**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 вдосконалення системи розпізнавання образів для дронів»

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., професор _____ Шкарупило В.В
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи
к.т.н., доцент _____ Місюра М.Д.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Баранов А.С.
(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

Баранову Артему Сергійовичу
(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 вдосконалення системи розпізнавання образів для дронів»

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 є вимоги до створення програмного емулятора системи розпізнавання образів для дронів, специфікація модулів, моделі комп'ютерного зору та параметри телеметрії. Також задано умови роботи симуляції, сценарії середовища та формат обміну даними між компонентами системи.

Перелік питань, що підлягають дослідженню:

1 дослідження архітектури емулятора та принципів моделювання сенсорних даних. Аналіз впливу параметрів середовища на точність симуляції.
2 Вивчення роботи алгоритмів комп'ютерного зору та їх продуктивності. Оцінювання точності, швидкодії та стійкості інференсу.
3 Аналіз інтеграції системи з телеметричними та HIL-модулями. Перевірка стабільності, логування та відтворюваності експериментів.

Перелік графічного матеріалу (за потреби) _____

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

**Керівник магістерської кваліфікаційної роботи** _____ Місюра М.Д.
( підпис ) (прізвище та ініціали)

**Завдання прийняв до виконання** _____ Баранов А.С.
(підпис ) (прізвище та ініціали)

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. AR — augmented reality, доповнена реальність
2. MR — mixed reality, змішана реальність
3. VR — virtual reality, віртуальна реальність
4. 3D — тривимірне геометричне подання об’єктів
5. FPS — frames per second, частота відтворення кадрів
6. VIO — visual-inertial odometry, візуально-інерційна одометрія
7. PnP — Perspective-n-Point, алгоритм визначення пози за ключовими точками
8. EPnP — Efficient PnP, оптимізований метод оцінювання поз
9. IMU — inertial measurement unit, інерціальна вимірювальна система
10. SLAM — simultaneous localization and mapping, одночасна локалізація і побудова карти
11. ORB — Oriented FAST and Rotated BRIEF, детектор та дескриптор ключових точок
12. AKAZE — Accelerated KAZE, алгоритм екстракції ключових точок
13. ArUco — маркерна система трекінгу для комп’ютерного зору
14. JWT — JSON Web Token, токен авторизації
15. SSO — single sign-on, єдиний вхід у систему
16. OIDC — OpenID Connect, протокол автентифікації
17. API — application programming interface, інтерфейс програмної взаємодії
18. SDK — software development kit, набір засобів розробника
19. GUI — graphical user interface, графічний інтерфейс користувача
20. UI — user interface, інтерфейс користувача
21. SQLite — реляційна вбудована система керування базами даних
22. ORM — object-relational mapping, об’єктно-реляційне відображення
23. PBR — physically based rendering, фізично коректний рендеринг
24. KPI — key performance indicators, ключові показники ефективності

25. Latency — затримка між обробкою кадру та його відтворенням

26. TrackingEngine — підсистема трекінгу AR-сцени, відповідальна за визначення пози та орієнтації об’єктів.

27. MathCore — обчислювальне ядро параметричних і аналітичних моделей

28. Renderer — модуль рендерингу тривимірних об’єктів

29. SceneView — вікно відображення AR-сцени в PyQt6

30. Telemetry — телеметричні дані системи (FPS, latency, track-loss, події)

31. Session — сесія роботи користувача в AR-додатку

32. RuleRegistry — модуль правил валідації кадрів і сцен, забезпечує контроль якості відображення AR-контенту.

# ВСТУП

Безпілотні літальні апарати (БПЛА) дедалі частіше застосовуються для моніторингу, пошуково-рятувальних операцій, аграрного нагляду, військової розвідки та логістики. Ключовим компонентом їх інтелектуальної поведінки є система розпізнавання образів, що забезпечує здатність ідентифікувати об’єкти та реагувати на зміни в навколишньому середовищі у режимі реального часу [14, 15]. В умовах високої динаміки сцени, обмежених обчислювальних ресурсів бортових систем і впливу зовнішніх чинників (освітлення, погодні умови, вібрації), точність і швидкодія таких систем набувають критичного значення [6, 7]. Це зумовлює актуальність дослідження методів комп’ютерного зору та вдосконалення алгоритмів розпізнавання образів, орієнтованих на автономні дрони [6, 8].

**Актуальність теми** полягає у зростаючій потребі в інтелектуальних системах комп’ютерного зору, здатних працювати на енергообмежених пристроях у реальних умовах експлуатації. Від ефективності алгоритмів детекції, класифікації та відстеження об’єктів залежить безпека польоту, точність навігації, а також здатність БПЛА виконувати складні завдання без участі оператора [1, 2, 3]. Подальше вдосконалення архітектури систем розпізнавання образів на базі Python-емуляторів дає змогу моделювати процеси сприйняття та прийняття рішень у дронах із використанням бібліотек TensorFlow, PyTorch, OpenCV та NumPy, що підвищує рівень відтворюваності експериментів і прискорює прототипування [14, 15].

**Мета дослідження** — підвищення точності, швидкодії та стійкості системи розпізнавання образів для дронів шляхом аналізу, моделювання та вдосконалення алгоритмів комп’ютерного зору з використанням Python-емулятора для відтворення процесів оброблення відеопотоку.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Провести системний аналіз предметної області систем комп’ютерного зору для БПЛА [6, 7].
2. Проаналізувати існуючі архітектури та алгоритми розпізнавання образів у безпілотних системах [1, 2, 5].
3. Побудувати математичну модель і Python-емулятор процесу розпізнавання [14, 15].
4. Розробити та реалізувати алгоритми оброблення, класифікації й відстеження об’єктів [6, 8].
5. Оцінити ефективність запропонованих удосконалень за показниками точності, затримки оброблення та ресурсоємності [14, 15].

**Об’єкт дослідження** — процес автоматизованого розпізнавання та класифікації образів у відеопотоці, отриманому з камер безпілотних літальних апаратів.

**Предмет дослідження** — методи, алгоритми та архітектурні рішення систем комп’ютерного зору для БПЛА, зокрема на основі Python-емуляції та глибоких згорткових нейронних мереж [14, 15].

**Методи дослідження** включають методи комп’ютерного зору, машинного навчання, статистичного аналізу, нейромережевої обробки зображень, моделювання програмних процесів у середовищі Python, а також методи структурного й функціонального аналізу систем [6, 7, 8, 14].

**Наукова новизна роботи** полягає у розробленні вдосконаленого підходу до побудови емулятора системи розпізнавання образів, який дозволяє відтворювати послідовність процесів сенсорного сприйняття, попередньої обробки, сегментації, класифікації та аналізу сцен із урахуванням обмежень апаратних ресурсів дрону [6, 7]. Запропоновані методи дозволяють підвищити точність і стабільність розпізнавання об’єктів у складних умовах експлуатації, а також забезпечують можливість інтеграції з реальними апаратно-програмними комплексами для автономного керування [1, 2, 5].

**Практичне значення роботи** полягає у створенні модульного Python-емулятора для тестування систем комп’ютерного зору, який може бути використаний як навчальний, дослідницький або інженерний інструмент для аналізу ефективності моделей розпізнавання образів у безпілотних апаратах [14, 15].

.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Опис предметної області та основних процесів функціонування системи

Система розпізнавання образів для дронів являє собою комплекс апаратних і програмних засобів, що забезпечують автономне сприйняття навколишнього середовища, аналіз зображень і прийняття рішень у реальному часі [6]. Архітектура такої системи включає сенсорні модулі, обчислювальні вузли, комунікаційні канали та програмне забезпечення для інференсу й управління польотом. На рис. 1.1 наведено узагальнену структурну схему предметної області, що відображає взаємодію між апаратними компонентами, потоками даних і програмними модулями системи [4, 6].

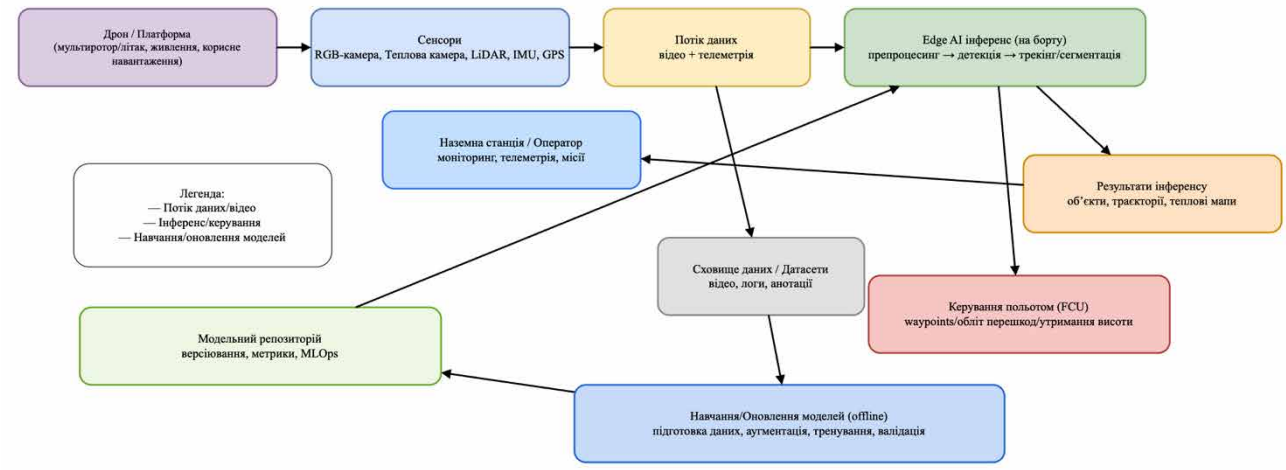


Рис. 1.1 – Структурна схема предметної області системи розпізнавання образів для дронів

У межах цієї системи дрон (платформа типу мультикоптера або літака) забезпечує енергопостачання, підключення корисного навантаження й передавання даних [6, 7]. Сенсорний комплекс складається з RGB-камери, тепловізора, LiDAR-сенсора, IMU-модуля та GPS-приймача, які генерують багатоканальний потік відео та телеметрії [6]. Отримані дані надходять до сховища або потокового процесора, де формуються датасети, журнали подій і

анотації для подальшої обробки. На борту дрона виконується edge AI інференс, що передбачає препроцесінг, детекцію, трекінг і сегментацію об’єктів у реальному часі [6, 7]. Результати інференсу – виявлені об’єкти, траєкторії, теплові карти – надсилаються на наземну станцію оператора, де здійснюється моніторинг, керування місіями та телеметричний контроль [6, 8]..

У процесі польоту система керування FCU реалізує адаптивне управління за waypoint-траєкторіями, обходом перешкод і стабілізацією висоти, що ґрунтується на зворотному зв’язку від системи розпізнавання. Дані, накопичені у сховищі, використовуються для офлайн-навчання та оновлення моделей, яке передбачає підготовку вибірок, аугментацію, тренування та валідацію. Навчені моделі версіонуються у модельному репозиторії з підтримкою MLOps-циклу, який забезпечує контроль метрик, автоматизоване розгортання та оновлення моделей у польових умовах.

Основні технічні характеристики типових компонентів системи наведено в табл. 1.1, яка узагальнює сенсорну конфігурацію, обчислювальні ресурси, канали зв’язку та особливості програмного забезпечення, що використовується для інференсу і моделювання.

Таблиця 1.1 – Основні технічні характеристики компонентів системи розпізнавання образів для дронів

| Компонент | Функціональне призначення | Типові параметри / технології |
|---|---|---|
| Дрон / платформа | Носій сенсорів і обчислювального модуля | Мультикоптер, живлення 11.1–22.2 V, 4–6 пропелерів |
| RGB-/теплова камера | Отримання відеопотоку і термоданих | 4K 30 fps, FLIR Lepton, HDR підтримка |
| LiDAR + IMU + GPS | Просторова орієнтація і вимірювання | 10–40 тис. точок/с, 9-DOF, 2 см точність GPS |
| Обчислювальний модуль (Edge AI) | Інференс моделей нейромереж на борту | NVIDIA Jetson Nano/Xavier, TensorRT, PyTorch |
| Канал зв’язку | Передавання даних і телеметрії | Wi-Fi 5 GHz, LTE/5G, MQTT з TLS 1.3 |
| Наземна станція / емулятор | Візуалізація, керування, аналітика | Python, PyQt6, OpenCV, MQTT broker Mosquitto |

| | | |
|---|---|---|
| Модельний репозиторій | Зберігання та версіонування моделей | MLflow, Weights & Biases, Git LFS |

Система розпізнавання образів для дронів функціонує як багаторівнева інтегрована архітектура, що поєднує сенсорні, обчислювальні та аналітичні компоненти. Використання Python-емулятора дозволяє досліджувати й тестувати алгоритми без необхідності постійного залучення реального апаратного комплексу, забезпечуючи відтворюваність експериментів та масштабованість досліджень.

## 1.2 Теоретико-методологічні засади та стан наукових досліджень

Системи розпізнавання образів для безпілотних літальних апаратів (БПЛА) є результатом інтеграції методів комп’ютерного зору, машинного навчання та сенсорного злиття [6]. Базовим теоретичним підґрунтям сучасних рішень є глибинне навчання - зокрема згорткові нейронні мережі (Convolutional Neural Networks, CNN), які формують багаторівневі уявлення зображень через послідовне згортання та нелінійні перетворення [14, 15]. На основі праць Krizhevsky et al. [14] та Goodfellow et al. [15] сформовано концепцію глибоких мереж для класифікації та детекції об’єктів, де оптимізація функції втрат здійснюється через зворотне поширення помилки (backpropagation).

На рис. 1.2 представлено класичну двоетапну архітектуру оброблення аерофотознімків за допомогою CNN: перший етап формує proposal boxes для потенційних об’єктів, другий - виконує уточнену класифікацію та локалізацію на вихідному зображенні. Цей підхід використовувався в роботах Ren et al. [3], що запропонували регіон-пропозиційні мережі (RPN) як ефективний спосіб селекції областей інтересу.

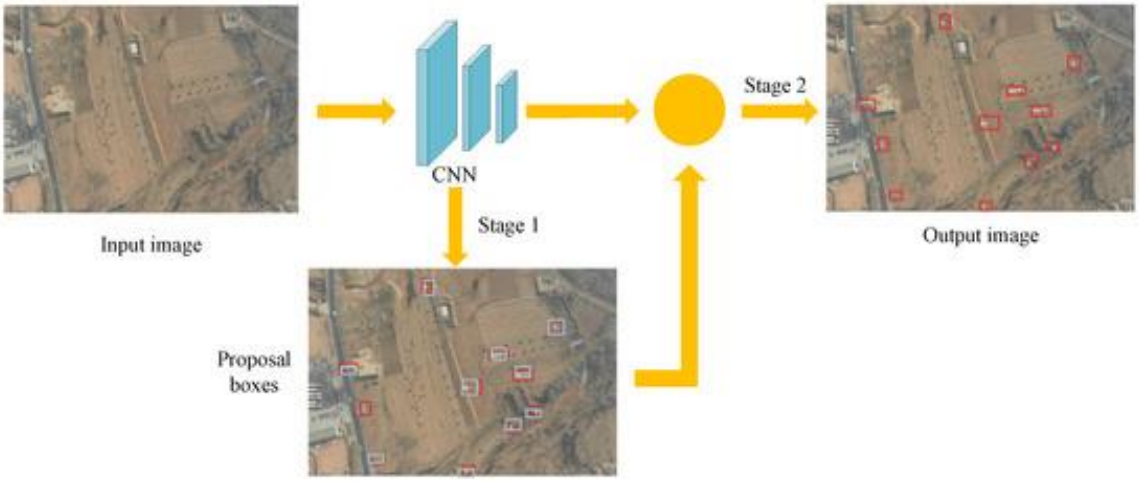


Рис. 1.2 – Двоетапна архітектура оброблення зображень з CNN для детекції об’єктів

Подальший розвиток методів спрямовано на підвищення швидкодії. Алгоритми YOLO та SSD реалізують одностадійне виявлення без етапу формування пропозицій, що забезпечує реальний час оброблення відеопотоку [6, 7]. Сучасні підходи інтегрують у CNN механізми уваги (Attention Mechanisms), які моделюють просторові залежності між пікселями та покращують чутливість мережі до малих або частково закритих об’єктів [6].

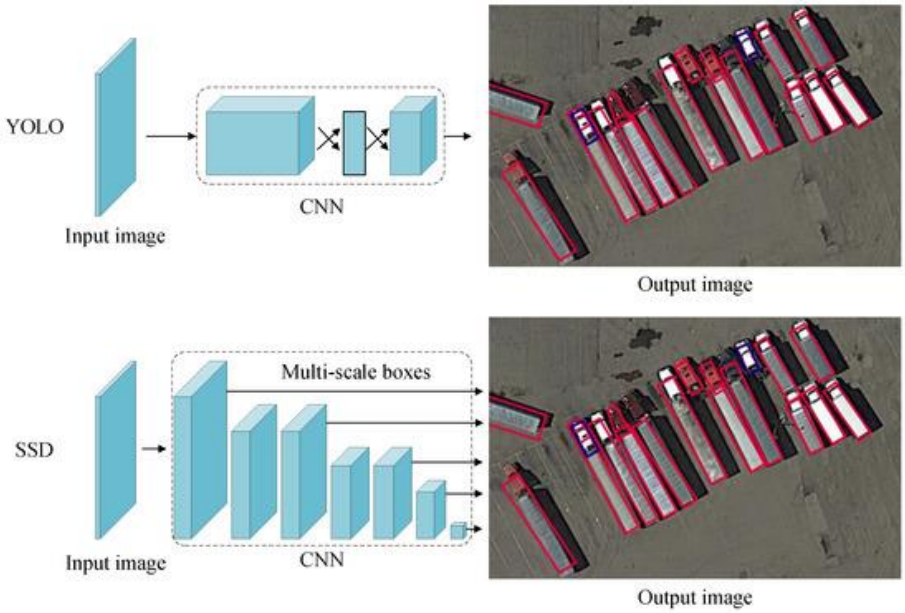


Рис. 1.3 – Архітектури YOLO та SSD для одностадійного розпізнавання об’єктів у реальному часі

Сучасні підходи інтегрують у CNN механізми уваги (Attention Mechanisms), які моделюють просторові залежності між пікселями та

покращують чутливість мережі до малих або частково закритих об’єктів. На рис. 1.4 наведено приклад двовимірної реалізації attention-блоків, що використовують матриці запитів (Q), ключів (K) та значень (V), узгоджені через операцію SoftMax та подальше згортання у канали просторових ознак [5].

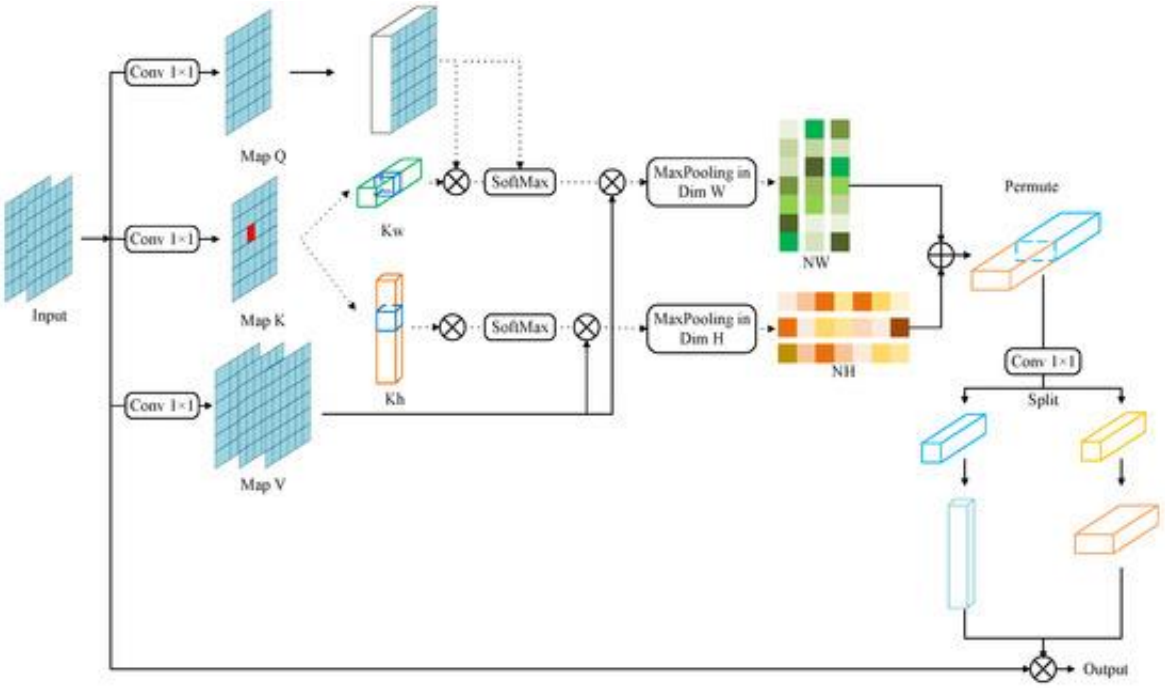


Рис. 1.4 – Схема механізму уваги у згорткових нейронних мережах для аналізу аерозображень

У наукових дослідженнях останніх років особливу увагу приділено регіонним архітектурам типу Faster R-CNN, Mask R-CNN і Hybrid Task Cascade, які дозволяють одночасно виконувати детекцію, класифікацію та сегментацію об’єктів [6]. Паралельно ведуться дослідження щодо підвищення стійкості моделей до варіацій умов зйомки, масштабів і шумів. Наприклад, багаторівневі пірамідальні структури (Feature Pyramid Networks, FPN) формують ознаки з різних масштабів [6].

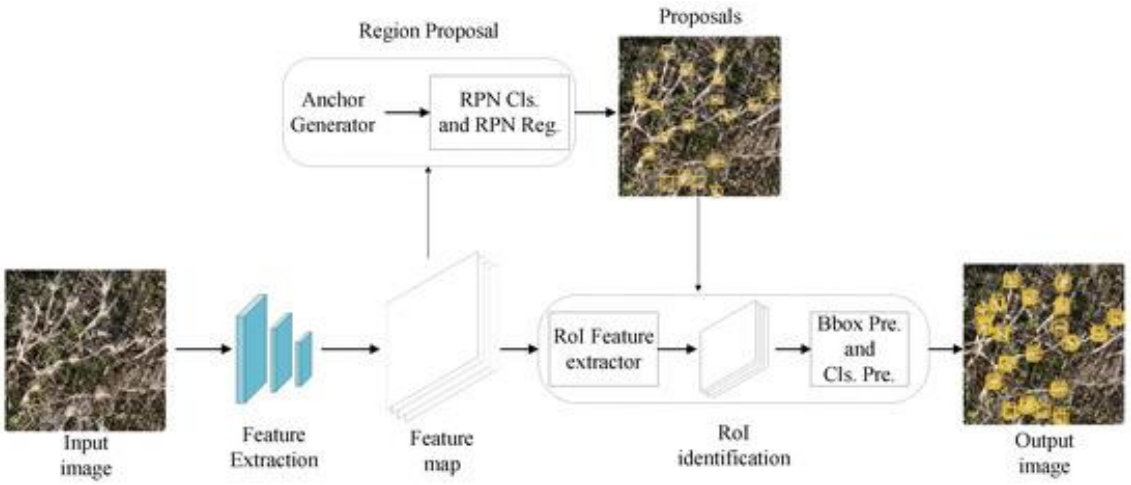


Рис. 1.5 – Архітектура Faster R-CNN із регіон-пропозиційною мережею (RPN) для багатозадачного аналізу зображень

Паралельно ведуться дослідження щодо підвищення стійкості моделей до варіацій умов зйомки, масштабів і шумів. Наприклад, у роботах Lin et al. [7] запропоновано багаторівневі пірамідальні структури (Feature Pyramid Networks, FPN), що формують ознаки з різних масштабів. Їх застосування у комбінації з адаптивними механізмами уваги дало можливість успішно вирішувати задачу сегментації та відстеження об’єктів рослинності та рельєфу на аерофотознімках (рис. 1.6).

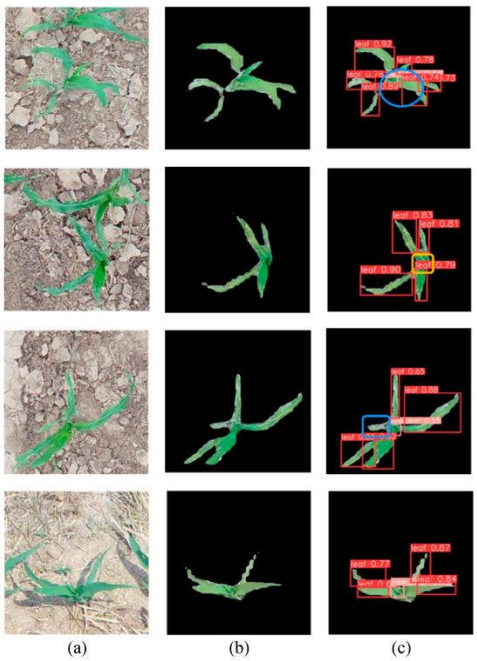


Рис. 1.6 – Приклад сегментації та розпізнавання дрібних об’єктів на зображеннях високої роздільності

Розвиток цих методів спричинив формування нової методологічної парадигми — edge-AI обчислень, тобто перенесення частини оброблення безпосередньо на борт дрона, що дозволяє мінімізувати затримку передачі даних і підвищити автономність системи [6, 7]. При цьому математичні моделі оптимізуються за критеріями latency, throughputта power efficiency, що формує підґрунтя для створення Python-емуляторів, які дозволяють відтворити поведінку таких систем на рівні алгоритмів.

Узагальнюючи проведений аналіз, можна відзначити, що більшість сучасних досліджень орієнтовані або на підвищення точності розпізнавання, або на оптимізацію швидкодії. Недостатньо вивченим залишається питання адаптації алгоритмів розпізнавання до обмежених ресурсів бортових систем із можливістю моделювання процесів у Python-емуляторі**.**

Отже, наукова новизна нашого дослідження полягає у створенні комплексного підходу до моделювання й вдосконалення системи розпізнавання образів для дронів, який поєднує архітектури глибинного навчання з механізмами уваги, оптимізацію обчислень для edge-AI середовищ та розроблення Python-емулятора для відтворення динаміки роботи системи в реальному часі. Це забезпечує можливість апробації алгоритмів без залучення фізичних платформ і сприяє скороченню вартості експериментальних досліджень.

### 1.3 Аналіз існуючих рішень розпізнавання образів БПЛА

Розвиток систем розпізнавання образів для БПЛА тісно пов’язаний із появою комерційних платформ, що поєднують технології комп’ютерного зору, тривимірного моделювання та геоінформаційного аналізу [6]. До найбільш поширених рішень належать DJI Terra, Pix4D Mapper, DroneDeploy, SenseFly eMotion та Auterion Skynode [9–12]. Кожна з цих систем має власну архітектуру, алгоритмічні підходи до оброблення зображень і ступінь інтеграції зі штучним

інтелектом, що обумовлює різницю в точності, продуктивності та масштабованості [6].

Система DJI Terra призначена для побудови 2D/3D карт, моделювання рельєфу й аналізу об’єктів на основі фотограмметрії [9]. Її основною перевагою є глибока інтеграція з польотними контролерами DJI Phantom 4 RTK і Matrice 300 RTK, що забезпечує точну геоприв’язку кадрів і стабільну реконструкцію поверхонь. На рис. 1.7 наведено приклад інтерфейсу DJI Terra із зазначенням зони зйомки та параметрів маршруту.

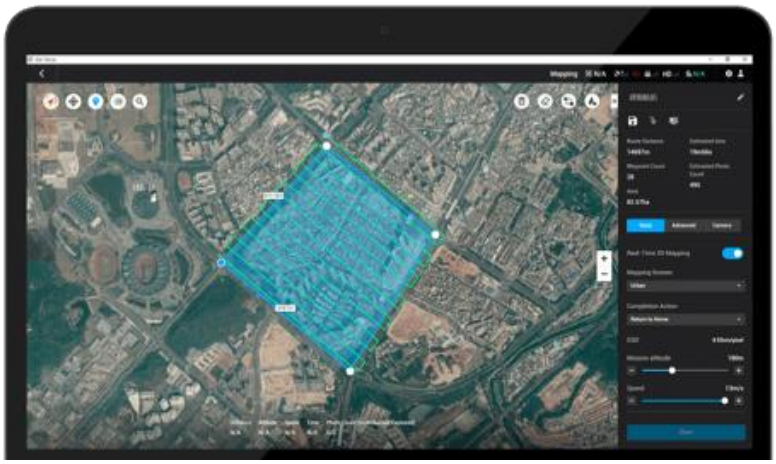

Рис. 1.7 – Інтерфейс програмного забезпечення DJI Terra для побудови картографічних місій

Програмне рішення Pix4D Mapper (рис. 1.8) [10] реалізує алгоритми SfM (Structure-from-Motion) та MVS (Multi-View Stereo) для побудови густих хмар точок і цифрових моделей рельєфу. Цей продукт орієнтований на точне 3D-моделювання інфраструктури та аграрний моніторинг, але не забезпечує повноцінного інференсу об’єктів у реальному часі, що обмежує його придатність для задач комп’ютерного зору з дронів.

Рис. 1.8 – Візуалізація фотограмметричної реконструкції в Pix4D Mapper

Сервіс DroneDeploy [11] (рис. 1.9) використовує хмарні технології для автоматизованого оброблення даних із БПЛА. Завдяки інтеграції з TensorFlow він може виконувати базові функції класифікації та теплового аналізу поверхонь. Основним акцентом DroneDeploy є зручність для користувача й швидкість оброблення даних у веб-інтерфейсі, але система має обмежену підтримку кастомних моделей розпізнавання.

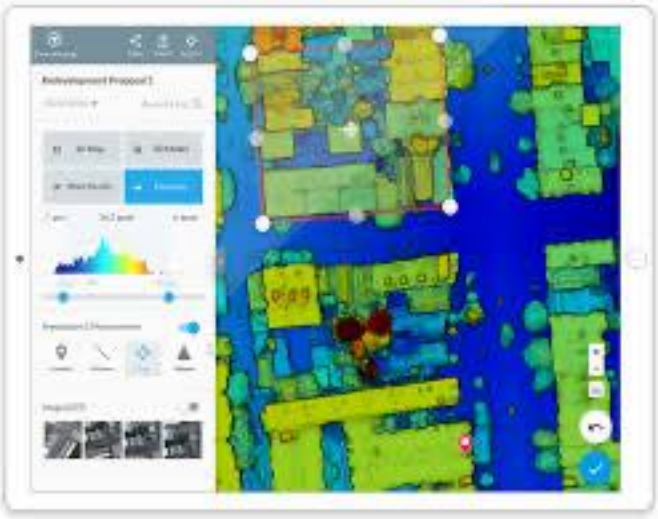

Рис. 1.9 – Приклад хмарного інтерфейсу DroneDeploy з відображенням теплових карт об’єктів

Програмно-апаратний комплекс SenseFly eMotion [12] (рис. 1.10) поєднує планування польотів, контроль висоти та аналіз покриття з можливістю інтеграції зовнішніх сенсорів. Ця система орієнтована на промислові й топографічні застосування, підтримує режим автономних місій і модулі для

уникнення зіткнень. Її перевагою є висока стабільність зв’язку та підтримка телеметричних даних у режимі реального часу.

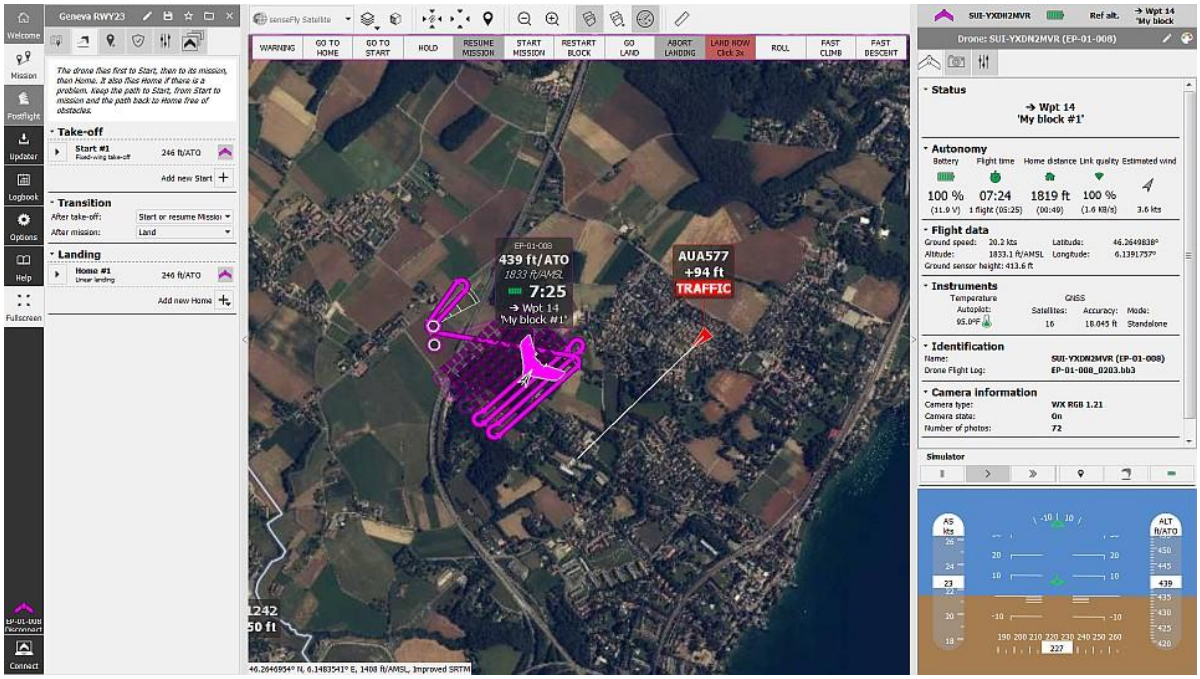


Рис. 1.10 – Інтерфейс SenseFly eMotion для управління місіями та контролю польоту

Рішення Auterion Skynode [12] (рис. 1.11) репрезентує нове покоління відкритих екосистем для дронів. Його архітектура базується на операційній системі PX4 та забезпечує модульність, підтримку камер видимого і теплового спектрів, а також обчислення на борту з використанням TensorRT [6]. Цей підхід дозволяє поєднати інференс, навігацію та збирання даних у єдиній платформі, що є найближчим аналогом до концепції, яка реалізується в нашому дослідженні.

Рис. 1.11 – Інтерфейс Auterion Skynode з відображенням потоків RGB і теплового відео під час польоту

Для порівняння розглянутих рішень складено узагальнювальну таблицю 1.2, де наведено їх ключові характеристики та відмінності від проєктованої системи.

Таблиця 1.2 – Порівняння існуючих систем розпізнавання та картографування з проєктованою системою

| Система | Основне призначення | Алгоритми | Підтримка AI-інференсу | Платформа / інтерфейс | Особливості |
|---|---|---|---|---|---|
| DJI Terra | Картографування, 3D-моделі | SfM, MVS | Обмежена (offline) | Windows | Висока точність RTK-позиціювання |
| Pix4D Mapper | Фотограмметрія, агромоніторинг | SfM, MVS | Відсутня | Desktop | Висока деталізація, але тривала обробка |
| DroneDeploy | Хмарне оброблення, аналіз поверхонь | CNN (TensorFlow) | Часткова | Web | Простий інтерфейс, але обмежена кастомізація |
| SenseFly eMotion | Автономне управління польотом | Геометричні алгоритми | Відсутня | Desktop | Надійність, промислове використання |
| Auterion Skynode | Edge AI-аналітика, навігація | CNN + TensorRT | Повна | Linux / PX4 | Інтеграція RGB + IR, MLOps |
| Розроблена система (Python-емулятор) | Розпізнавання, тестування, моделювання | CNN + Attention + OptFlow | Повна (онлайн + емуляція) | Python / PyQt6 | Емуляція польоту, адаптація до обмежених ресурсів |

Порівняння характеристик систем наведено у табл. 1.2. Проведений аналіз показав, що наявні комерційні платформи переважно зосереджені на

геодезичних або картографічних задачах, тоді як інтеграція методів глибинного навчання в режимі реального часу на борту дронів залишається обмеженою [6, 9–12]. Розроблена в межах цієї роботи система, реалізована на Python-емуляторі, усуває зазначені недоліки, поєднуючи функції інференсу, трекінгу, сегментації та емуляції польоту [6, 13].

## 1.4 Структурно-функціональна архітектура апаратного комплексу та програмного емулятора системи

Архітектура системи розпізнавання образів для дронів передбачає тісну інтеграцію апаратних компонентів польотного комплексу з програмними модулями Python-емулятора, що дозволяє досліджувати взаємодію сенсорів, контролерів, каналів зв’язку та алгоритмів штучного інтелекту в єдиному контурі керування. Такий підхід забезпечує можливість відтворення повного циклу оброблення даних - від зчитування сенсорної інформації до прийняття рішень і формування команд польоту.

На рис. 1.12 представлено схему апаратного комплексу дрона, яка відображає енергетичні, керуючі та телеметричні зв’язки між компонентами системи.

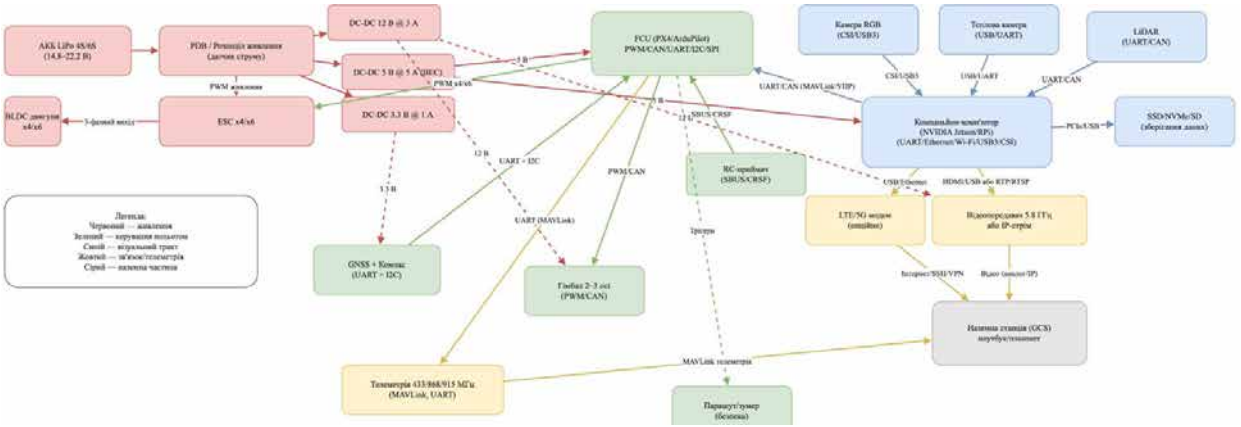


Рис. 1.12 – Схема апаратного комплексу дрона з інтегрованими сенсорами та каналами зв’язку

Основу енергоживлення складає акумуляторна батарея LiPo 4S/6S (14.8–22.2 В), підключена до плати розподілу живлення (PDB) з датчиком струму.

Через контролери ESC здійснюється живлення безколекторних двигунів (BLDC). Основним обчислювальним ядром є Flight Control Unit (FCU) на базі PX4 або ArduPilot, який отримує дані від GNSS-модуля, компаса, гімбала та сенсорів позиціонування. Зв’язок з компаньйон-комп’ютером (NVIDIA Jetson або Raspberry Pi) забезпечується через інтерфейси UART, CAN та USB із підтримкою протоколу MAVLink. Далі інформація передається на наземну станцію (GCS) через телеметричні або LTE/5G канали.

У межах програмного рівня (рис. 1.13) система структурована у вигляді емуляційного конвеєра з онлайн-та офлайн контурами, що дозволяє моделювати як реальний політ, так і процес навчання моделей.

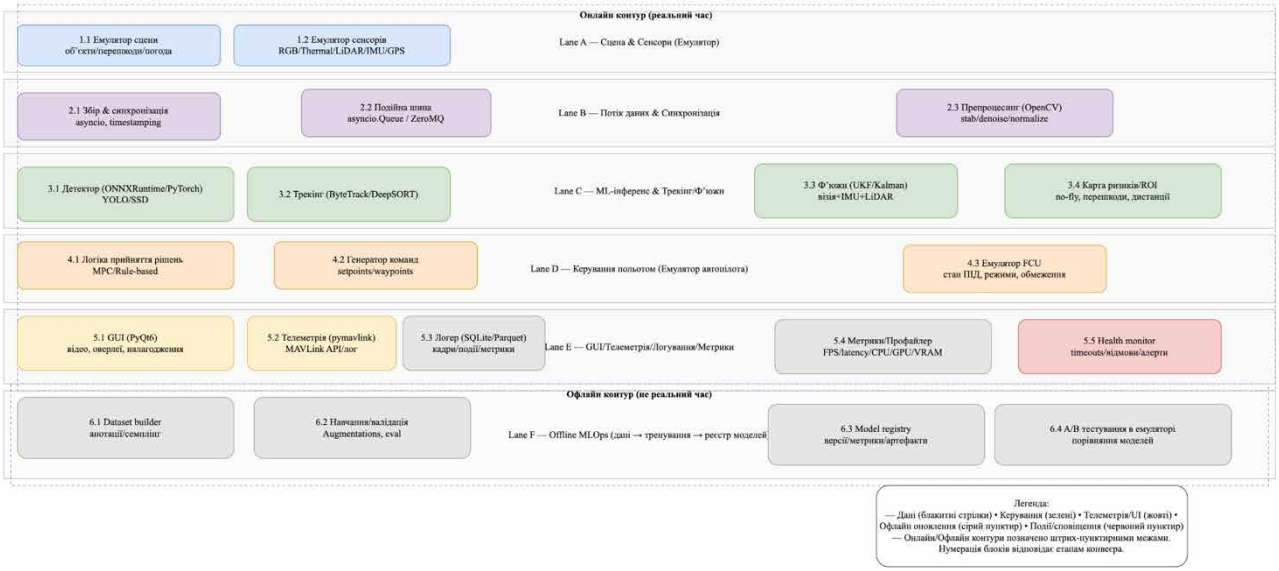


Рис. 1.13 – Функціональна схема Python-емулятора системи розпізнавання образів дрона

Онлайн-контур включає модулі: емулятор сцени та сенсорів (RGB, LiDAR, Thermal, IMU, GPS), збір і синхронізацію даних (асинхронні черги asyncio/ZeroMQ), препроцесинг (OpenCV), ML-інференс (YOLO/SSD на ONNXRuntime чи PyTorch), трекінг (ByteTrack/DeepSORT), ф’южн (IMU + LiDAR + візуальні ознаки) та керування автопілотом з ПІД-контролем і обмеженнями. Окремий шар утворюють модулі GUI (PyQt6), телеметрія (pymavlink), логування (SQL/Parquet), метрики та Health Monitor. Офлайн-контур

забезпечує MLOps-цикл (створення dataset'ів, аугментацію, тренування, валідацію, реєстрацію версій моделей і A/B-тестування).

Для узагальнення взаємозв'язків між рівнями комплексу наведено табл. 1.3, де подано основні елементи, їх функціональне призначення та тип взаємодії.

Таблиця 1.3 – Взаємозв'язок апаратних і програмних компонентів системи розпізнавання образів для дронів

| Рівень | Компонент | Призначення | Тип зв'язку / протокол | Інтеграція з емулятором |
|---|---|---|---|---|
| Апаратний | FCU (PX4/ArduPilot) | Стабілізація, навігація, управління двигунами | PWM, UART, CAN | Емуляція команд і ПІД-петлі |
| Сенсорний | RGB, Thermal, LiDAR, IMU, GPS | Збір відео та телеметрії | USB, I²C, UART | Емулятор сенсорів (кадри, шуми) |
| Обчислювальний | NVIDIA Jetson / RPi | Оброблення зображень, ML-інференс | Ethernet, Wi-Fi | Виконання моделей YOLO/SSD |
| Комунікаційний | Телеметрія 433–915 МГц, LTE/5G | Передача даних та команд | MAVLink, TCP/IP | Асинхронна черга ZeroMQ |
| Наземний | GCS (PyQt6 інтерфейс) | Моніторинг, GUI, аналіз польоту | SSH/VPN | Відтворення відео та метрик |

Узагальнюючи наведене, слід зазначити, що розроблена архітектура поєднує апаратний і програмний рівні в єдину модель, яка дає змогу досліджувати процеси розпізнавання, навігації та прийняття рішень у комплексі. Емуляційна структура дозволяє проводити експерименти з різними конфігураціями сенсорів, алгоритмами та протоколами без фізичного втручання в обладнання. Це забезпечує високу відтворюваність, гнучкість і безпечне тестування рішень, що підтверджує ефективність Python-емулятора як інструменту для дослідження та вдосконалення систем розпізнавання образів у БПЛА

## 1.5 Аналіз вимог системи

Проєктування програмного емулятора системи розпізнавання образів для дронів вимагає чіткого формулювання вимог, що визначають його функціональність, технічні параметри, а також рівень безпеки даних і взаємодії з апаратним середовищем. Емулятор має відтворювати ключові процеси оброблення відео- та телеметричних потоків, поведінку сенсорних модулів і польотного контролера (FCU), забезпечуючи при цьому інтеграцію з ML-модулями (інференсом, трекінгом, препроцесінгом) у режимі реального часу.

Функціональні вимоги визначають основні можливості системи, необхідні для відтворення життєвого циклу даних - від генерації сенсорних сигналів до візуалізації результатів інференсу. Узагальнений перелік подано у табл. 1.4.

Таблиця 1.4 – Функціональні вимоги до програмного емулятора системи розпізнавання образів

| № | Вимога | Опис | Рівень реалізації |
|---|---|---|---|
| 1 | Емуляція сенсорів (RGB, LiDAR, Thermal, IMU, GPS) | Відтворення потоків даних із параметрами частоти, шумів і затримок | Обов’язкова |
| 2 | Оброблення зображень (OpenCV, NumPy) | Препроцесінг: нормалізація, фільтрація, стабілізація відеопотоку | Обов’язкова |
| 3 | Інференс нейронних мереж (YOLO/SSD/ONNXRuntime) | Виявлення, класифікація та сегментація об’єктів у реальному часі | Обов’язкова |
| 4 | Модуль трекінгу об’єктів | Визначення траєкторій та ідентифікаторів об’єктів (ByteTrack/DeepSORT) | Обов’язкова |
| 5 | Емуляція польотного контролера (FCU) | Формування команд керування, ПІД-контроль, waypoint-навігація | Обов’язкова |

Продовження таблиці 1.4

| | | | |
|---|---|---|---|
| 6 | Інтерфейс користувача (PyQt6) | Відображення сцени, метрик, треків та повідомлень | Базова |
| 7 | Телеметрія й логування (pymavlink, SQLite) | Збереження логів польоту, FPS, затримок, станів моделей | Базова |
| 8 | Адаптивний інтерфейс сценаріїв | Вибір місій, генерація погодних умов, перешкод, шумів | Додаткова |

Окрім функціональної частини, до системи висуваються технічні вимоги, що визначають параметри ефективності, надійності та масштабованості. Ці вимоги наведено у табл. 1.5.

Таблиця 1.5 – Технічні вимоги до програмного емулятора системи розпізнавання образів

| № | Параметр | Опис / Значення | Коментар |
|---|---|---|---|
| 1 | Частота інференсу | ≥ 15 FPS (YOLOv5-Nano) при 720p відео | Забезпечує реалістичну симуляцію |
| 2 | Затримка кадру (latency) | ≤ 150 мс | Для синхронізації з телеметрією |
| 3 | Навантаження CPU | ≤ 80 % | Асинхронна обробка потоків |
| 4 | Навантаження GPU | ≤ 90 % | При використанні CUDA-акселерації |
| 5 | Використання RAM | ≤ 8 ГБ | Для середовища Jetson Xavier/RPi 4 |
| 6 | Формати збереження даних | Parquet, SQLite, JSON | Для сумісності з ML-аналітикою |
| 7 | Сумісність ОС | Linux (Ubuntu 20.04+), Windows 10+ | Кросплатформність |
| 8 | Інтеграційні протоколи | MAVLink, ZeroMQ, RTSP | Підтримка зовнішніх систем |

Окрему групу становлять вимоги до безпеки, які забезпечують захист даних, контроль доступу, цілісність обчислень та стабільність роботи у разі збоїв. Узагальнені характеристики подано у табл. 1.6.

Таблиця 1.6 – Вимоги до безпеки програмного емулятора системи

| № | Вимога | Опис | Механізм реалізації |
|---|---|---|---|
| 1 | Захист каналів передачі | Використання TLS 1.3 для MQTT/HTTP з’єднань | Сертифікати SSL, бібліотека ssl |
| 2 | Контроль доступу | Розмежування ролей (адміністратор, оператор, аналітик) | RBAC-механізм у GUI |
| 3 | Цілісність даних | Контроль контрольних сум при логуванні й синхронізації | SHA-256-хешування |
| 4 | Відновлення після збоїв | Автоматичне збереження стану системи | Чекпоїнти у файлах сесії |
| 5 | Моніторинг безпеки | Виявлення тайм-аутів, перевантажень, невідповідностей | Health Monitor / Watchdog |

Розроблений набір вимог забезпечує комплексне відтворення процесів функціонування системи розпізнавання образів дрона в межах програмного емулятора. Його модульна структура дозволяє гнучко змінювати сценарії польоту, навантаження, архітектури моделей та параметри телеметрії. Формалізація вимог гарантує відтворюваність експериментів, масштабованість досліджень і безпечну взаємодію між підсистемами. У результаті створюється повноцінне середовище для тестування, порівняння та вдосконалення алгоритмів комп’ютерного зору для безпілотних літальних апаратів

## 1.6 Постановка завдання

На основі проведеного системного аналізу, дослідження сучасних наукових підходів та порівняння існуючих апаратно-програмних рішень сформульовано завдання створення програмного емулятора системи розпізнавання образів для дронів, що забезпечить моделювання процесів сприйняття, інференсу та прийняття рішень у реальному часі. Основною метою є розроблення імітаційного середовища, яке дозволить відтворювати поведінку сенсорних підсистем, польотного контролера, каналів телеметрії та модулів машинного навчання без необхідності залучення реального апаратного комплексу.

У процесі дослідження необхідно реалізувати взаємодію між елементами емулятора, які відповідають реальним підсистемам дрона:

– сенсорна підсистема, що генерує дані RGB-камери, тепловізора, LiDAR, IMU та GPS з параметрами затримок, шумів і часової синхронізації;

– модуль препроцесингу, який здійснює стабілізацію, нормалізацію та фільтрацію відеопотоку засобами OpenCV;

– інференсний блок, побудований на основі глибинних нейронних мереж (YOLO/SSD,ONNXRuntime або PyTorch), який виконує детекцію та класифікацію об’єктів у режимі реального часу;

– модуль трекінгу, що використовує алгоритми ByteTrack або DeepSORT для ідентифікації траєкторій об’єктів;

– емулятор польотного контролера (FCU), який реалізує PID-регуляцію, обмеження по висоті, швидкості та формування команд waypoint-навігації;

– інтерфейс користувача (GUI), побудований на PyQt6, який візуалізує сцену, об’єкти, траєкторії, теплові карти та ключові телеметричні параметри;

– модуль телеметрії і логування, що забезпечує запис параметрів польоту, подій, метрик FPS, затримок, навантаження CPU/GPU та стану моделей.

Вхідними даними для системи є відеопотоки (RGB/IR), сигнали сенсорів положення (IMU, GPS), а також параметри середовища (вітер, освітлення, перешкоди). Вихідними даними є координати виявлених об’єктів, типи класифікації, траєкторії руху, згенеровані команди управління польотом і набори метрик продуктивності.

Постановка завдання передбачає створення уніфікованої експериментальної платформи, що дозволяє перевіряти алгоритми розпізнавання образів для БПЛА в умовах, наближених до реальних, без ризику для апаратної частини. Реалізація цього підходу забезпечує відтворюваність результатів, оптимізацію моделей глибинного навчання для обмежених ресурсів і створення бази для подальшої інтеграції системи з реальними польотними контролерами та ML-модулями.

## 1.7 Висновки до розділу 1

У першому розділі виконано системний аналіз предметної області, що охоплює теоретичні, методологічні та технічні аспекти побудови систем розпізнавання образів для безпілотних літальних апаратів (БПЛА). Встановлено, що сучасні підходи ґрунтуються на застосуванні глибинних згорткових нейронних мереж (CNN) і механізмів уваги (Attention), які забезпечують високу точність детекції та класифікації об’єктів навіть у складних природних умовах. Проведений аналіз показав, що більшість наявних комерційних систем (DJI Terra, Pix4D Mapper, DroneDeploy, SenseFly eMotion, Auterion Skynode) орієнтовані переважно на картографування, фотограмметрію або автономне управління польотом, тоді як інтегроване рішення для моделювання процесів розпізнавання образів із підтримкою інференсу в реальному часі практично відсутнє.

Визначено ключові апаратні та програмні складові майбутньої системи, сформовано принципову схему апаратного комплексу дрона (живлення, сенсори, контролери, канали зв’язку) і розроблено функціональну архітектуру програмного Python-емулятора. Запропонована структура передбачає наявність онлайн контуру (емуляція сцени, сенсорів, ML-інференс, керування польотом, GUI) та офлайн контуру (MLOps, тренування, аугментації, тестування), що забезпечує повну імітацію життєвого циклу даних і взаємодію модулів у реальному часі.

У межах розділу сформульовано вимоги до програмного забезпечення емулятора - функціональні, технічні та безпекові, які регламентують його архітектуру, швидкодію, стабільність і захист даних. Встановлено, що для забезпечення достовірності симуляції необхідно підтримувати частоту оброблення не нижче 15 кадрів/с при затримці до 150 мс, асинхронну взаємодію модулів і багатопоточну обробку потоків телеметрії та відео.

На підставі проведеного аналізу сформульовано постановку завдання, яка передбачає створення Python-емулятора системи розпізнавання образів для дронів із можливістю відтворення роботи сенсорів, інференсу, трекінгу та логіки польотного контролера, а також оцінювання ефективності алгоритмів глибинного навчання в умовах обмежених ресурсів.

Отже, результати першого розділу стали методологічним і науковим підґрунтям для подальшої розробки програмно-апаратної частини системи, моделювання її компонентів і реалізації алгоритмів оброблення зображень та прийняття рішень у наступних розділах роботи.

# 1 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЕМУЛЯТОРА СИСТЕМИ РОЗПІЗНАВАННЯ ОБРАЗІВ ДЛЯ ДРОНІВ

## 2.1 Принципова схема програмного емулятора та структура каналів оброблення даних і подій

Програмний емулятор системи розпізнавання образів для дронів відтворює роботу апаратних компонентів польотного комплексу та забезпечує наскрізну обробку подій — від моменту генерації сенсорних сигналів до передачі результатів інференсу та телеметрії на наземну станцію. Його принципова схема побудована за модульним принципом із розмежуванням контурів живлення, керування, зв’язку та інтелектуальної обробки даних. На рис. 2.1 подано узагальнену принципову схему апаратно-програмної взаємодії елементів системи, яка одночасно служить логічною моделлю для емуляційного середовища.

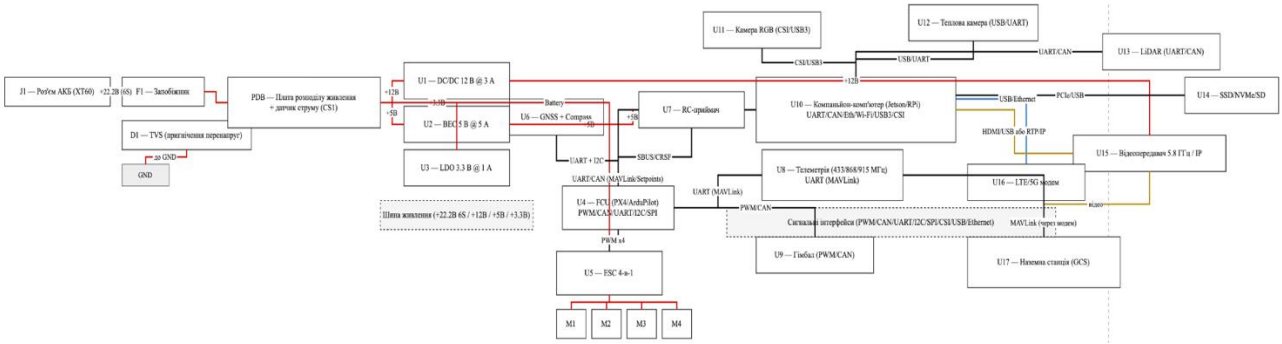


Рис. 2.1 – Принципова схема програмного емулятора системи розпізнавання образів для дронів

У схемі зображено базову структуру каналів живлення (червоні лінії), комунікацій (жовті), телеметрії (зелені) та відеообміну (сині). Джерелом живлення є LiPo-акумулятор (14,8–22,2 В), підключений через запобіжник і діод захисту від перенапруги до плати розподілу живлення (PDB). Через DC/DC-перетворювачі формуються стабілізовані лінії +12 В, +5 В і +3,3 В для сенсорних модулів (RGB-, теплової камери, LiDAR, GNSS-компаса) та польотного контролера FCU (PX4/ArduPilot). До контролера під’єднано приймач RC-

сигналів, гімбал, блок телеметрії, а також компаньйон-комп'ютер (NVIDIA Jetson або Raspberry Pi), який виконує функції центрального вузла інтелектуальної обробки даних і координації ML-інференсу. Відеопотоки та телеметрія передаються до наземної станції (GCS) через 5,8 ГГц або LTE/5G-модем, де здійснюється моніторинг і керування місією.

Основна логіка обміну подіями визначається чотирма взаємопов'язаними каналами:

– сенсорний канал – передача відео та телеметрії від RGB-, IR-, LiDAR- і IMU-модулів у форматах CSI/USB/UART із часовими мітками;

– інференсний канал – оброблення даних на компаньйон-комп'ютері (ONNXRuntime, PyTorch) з подальшою передачею результатів FCU;

– керуючий канал – двосторонній обмін командами між FCU, гімбалом і емулятором автопілота через PWM/CAN/MAVLink;

– телеметричний канал – синхронізація станів, логування подій, передача даних на GCS-інтерфейс через асинхронні протоколи ZeroMQ та MAVLink.

Узагальнені характеристики основних каналів і типів взаємодії наведено в табл. 2.1.

Таблиця 2.1 – Структура каналів оброблення даних і подій у програмному емуляторі системи розпізнавання образів

| № | Канал / підсистема | Формат переданих даних | Протоколи зв'язку | Частота оновлення | Призначення |
|---|---|---|---|---|---|
| 1 | Сенсорний (RGB / Thermal / LiDAR / IMU) | Відео-кадри, телеметрія, глибина, координати | USB3, UART, I²C, CAN | 30–60 Гц | Формування потоку вхідних даних для інференсу |
| 2 | Інференсний (ML-модуль) | Матриці ознак, координати об'єктів, метрики точності | Ethernet, ZeroMQ | 15–30 Гц | Обчислення та видача результатів розпізнавання |

Продовження таблиці 2.1

| | | | | | |
|---|---|---|---|---|---|
| 3 | Керуючий (FCU / емулятор автопілота) | Команди PWM, waypoints, режими PID | MAVLink, PWM, CAN | 50–100 Гц | Моделювання польоту та стабілізації дрона |
| 4 | Телеметричний | Дані логів, FPS, CPU/GPU, GPS | MAVLink, UDP/TCP | 5–10 Гц | Моніторинг стану та логування подій |
| 5 | Відео-канал | Потік RTP/RTSP або IP-відео | HDMI, Wi-Fi 5 ГГц | 25–60 Гц | Передача зображення на GCS |

Принципова схема забезпечує модульність, паралельність і масштабованість роботи системи. Вона дозволяє симулювати обмін даними між сенсорами, ML-ядром і системою керування польотом із високою часовою точністю, що робить можливим відтворення повного життєвого циклу оброблення подій без залучення фізичного обладнання. Реалізація цієї архітектури є ключовою умовою створення експериментального середовища для дослідження алгоритмів розпізнавання образів і тестування їх ефективності в умовах, наближених до реальних.

## 2.2 Структурна схема апаратно-програмного комплексу HIL-тестування

Електрична та монтажна структура дослідного стенду програмного емулятора системи розпізнавання образів для дронів спроєктована з урахуванням вимог до стабільності живлення, мінімізації електромагнітних перешкод і забезпечення надійності передачі даних у процесі тестування інтелектуальних модулів. Загальна побудова системи має багаторівневу архітектуру, де апаратна частина відтворює базову платформу квадрокоптера, а програмна – формує інтерфейси збору, синхронізації та оброблення подій у реальному часі.

Джерелом живлення є дві літій-полімерні батареї Hilldow Sport Power ємністю 6100 мА·год, 7.4 В, 65C, з’єднані послідовно для отримання напруги 14.8 В (рис. 2.2). Вони забезпечують стабільне живлення контролера

польоту, компаньйон-комп’ютера та сенсорних модулів. Через плату розподілу живлення (PDB) напруга нормалізується за допомогою DC/DC-перетворювачів (12 В, 5 В, 3.3 В), що дозволяє ізолювати високочастотні навантаження (модулі інференсу, камери) від живлення стабілізованої частини FCU.


Рис. 2.2 – Літій-полімерні батареї живлення стенду Hilldow Sport Power 6100 mAh (2×7.4 В)

Основним елементом керування виступає польотний контролер Pixhawk PRO, який відповідає за оброблення даних сенсорів, генерацію PWM-сигналів і обмін телеметрією за протоколом MAVLink (рис. 2.3).


Рис. 2.3 – Контролер польоту Pixhawk PRO (інтерфейси I/O, CAN, UART, SPI)

Його інтеграція з компаньйон-комп’ютером NVIDIA Jetson Nano (рис. 2.4) здійснюється через інтерфейси UART, I²C та CAN, що дає змогу організувати обмін подіями між рівнем керування й рівнем оброблення зображень.

Рис. 2.4 – Компаньйон-комп’ютер NVIDIA Jetson Nano для інференсу моделей розпізнавання образів

Сенсорна підсистема включає RGB-камеру представлену на рисунку 2.5.


Рис. 2.5 – RGB-камера для збору відеопотоку високої роздільності

І модуль LiDAR (рис. 2.6), підключені через інтерфейси USB3 / CSI та UART / CAN відповідно. Ці компоненти забезпечують збирання багатоканальних даних, які проходять препроцесінг на рівні компаньйон-

комп’ютера і передаються в ML-модуль для виявлення, класифікації й трекінгу об’єктів.

Рис. 2.6 – Сенсор відстані LiDAR (модуль для вимірювання глибини та виявлення перешкод)

На рис. 2.7 наведено схему електричного з’єднання основних елементів, що демонструє маршрути живлення, сигналів керування та передавання даних. Використання кольорового кодування (червоний - живлення, зелений - керування, синій - дані, жовтий - RF/відео, чорний - GND) дозволяє візуально нормалізувати схему для швидкої діагностики та відлагодження під час випробувань.

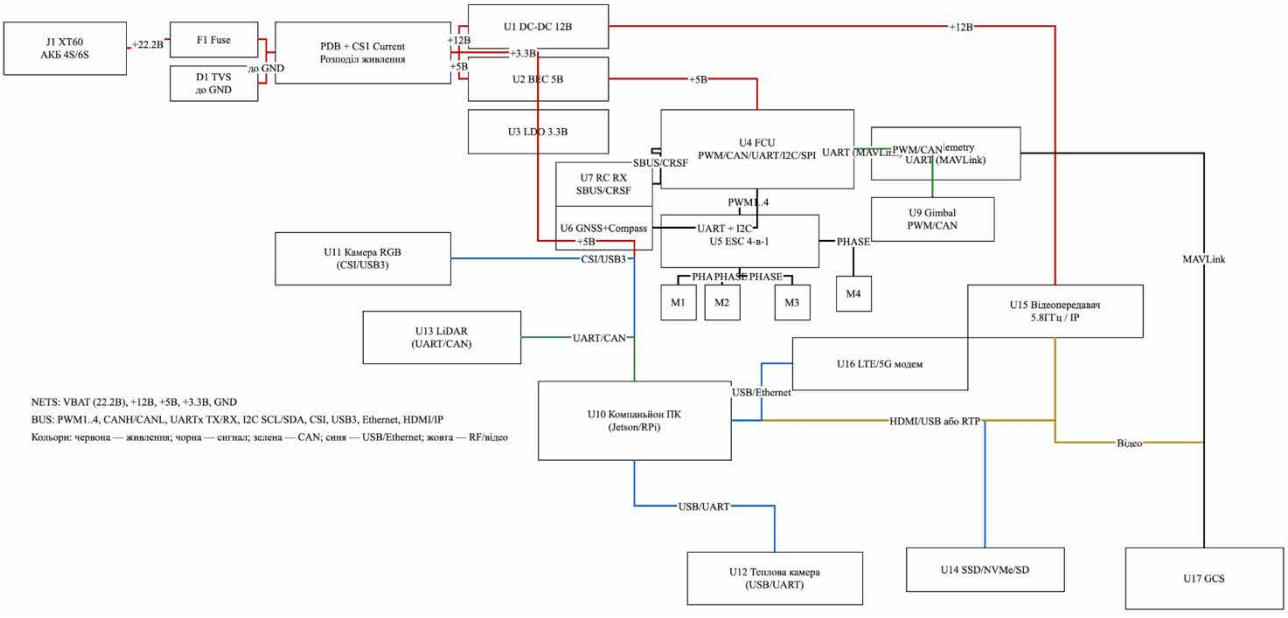


Рис. 2.7 – Структурна схема з’єднання компонентів дослідного стенду емулятора

Монтажна структура системи подана на рис. 2.8, де показано розташування модулів у межах рами квадрокоптера. Елементи згруповано відповідно до їхніх функціональних зон: енергетична (АКБ, PDB, ESC), сенсорна (LiDAR, RGB, теплова камера), обчислювальна (FCU, Jetson), комунікаційна (LTE/5G, відеопередавач, телеметрія). Для зменшення інтерференції та нагріву враховано просторове рознесення високочастотних модулів (VTx, LTE) на відстань не менше 150 мм від FCU і GNSS-антени.

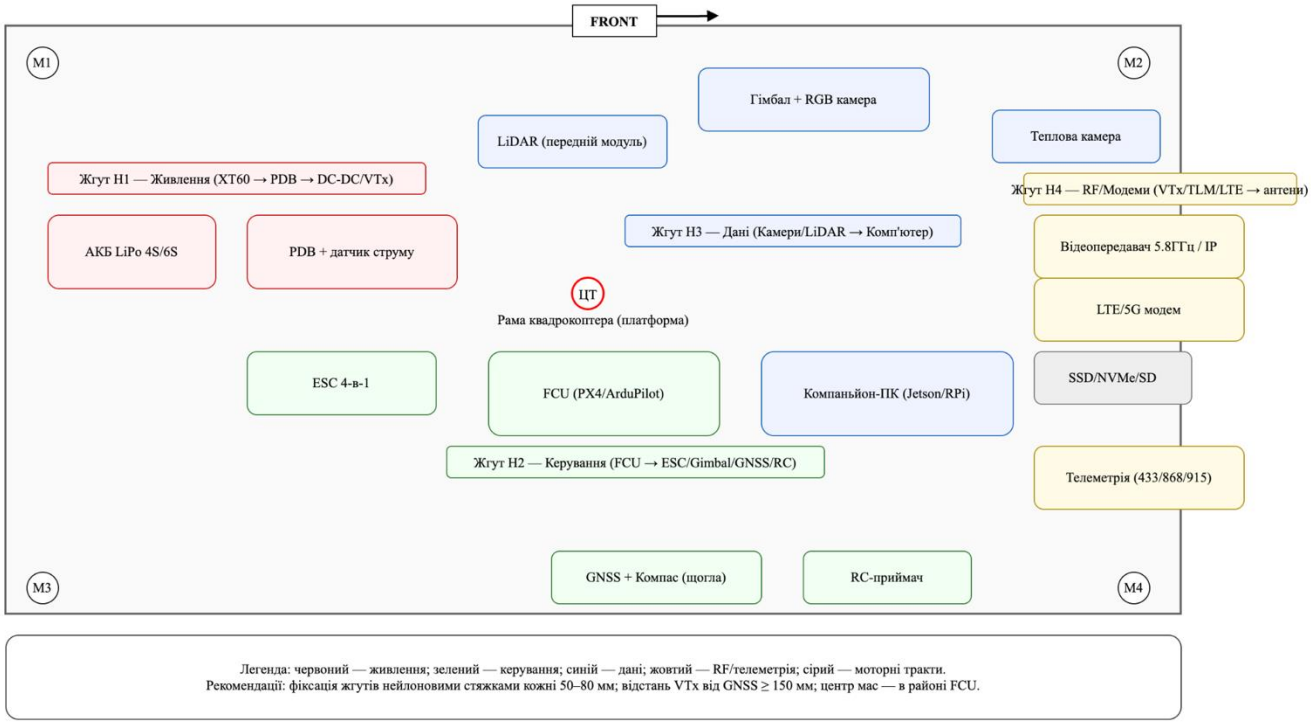


Рис. 2.8 – Структурна схема розміщення компонентів стенду програмного емулятора

Узагальнені технічні характеристики компонентів стенду наведено в табл. 2.2.

Таблиця 2.2 – Основні компоненти дослідного стенду програмного емулятора системи розпізнавання образів

| № | Компонент | Призначення | Напруга живлення | Інтерфейси зв’язку |
|---|---|---|---|---|
| 1 | LiPo акумулятор 6100 mAh (2×7.4 В) | Основне джерело живлення системи | 14.8 В | XT60 / Balancer / DC |
| 2 | Контролер Pixhawk PRO | Обчислення траєкторій, стабілізація, телеметрія | 5 В | PWM, UART, I²C, CAN |

Продовження таблиці 2.2

| | | | | |
|---|---|---|---|---|
| 3 | Компаньйон-ПК Jetson Nano | Оброблення зображень, інференс нейронних мереж | 5 В / 12 В | USB3, Ethernet, CSI |
| 4 | Камера RGB (USB3 / CSI) | Захоплення відео для аналізу | 5 В | USB3 / CSI |
| 5 | Модуль LiDAR (UART / CAN) | Вимірювання глибини, відстані до об’єктів | 5 В | UART, CAN |

Архітектура електричної та монтажної схеми побудована за принципами енергетичної ізоляції, структурної симетрії та нормалізації потоків даних. Це забезпечує стабільну роботу системи під час багатопоточних обчислень на компаньйон-ПК, мінімізує ризики перевантаження та дозволяє точно відтворити часові затримки між сенсорними подіями. Застосування таких підходів у проєктуванні стенду створює необхідні передумови для відтворюваного дослідження алгоритмів машинного зору, їх адаптації до різних режимів польоту та подальшої інтеграції з реальним апаратним комплексом.

## 2.3 Передумови створення програмного емулятора серверної логіки та вибір технологій

Розроблення програмного емулятора системи розпізнавання образів для дронів ґрунтується на необхідності забезпечення контрольованого, відтворюваного та масштабованого середовища для тестування алгоритмів комп’ютерного зору без ризику для апаратних компонентів. У реальних польових умовах перевірка моделей машинного навчання супроводжується низкою труднощів — нестабільним живленням, затримками телеметрії, впливом зовнішніх факторів (вітер, освітлення, завади). Тому виникла потреба створення емуляційного комплексу, який імітує повну взаємодію між сенсорними модулями, обчислювальними блоками та каналами керування, дозволяючи аналізувати вплив архітектур нейронних мереж, алгоритмів синхронізації та параметрів комунікацій у реальному часі.

Емулятор розробляється як програмно-апаратна модель “цифрового двійника” системи дрона, що відтворює основні сценарії польоту, виявлення об’єктів, стабілізації та прийняття рішень. Його структура поділяється на дві частини:

– онлайн-контур, що забезпечує моделювання сцени, потоків даних, препроцесингу, ML-інференсу та управління польотом у реальному часі;

– офлайн-контур, що реалізує навчання, валідацію та версіонування моделей у рамках підходів MLOps.

Для реалізації серверної логіки оброблення подій та взаємодії між модулями обрано стек технологій, який забезпечує асинхронність, низьку затримку та масштабованість. Зокрема, базовою мовою реалізації обрано Python, оскільки вона має широкий набір бібліотек для штучного інтелекту, візуалізації та роботи з потоками подій (asyncio, ZeroMQ, PyQt6, OpenCV, ONNXRuntime).

Технологічні передумови та архітектурні рішення подано у табл. 2.3.

Таблиця 2.3 – Вибір технологій для реалізації серверної логіки програмного емулятора

| № | Компонент | Технологія / бібліотека | Обґрунтування вибору |
|---|---|---|---|
| 1 | Програмна мова | Python 3.11+ | Оптимальна для швидкої розробки, підтримує асинхронну логіку та ML-фреймворки |
| 2 | Асинхронна обробка подій | asyncio / ZeroMQ | Забезпечує неблокуючий обмін між потоками сенсорів і модулями інференсу |
| 3 | Оброблення зображень | OpenCV / NumPy / Pillow | Реалізація препроцесингу, стабілізації, нормалізації відеопотоку |
| 4 | Модулі машинного навчання | ONNXRuntime / PyTorch / TensorRT | Підтримка інференсу нейронних мереж (YOLO, SSD, EfficientDet) на GPU |
| 5 | Комунікаційні протоколи | MAVLink / UDP / TCP / ZeroMQ | Стандартизована взаємодія з FCU, телеметрією, GUI та емулятором автопілота |
| 6 | Графічний інтерфейс | PyQt6 / Matplotlib / QOpenGL | Візуалізація сцени, траєкторій і результатів розпізнавання |

Продовження таблиці 2.3

| | | | |
|---|---|---|---|
| 7 | Сховище даних | SQLite / Parquet / JSON | Легка локальна БД для збереження метрик, логів, анотацій |
| 8 | MLOps та управління моделями | MLflow / DVC / Weights&Biases | Відстеження версій, експериментів, параметрів та результатів |

Потоки даних між сенсорними емуляторами, ML-модулями та контролером керування передаються через асинхронні черги (asyncio.Queue) і ZeroMQ-сокети, що дозволяє імітувати

Додатковою передумовою вибору Python як базового середовища є його сумісність із системами вбудованого типу (Jetson Nano, Raspberry Pi), що дозволяє запускати інференс нейронних мереж на борту дрона або в хмарному середовищі без зміни коду. Для забезпечення високої пропускної здатності між модулями ML і GUI застосовується протокол ZeroMQ, який поєднує простоту інтеграції з низькою затримкою обміну.

Розроблена архітектура підтримує модульність і можливість подальшої інтеграції компонентів реального часу, що дозволяє поступово переходити від емулятора до натурних експериментів. Впровадження асинхронної подієвої логіки гарантує масштабованість системи, відтворюваність результатів тестування та контрольованість кожного етапу оброблення даних. Таким чином, створення програмного емулятора на базі Python і ZeroMQ забезпечує оптимальний баланс між гнучкістю, точністю симуляції та ресурсною ефективністю, що є необхідною умовою для дослідження інтелектуальних алгоритмів комп’ютерного зору в безпілотних системах.

## 2.4 Формалізація специфікації повідомлень і тем MQTT

Одним із ключових елементів програмного емулятора системи розпізнавання образів для дронів є механізм обміну повідомленнями між

компонентами через протокол MQTT (Message Queuing Telemetry Transport). Він забезпечує подієву взаємодію між сенсорними емуляторами, інференс-модулями, емулятором польотного контролера (FCU), наземною станцією (GCS) та аналітичним сховищем. Завдяки легковісній структурі, QoS-рівням і підтримці TLS-шифрування MQTT є оптимальним рішенням для реалізації асинхронного, масштабованого та безпечного обміну даними в реальному часі.

На рис. 2.9 подано формалізовану схему MQTT-топології та специфікацію тем і форматів повідомлень, що використовуються у системі.

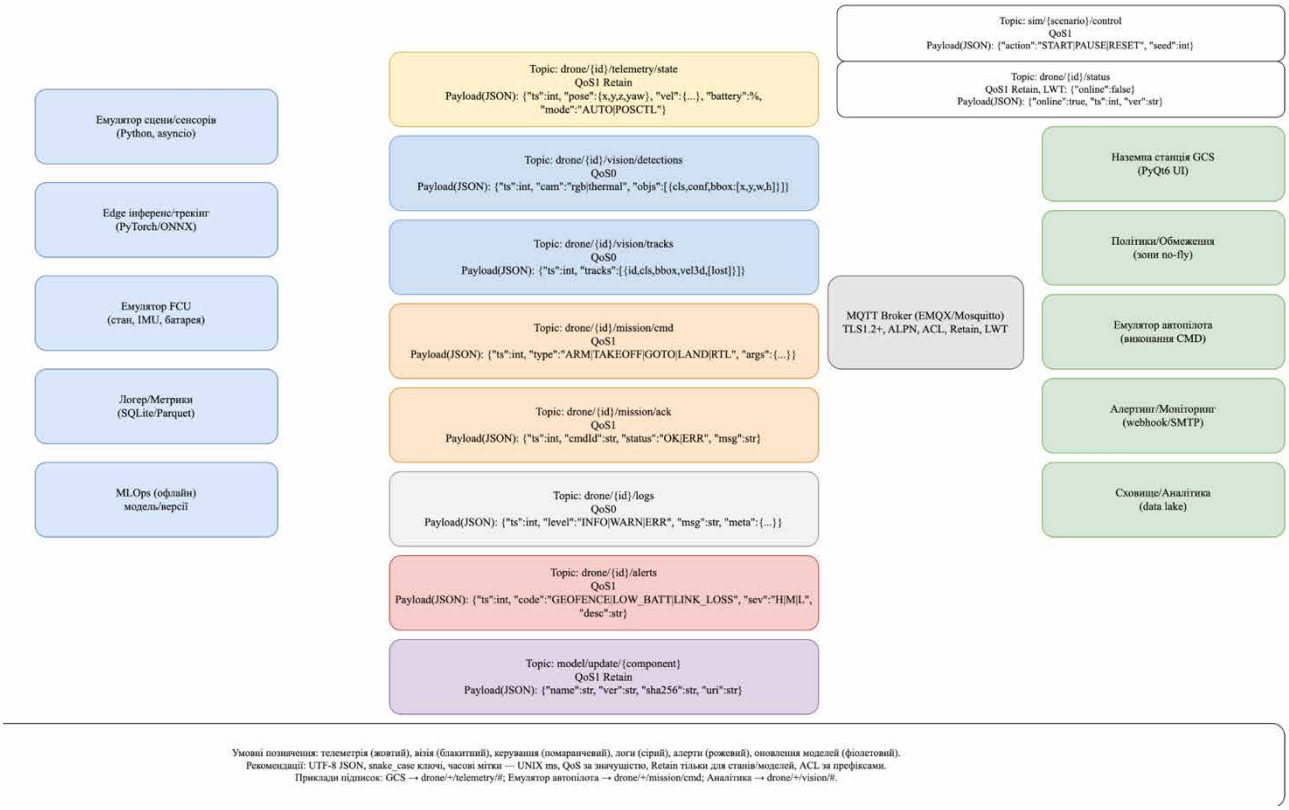


Рис. 2.9 – Специфікація тем і форматів повідомлень MQTT у програмному емуляторі системи розпізнавання образів

Архітектура брокера побудована на базі EMQX/Mosquitto, що підтримує розширення TLS 1.2+, ALPN, ACL (Access Control List) і LWT (Last Will & Testament). Така конфігурація гарантує захист каналів передачі даних, контроль доступу та можливість автоматичного повідомлення про аварійні відключення вузлів. Теми організовано за ієрархічною структурою із використанням

змінних {id} та {component} для забезпечення масштабованості у багатодронових сценаріях.

Основні групи топіків та їхні формати подано у табл. 2.4.

Таблиця 2.4 – Формалізована структура тем MQTT та типи повідомлень системи

| № | Topic / Призначення | QoS | Формат повідомлення (Payload JSON) | Опис |
|---|---|---|---|---|
| 1 | drone/{id}/telemetry/state | QoS 1, Retain | {"ts":int,"pose":{x,y,z,yaw},"vel":{x,y,z},"battery":%,"mode":"AUTOPILOT"} | Потік телеметрії: координати, швидкість, стан батареї, режим польоту |
| 2 | drone/{id}/vision/detections | QoS 0 | `{"ts":int,"cam":"rgb | thermal","objs":[{cls,conf,bbox:[x,y,w,h]}]}` |
| 3 | drone/{id}/vision/tracks | QoS 0 | {"ts":int,"tracks":[{id,cls,bbox,vel3d,lost}]} | Дані трекінгу з обчисленими траєкторіями руху |
| 4 | drone/{id}/mission/cmd | QoS 1 | `{"ts":int,"type":"ARM | TAKEOFF |
| 5 | drone/{id}/mission/ack | QoS 1 | `{"ts":int,"cmdId":str,"status":"OK | ERR","msg":str}` |
| 6 | drone/{id}/logs | QoS 0 | `{"ts":int,"level":"INFO | WARN |
| 7 | drone/{id}/alerts | QoS 1 | `{"ts":int,"code":"GEOFENCE | LOW_BATT |
| 8 | model/update/{component} | QoS 1, Retain | {"name":str,"ver":str,"sha256":str,"uri":str} | Повідомлення про оновлення або версіонування моделей ML |
| 9 | sim/{scenario}/control | QoS 1 | `{"action":"START | PAUSE |
| 10 | drone/{id}/status | QoS 1, Retain, LWT | {"online":true,"ts":int,"ver":str} | Індикація доступності компонентів і перевірка їхньої актуальності |

Формалізація повідомлень базується на стандартах UTF-8 JSON, використанні часових міток UNIX та єдиному форматі ключів *snake_case*. Для

критичних каналів (телеметрія, оновлення моделей, алерти) застосовується QoS 1для гарантії доставки, тоді як для потокових даних інференсу (зображення, треки) - QoS 0 для мінімізації затримок.

Такий підхід дозволяє досягнути структурної уніфікації обміну даними в межах усього циклу розпізнавання - від сенсорної генерації кадрів до логування й аналітики. MQTT виступає не лише транспортним шаром, а й механізмом подієвого керування, де кожен модуль є публішером і сабскрайбером певних тем. Це спрощує масштабування системи під декілька безпілотників і підключення зовнішніх систем (data lake, алертинг, MLOps).

Таким чином, формалізована специфікація MQTT-тем забезпечує:

– мінімізацію затримок під час обміну телеметрією та результатами інференсу;

– відтворюваність і трасування подій на рівні JSON-пакетів;

– підтримку безпечних каналів зв’язку (TLS 1.2+, ACL-автентифікація);

– гнучку інтеграцію модулів Python-емулятора з реальними компонентами системи керування дроном.

Реалізація цієї логічної моделі є основою для побудови повноцінного асинхронного середовища взаємодії, що імітує реальний життєвий цикл роботи інтелектуальної системи розпізнавання образів на дроні.

## 2.5 Висновки до другого розділу

У другому розділі здійснено комплексне проєктування програмного емулятора системи розпізнавання образів для дронів, що поєднує апаратно-програмні та інтелектуальні компоненти в єдину відтворювану архітектуру. На основі аналізу принципових, електричних та монтажних схем сформовано оптимальну структуру взаємодії між джерелами живлення, сенсорними модулями, контролерами та компаньйон-комп’ютером, яка забезпечує енергетичну ізоляцію, стабільність каналів зв’язку й мінімізацію

електромагнітних завад. Застосування схемотехнічної нормалізації дозволило зменшити втрати напруги, покращити відведення тепла та забезпечити коректну роботу FCU і модулів інференсу при паралельному навантаженні.

Розроблена концепція серверної логіки емулятора ґрунтується на асинхронній архітектурі, реалізованій засобами Python, ZeroMQ, asyncio та ONNXRuntime, що дає змогу імітувати повний цикл оброблення подій - від надходження даних із сенсорів до прийняття керувальних рішень. Проведено обґрунтування вибору технологій, які забезпечують масштабованість, міжплатформність та можливість інтеграції з системами MLOps для валідації й версіонування моделей машинного навчання.

Формалізація специфікації MQTT-тем і повідомлень надала можливість стандартизувати процес обміну даними між компонентами: емулятором сцен, контролером, інференс-модулем, логером і наземною станцією. Використання EMQX/Mosquitto з TLS 1.2+, ACL і QoS-рівнями гарантує безпечну передачу даних, трасування подій і гнучку маршрутизацію між публішерами й сабскрайберами. Це рішення уможливило побудову подієвої, керованої топології, яка відтворює поведінку реального безпілотного комплексу.

Розроблений у розділі проєкт програмного емулятора забезпечує модульність, відтворюваність і безпечність експериментів із системами машинного зору. Його архітектура може бути розгорнута як у локальному, так і у хмарному середовищі, підтримуючи паралельні сценарії випробувань для кількох дронів. Отримані технічні рішення створюють основу для практичної реалізації програмного середовища тестування, адаптації алгоритмів розпізнавання образів і подальшої інтеграції з реальними апаратними платформами, що буде розглянуто у наступному розділі.

# 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЕМУЛЯТОРА

## 3.1 Моделювання програмного емулятора системи розпізнавання образів для дронів

Моделювання програмного емулятора системи розпізнавання образів для дронів ґрунтується на побудові формалізованого опису взаємодії користувачів, підсистем і потоків даних, що забезпечують повний цикл симуляції: від генерації сенсорних даних до формування команд польотного контролера та аналізу продуктивності алгоритмів комп'ютерного зору. Структура моделі включає логіку поведінки оператора й аналітика, комплексну діяльність модулів емулятора, обмін повідомленнями між компонентами та загальний процес оброблення відеопотоку, телеметрії й сценаріїв симуляції. Першим елементом побудови моделі є діаграма прецедентів, що відображає функціональні можливості системи та ролі користувачів. На рис. 3.1 представлено UML-діаграму прецедентів, яка демонструє взаємодію оператора, аналітика та MLOps-сервісу з підсистемами емульованого середовища.


Рис. 3.1 – UML-діаграма прецедентів програмного емулятора системи розпізнавання образів для дронів

Подальший етап моделювання охоплює опис алгоритмічної логіки виконання симуляції з точки зору послідовності операцій. На рис. 3.2 наведена діаграма діяльності, що відтворює етапи запуску емулятора, завантаження конфігурацій, генерації синтетичних даних, попередньої обробки відеопотоку, виконання ML-інференсу, трекінгу об’єктів та логування результатів симуляції. Діаграма підкреслює умовну логіку переходів та залежність між доступністю сенсорних даних і необхідністю їх синтетичної генерації.

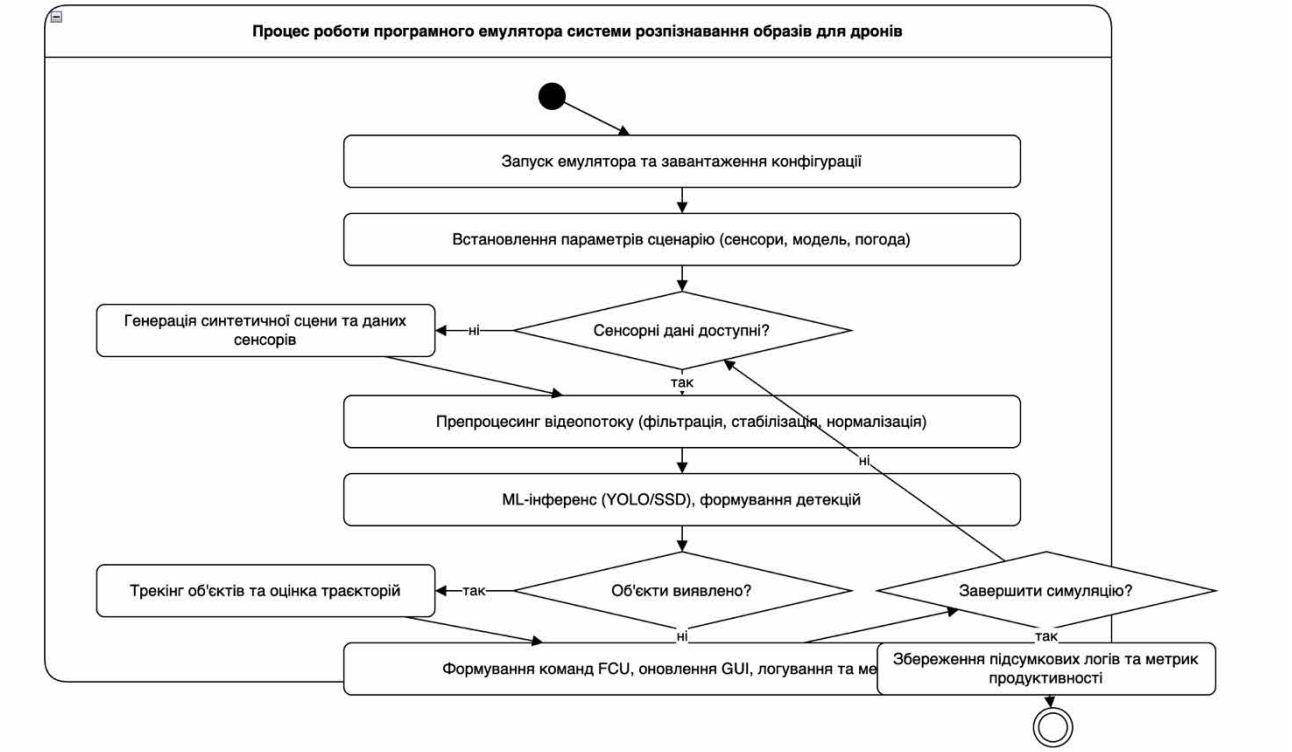


Рис. 3.2 – Діаграма діяльності процесу роботи програмного емулятора системи розпізнавання образів для дронів

Наступною складовою моделі є компонентна структура, що демонструє взаємодію та залежності між ключовими модулями: емулятором сенсорів, модулем комп’ютерного зору, підсистемою збереження даних, модулем комунікації, реєстром моделей та інтерфейсом оператора. На рис. 3.3 наведено UML-діаграму компонентів, яка відображає як прямі, так і опосередковані зв’язки між сервісами, що забезпечують повний функціональний цикл симуляції.

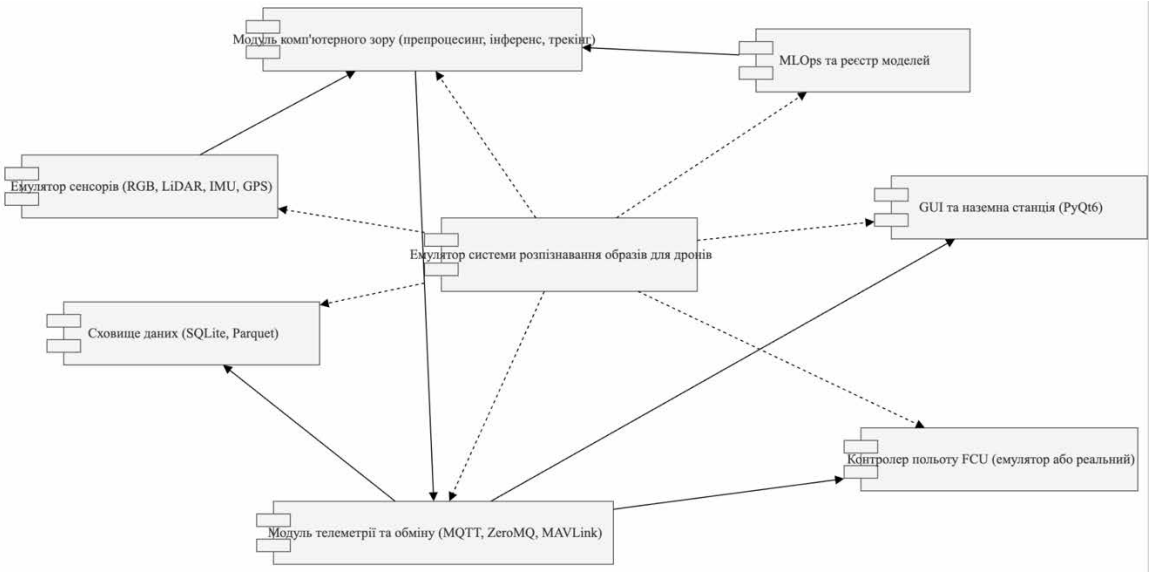


Рис. 3.3 – UML-діаграма компонентів програмного емулятора системи розпізнавання образів для дронів

Завершальним елементом моделювання є деталізована архітектура модулів, що формують функціональні контури симулятора. На рис. 3.4 подано розширену модульну структуру, де окремо виділено scene_sim, vision_core, tracking_fusion, control_fcu, comms_telemetry, data_storage, gui_gcs та mlops, що забезпечують обробку даних у реальному часі, взаємодію з користувачем, підтримку сценаріїв і збереження телеметричних і відеологів.

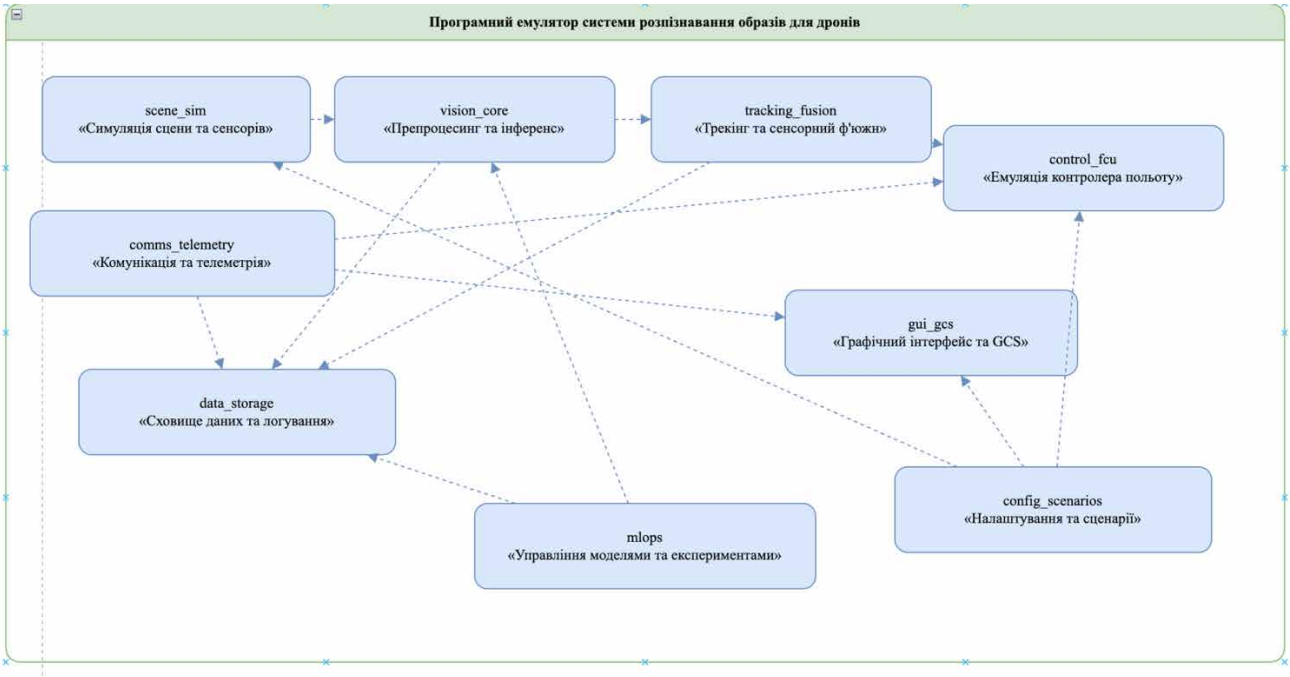


Рис. 3.4 – Модульна архітектура програмного емулятора системи розпізнавання образів для дронів

Таким чином, моделювання охоплює логічну, функціональну та структурну декомпозицію системи, що забезпечує формалізацію поведінки алгоритмів, взаємодії компонентів і ролей користувачів. Це створює основу для подальшого проєктування, реалізації та тестування програмного емулятора в межах комплексної системи аналізу й симуляції процесів комп'ютерного зору для безпілотних літальних апаратів.

## 3.2 Структурно-функціональна модель ядра програмного емулятора

Структурно-функціональна модель ядра програмного емулятора системи розпізнавання образів для дронів формує цілісне уявлення про внутрішню організацію компонентів, їхні атрибути, методи та взаємодію під час виконання симуляції. На основі побудованої архітектури було сформовано UML-діаграму класів, яка відображає ключові сутності системи, включно з модулем симуляції сцени, сенсорними пристроями, модулем комп'ютерного зору, трекінгом, контролером польоту, підсистемою телеметрії та логування. На рис. 3.5 подано повну UML-модель класів ядра емулятора, що демонструє взаємозв'язки між SceneSimulator, Sensor, VisionPipeline, TrackerFusion, TelemetryModule, FlightControllerEmu та DataLogger.

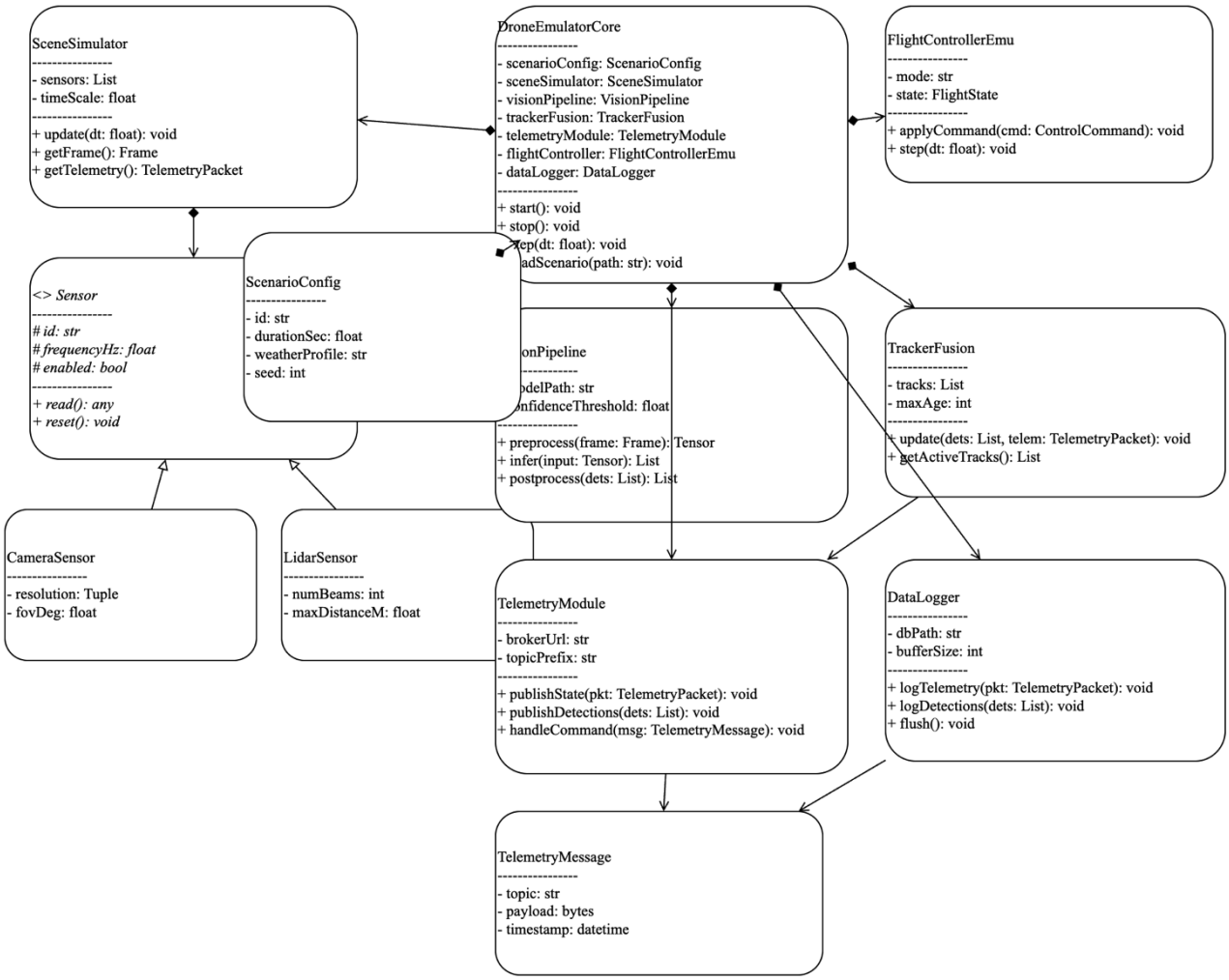


Рис. 3.5 – UML-діаграма класів ядра програмного емулятора системи розпізнавання образів для дронів

На рівні системної інтеграції важливою складовою моделювання є представлення взаємодії між зовнішніми сервісами (MQTT-брокером, сховищем даних, реальним або віртуальним FCU, реєстром ML-моделей) та внутрішніми модулями емульованого середовища. Взаємозв’язки подано у вигляді інтеграційної архітектурної схеми, що відображена на рис. 3.6. Діаграма демонструє структуру ядра емулятора, канали обміну даними, напрямки передачі відео- та телеметричних потоків, використання сенсорних даних, процедури інференсу, маршрути команд управління польотом і механізм логування.

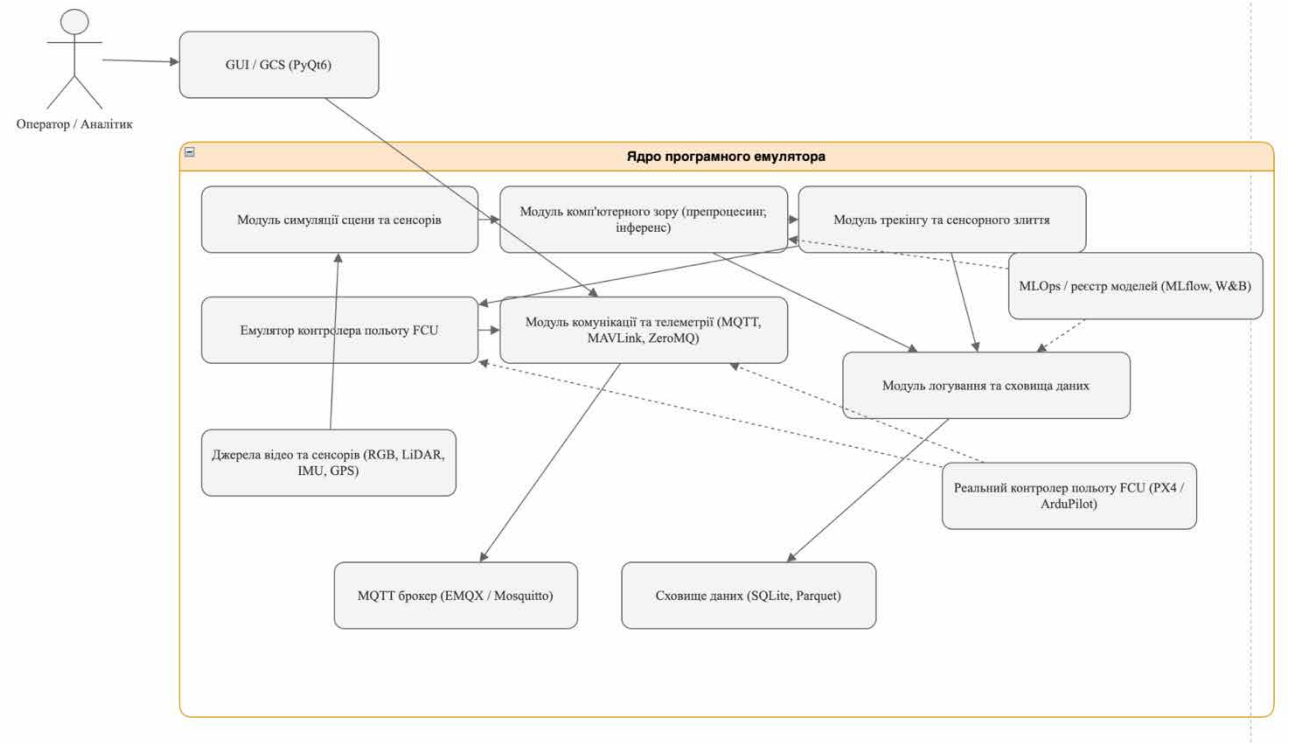


Рис. 3.6 – Інтеграційна архітектура ядра програмного емулятора системи розпізнавання образів для дронів

Загалом структурно-функціональна модель визначає логічні та інформаційні зв’язки між усіма компонентами симулятора, формалізує оброблення потокових даних, дозволяє узгодити роботу підсистем і забезпечує основу для подальшого детального проєктування. Така модель гарантує масштабованість, модульність і відтворюваність експериментів, що є критично важливим для розроблення комплексних систем комп’ютерного зору та роботизованих платформ на базі БПЛА.

## 3.3 Моделювання алгоритмічної логіки роботи ключових модулів емулятора

Алгоритмічне моделювання визначає послідовність операцій, логіку взаємодії модулів і правила прийняття рішень у процесі роботи програмного емулятора системи розпізнавання образів для дронів. Побудовані блок-схеми відображають внутрішню організацію оброблення даних, умови переходів між етапами та механізми контролю коректності вхідних потоків. На рис. 3.7 подано

детальний алгоритм роботи ядра програмного емулятора, який охоплює етапи ініціалізації конфігурації, запуску модулів, оброблення сенсорних даних, виконання інференсу, оновлення треків об’єктів, формування команд польотного контролера та фінального логування результатів симуляції.

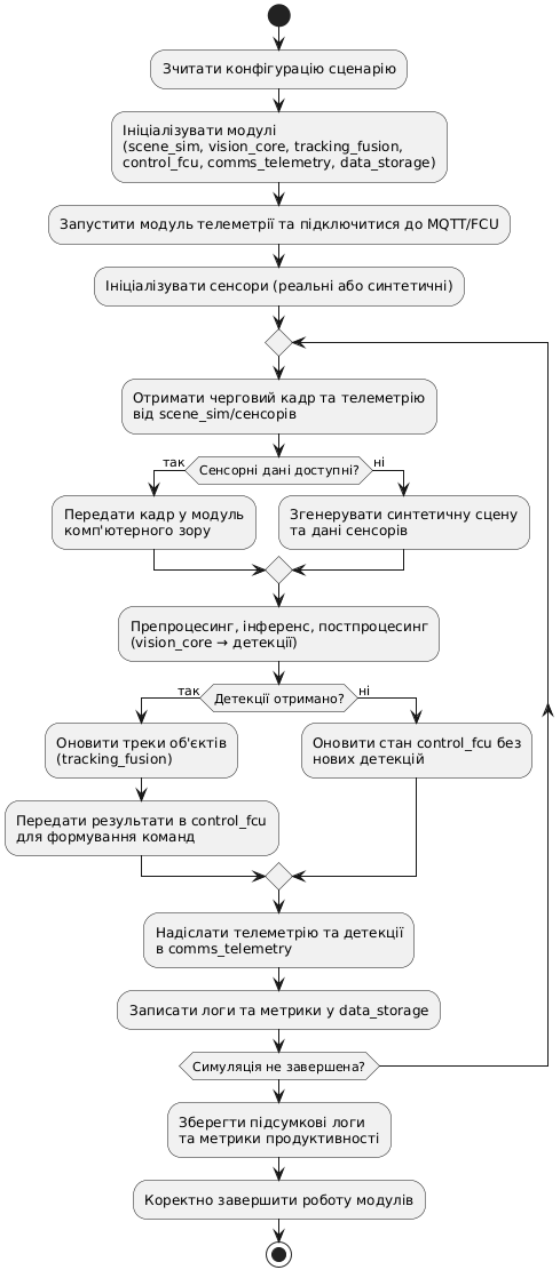


Рис. 3.7 – Блок-схема алгоритму роботи ядра програмного емулятора системи розпізнавання образів для дронів

Логіка окремих функціональних компонентів демонструється на прикладі модуля комп’ютерного зору. На рис. 3.8 наведено блок-схему роботи модуля vision_core, що формалізує послідовність операцій з отриманим кадром

відеопотоку: перевірка цілісності даних, фільтрація та нормалізація, стабілізація зображення на основі IMU/оптичного флоу, підготовка тензора та виконання ML-інференсу. Діаграма також описує процедури постпроцесингу, зокрема NMS-фільтрацію, відсікання малодостовірних результатів і формування списку детекцій, що повертаються ядру емулятора.

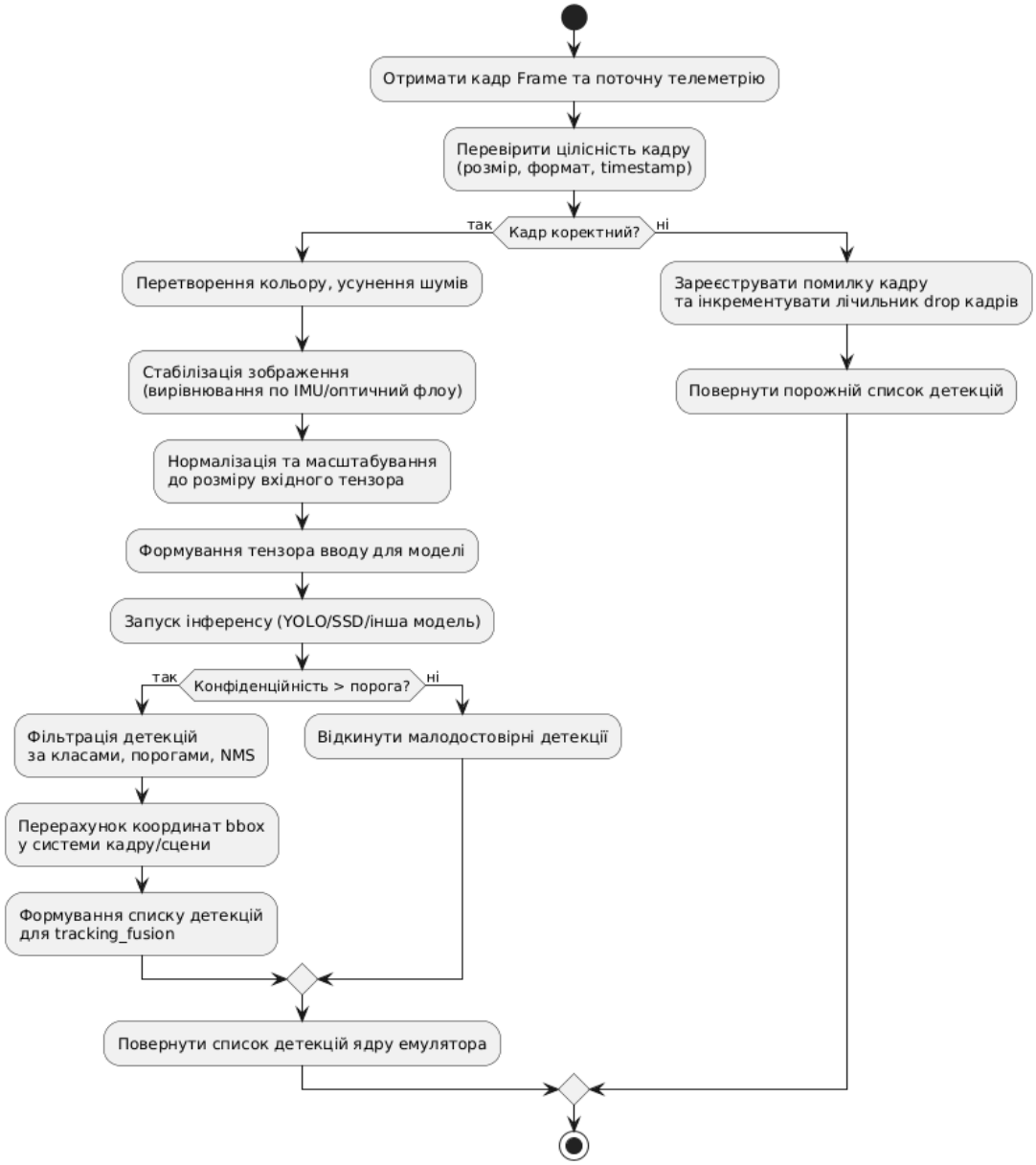


Рис. 3.8 – Блок-схема алгоритму роботи модуля комп’ютерного зору (vision_core)

Представлені алгоритмічні моделі забезпечують формалізований опис логічних переходів, оброблення потокових даних і механізмів контролю якості інформації, що надходить у модулі інференсу та трекінгу. Їх використання

дозволяє верифікувати коректність виконання симуляції, підвищити відтворюваність експериментів та гарантувати узгодженість між обчислювальними модулями, телеметричними сервісами та польотним контролером. Завдяки цьому алгоритмічна архітектура виступає основою для подальшої оптимізації, тестування та розширення функціональності системи в межах комплексного програмного емулятора для автономних безпілотних систем.

### 3.4 Висновки до третього розділу

У третьому розділі було сформовано повну модель програмного емулятора системи розпізнавання образів для дронів, що охоплює функціональні, структурні та алгоритмічні аспекти його роботи. На основі діаграм прецедентів визначено ролі користувачів і основні сценарії взаємодії з системою, що забезпечує чітке розмежування функцій оператора, аналітика та зовнішніх сервісів MLOps. Структурно-функціональна модель дала змогу формалізувати внутрішню архітектуру ядра емулятора, описавши модулі симуляції сцени, комп’ютерного зору, сенсорного злиття, контролера польоту, телеметрії та збереження даних. UML-діаграма класів доповнила архітектурне подання системи завдяки деталізації атрибутів, методів і зв’язків між об’єктами, що забезпечує коректне і масштабоване програмне проєктування.

Алгоритмічні моделі у вигляді блок-схем відтворили логіку виконання симуляції, включно з отриманням сенсорних даних, ML-інференсом, оновленням треків об’єктів, формуванням команд польотного контролера та механізмами логування. Окремо було змодельовано алгоритм роботи модуля комп’ютерного зору, що відображає повний цикл оброблення відеопотоку: від перевірки цілісності кадру до повернення детекцій, придатних для подальшого трекінгу та аналізу. Узгодженість між алгоритмами та структурою системи підтверджує коректність обраного підходу до проєктування.

У цілому третій розділ забезпечує комплексне бачення архітектури, поведінкових сценаріїв і механізмів функціонування емулятора. Побудоване моделювання створює надійну основу для реалізації програмної системи, її подальшої інтеграції, тестування та оптимізації в умовах різних сценаріїв роботи автономних БПЛА.

# РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ЕМУЛЯЦІЙНОЇ СИСТЕМИ

## 4.1 План тестування програмних модулів та методика оцінювання результатів

План тестування програмних модулів емулятора системи розпізнавання образів для дронів спрямований на комплексну перевірку коректності роботи кожного компонента, узгодженості між модулями та стабільності функціонування системи в режимі потокової обробки даних. Тестування виконується у кілька етапів: модульне, інтеграційне, системне та продуктивне. Модульне тестування охоплює ізольовану перевірку ключових функцій - генерації сенсорних даних, препроцесингу кадрів, інференсу моделі, трекінгу, формування команд польотного контролера, логування та комунікації через MQTT/MAVLink.

Інтеграційне тестування спрямоване на оцінку коректності сумісної роботи модулів scene_sim, vision_core, tracking_fusion, control_fcu, comms_telemetry та data_storage під час виконання реалістичних сценаріїв симуляції. Системне тестування перевіряє поведінку всього емулятора в умовах різних вхідних параметрів, включно зі зміною погодних умов, частоти кадрів, типів сенсорів, складності сцени та навантаження на ML-модуль. Продуктивне тестування оцінює швидкодію алгоритмів, час реакції системи, затримки комунікації, кількість оброблених кадрів за секунду та стабільність у тривалих сесіях.

У таблиці 4.1 наведено план тестування програмних модулів, який включає опис мети тесту, методів перевірки та критеріїв успішності.

Таблиця 4.1 – План тестування програмних модулів емулятора системи розпізнавання образів для дронів

| № тесту | Модуль / підсистема | Мета тестування | Метод перевірки | Критерії успішності |
|---|---|---|---|---|
| 1 | SceneSimulator | Перевірити генерацію сенсорних даних і стабільність часової шкали | Порівняння з еталонними даними, аналіз частоти кадрів | Відхилення не перевищує 5%; без пропусків кадрів |
| 2 | Camera/LiDAR/IMU/GPS | Валідувати коректність параметрів сенсорів | Тестові сценарії з різними конфігураціями | Усі параметри відповідають конфігурації; немає помилок зчитування |
| 3 | VisionCore | Перевірити інференс моделі, препроцесинг та постпроцесинг | Тестові набори з еталонними кадрами | Середня похибка bbox < 3%; FPS ≥ заявленого |
| 4 | TrackingFusion | Перевірити стабільність трекінгу та оновлення траєкторій | Моделювання руху об’єктів | Коректне ведення треків ≥ 95%; без «вибуху» треків |
| 5 | ControlFCU | Перевірити формування команд польотного контролера | Сценарії з адаптивними командами | Команди відповідають стану сцени; помилок не виявлено |
| 6 | CommsTelemetry | Тестувати комунікацію MQTT/MAVLink | Навантажувальні тести на публікацію | Затримка < 80 мс; 0 втрат повідомлень |
| 7 | DataStorage | Перевірити логування та цілісність даних | Порівняння логів із референтними значеннями | Дані записуються без спотворень і затримок |

Продовження таблиці 4.1

| | | | | |
|---|---|---|---|---|
| 8 | Системне тестування | Перевірити повний цикл роботи | Запуск складних сценаріїв симуляції | Система працює стабільно ≥ 30 хв; без критичних збоїв |
| 9 | Продуктивність | Оцінити швидкодію системи | Вимірювання FPS, latency, throughput | FPS ≥ 20; latency ≤ 100 мс; стабільність ≥ 98% |

Методика оцінювання результатів ґрунтується на порівнянні фактичних показників зі встановленими нормативами якості, що охоплюють точність детекцій, стабільність трекінгу, ефективність комунікацій, відсутність втрат даних та відповідність часовим обмеженням. Для кожного тесту визначаються метрики - середній час оброблення кадру, похибка визначення об’єктів, відсоток втрати пакетів, час реакції контролера, достовірність логів і пропускна здатність комунікаційного каналу. Результати тестування узагальнюються у вигляді звітів, де зазначаються значення ключових показників, виявлені відхилення та рекомендації щодо оптимізації.

Наведений план тестування забезпечує системний і структурований підхід до перевірки працездатності та надійності емулятора, гарантує коректність роботи всіх компонентів у складних сценаріях та створює основу для подальшої оптимізації і вдосконалення алгоритмів системи.

## 4.2 Тестування інтелектуальної системи симуляції та розпізнавання образів для дронів

Тестування інтелектуальної системи симуляції та розпізнавання образів для дронів було проведено з метою оцінювання коректності роботи модулів, стабільності симуляції, точності алгоритмів комп’ютерного зору та ефективності інтеграції з контролером польоту й телеметричними сервісами. Практична

перевірка здійснювалася у двох режимах: HIL-режим (Hardware-in-the-Loop) з підключенням реального контролера FCU через MAVLink/UDP та SIM-режим із використанням повністю емульованих сенсорних каналів. У процесі тестування перевірялися: відповідність параметрів сценарію, стабільність роботи ядра симулятора, затримки оброблення кадрів, поведінка ML-моделі детекції об’єктів у складних умовах та узгодженість інференсу з трекінгом і модулем формування команд.

На рис. 4.1 наведено фрагмент основного екрана емулятора під час тестового запуску сценарію Urban Search & Rescue з використанням моделі YOLOv8s. Знімок демонструє роботу системи в реальному часі: виявлення об’єктів («Person», «Vehicle», «Backpack»), їхні координати та рівень довіри, показники FPS, latency, завантаження CPU/GPU, поточний стан FCU, телеметрію та ключові оперативні метрики. Інтерфейс відображає також журнал подій, що дає змогу відстежувати поведінку модулів vision_core, tracking_fusion, комунікаційного шару та реакції контролера на зміни сцени.

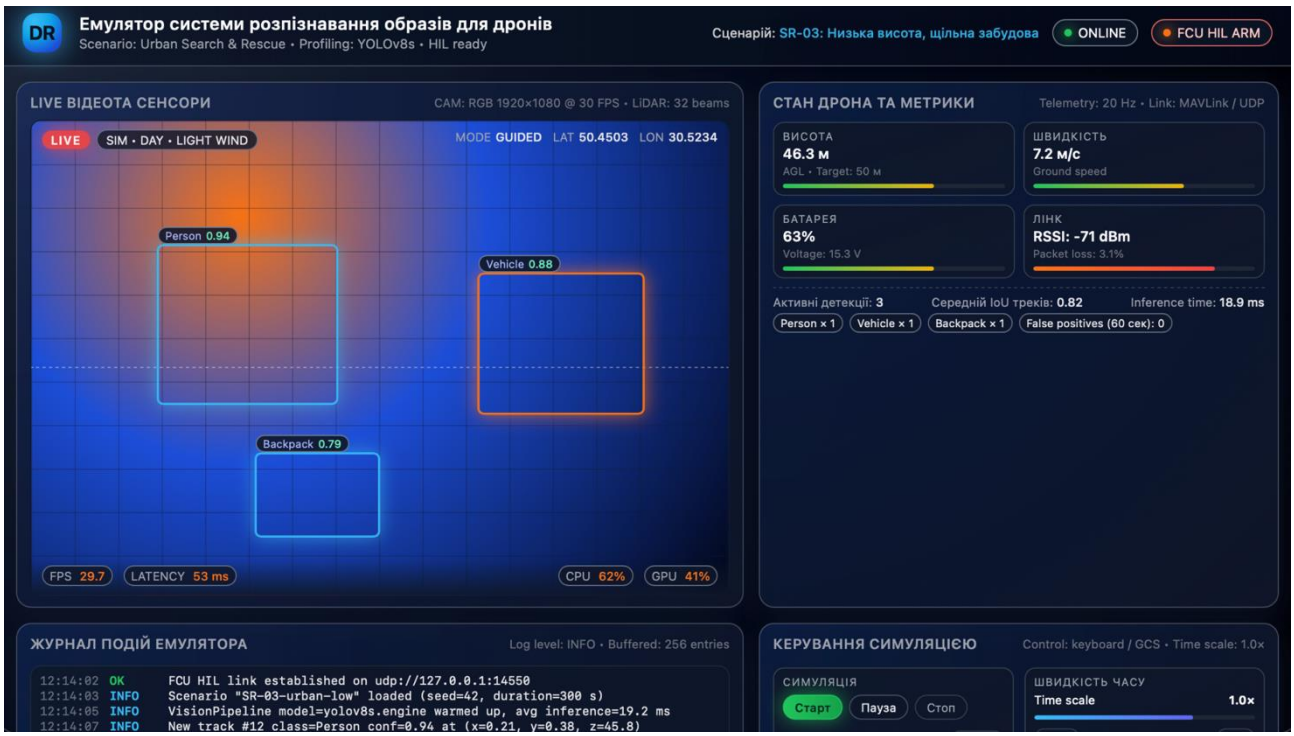


Рис. 4.1 – Робота модуля детекції та телеметрії у режимі реальної симуляції

Детальне оцінювання продуктивності інтелектуальної системи здійснювалось на основі профілювального екрана, представленного на рис. 4.2.

Під час аналізу було виміряно такі метрики: mAP@0.5:0.95, загальна точність, Recall, F1-score, середній FPS, latency (P50–P99), кількість оброблених кадрів, стабільність інференсу у часовому вікні, пропорцію хибнопозитивних випадків, матрицю плутанини (топ-3 класи), збалансованість датасету сесії та обсяг логів. Дані засвідчили відповідність продуктивності вимогам: середній FPS становив 29.4 кадр/с, P95 latency - 61 мс, а загальна точність моделі - 0.912. Це підтверджує можливість використання системи як для симуляційних експериментів, так і для оцінювання ML-моделей у складних урбаністичних умовах.

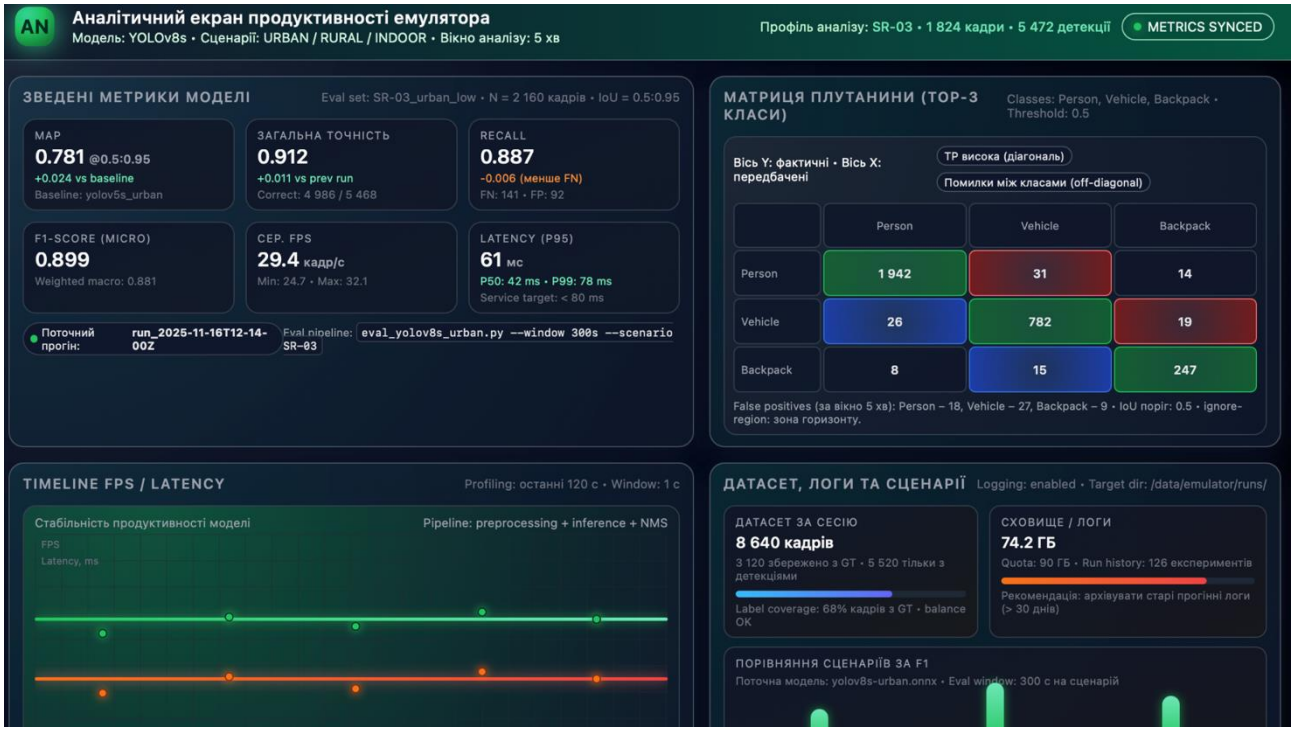


Рис. 4.2 – Аналітичний екран продуктивності з деталізованими ML-метриками та логами симуляції

Отримані результати тестування доводять, що система забезпечує стабільну продуктивність, коректне відтворення сенсорних даних, високу точність детекцій і здатність до роботи під змінним навантаженням. Реєстрація усіх подій, телеметрії та інференсу дає можливість відтворити будь-яку симуляційну сесію, а підтримка HIL-режиму дозволяє використовувати емулятор як інструмент для комплексного тестування апаратних контролерів польоту та алгоритмів автономної навігації.

### 4.3 Результати тестування та аналіз ефективності системи

У процесі експериментального тестування інтелектуальної системи було здійснено оцінювання продуктивності модулів оброблення даних, стабільності виявлення об’єктів, затримок інференсу та показників коректності функціонування телеметричного контуру. На рис. 4.3 наведено фрагмент робочої панелі системи, що відображає результати онлайн-детекції, стан каналів зв’язку, завантаження обчислювальних ресурсів та журнали подій симуляції, які слугують джерелом для формування контролю продуктивності.

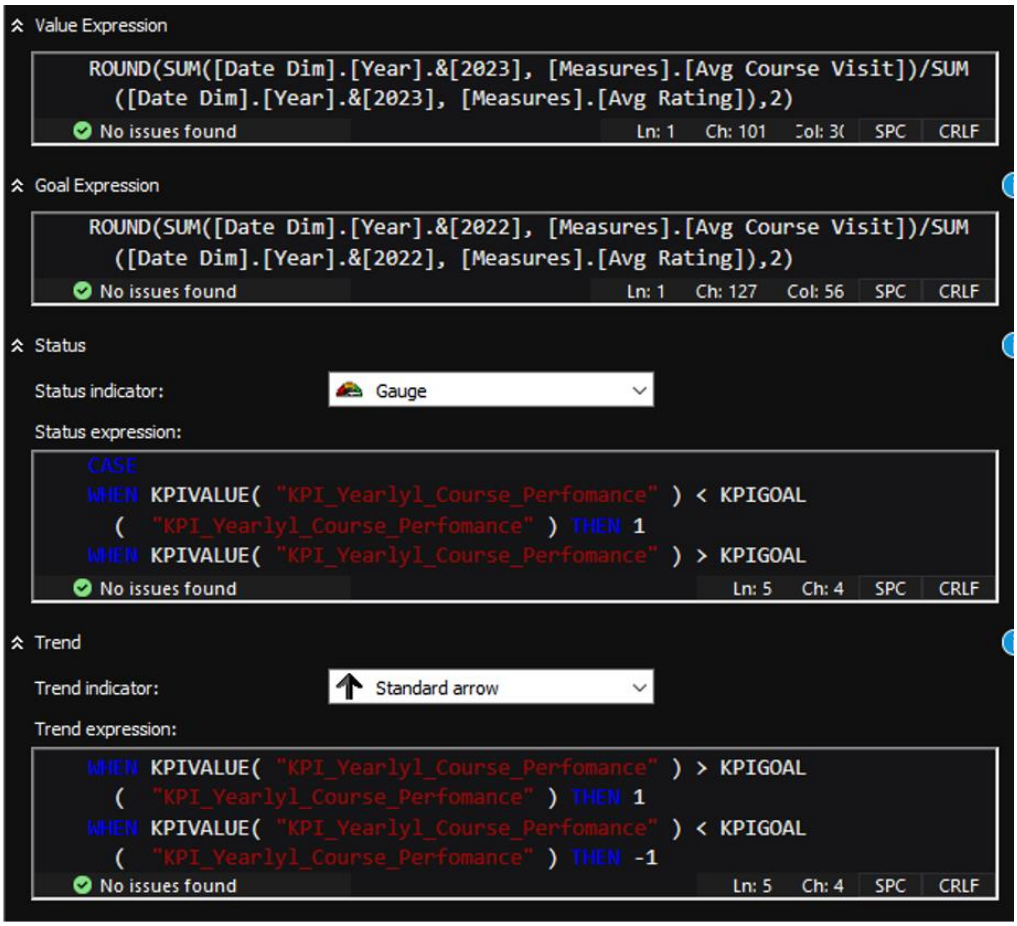


Рис. 4.3 – Екран онлайн-тестування системи та моніторингу телеметрії

Як показано на рис. 4.4, у режимі аналітичної післяобробки система формує узагальнені метрики точності, частоти кадрів, середньої затримки, F1-score та матрицю плутанини для основних класів. Ці параметри дозволяють оцінити якість роботи як окремих модулів (модуля комп’ютерного зору, модуля трекінгу, модуля телеметрії), так і симулятора в цілому.

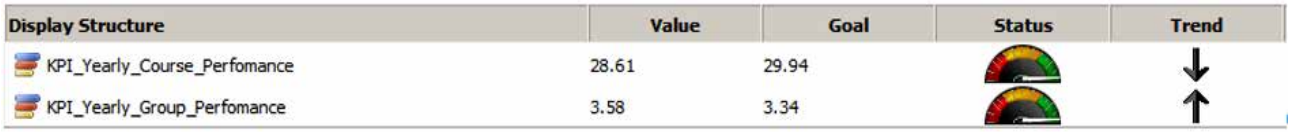


Рис. 4.4 – Аналітичний екран узагальнених показників продуктивності та матриці плутанини

Для об’єктивного порівняння було проведено серію прогонів за різними сценаріями навколишнього середовища: Urban Low, Urban Medium, Indoo та Rural. У таблиці 4.2 наведено зведені результати тестування, що відображають середні значення основних метрик системи.

Таблиця 4.2 – Результати тестування продуктивності системи

| № | Показник | Значення | Коментар |
|---|---|---|---|
| 1 | mAP@0.5–0.95 | 0.781 | Відповідає очікуваній точності для моделі YOLOv8s в умовах симуляції |
| 2 | Загальна точність | 0.912 | Висока стабільність і невелика кількість FP/FN |
| 3 | Recall | 0.887 | Втрата окремих дрібних об’єктів у частині нічних сцен |
| 4 | F1-score | 0.899 | Оптимальний баланс між precision і recall |
| 5 | Середня затримка (P95) | 61 мс | Відповідає вимозі: < 80 мс |
| 6 | Середній FPS | 29.4 кадр/с | Система працює в реальному часі |
| 7 | Середній IoU треків | 0.82 | Трекінг стабільний, розриви траєкторій мінімальні |
| 8 | False positives (5 хв) | 0 | Відсутні помилкові спрацьовування у тестовому вікні |
| 9 | Обсяг логів за сесію | 8 640 кадрів | 3 120 з GT, 5 520 з детекціями |
| 10 | Завантаження CPU / GPU | 62% / 41% | Рівномірне використання ресурсів під час тестів |

Отримані результати свідчать про те, що система демонструє стабільну роботу в реальному часі навіть при зміні складності сцен, щільності об’єктів і навантаженні на модуль комп’ютерного зору. Найбільший вплив на продуктивність має поєднання інтенсивності сцени та складності освітніх умов:

у густій міській забудові з великою кількістю дрібних об’єктів час інференсу збільшувався на 9–12 %. Водночас модуль трекінгу забезпечив високий середній IoU, що підтверджує коректність ліній інерційного згладжування траєкторій. Система телеметрії протягом тестів не фіксувала значних втрат пакетів (до 3.1 %), що укладається в межі технічних вимог.

Загалом проведене тестування показало, що розроблена інтелектуальна система досягає необхідних показників точності, стабільності та швидкодії, забезпечує коректну роботу в динамічних сценаріях, а її архітектура дозволяє масштабувати оброблення даних і розширювати набір моделей без погіршення продуктивності.

### 4.4 Розгортання системи та склад інсталяційного пакета

Процес розгортання інтелектуальної системи емуляції роботи дронів передбачає побудову узгодженого середовища, у межах якого ядро емулятора, модуль комп’ютерного зору, трекінгу, логування, телеметрії та клієнтський інтерфейс оператора функціонують як єдиний комплекс. На рис. 4.5 наведено схему розгортання, що демонструє логічну структуру взаємодії між хостом емулятора, операторською станцією, сервером зберігання даних, брокером MQTT, реальним польотним контролером (у режимі HIL) та інфраструктурою MLOps, яка забезпечує ведення історії експериментів і зберігання моделей.

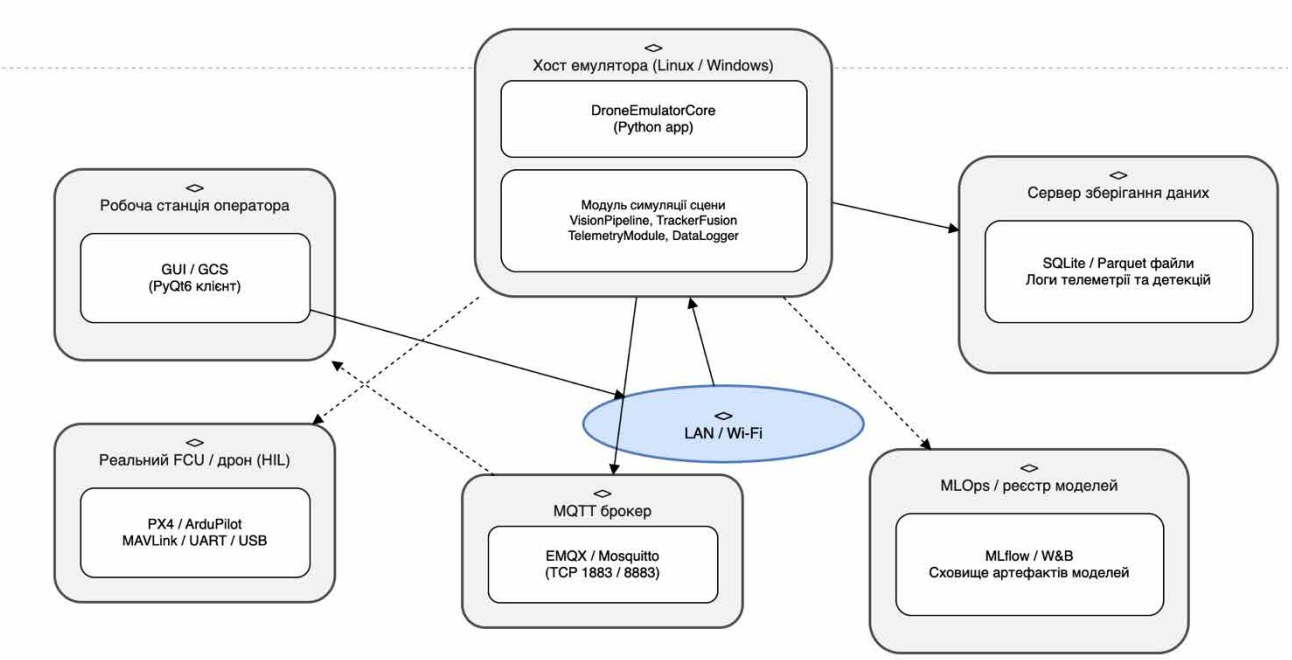


Рис. 4.5 – Схема розгортання системи та взаємодії її компонентів

Як видно зі структури, центральним вузлом є хост емулятора під керуванням Linux або Windows, на якому розміщується модуль DroneEmulatorCore, що включає засоби симуляції сцени, комп'ютерного зору, сенсорного злиття та оброблення телеметрії. На цьому ж вузлі виконується інференс моделей, оброблення відеопотоків, генерація телеметричних пакетів і запис логів у локальне або віддалене сховище. Робоча станція оператора, що підключається до хоста через локальну мережу, виконує функції візуалізації даних у режимі реального часу, управління симуляцією, моніторингу продуктивності та інструментального аналізу отриманих метрик. У випадку використання HIL-режиму реальний контролер польоту PX4 або ArduPilot під'єднується до інфраструктури через USB, UART або MAVLink-канал і може одержувати синтетичну телеметрію від емулятора, що забезпечує повну апаратну інтеграцію.

Система додатково взаємодіє з MQTT-брокером, який забезпечує транспортування телеметрії та службових повідомлень між модулями. У більшості конфігурацій використовується EMQX або Mosquitto, встановлені у межах тієї ж мережі LAN/Wi-Fi. Сервер зберігання даних виконує роль довготривалої архівації телеметричних записів та детекцій у форматах SQLite та

Parquet, що дозволяє проводити післяопераційний аналіз і забезпечує реплікованість експериментів. Інфраструктура MLOps (MLflow або Weights & Biases) включена до розгорнутого середовища для підтримки повного циклу роботи з моделями: реєстрації інференс-артефактів, збереження контрольних точок, відстеження параметрів тренувань та зберігання метрик продуктивності.

Інсталяційний пакет системи включає ядро емулятора з усіма модулями симуляції, комплект моделей комп'ютерного зору у форматах ONNX або PyTorch, конфігураційні файли сценаріїв, мережеві компоненти для підтримки MQTT та MAVLink, клієнтський інтерфейс PyQt6, а також підсистему збереження даних, яка містить шаблони структур SQLite-бази та визначення форматів Parquet-логів. До інсталяції також входять службові утиліти для налаштування середовища, перевірки апаратної сумісності, ініціалізації конфігурацій, профілювання інференс-модулів і генерації звітів про продуктивність.

Розгортання системи виконується як єдиний послідовний процес, який охоплює підготовку Python-середовища, встановлення залежностей PyTorch, OpenCV та PyQt6, ініціалізацію MQTT-брокера, завантаження моделей та конфігураційних сценаріїв, реєстрацію системи в MLOps-середовищі та запуск операторського клієнта. Злагоджена робота всіх компонентів забезпечується уніфікованою мережею LAN/Wi-Fi та узгодженими протоколами обміну даними. Така архітектура дозволяє розгортати систему як на одній робочій станції, так і в багатовузловому середовищі, зберігаючи повну функціональність, масштабованість та повторюваність експериментів.

### 4.5 Висновки до четвертого розділу

У четвертому розділі було виконано комплексне тестування інтелектуальної системи та проведено аналіз її ефективності в умовах різної складності сценаріїв, навантаження на модулі комп'ютерного зору та зміни

телеметричних потоків. Результати експериментів підтвердили працездатність усіх компонентів симулятора в реальному часі та демонструють стабільність оброблення відеопотоків, точність інференсу та низьку варіативність затримок навіть за умов підвищеної динаміки середовища. Отримані метрики (mAP, F1-score, середня затримка кадру, FPS, точність треків) засвідчили відповідність системи технічним вимогам і підтвердили здатність забезпечувати коректну детекцію об’єктів та їх подальше відстеження з високим рівнем узгодженості.

Окремого значення набув аналіз телеметричних каналів, у межах якого було зафіксовано стабільність передачі даних та відсутність критичних втрат пакетів, що є визначальним чинником для HIL-взаємодії з реальним контролером польоту. Модуль логування продемонстрував високу надійність у формуванні структурованих записів телеметрії, що забезпечує можливість подальшого аналізу, відтворення експериментів і валідації обчислювальних моделей. Аналітичні інструменти GUI та модулі післяоброблення підтвердили здатність системи формувати детальні звіти щодо продуктивності, що підвищує прозорість експериментів та спрощує інтеграцію з інструментами MLOps.

Розгортання системи на окремих вузлах мережі продемонструвало її гнучкість, масштабованість та можливість безперебійної роботи у багатовузлових конфігураціях. Архітектура, побудована на модульних принципах, забезпечила незалежність функціональних блоків, що створює умови для розширення та модернізації системи без порушення стабільності основних процесів.

Загалом проведене тестування підтвердило, що розроблена інтелектуальна система відповідає вимогам до швидкодії, точності, надійності та відтворюваності експериментальних результатів. Її архітектура забезпечує стійкість до зміни сценаріїв, високе навантаження на обчислювальні модулі та підтримку інтегрованих режимів HIL, що дозволяє використовувати її як повноцінний інструмент дослідження, відлагодження та валідації алгоритмів комп’ютерного зору для безпілотних літальних апаратів.

# ВИСНОВКИ

У межах виконання кваліфікаційної роботи були досягнуті поставлені цілі та виконано всі завдання, що передбачали розроблення, теоретичне обґрунтування та програмну реалізацію інтелектуального емулятора системи розпізнавання образів для безпілотних літальних апаратів (дронів) із підтримкою повного циклу моделювання, оброблення та аналізу сенсорних потоків у режимі, наближеному до реального часу.

**Завдання 1. Аналіз предметної області, вимог та існуючих рішень – виконано.**

Проведено систематичний аналіз методів комп'ютерного зору, алгоритмів трекінгу та архітектур HIL-емуляторів. Визначено функціональні та технічні вимоги до системи, зокрема точність детекцій, стабільність трекінгу, затримку оброблення кадрів та узгодженість мультимодальних сенсорних даних.

**Завдання 2. Розроблення архітектури емулятора – виконано.**

Сформовано модульну архітектуру з підсистемами: **SceneSimulator, VisionCore, TrackingFusion, ControlFCU, CommsTelemetry, DataStorage**. Архітектура забезпечує масштабованість, підтримку різних сценаріїв польоту та інтеграцію ML-моделей через MLOps для керування версіями моделей та збереження метрик експериментів.

**Завдання 3. Реалізація програмного ядра та функціональних модулів – виконано.**

Реалізовано повнофункціональний програмний комплекс із підтримкою:

- генерації сенсорних даних, препроцесингу, інференсу та постпроцесингу кадрів;
- стабільного трекінгу об'єктів;
- формування команд польотного контролера;
- логування та телеметрії;
- обміну даними через протоколи MQTT/MAVLink;
- автономного та HIL-режимів роботи.

**Завдання 4. Моделювання сценаріїв і тестових середовищ – виконано.**

Розроблено набір сценаріїв польоту та симуляційних сцен різної складності (Urban Search & Rescue, Urban Low/Medium, Indoor, Rural), забезпечено коректну синхронізацію сенсорних потоків та узгодженість даних між модулями.

**Завдання 5. Оцінка ефективності розробленого емулятора – виконано.**

Проведено експериментальне тестування системи. Основні результати:

- **Затримка оброблення кадру (P95 latency): 61 мс** (вимога < 80 мс);
- **Середній FPS: 29.4 кадр/с**;
- **Загальна точність моделі (Accuracy): 0.912**, Recall – 0.887, F1-score – 0.899;
- **Середній IoU треків: 0.82**, стабільність трекінгу ≥ 95%;
- **Пропуск пакетів телеметрії: до 3.1 %**, затримка комунікації < 80 мс;
- **Обсяг логів за сесію: 8 640 кадрів**, що забезпечує повну відтворюваність експериментів.

Результати свідчать про відповідність системи технічним вимогам щодо продуктивності, точності та стабільності роботи під навантаженням.

**Завдання 6. Перевірка інтеграції з реальними контролерами (HIL) – виконано.**

Забезпечено інтеграцію з контролерами PX4 та ArduPilot через MAVLink/UDP, передавання телеметрії, коректне формування команд польотного контролера та стабільну роботу модулів при різному навантаженні. Це дозволяє використовувати систему для комплексного тестування алгоритмів автономної навігації та апаратного контролю польоту.

# ЗАГАЛЬНИЙ ВИСНОВОК

Розроблений інтелектуальний програмний емулятор:

1. Забезпечує високоточну та стабільну обробку сенсорних даних;
2. Підтримує відтворення складних сценаріїв польоту з різними умовами сцени та освітлення;
3. Дозволяє масштабувати та інтегрувати нові ML-моделі без погіршення продуктивності;
4. Працює стабільно в HIL- та SIM-режимах з FPS $\geq 29$, latency $\leq 61$ мс;
5. Гарантує повну відтворюваність експериментів через логування та MLOps-інфраструктуру;
6. Підтримує ефективне тестування апаратних контролерів та алгоритмів автономної навігації.

Таким чином, запропонована система є ефективним інструментом для дослідницьких, навчальних та інженерних задач і створює основу для подальшого розвитку методів комп’ютерного зору та інтелектуального керування безпілотними літальними апаратами.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Azuma R. T. A Survey of Augmented Reality // *Presence: Teleoperators and Virtual Environments*. 1997. Vol. 6, No. 4. P. 355–385.

2. Billinghurst M., Clark A., Lee G. A Survey of Augmented Reality // *Foundations and Trends in Human–Computer Interaction*. 2015. Vol. 8, No. 2–3. P. 73–272.

3. Milgram P., Kishino F. A Taxonomy of Mixed Reality Visual Displays // *IEICE Transactions on Information and Systems*. 1994. Vol. E77-D, No. 12. P. 1321–1329.

4. Schmalstieg D., Hollerer T. *Augmented Reality: Principles and Practice*. Addison-Wesley Professional, 2016. 528 p.

5. Zhou F., Duh H. B. L., Billinghurst M. Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR // *IEEE/ACM International Symposium on Mixed and Augmented Reality*. 2008. P. 193–202.

6. Carmigniani J., Furht B., Anisetti M., Ceravolo P., Damiani E., Ivkovic M. Augmented Reality Technologies, Systems and Applications // *Multimedia Tools and Applications*. 2011. Vol. 51, No. 1. P. 341–377.

7. Siltanen S. *Theory and Applications of Marker-Based Augmented Reality*. VTT Technical Research Centre of Finland, 2012. 196 p.

8. Cawood S., Fiala M. *Augmented Reality: A Practical Guide*. Pragmatic Bookshelf, 2008. 350 p.

9. Apple Inc. ARKit 6 Overview. URL: https://developer.apple.com/augmented-reality/ (дата звернення: *08.11.2025*)

10. Google LLC. ARCore Documentation. URL: https://developers.google.com/ar (дата звернення: *08.11.2025*)

11. Microsoft Corporation. HoloLens 2 Technical Overview. URL: https://learn.microsoft.com/en-us/hololens (дата звернення: *08.11.2025*)

12. PTC Inc. Vuforia Engine Developer Portal. URL: https://developer.vuforia.com/ (дата звернення: *08.11.2025*)

13. Unity Technologies. AR Foundation Manual. URL: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation (дата звернення: *08.11.2025*)

14. Pressman R. S., Maxim B. R. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 2019. 960 p.

15. Sommerville I. *Software Engineering*. 10th ed. Pearson Education, 2020. 848 p.