має низьку ремонтопридатність. При виході з ладу одного компонента (напр., згорів BEC 5V або один порт UART) заміни потребує вся плата ($40–$60).

Предметно-орієнтована система має високу ремонтопридатність. При збої, наприклад, GPS-модуля, замінюється лише цей модуль (вартістю $5 – $7).

Для одиничного цивільного проєкту комерційна система є економічно доцільнішою, оскільки висока вартість компонентів компенсується нульовою вартістю розробки.

Однак для військової сфери та серійного виробництва поліпшена предметно-орієнтована система є значно вигіднішою. Висока вартість NRE (розробки) є одноразовою інвестицією. Після того, як "Модель 3" розроблена та відтестована, вартість її серійного відтворення стає на 50-70% нижчою за комерційні аналоги.

Це, у поєднанні з високою ремонтопридатністю та кастомною безпекою (п. 3.5), робить її ідеальним кандидатом для виробництва масових, дешевих та стійких до РЕБ безпілотних літальних апаратів.

ВИСНОВОК

У магістерській роботі було поставлено та вирішено актуальну науково-технічну задачу: дослідження та теоретичне обґрунтування шляхів поліпшення підсистем керування та телеметрії БПЛА. Мета роботи була досягнута шляхом глибокого теоретичного порівняльного аналізу трьох моделей: базової предметно-орієнтованої на основі бакалаврської роботи, типової комерційної на базі F405 + INAV та поліпшеної предметно-орієнтованої.

Проведений аналіз виявив фундаментальне протиріччя. Типові комерційні системи пропонують високу інтеграцію, багату функціональність "з коробки", як-от готові GPS-режими, EKF, OSD, та високу надійність. Однак вони є "чорною скринькою", що обмежує гнучкість та демонструє критичні вразливості, оскільки їхні стандартні протоколи не шифрують телеметрію, що призводить до ризику витоку координат оператора. З іншого боку, базова предметно-орієнтована система демонструє повну гнучкість, але є функціонально неконкурентоспроможною через відсутність OSD, примітивні фільтри та необхідність налаштування через код.

Для вирішення цього протиріччя в роботі була розроблена та теоретично обґрунтована поліпшена предметно-орієнтована модель. Ключові поліпшення включали не написання коду "з нуля", а портування повноцінної прошивки INAV на кастомну апаратну базу STM32F401, а також апаратну інтеграцію модуля MinimOSD та впровадження методів захисту з курсової роботи.

Порівняльний аналіз довів, що запропоновані поліпшення теоретично ефективні.

За функціональністю поліпшена модель досягає приблизно 100% паритету з комерційною, отримуючи повний навігаційний стек, що включає RTH та Waypoint, і зручний графічний інтерфейс INAV Configurator.

За продуктивністю поліпшена модель є життєздатною, але працює на межі апаратних можливостей свого заліза – STM32F401, 256КБ Flash. Це вимагає кастомної компіляції прошивки для економії ресурсів та накладає обмеження, наприклад, відмову від DShot.

За стійкістю до загроз поліпшена модель демонструє кардинальну перевагу. На відміну від вразливої комерційної системи, її гнучкість дозволяє вбудувати нестандартні методи захисту, як-от шифрування телеметрії AES та ML-аналіз спуфінгу, що робить її стійкою до цілеспрямованих атак.

За економікою, при високій вартості розробки – часу інженера, – вартість компонентів поліпшеної системи є на 50-70% нижчою за комерційну, а модульність забезпечує високу ремонтопридатність.

На основі проведеного дослідження сформульовано наступні рекомендації. Щодо вибору архітектури: комерційні системи COTS рекомендовані для цивільних завдань та швидкого прототипування, де швидкість розгортання є пріоритетом. Поліпшені предметно-орієнтовані системи рекомендовані для науково-дослідних робіт та завдань військової сфери.

Щодо впровадження поліпшень: визнано недоцільним створення базових систем з нуля, наприклад, на Arduino IDE. Рекомендовано використовувати гібридний підхід: портування існуючих відкритих екосистем INAV на власну, модульну та дешеву апаратну базу. Саме ця комбінація – низька вартість компонентів, висока ремонтопридатність та нестандартна кастомна безпека – робить поліпшені предметно-орієнтовані системи оптимальним кандидатом для масового серійного виробництва у сферах, де стійкість до РЕБ є критичним фактором.

Подальші дослідження можуть бути спрямовані на практичну реалізацію розробленої поліпшеної моделі та експериментальну перевірку її стійкості до реальних загроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Книш Б.П., Кулик Я.А., Барабан М.В. Класифікація безпілотних літальних апаратів та їх використання для доставки товарів // Вісник Хмельницького національного університету. – 2018. – №3(261). – С. 246–252. URL: https://journals.khnu.km.ua/vestnik/pdf/tech/pdfbase/2018/2018_3/jrn/pdf/42.pdf (дата звернення: 05.11.2024).

2. ДП «Антонов» розробляє нові безпілотні авіакомплекси. ОПК. URL: https://opk.com.ua/%D0%B4%D0%BF-%D0%B0%D0%BD%D1%82%D0%BE%D0%BD%D0%BE%D0%B2-%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BB%D1%8F%D1%94-%D0%BD%D0%BE%D0%B2%D1%96-%D0%B1%D0%B5%D0%B7%D0%BF%D1%96%D0%BB%D0%BE%D1%82/ (дата звернення: 05.11.2024).

3. MALE and HALE Drone Developments: Evolving Existing Systems and Introducing New Aircraft. European Security & Defence. 2025. URL: https://euro-sd.com/2025/06/articles/44958/male-and-hale-drone-developments-evolving-existing-systems-and-introducing-new-aircraft/ (дата звернення: 10.06.2025).

4. БПЛА «Сокіл-300». Промисловий Дайджест. URL: https://www.psdinfo.pro/post/%D0%B1%D0%BF%D0%BB%D0%B0-%D1%81%D0%BE%D0%BA%D1%96%D0%BB-300 (дата звернення: 07.11.2024).

5. Класифікація дронів: які види та типи бувають?. Bezpeka-shop.com. URL: https://www.bezpeka-shop.com/ua/blog/poleznye-sovety/klassifikatsiya-dronov-kakie-vidy-i-tipy-byvayut-chast-pervaya/ (дата звернення: 10.11.2024).

6. Гуцул, Т. ., Жежера, І., Ткач, В. Особливості класифікації та методів вибору БПЛА // Технічні науки та технології. – 2023. – №4(30). – С. 201–212. –

DOI: https://doi.org/10.25140/2411-5363-2022-4(30)-201-212. (дата звернення: 10.11.2024).

7. Hassanalian M., Abdelkefi A. Classifications, applications, and design challenges of drones: A review. Aerospace Science and Technology. 2017. Т. 66. С. 1–21. URL: https://www.sciencedirect.com/science/article/abs/pii/S0376042116301348 (дата звернення: 12.11.2024).

8. Criollo L., Mena-Arciniega C., Xing S. Classification, military applications, and opportunities of unmanned aerial vehicles. Aviation. 2024. Т. 28, № 2. С. 115–127. URL: https://journals.vilniustech.lt/index.php/Aviation/article/view/21672 (дата звернення: 12.11.2024).

9. Hlotov V., Hunina A., Kniaziev S., Kolesnichenko V., Prokhorchuk O. Analysis of application of the UAVs for military tasks. Photogrammetry, Geoinformation Systems and Cartography. Lviv Polytechnic National University. URL: https://ena.lpnu.ua:8443/server/api/core/bitstreams/be4f296b-96e9-4656-a12d-1759463702ee/content (дата звернення: 15.11.2024).

10. Asmaa I., Boukhdir K., Medromi H. UAV Control Architecture: Review. International Journal of Advanced Computer Science and Applications (IJACSA). 2019. Т. 10, №. 11. С. 674–683. URL: https://thesai.org/Downloads/Volume10No11/Paper_86-UAV_Control_Architecture_Review.pdf (дата звернення: 15.11.2024).

11. Zhukov I., Dolintse B. Enhancing accuracy of information processing in onboard subsystems of UAVs. Technology Audit and Production Reserves. 2023. Т. 5, № 2(73). С. 6–10. URL: https://journals.uran.ua/tarp/article/view/287700 (дата звернення: 20.01.2025).

12. Lu X. Design of UAV Flight Control System. International Journal of Frontiers in Engineering Technology. 2022. Т. 4, Issue 4. С. 89–92. doi: 10.25236/IJFET.2022.040412. URL: https://francis-press.com/papers/6391 (дата звернення: 20.01.2025).

13. Frigioescu T., Dombrovschi M., Badea G., Cican G., Condruz R., Crunteanu D.-E. Development of an UAV Platform for Autopilot Implementation and Validation. TURBO. 2023. T. X, № 2. C. 6–16. URL: https://journals.indexcopernicus.com/api/file/viewByFileId/1984467 (дата звернення: 21.01.2025).

14. Betaflight, INAV vs ArduPilot: Which FC Firmware is Right for You? MEPS FPV. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mepsking.shop/blog/betaflight-ardupilot-inav-fc-firmware.html (дата звернення: 20.11.2024).

15. Ebeid E.S.M., Skriver M., Terkildsen K.H., Jensen K., Schultz U. P. A Survey of Open-Source UAV Flight Controllers and Flight Simulators. Microprocessors and Microsystems. 2018. T. 61. C. 11–20. URL: https://findresearcher.sdu.dk/ws/portalfiles/portal/143278283/A_Survey_of_Open_Source_UAV_Flight_Controllers_and_Flight_Simulators.pdf (дата звернення: 20.11.2024).

16. ArduPilot Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://ardupilot.org/ (дата звернення: 25.01.2025).

17. Lienkov S., Myasischev A., Sieliukov O., Pashkov O., Zhyrov G., Zinchyk A. Checking the Flight Stability of a Rotary UAV in Navigation Modes for Different Firmware // CEUR Workshop Proceedings. – 2022. – T. 3126. – C. 70–74. – URL: https://ceur-ws.org/Vol-3126/paper7.pdf (дата звернення: 25.01.2025).

18. Flight Modes — Plane documentation. ArduPilot Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://ardupilot.org/plane/docs/flight-modes.html (дата звернення: 26.01.2025).

19. Copter Flight Modes — Copter documentation. ArduPilot Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://ardupilot.org/copter/docs/flight-modes.html (дата звернення: 26.01.2025).

20. ArduPilot VS PX4 | Which Is Better? Dojo for Drones. 2023. [Електронний ресурс] – Режим доступу до ресурсу: https://dojofordrones.com/ardupilot-vs-px4/ (дата звернення: 05.04.2025).

21. PX4 vs ArduPilot: Build Powerful Drone Control Apps. A-bots. 2025. [Електронний ресурс] – Режим доступу до ресурсу: https://a-bots.com/blog/PX4-vs-ArduPilot (дата звернення: 12.06.2025).

22. Flight Controller Explained: How to Choose the Best FC for FPV Drone. OscarLiang.com. 21.08.2025. URL: https://oscarliang.com/flight-controller/ (дата звернення: 22.08.2025).

23. FC Firmware List. OscarLiang.com. 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://oscarliang.com/fc-firmware/ (дата звернення: 25.11.2024).

24. Betaflight: Open Source Flight Controller Firmware. GitHub. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/betaflight/betaflight (дата звернення: 28.01.2025).

25. Betaflight PID Tuning Guide. Betaflight Docs [Електронний ресурс] – Режим доступу до ресурсу: https://www.betaflight.com/docs/wiki/guides/current/PID-Tuning-Guide (дата звернення: 28.01.2025).

26. Setting Up Waypoints in INAV. SpeedyBee Zendesk [Електронний ресурс] – Режим доступу до ресурсу: https://speedybee.zendesk.com/hc/en-us/articles/19630162452507-Setting-Up-Waypoints-in-INAV (дата звернення: 30.01.2025).

27. Gunes E., Duzkaya H. Arduino Based Flight Control Card Design and Quadcopter Integration. Gazi University Journal of Science, Part A: Engineering and Innovation. 2024. Т. 11, № 2. С. 392–406. URL: https://dergipark.org.tr/tr/download/article-file/3913820 (дата звернення: 05.02.2025).

28. Оптоволоконні FPV-дрони, нова загроза на полі бою в російсько-українській війні. Hackyourmom.com. [Електронний ресурс] – Режим доступу до ресурсу: https://hackyourmom.com/drony/optovolokonni-fpv-drony-nova-zagroza-na-poli-boyu-v-rosijsko-ukrayinskij-vijni/. (дата звернення: 10.04.2025).

29. Hardware Compatibility [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ZZ-Cat/CRSFforArduino/wiki/Hardware-Compatibility (дата звернення: 10.02.2025).

30. STM32CubeProgrammer [Електронний ресурс] – Режим доступу до ресурсу: https://www.st.com/en/development-tools/stm32cubeprog.html (дата звернення: 10.02.2025).

31. Arduino Core for STM32 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/stm32duino/Arduino_Core_STM32 (дата звернення: 10.02.2025).

32. rfetick/Kalman. GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/rfetick/Kalman (дата звернення: 15.02.2025).

33. Topic: MPU6050 KalmanFilter. Arduino Forum [Електронний ресурс] – Режим доступу до ресурсу: https://forum.arduino.cc/t/mpu6050-kalmanfilter/316127 (дата звернення: 15.02.2025).

34. nut-code-monkey/KalmanFilter-for-Ardui № GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/nut-code-monkey/KalmanFilter-for-Arduino (дата звернення: 15.02.2025).

35. Polyotnyi Kontroller SpeedyBee F405 Wing APP [Електронний ресурс] – Режим доступу до ресурсу: https://modelistam.com.ua/polyotnyi-kontroller-speedybee-f405-wing-app-p-47950/ (дата звернення: 20.02.2025).

36. SpeedyBeeF4 V3 Documentation. ArduPilot.org [Електронний ресурс] – Режим доступу до ресурсу: https://ardupilot.org/copter/docs/common-speedybeef4-v3.html (дата звернення: 20.02.2025).

37. SpeedyBee F405 V4 Stack Manual [Електронний ресурс] – Режим доступу до ресурсу: https://store-fhxxhuiq8q.mybigcommerce.com/product_images/img_SpeedyBee_F405_V4_Stack/SpeedyBee_F405_V4_Stack_Manual_EN.pdf (дата звернення: 20.02.2025).

38. FPV Drone Flight Controllers Explained (F1, F3, F4, F7, H7): Latest Version 2024. MEPS FPV. URL: https://www.mepsking.shop/blog/fpv-drone-flight-

controllers-explained-f1-f3-f4-f7-h7-latest-version-2024.html (дата звернення: 25.02.2025).

39. iNavFlight/inav. GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/iNavFlight/inav (дата звернення: 01.03.2025).

40. UAV Tech PID Tuning Principles. TheUAVTech.com [Електронний ресурс] – Режим доступу до ресурсу: https://theuavtech.com/wp-content/uploads/2020/10/UAV-Tech-PID-Tuning-Principles.pdf (дата звернення: 05.03.2025).

41. INAV Modes. Scribd. [Електронний ресурс] – Режим доступу до ресурсу: https://www.scribd.com/document/709118855/INAV-Modes (дата звернення: 05.03.2025).

42. ArduPilot Navigation: Extended Kalman Filter Overview. ArduPilot.org. [Електронний ресурс] – Режим доступу до ресурсу: https://ardupilot.org/copter/docs/common-apm-navigation-extended-kalman-filter-overview.html (дата звернення: 10.03.2025).

43. Kalman Filter vs Complementary Filter: Which Reduces Drift? Patsnap. [Електронний ресурс] – Режим доступу до ресурсу: https://eureka.patsnap.com/report-kalman-filter-vs-complementary-filter-which-reduces-drift (дата звернення: 10.03.2025).

ДОДАТОК А

Код програмного забезпечення системи

```
#include <AlfredoCRSF.h>
#include <HardwareSerial.h>
#include <HardwareTimer.h>
#include <Ardui №h>
#include <Servo.h>
#include <TinyGPS++.h>
#include <Adafruit_MPU6050.h>
#include <MPU6050_light.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

#define PIN_LED PC13  // відповідає за вбудований LED
#define RX_PIN PB7 // Rx пін для RP3
#define TX_PIN PB6 // Tx пін для RP3
#define SRX_PIN PA3 // Rx пін для NEO7-M
#define STX_PIN PA2 // Tx пін для NEO7-M

#define PIN_SNS_VIN PA7 // аналоговий пін для зчитування параметрів акумулятора
#define PWM_PIN PB10  // пін генерації ШІМ сигналу
#define LG1_PIN PB1 // пін управління кнопкою FR
#define LG2_PIN PB2 // пін управління кнопкою CH
#define CTRL_PIN PB0 // контролюючий пін


// Ініціалізація UART портів
HardwareSerial gpsSerial(SRX_PIN, STX_PIN);
HardwareSerial crsfSerial(RX_PIN, TX_PIN);

// Ініціалізація об'єктів для роботи з датчиками
TinyGPSPlus gps;
Adafruit_MPU6050 mpu;
MPU6050 mpuLight(Wire);

// Ініціалізація об'єкта для роботи з CRSF
AlfredoCRSF crsf;

// Ініціалізація об'єктів для роботи з мотором та сервоприводами
Servo flmotor;
Servo rightServo;
Servo leftServo;
```

```
      // Глобальні змінні для таймера
      TIM_TypeDef *Instance = TIM2;
      HardwareTimer *timer;

      // Ініціалізація змінних
      sensors_event_t a, g, temp;
      int AVin, charge, motor, CH1_value, CH2_value, CH3_value;
      int    aileronInput,    elevatorInput,    rightServoOutput,
leftServoOutput;
      int CH1, CH2, CH3, CH4, CH5, CH6, CH7, CH8, CH9;
      float deg_to_rad, batteryVoltage;
      float angleX_rad, angleY_rad, angleZ_rad;
      bool mpuConnected = false;


      void setup() {
        Serial.begin(9600);
        Serial.println("Ardupilot initialized");

        crsfSerial.begin(420000);    //  Ініціалізація  UART1  для
зв'язку з AlfredoCRSF бібліотекою
        gpsSerial.begin(9600);     // Ініціалізація UART2 для GPS
модуля

        pinMode(PIN_LED, OUTPUT);
        pinMode(LG1_PIN, OUTPUT);
        pinMode(LG2_PIN, OUTPUT);
        pinMode(CTRL_PIN, OUTPUT);
        digitalWrite(CTRL_PIN, HIGH);

        // Ініціалізація таймеру
        timer = new HardwareTimer(Instance);
        timer->setOverflow(6600);
        timer->setMode(3, TIMER_OUTPUT_COMPARE_PWM1, PWM_PIN);
        timer->setCaptureCompare(3, 0);
        timer->resume();

        digitalWrite(PIN_LED, LOW);   // Вимикаємо світлодіод на
старті

        if (!crsfSerial)
          while     (1)     Serial.println("Invalid     crsfSerial
configuration");
        crsf.begin(crsfSerial);

        if (!gpsSerial)
          while     (1)     Serial.println("Invalid     gpsSerial
configuration");

        //Налаштування I2C
        Wire.setSDA(PB9);
        Wire.setSCL(PB8);
```

```
  Wire.begin();

  //Налаштування параметрів MPU6050
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  byte status = mpuLight.begin();
  if (!mpu.begin() && status != 0) {
    Serial.println(F("Failed to connect to MPU6050"));
  } else {
    delay(1000);

    // Обчислити офсети для гіроскопа та акселерометра
    mpuLight.calcOffsets();
    Serial.println("Done!\n");
  }

  deg_to_rad = 3.14 / 180.0;

  flmotor.attach(PA1);
  //Автоматичне налаштування THROTTLE
  flmotor.writeMicroseconds(2300);
  delay(2000);
  flmotor.writeMicroseconds(800);
  delay(6000);

  rightServo.attach(PB4);
  leftServo.attach(PB3);
}

void loop() {
  crsf.update();
  mpuLight.update();

  gpsStatus();

  mpuSatus();

  BatteryData();

  ControlPanel();

  updateLinkStatusLed();
}

//ФУНКЦІЇ ОТРИМАННЯ, ФОРМУВАННЯ ТА ОБРОБЛЕННЯ ДАНИХ

void gpsStatus() {
  while (gpsSerial.available() > 0) {
    gps.encode(gpsSerial.read());
    if (gps.location.isUpdated()) {
```

```
            //gpsTextInfo();

            sendGps(gps.location.lat(),       gps.location.lng(),
gps.speed.kmph(),    gps.course.deg(),    gps.altitude.meters(),
gps.satellites.value());
          }
        }
      }


      void mpuSatus() {
        mpu.getEvent(&a, &g, &temp);

        // Отримання кутів у градусах
        float angleX_deg = mpuLight.getAngleX();
        float angleY_deg = mpuLight.getAngleY();
        float angleZ_deg = mpuLight.getAngleZ();

        // Перетворення кутів з градусів у радіани
        angleX_rad = angleX_deg * deg_to_rad;
        angleY_rad = angleY_deg * deg_to_rad;
        angleZ_rad = angleZ_deg * deg_to_rad;

        //mpuTextInfo();

        // Виклик функцій sendAttitude та sendMySensor для
відправки даних
        sendAttitude(angleY_rad, angleX_rad, angleZ_rad);
        sendMySensor(temp.temperature);

        Serial.println("");
      }


      void BatteryData() {
        //Зчитування напруги
        AVin = analogRead(PIN_SNS_VIN);
        //Перетворення даних у відповідний тип
        batteryVoltage = ((float)AVin / 251.97 * 2);
        //Перетворення у відсотки заряду
        charge = map(AVin, 554, 1058, 0, 100);

        sendBattery(batteryVoltage, 1.2, 10, charge);
      }


      //ФУНКЦІЇ ФОРМУВАННЯ ПАКЕТІВ

      void  sendGps(float  latitude,  float  longitude,  float
groundspeed, float heading, float altitude, float satellites) {
        crsf_sensor_gps_t crsfGps = { 0 };
```

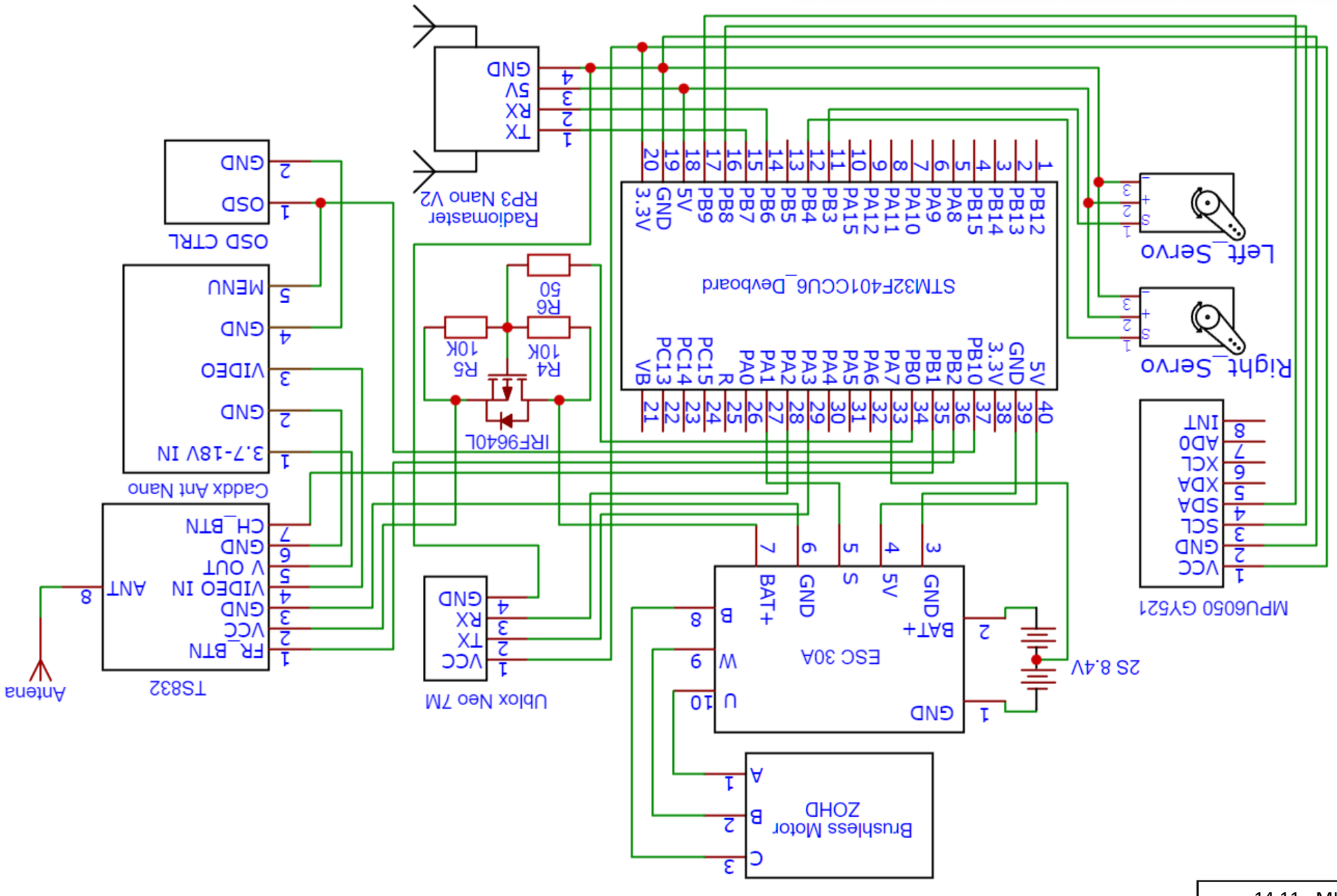
```
        crsfGps.latitude    =    htobe32((int32_t)(latitude    *
10000000.0));
        crsfGps.longitude    =    htobe32((int32_t)(longitude    *
10000000.0));
        crsfGps.groundspeed  =  htobe16((uint16_t)(groundspeed  *
10.0));
        crsfGps.heading = htobe16((int16_t)(heading * 1000.0));
        crsfGps.altitude = htobe16((uint16_t)(altitude + 1000.0));
        crsfGps.satellites = (uint8_t)(satellites);
        crsf.queuePacket(CRSF_SYNC_BYTE,      CRSF_FRAMETYPE_GPS,
&crsfGps, sizeof(crsfGps));
      }

      void sendAttitude(float pitch, float roll, float yaw) {
        crsf_sensor_attitude_t crsfAttitude = { 0 };

        crsfAttitude.pitch = htobe16((int16_t)(pitch * 10000.0));
        crsfAttitude.roll = htobe16((int16_t)(roll * 10000.0));
        crsfAttitude.yaw = htobe16((int16_t)(yaw * 10000.0));
        crsf.queuePacket(CRSF_SYNC_BYTE, CRSF_FRAMETYPE_ATTITUDE,
&crsfAttitude, sizeof(crsfAttitude));
      }

      void sendMySensor(float temp) {
        crsf_sensor_mysensor_t crsfMySensor = { 0 };

        crsfMySensor.temp = htobe16((int16_t)(temp * 100.0));
        crsf.queuePacket(CRSF_SYNC_BYTE,
CRSF_FRAMETYPE_MY_SENSOR, &crsfMySensor, sizeof(crsfMySensor));
      }

      static void sendBattery(float voltage, float current, float
capacity, float remaining) {
        crsf_sensor_battery_t crsfBatt = { 0 };

        crsfBatt.voltage = htobe16((uint16_t)(voltage * 10.0));
        crsfBatt.current = htobe16((uint16_t)(current * 10.0));
        crsfBatt.capacity = htobe16((uint16_t)(capacity)) << 8;
        crsfBatt.remaining = (uint8_t)(remaining);
        crsf.queuePacket(CRSF_SYNC_BYTE,
CRSF_FRAMETYPE_BATTERY_SENSOR, &crsfBatt, sizeof(crsfBatt));
      }

      //ФУНКЦІЇ УПРАВЛІННЯ КРИЛОМ

      void ControlPanel() {
        CH1 = crsf.getChannel(1);
        CH2 = crsf.getChannel(2);
        CH3 = crsf.getChannel(3);
        CH4 = crsf.getChannel(4);
        CH5 = crsf.getChannel(5);
        CH6 = crsf.getChannel(6);
```

```
CH7 = crsf.getChannel(7);
CH8 = crsf.getChannel(8);
CH9 = crsf.getChannel(9);

if (CH5 == 2000) {
  if (CH1 < 1000) {
    timer->setCaptureCompare(3, 1440);
  } else if (CH1 > 2000) {
    timer->setCaptureCompare(3, 3440);

  } else if (CH2 < 1000) {
    timer->setCaptureCompare(3, 440);
  } else if (CH2 > 2000) {
    timer->setCaptureCompare(3, 2440);

  } else if (CH4 < 1000) {
    digitalWrite(LG1_PIN, LOW);
  } else if (CH4 > 2000) {
    digitalWrite(LG2_PIN, LOW);

  } else if (CH9 == 2000) {
    timer->setCaptureCompare(3, 4040);
  } else {
    timer->setCaptureCompare(3, 0);
    digitalWrite(LG1_PIN, HIGH);
    digitalWrite(LG2_PIN, HIGH);
  }
} else {
  timer->setCaptureCompare(3, 0);
}

if (CH6 == 1503) {
} else {
  FLMotor();
}
if (CH7 == 1503) {
  rightServo.writeMicroseconds(1350);
  leftServo.writeMicroseconds(1600);
} else {
  FLWings();
}

//FlControlsInfo();

if (CH8 == 2000) {
  digitalWrite(CTRL_PIN, LOW);
} else {
  digitalWrite(CTRL_PIN, HIGH);
}


//ControlPanelInfo();
```

```
}

void FLMotor() {
  //Обмеження потужності до 70%
  if (CH6 == 2000) {
    CH3_value = constrain(CH3, 994, 1704);
  } else {
    CH3_value = constrain(CH3, 994, 2008); //потужність 100%
  }
  //Перетворення значень джойстика
  motor = map(CH3_value, 994, 2008, 800, 2300);
  constrain(motor, 800, 2300);
  //Подання даних драйверу
  flmotor.writeMicroseconds(motor);
}

void FLWings() {
  CH1_value = constrain(CH1, 1000, 2000);
  CH2_value = constrain(CH2, 1000, 2000);

  //Зчитування даних джойстика
  aileronInput = map(CH1_value, 1000, 2000, 500, -500);
  elevatorInput = map(CH2_value, 1000, 2000, 500, -500);

  //Формула Синхронізації закрилок
  rightServoOutput = constrain(1500 + aileronInput -
elevatorInput + 40, 600, 2100);
  leftServoOutput = constrain(1500 + aileronInput +
elevatorInput - 80, 600, 2100);

  //Подання даних на севомотори
  rightServo.writeMicroseconds(rightServoOutput);
  leftServo.writeMicroseconds(leftServoOutput);
}

//Функція перевірки підключення
void updateLinkStatusLed() {
  if (crsf.isLinkUp()) {
    digitalWrite(PIN_LED, HIGH);
  } else {
    digitalWrite(PIN_LED, LOW);
    // Perform the failsafe action
  }
}


// ФУНКЦІЇ ВИВОДУ ІНФОРМАЦІЇ В SERIAL MONITOR

void gpsTextInfo() {
  Serial.print("Latitude: ");
  Serial.println(gps.location.lat(), 6);
  Serial.print("Longitude: ");
```

```
        Serial.println(gps.location.lng(), 6);
        Serial.print("Altitude: ");
        Serial.println(gps.altitude.meters());
        Serial.print("Speed: ");
        Serial.println(gps.speed.kmph());
        Serial.print("Heading: ");
        Serial.println(gps.course.deg());
        Serial.print("Satellites: ");
        Serial.println(gps.satellites.value());   // Відображення
кількості супутників
        Serial.print("Date/Time: ");
        Serial.print(gps.date.day());
        Serial.print("/");
        Serial.print(gps.date.month());
        Serial.print("/");
        Serial.print(gps.date.year());
        Serial.print(" ");
        Serial.print(gps.time.hour());
        Serial.print(":");
        Serial.print(gps.time.minute());
        Serial.print(":");
        Serial.print(gps.time.second());
        Serial.println("");
      }

      void mpuTextInfo() {

        Serial.print("Angle X: ");
        Serial.print(angleX_rad);
        Serial.print("\tAngle Y: ");
        Serial.print(angleY_rad);
        Serial.print("\tAngle Z: ");
        Serial.println(angleZ_rad);
        Serial.print("Temperature: ");
        Serial.print(temp.temperature);
        Serial.println(" degC");
      }

      void dataTextInfo() {

        Serial.print("Analog: ");
        Serial.println(AVin);
        Serial.print("Real: ");
        Serial.println(batteryVoltage);
      }

      void ControlPanelInfo() {

        Serial.print(CH1);
        Serial.print(", ");
        Serial.print(CH2);
        Serial.print(", ");
```

```
  Serial.print(CH3);
  Serial.print(", ");
  Serial.print(CH4);
  Serial.print(", ");
  Serial.print(CH5);
  Serial.print(", ");
  Serial.print(CH6);
  Serial.print(", ");
  Serial.print(CH7);
  Serial.print(", ");
  Serial.print(CH8);
  Serial.print(", ");
  Serial.print(CH9);
  Serial.println(" ");
}

void FlControlsInfo() {
  Serial.print("Controller: ");
  Serial.print(CH3_value);
  Serial.print(", Motor: ");
  Serial.println(motor);

  Serial.print("Controller: ");
  Serial.print(CH1_value);
  Serial.print(", Right: ");
  Serial.println(rightServoOutput);

  Serial.print("Controller: ");
  Serial.print(CH2_value);
  Serial.print(", Left: ");
  Serial.println(leftServoOutput);
}
```



Змн.	Арк.	№ докум.	Підп.	Дата	15.03 - МКР.1941 "С" 24.10.29.02 Е4				
Розроб.		В.Ю. Вернигора							
Перевір.		В.В. Шкарупило							
Т. контр.					Розміщення та підключення периферії безпілотного літального апарату				
Н. контр.		В.В. Шкарупило							
Затверд.		Д.Ю. Касаткін				Літ.	Маса	Масшт.	
								1:1	
					Монтажна схема з'єднань системи БПЛА	Арк. 101		Аркушів 1	
						НУБіП КСІМ-24006м			

14.11 - МКР.1941 "С"
24.10.29.02.Е4