Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютерна система доставки вантажу безпілотним літальним апаратом*

Виконав: студент 4 курсу, групи СІ-42
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

Манько А. М.
(підпис) (прізвище та ініціали)

Керівник Лещишин Ю. З.
(підпис) (прізвище та ініціали)

Нормоконтроль Тиш Є. В.
(підпис) (прізвище та ініціали)

Завідувач кафедри Осухівська Г.М.
(підпис) (прізвище та ініціали)

Рецензент Михилик Д. М.
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
«25» квітня 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

` Манька Андрія Михайловича
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютерна система доставки вантажу безпілотним літальним апаратом*

Керівник роботи *к.т.н. доц. Лещишин Юрій Зіновійович*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від
« 24 » квітня 2026 року № 4/9-188 . « 24 » квітня 2026 року № 4/9-189 .
2. Термін подання студентом завершеної роботи *17.06.2026*
3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз технічного завдання
2. Проєктна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці.
Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Схема електрична структурна (загальний вигляд)
2. Схема електрична структурна (розробляючої системи)
3. Схема електрична принципова
4. Блок схема алгоритму

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*Сенчишин В. С.*		

7. Дата видачі завдання *25.04.2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*26.01 – 02.02*	*Викон.*
2.	*Робота над першим розділом «Аналіз технічного завдання»*	*03.02 – 15.02*	*Викон.*
3.	*Робота над другим розділом «Проєктна частона»*	*20.04 – 10.05*	*Викон.*
4.	*Робота над третім розділом «Практична частина»*	*11.05 – 24.05*	*Викон.*
5.	*Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»*	*25.05 – 31.05*	*Викон.*
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*01.06 – 7.06*	*Викон.*
7.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*8.06 – 14.06*	*Викон.*
8.	*Попередній захист кваліфікаційної роботи бакалавра*	*15.06 – 21.06*	*Викон.*
9.	*Захист кваліфікаційної роботи бакалавра*	*24.06*	*Викон.*

Студент _____ *Манько А. М.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Лещишин Ю. З.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Манько А.М. Комп'ютерна система доставки вантажу безпілотним літальним апаратом: робота на здобуття кваліфікаційного ступеня бакалавра: спец. 123 — комп'ютерна інженерія. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2026.

Ключові слова: безпілотний літальний апарат, точне приземлення, ArUco-маркер, Raspberry Pi, optical flow, комп'ютерний зір, ArduPilot

Кваліфікаційна робота присвячена розробці комп'ютерної системи доставки вантажу безпілотним літальним апаратом, орієнтованої на автономне точне приземлення на спеціалізовану станцію видачі замовлень.

У першому розділі проаналізовано технічне завдання, виконано огляд наявних рішень у сфері дронової доставки та методів точного приземлення, сформульовано вимоги до системи. У другому — розроблено узагальнену структуру, обґрунтовано вибір апаратного та програмного забезпечення, спроєктовано схему електричну принципову, UML-діаграму взаємодії компонентів та блок-схему алгоритму роботи. У третьому розділі описано налаштування прошивки ArduPilot для точного приземлення та розроблено програмне забезпечення бортового комп'ютера, що реалізує детекцію ArUco-маркерів засобами комп'ютерного зору і керування скиданням вантажу. У четвертому розділі розглянуто питання безпеки життєдіяльності та основ охорони праці при експлуатації автономної системи доставки.

ANNOTATION

Manko A. Computer System for Cargo Delivery Using an Unmanned Aerial Vehicle: Bachelor's Graduation Thesis: speciality 123 — computer engineering. Ternopil: Ternopil Ivan Puluj National Technical University, 2026.

Keywords: unmanned aerial vehicle, precision landing, ArUco marker, Raspberry Pi, optical flow, computer vision, ArduPilot

The first section analyses the technical specification, reviews existing solutions in the field of drone delivery and methods of precision landing, and formulates requirements for the system. The second section develops the generalised system structure, justifies the choice of hardware and software, and presents the electrical schematic, UML interaction diagram, and the block diagram of the operation algorithm. The third section describes the configuration of the ArduPilot firmware for precision landing and the development of the onboard computer software, which implements ArUco marker detection by means of computer vision and control of the cargo release mechanism. The fourth section addresses life safety and occupational health considerations related to the operation of an autonomous delivery system.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

БПЛА – безпілотний літальний апарат

FC (Flight Controller) – польотний контролер

GPS (Global Positioning System) – глобальна система позиціонування

LTE (Long-Term Evolution) – стандарт стільникового зв'язку 4-го покоління

MAVLink (Micro Air Vehicle Link) – протокол обміну даними з БПЛА

LiDAR (Light Detection and Ranging) – оптичний дальномір

RPi (Raspberry Pi) – одноплатний комп'ютер

ESC (Electronic Speed Controller) – електронний регулятор швидкості

PWM (Pulse Width Modulation) – широтно-імпульсна модуляція

OpenCV (Open Source Computer Vision Library) – бібліотека комп'ютерного зору

ArUco – тип візуальних маркерів для роботизованих систем

ПЗ – програмне забезпечення

ВСТУП

Стрімкий розвиток електронної комерції та збільшення обсягу замовлень створюють зростаюче навантаження на системи логістики, особливо на її ділянку доставку від складу безпосередньо до отримувача. У щільно забудованих міських районах класична доставка наземним транспортом часто є повільною, дорогою та залежною від дорожньої ситуації. Як альтернатива в світі активно розвивається доставка безпілотними літальними апаратами (БПЛА), яку вже у комерційному масштабі впровадили компанії Meituan у Шеньчжені, Amazon Prime Air, Wing, Zipline та інші.

Існуючі рішення дронової доставки демонструють принципову технічну здійсненність концепції, проте здебільшого є пропрієтарними та закритими для модифікації, мають високу вартість впровадження та орієнтовані на специфічну логістичну інфраструктуру виробника. Це створює потребу в розробці відкритих, гнучких та доступних архітектурних рішень на базі вільнодоступних апаратних та програмних компонентів.

Ключовою інженерною проблемою у дроновій доставці є забезпечення точного автономного приземлення БПЛА на станцію видачі замовлень. Точність стандартної GPS-навігації у міському середовищі недостатня (від 10 до 30 м у разі багатопроменевого розповсюдження сигналу), що унеможливлює доставку без додаткових систем позиціонування. Тому актуальним є розробка комбінованої системи, яка поєднує GPS, оптичний сенсор потоку, ультразвукові датчики та комп'ютерний зір для розпізнавання спеціалізованих візуальних маркерів станції видачі.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз вимог до системи доставки безпілотним літальним апаратом

Створення автономної системи доставки вантажів за допомогою БПЛА із приземленням на спеціалізований пункт видачі ставить перед розробником сукупність взаємопов'язаних вимог, які охоплюють функціональні характеристики апарата, його взаємодію із наземною інфраструктурою, рівень автономності, безпеку польотів та відповідність нормативно-правовим актам. Формалізація цих вимог на початковому етапі проєктування дозволяє визначити критичні параметри системи та уникнути конфліктів між окремими підсистемами під час реалізації.

Функціональні вимоги визначаються основним призначенням системи — автоматизованою доставкою вантажу від складу до кінцевого одержувача. До них належать корисне навантаження апарата, дальність польоту з повним завантаженням, режими автономної навігації, точність приземлення на цільовий об'єкт та механізм передачі вантажу. Для сценарію доставки до пункту видачі, який має обмежені габарити приймальної комірки, точність приземлення набуває критичного значення: відхилення на кілька десятків сантиметрів може зробити операцію неможливою. Це обумовлює потребу в комбінованих методах позиціонування: супутникового, оптичного та інерціального. Оскільки стандартний супутниковий приймач забезпечує точність порядку кількох метрів, чого недостатньо для роботи з компактним місцем приземлення [1-3].

Технічно-експлуатаційні вимоги стосуються здатності апарата виконувати завдання у різноманітних умовах експлуатації. Сюди належать допустимі діапазони температур, стійкість до вітрового навантаження,

тривалість польоту, стабільність роботи систем зв’язку у міській забудові та запас енергії, достатній для повернення в разі відхилення від плану місії. Енергетичний баланс також має враховувати резерв для виконання процедур безпеки — повернення додому або аварійного приземлення в безпечну зону [10]. Час реакції бортових систем керування й обробки відеопотоку також впливає на стабільність утримання курсу під час підходу до місця призначення та посадки.

Окрему групу формують вимоги безпеки [11]. Система повинна забезпечувати уникнення зіткнень зі статичними та динамічними перешкодами, контроль геозон, обробку втрати каналу зв’язку, перехід у режим аварійного приземлення в разі критичних відмов та інформування диспетчерського центру про нештатні події. Для апарата, який оперує в межах населеного пункту, вимоги до резервування критичних компонентів як джерел живлення, каналів передачі телеметрії, систем визначення положення суттєво зростають порівняно із застосуванням у відкритій місцевості.

Інтеграційні вимоги випливають із того, що БПЛА є лише однією зі складових ширшої системи, до якої входять складська програма керування замовленнями, сервер та мобільний застосунок для кінцевого користувача [12]. Кожен компонент має чітко визначений інтерфейс взаємодії: склад передає на сервер інформацію про замовлення, сервер планує місію і передає її БПЛА, апарат регулярно повертає телеметрію, а мобільний застосунок інформує користувача про статус доставки.

Нормативно-правові вимоги в умовах України регулюються Повітряним кодексом, авіаційними правилами щодо польотів безпілотних повітряних суден та обмеженнями, пов’язаними з режимом воєнного стану. Вони визначають дозволені висоти, зони польотів, процедури отримання дозволів, обов’язкову ідентифікацію апарата та допустимі режими автономного функціонування.

1.2 Огляд існуючих рішень у сфері доставки безпілотними літальними апаратами

Комерційна доставка вантажів безпілотними літальними апаратами за останнє десятиліття пройшла шлях від експериментальних пілотів до регулярних операційних сервісів у кількох географічних регіонах. Аналіз цих рішень дозволяє виявити закономірності в архітектурі систем, форматах доставки, способах приземлення та обмеженнях, які залишаються невирішеними в наявних реалізаціях.

Найбільш відомою західною комерційною ініціативою є Prime Air компанії Amazon, яка отримала сертифікацію Part 135 від Федеральної авіаційної адміністрації США у 2020 році [14]. Поточна модель MK30 призначена для доставки пакунків вагою до 2.2 кілограма у радіусі 16–24 кілометрів від хабу та використовує технологію sense-and-avoid із лідаром для польотів за межами візуального контакту (BVLOS). Сервіс розгортається у конкретних географічних зонах — спочатку в Орегоні, згодом в Аризоні та Техасі і потребує від клієнта розміщення фізичного маркера на дворі для позначення точки приземлення [15]. Цей підхід демонструє концептуально схожу з нашою задачею модель: апарат повинен сісти у визначеній наземній точці, ідентифікованій візуальним маркером. Водночас Amazon оперує великими апаратами зі злітною масою близько 37,7 кг [16], що обмежує можливості польотів у щільній забудові та породжує суворі вимоги до інфраструктури.

Принципово інша архітектура реалізована компанією Wing, дочірньою структурою Alphabet. Wing застосовує легкі апарати, спроєктовані з розрахунку на руйнування при ударі, що знижує загрозу для людей та об'єктів на землі, та працює в умовах вітру до 10 метрів/секунду із поривами до 13. Доставка здійснюється шляхом зависання над цільовою зоною та опускання вантажу на тросі, тобто без приземлення апарата як такого. Такий

підхід виключає задачу точного посадкового позиціонування, проте обмежує сценарії використання: вантаж не може бути захищений від погоди чи стороннього доступу під час доставки, і не передбачає інтеграції зі стаціонарною приймальною інфраструктурою.

Компанія Zipline — один із світових лідерів у сегменті, який починав з фіксованокрилих апаратів для постачання медичних препаратів у віддалені регіони Африки. До 2023 року Zipline виконала понад 600 000 комерційних доставок, переважно медичних товарів, а у 2023 році представила нову платформу Platform 2 з гібридною архітектурою: основний апарат залишається на висоті, а пакунок опускається на тросі до невеликого керованого «дроїда», який забезпечує точне розміщення вантажу [17]. Це непрямо підтверджує ідею про корисність наземного компонента в системі точної доставки, хоча реалізація Zipline переносить інтелект на бортовий комп’ютер, а не на стаціонарний об'єкт на землі.

Найдовший досвід доставки в межах міст накопичила компанія Matternet, що з 2017 року експлуатує мережі для транспортування медичних зразків між лікарнями у Швейцарії в партнерстві зі Swiss Post та у США разом із UPS [18]. Безпілотник M2 має паспортну дальність понад 20 км, проте реальні маршрути обмежуються нормативними вимогами до польотів над населеними територіями. У 2022 році Matternet разом зі Stadspital Zürich запустила п'ятикілометровий маршрут BVLOS між лікарнями Triemli і Waid, який на момент запуску був найдовшим маршрутом доставки безпілотником над великим містом у світі [19]. Принципово важливим у підході Matternet є саме використання стаціонарних термінальних станцій, Matternet Station, на території лікарень: апарат не приземляється у довільному місці, а взаємодіє зі спеціалізованою інфраструктурою, що автоматизує передачу й прийняття вантажу. Така концепція є найближчим за духом аналогом задачі, що розглядається в цій кваліфікаційній роботі, хоча Matternet орієнтована на закриту інституційну логістику, а не на доставку до кінцевих споживачів.

В азіатському регіоні швидке зростання демонструє китайська платформа Meituan, яка розпочала проєкт безпілотної доставки у 2017 році й до кінця 2024 року виконала понад 450 000 комерційних доставок на 53 маршрутах у містах Шеньчжень, Пекін, Шанхай, Гуанчжоу та Нанкін, а її четверте покоління апаратів отримало від Адміністрації цивільної авіації КНР перший національний сертифікат на низьковисотну логістику [20]. Архітектура Meituan базується на мережі стаціонарних злітно-посадкових майданчиків (drone airports), розміщених у торгових та офісних центрах, з яких споживач забирає замовлення особисто. Це проміжна модель між класичною і пунктами видачі.

Окремим напрямом досліджень є інтеграція безпілотників із паркоматами та автоматизованими відділеннями. У роботах [20] запропоновано моделі мережі «пункт видачі–дрон», у яких пункт видачі виконує роль одночасно складу, точки зарядки і місця доставки. Така архітектура — locker–drone logistics system (LDLS) — забезпечує наскрізне поводження з пакунком, включно зі зберіганням, автономним завантаженням і вивантаженням, а живлення може здійснюватися від інтегрованих сонячних панелей. Подібну концепцію розвиває проєкт Smart Autonomous Drone Delivery Station, у якому посадковий майданчик з автонівелюванням, RFID-аутентифікацією та мобільним застосунком розрахований на встановлення на балконах або дахах житлових будинків [21]. Ці розробки переважно знаходяться на рівні концептів або вузьких пілотів і не утворюють завершеної комерційної екосистеми, однак саме до цього класу систем належить рішення, яке розробляється у цій роботі.

Окрему категорію складають апарати важкої комерційної логістики — Wingcopter, DJI FlyCart, EHang та подібні платформи, орієнтовані на перевезення вантажів масою від кількох до десятків кілограмів [22]. Вони використовують переважно класичне приземлення на наземних майданчиках, а сценарій доставки кінцевому клієнту реалізується через

посередника (наземний транспорт або логістичний термінал). Точне автономне приземлення у цих системах вирішене переважно через RTK GNSS і власні візуальні маркери виробника, що не є відкритою стандартизованою практикою.

Порівняння ключові характеристики розглянутих систем зведено у таблиці 1.1.

Таблиця 1.1 – Порівняння наявних систем безпілотної доставки

Система	Тип апарата	Маса вантажу	Дальність	Спосіб передачі вантажу	Цільовий ринок
Amazon Prime Air (MK30)	Гібрид VTOL	до 2,3 кг	12–24 км	Приземлення на маркер у дворі	Роздрібна доставка, США
Wing	Легкий мультикоптер	до 1,5 кг	до 19 км	Опускання на тросі у зависанні	Роздрібна доставка, передмістя
Zipline Platform 2	Фіксованокрилий + дроїд	до 3,6 кг	до 16 км	Опускання дроїда на тросі	Медицина, рітейл
Matternet M2	Мультикоптер	до 2 кг	до 20 км	Стаціонарна станція	Медична логістика B2B
Meituan (4-те покоління)	Мультикоптер	до 2,5 кг	до 5 км	Стаціонарні посадкові майданчики	Доставка їжі у мегаполісах
LDLS-концепти	Мультикоптер	1–5 кг	до 5 км	Автоматизований пункт видачі з посадкою	Дослідницький рівень

Аналіз показує, що жодне з масово впроваджених рішень не поєднує одночасно три ключові ознаки, важливі для цієї задачі: автономне точне приземлення безпосередньо на компактну приймальну інфраструктуру з фізично контрольованим доступом; інтеграцію з системою керування замовленнями складу; та доступну для повторного впровадження архітектуру на базі відкритих компонентів і протоколів.

1.3 Аналіз методів точного автономного приземлення БПЛА

Точність приземлення безпілотного літального апарата на цільову точку є одним із найкритичніших параметрів системи доставки, оскільки помилки у позиціонуванні на цьому етапі польоту не можуть бути скомпенсовані пізнішими діями. У сучасній практиці сформувалося кілька основних методів автономного точного приземлення, кожен із яких базується на різних фізичних принципах і має власні діапазони точності та обмежень застосування.

Класичний підхід полягає у використанні бортового GNSS-приймача, який відстежує сигнали супутникових сузір'їв GPS, GLONASS, Galileo і BeiDou та обчислює положення апарата у геодезичній системі координат. Точність такого приймача в автономному режимі становить від одного до трьох метрів за відсутності завад [23], що достатньо для маршрутної навігації, проте принципово недостатньо для приземлення на компактну ціль. Підвищити точність до сантиметрового рівня дозволяє кінематика реального часу (Real-Time Kinematic, RTK): нерухома базова станція, координати якої відомі, передає на апарат поправки до фазових вимірювань, що компенсують атмосферні та орбітальні похибки. У результаті відносна точність позиціонування досягає одиниць сантиметрів [23]. Недоліком RTK є необхідність постійного каналу зв'язку з базою, чутливість до затінення супутників міською забудовою та витрати на обладнання другого приймача. Принципово важливим є й те, що RTK дає геодезично-точне положення апарата, але не вирішує задачі прив'язки до фізичного об'єкта приземлення, якщо координати самої цілі відомі з гіршою точністю.

Метод інфрачервоного маяку базується на розміщенні на цілі активного інфрачервоного випромінювача та сприйнятті його сигналу спеціалізованою камерою, чутливою до ІЧ-діапазону. Найвідомішою комерційною реалізацією є система IR-LOCK, інтегрована у польотний

контролер ArduPilot як рідний режим точного приземлення. Згідно з документацією ArduPilot, у поєднанні з далекоміром (ультразвуковий або лідар) система забезпечує приземлення у радіусі приблизно 30 см від маяка, який рухається зі швидкістю до 1 м/с [24]. У дослідницьких роботах повідомляється про точність 5–30 см залежно від умов освітлення та діапазону [25]. ІЧ-маяк має суттєву перевагу — здатність працювати у темряві та при низькому контрасті ґрунту, проте йому притаманні й значні обмеження: невелика дальність впевненої детекції (15–20 м для серійних рішень), погіршення характеристик при яскравому сонячному світлі через насичення сенсора, та необхідність активного джерела живлення на наземній стороні [26]. Для системи, що проєктується, активне живлення пункту видачі є прийнятним, проте денна експлуатація під відкритим небом ставить під сумнів стабільність ІЧ-каналу.

Клас методів фідуціальні маркерів базується на нанесенні на ціль спеціально розробленого друкованого маркера із внутрішнім бінарним кодом, що дозволяє однозначно його ідентифікувати на зображенні з бортової камери. Найпоширенішими системами маркерів є ArUco, AprilTag, ARTag, STag, CCTag, WhyCode та похідні від них. У серії порівняльних досліджень [21, 22] показано, що ArUco і AprilTag демонструють близьку точність детекції, при цьому ArUco є обчислювально ефективнішим, а AprilTag має дещо вищу стійкість до рівня шуму у вхідних даних. У роботі [27] автори застосували ArUco-маркер розміром 56 × 56 см і досягли середнього відхилення приземлення 11 см при злеті з висоти до 30 м, тоді як стандартний GPS забезпечував похибку 1–3 м у тих самих умовах. У роботі [28] із використанням ArUco у симуляторі та реальному польоті отримано точність приземлення 19,75 мм та 21,2 мм відповідно — це свідчить про потенціал методу за умови коректної геометричної калібровки камери та якісного зображення. Розширення робочого діапазону вирішується за допомогою вкладених або багаторівневих маркерів [29]. Такий підхід

дозволяє уникнути «сліпої зони», коли при низькій висоті основний маркер вже не вміщується у кадрі. Принциповою перевагою фідуціальних маркерів є можливість безпосереднього обчислення відносного положення та орієнтації апарата щодо цілі — задача Perspective-n-Point, — що повністю усуває залежність від глобальних координат. До недоліків належать чутливість до освітлення, до забруднення поверхні маркера та до сильного бокового вітру, який може випадково винести апарат за межі поля зору камери. У польотному контролері ArduPilot інтеграція реалізується через MAVLink-повідомлення LANDING_TARGET, які передаються бортовим комп'ютером із частотою не нижче 1 Гц [30].

Сенсори оптичного потоку (як PMW3901 у складі Matek 3901-L0X або PixArt PAA3905E1) працюють за принципом аналогічним до оптичної комп'ютерної миші: вони фіксують переміщення характерних рис ґрунту між послідовними кадрами і обчислюють лінійну швидкість апарата щодо поверхні. У поєднанні з лазерним або ToF-далекоміром цей сигнал стає метрично калібрований. Робочий діапазон висот серійних сенсорів сягає 2–5 м для бюджетних рішень і 10–15 м для професійних [31]. Оптичний потік сам собою не вирішує задачі приземлення на конкретну ціль, проте є важливим допоміжним сенсором у близькій до землі зоні, де відбувається втрата сигналу GNSS через відбиття та перевідбиття, а пориви повітря від гвинтів спричиняють збурення положення. Поєднання оптичного потоку, ІЧ-далекоміра та EKF-фільтра польотного контролера є стандартною практикою для стабільного зависання на висоті 0,3–2 м перед фінальною фазою посадки [32].

Жоден із наведених методів окремо не забезпечує надійного приземлення у всіх ділянках траєкторії, тому в реальних системах застосовується каскадна архітектура. На крейсерській висоті використовується GNSS, на проміжній — детекція великого фідуціального маркера, у близькій фазі — малий внутрішній маркер з оптичним потоком,

який стабілізує бокове знесення. У дослідницьких роботах [33] запропоновано додаткові варіанти, зокрема комбінування візуальних і ІЧ-маркерів на одній цілі: ІЧ-канал використовується вночі або при тумані, а ArUco — у звичайних денних умовах. Окремо досліджуються підходи на основі глибокого навчання, де нейронна мережа детектує не штучний маркер, а сам об'єкт приземлення безпосередньо за його зовнішнім виглядом [34].

Для зведення характеристик методів у таблиці 1.2 наведено порівняння їх ключових параметрів у застосуванні до задачі точного приземлення на компактну приймальну інфраструктуру.

Таблиця 1.2 – Порівняння методів точного приземлення БПЛА

Метод	Типова точність	Робоча дальність	Залежність від освітлення	Потреба в інфраструктурі
GNSS	1–3 м	Без обмежень	Відсутня	Відсутня
GNSS RTK	1–3 см	Потрібен зв'язок з базою	Відсутня	Базова станція, канал зв'язку
ІЧ-маяк (IR-LOCK)	5–30 см	До 15–20 м	Низька	Активний маяк із живленням
ArUco / AprilTag	5–20 см	До 30 м (від розміру маркера)	Висока	Друкований маркер
Оптичний потік + лідар	5–15 см	0,3–10 м	Помірна (потрібна текстура)	Відсутня

З огляду на специфіку задачі — приземлення на стаціонарний приймальний пункт видачі у міському середовищі за денних умов, з вимогою низької вартості та відкритих компонентів — найбільш раціональним є гібридний підхід, ядром якого є детекція ArUco-маркера бортовим комп'ютером із передачею координат у польотний контролер через MAVLink LANDING_TARGET. ArUco забезпечує сантиметрову точність, не

потребує додаткового активного обладнання на цілі, працює зі стандартним RGB-сенсором та підтримується бібліотекою OpenCV. Допоміжно застосовуються оптичний потік і лазерний далекомір (модуль Matek 3901-L0X), що стабілізують положення на малих висотах, та GNSS-приймач для підльоту з крейсерської висоти.

1.4 Формулювання вимог до розроблюваної системи

На основі аналізу, проведеного у підрозділах 1.1–1.3, сформовано перелік вимог до системи автономної доставки безпілотним літальним апаратом на приймальний пункт видачі. Вимоги розділено за категоріями, що відповідають архітектурним рівням системи.

Функціональні вимоги:

– апарат повинен виконувати автономну місію за заздалегідь сформованим планом польоту від точки старту над складом до цільового пункту видачі;

– точність приземлення на маркер пункту видачі має становити не більше 20 см від центру приймальної комірки;

– система повинна забезпечувати автоматичну ідентифікацію цільового пункту видачі серед інших однотипних об'єктів за унікальним кодом маркера;

– передбачено двосторонній обмін статусом місії між БПЛА, сервером і мобільним застосунком отримувача в режимі реального часу;

– у разі втрати маркера у фінальній фазі підльоту виконується процедура повторної спроби з підйомом на проміжну висоту.

Технічно-експлуатаційні вимоги:

– робоча дальність — не менше 3 км в одному напрямку із запасом енергії на повернення;

– робочий діапазон температур — від −5 до +35 °C;

– стійкість до бічного вітру до 8 м/с у режимі підльоту і до 5 м/с у режимі точного приземлення;

– корисне навантаження — до 1 кг;

– загальний час від отримання замовлення сервером до завершення доставки — до 20 хв.

Архітектурні вимоги:

– польотний контролер реалізований на відкритій платформі, сумісній із прошивкою ArduPilot;

– бортовий комп'ютер виконує детекцію маркера засобами OpenCV і взаємодіє з контролером через MAVLink-повідомлення LANDING_TARGET;

– канал зв'язку з сервером — LTE, з резервним телеметричним каналом для аварійних ситуацій;

– серверна частина побудована за клієнт-серверною архітектурою з REST-інтерфейсом для складської системи та мобільного застосунку;

– програмне забезпечення розроблено з використанням мов і бібліотек з відкритою ліцензією, що допускає вільне використання компонентів.

Вимоги безпеки:

– реалізація відмовостійких режимів Return-to-Launch (RTL) та Land у разі втрати зв'язку, критичного розряду акумулятора або відмови ключових сенсорів;

– визначення геозон, у межах яких дозволено політ, із автоматичною блокуванням виходу за їх межі;

– реєстрація телеметричних даних польоту для подальшого аналізу нештатних ситуацій;

– фізичний доступ до вмісту пункту видачі лише після автентифікації отримувача через мобільний застосунок.

Сформовані вимоги становлять основу технічного завдання на кваліфікаційну роботу (Додаток А) і визначають вибір апаратного та програмного забезпечення, що детально обґрунтовано у розділі 2

РОЗДІЛ 2 ПРОЄКТНА ЧАСТИНА

2.1 Розробка схеми електричної структурної системи

Перед детальним обґрунтуванням компоненті представлено узагальнену структуру системи автономної доставки безпілотним літальним апаратом, яка визначає склад елементів та характер їх взаємодії. Розроблена структурна схема відображає систему на двох рівнях деталізації: системному, що показує зовнішні підсистеми і логічні потоки даних між ними, та бортовому, що розкриває склад електроніки безпілотного літального апарата.

Системний рівень охоплює чотири взаємодіючі компоненти: серверну частину, що виконує функцію координатора замовлень; складську підсистему, що готує вантаж до видачі; БПЛА, що здійснює фізичне переміщення вантажу; та користувача з мобільним застосунком, який ініціює замовлення і отримує посилку. Сервер є центральним вузлом обміну: він приймає від користувача замовлення з прив'язкою до конкретного приймального пункту видачі, надсилає на склад команду підготувати вантаж, передає на БПЛА координати точки доставки та команду на старт місії, а потім транслює користувачу інформацію про статус доставки в режимі реального часу. БПЛА у відповідь повертає телеметрію і статус місії, склад — підтвердження готовності замовлення.

Узагальнена структурна схема системного рівня наведена на рис. 2.1.

Серверна частина та мобільний застосунок як окремі програмні продукти не є предметом проєктування у цій кваліфікаційній роботі. Вони розглядаються як зовнішні системи зі стабільним інтерфейсом обміну повідомленнями, з яким взаємодіє бортовий комп'ютер БПЛА через канал

LTE. Основну увагу зосереджено на бортовій електроніці апарата та реалізації автономного приземлення на приймальний пункт видачі, оскільки саме ця частина становить технічний внесок роботи.

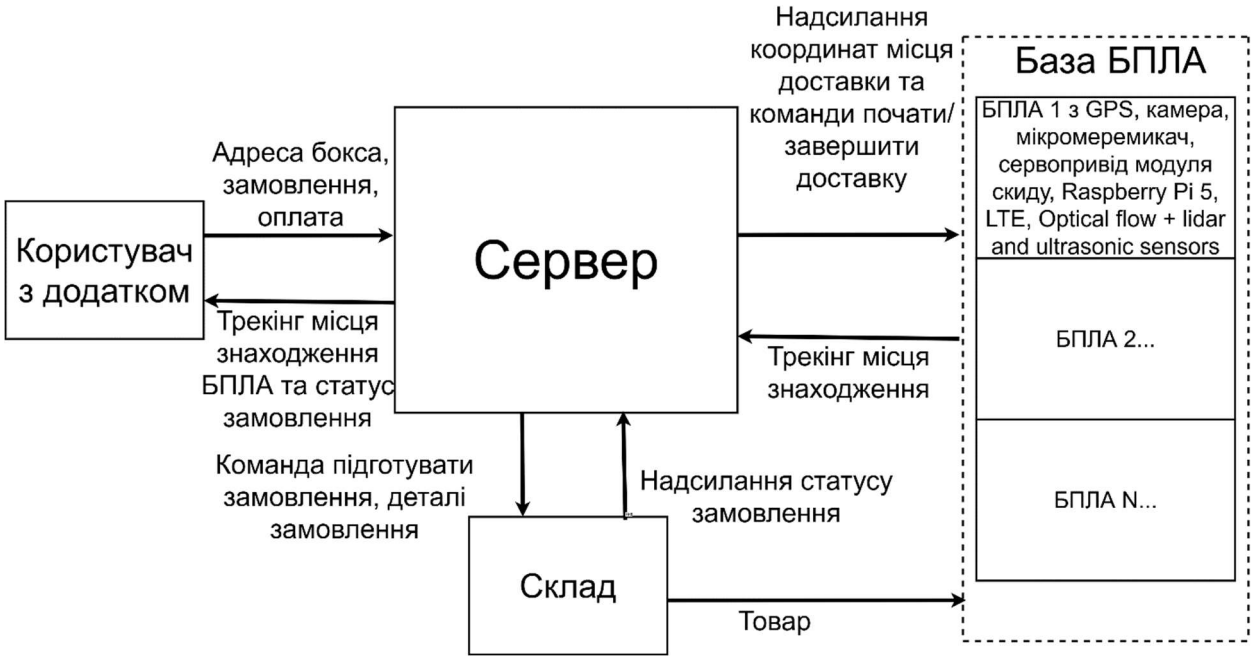

Рисунок 2.1 – Схема структурна системи доставки на рівні взаємодії підсистем

Бортовий рівень розкриває внутрішню структуру електроніки безпілотного літального апарата. Логічним центром бортової системи виступає бортовий комп'ютер Raspberry Pi 5, який виконує функції керування місією: підтримує зв'язок із сервером, обробляє відеопотік з камери для детекції ArUco-маркера та формує команди керування для польотного контролера. До Raspberry Pi через інтерфейс CSI підключено камеру, через USB — LTE-модем для зв'язку з сервером, а через GPIO — мікроперемикач механізму утримання вантажу, що дозволяє контролювати наявність вантажу на борту під час польоту.

Польотний контролер виконує функції тактичного рівня: стабілізує апарат у просторі, обробляє дані навігаційних сенсорів та керує виконавчими механізмами. До нього підключено GNSS-модуль за інтерфейсом UART для глобального позиціонування, модуль оптичного потоку з вбудованим

лазерним далекоміром за UART для стабілізації на малій висоті, чотири ультразвукові сенсори за інтерфейсом I2C, розташовані по чотирьох сторонах апарата для виявлення перешкод у горизонтальній площині, а також безколекторні двигуни через електронні регулятори швидкості (ESC) за цифровим протоколом Bi-direction DShot. Сервопривід механізму скидання вантажу керується ШІМ-сигналом з допоміжного виходу польотного контролера.

Raspberry Pi обмінюється з польотним контролером даними за протоколом MAVLink: отримує телеметрію стану апарата та передає у польотний контролер координати виявленого маркера у вигляді повідомлень LANDING_TARGET [30]. Такий поділ функцій між бортовим комп'ютером і польотним контролером відповідає рекомендованій архітектурі ArduPilot для систем, що поєднують жорсткий реальний час керування з ресурсомісткими завданнями обробки даних [35]: польотний контролер працює у реальному часі з жорсткими часовими обмеженнями і не пристосований до обробки відеопотоку, тоді як Raspberry Pi має достатню обчислювальну потужність для виконання детекції маркера засобами OpenCV, але не може забезпечити гарантовану реакцію на події керування двигунами.

Живлення бортової електроніки реалізовано від основного літій-іоного акумулятора БПЛА через знижувальний DC/DC-перетворювач, що формує стабілізовану напругу +5 В для Raspberry Pi, польотного контролера та сенсорів. Електронні регулятори швидкості моторів живляться напряму від акумулятора напругою +24 В.

Бортова структурна схема наведена на рис. 2.2.

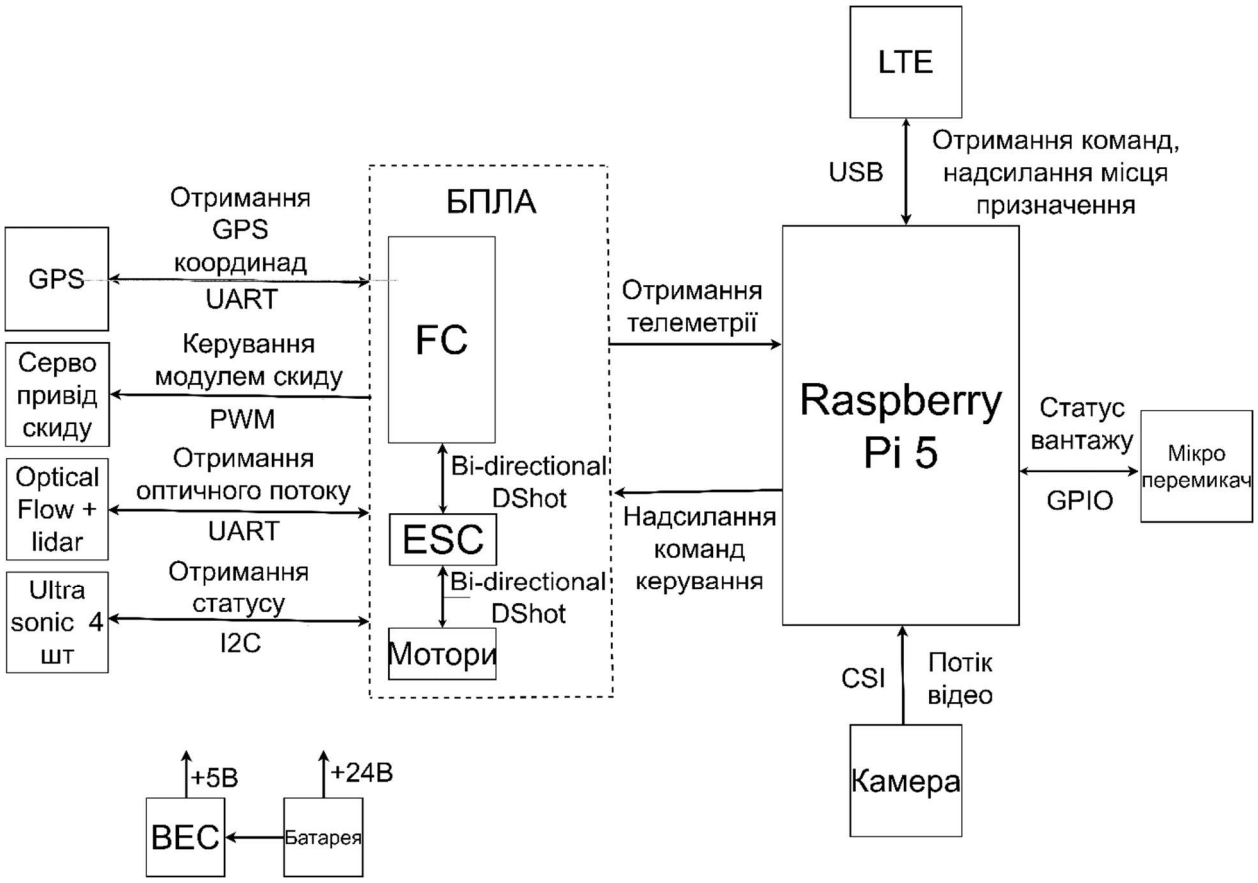

Рисунок 2.2 – Схема структурна бортової електроніки безпілотного літального апарата

Запропонована архітектура має кілька принципових властивостей. По-перше, розподіл функцій між Raspberry Pi і польотним контролером дозволяє використовувати готову прошивку польотного контролера ArduPilot без модифікацій її ядра, обмежуючись налаштуванням параметрів. Уся логіка комп'ютерного зору винесена на бортовий комп'ютер, де вона реалізується мовою Python зі стандартними бібліотеками. По-друге, взаємодія між підсистемами використовує стандартизований протокол MAVLink, що забезпечує сумісність із наявною інфраструктурою керування БПЛА. По-третє, всі апаратні компоненти бортового рівня є серійно доступними і не потребують власної розробки.

2.2 Обґрунтування вибору апаратного забезпечення

Реалізація системи автономної доставки вимагає підбору апаратних компонентів, які забезпечують виконання функцій навігації, точного приземлення, сумісності, утримання вантажу, протоколів зв'язку із зовнішніми та внутрішніми складовими системи. У цьому підрозділі обґрунтовано вибір кожного компонента бортової електроніки з урахуванням вимог, сформульованих у розділі 1, та архітектури системи, представленої у підрозділі 2.1.

2.2.1 GNSS-модуль u-blox NEO-M9N

GNSS-модуль виконує функцію визначення глобальних координат апарата у геодезичній системі WGS-84 та передає їх польотному контролеру для виконання запрограмованої місії. Для системи доставки точність GNSS визначає, наскільки близько до приймального пункту видачі БПЛА вийде на фінальну фазу зниження — фазу, в якій далі підключається ArUco-детекція. Чим точніше попереднє позиціонування, тим менший простір потрібен для пошуку маркера камерою.

Обрано готовий модуль на базі чіпа u-blox NEO-M9N, що поставляється у комплекті з інтерфейсною платою та активною керамічною антеною розмірами 25×25 мм [36]. Модуль підтримує одночасний прийом сигналів шести супутникових систем — GPS, GLONASS, Galileo, BeiDou, QZSS та SBAS, що забезпечує стійкість позиціонування навіть у складних умовах прийому, таких як міська забудова або зони з частковим затіненням небосхилу. Частота оновлення координат сягає 25 Гц (рис. 2.3).

З технічного боку модуль має такі характеристики: робоче живлення 3,0–5,0 В; вбудована EEPROM для збереження налаштувань після вимкнення живлення; резервний акумулятор, який зберігає альманах супутників і поточний час між сеансами роботи; світлодіодний індикатор фіксації положення; вага модуля близько 16 грамів [36].

Рисунок 2.3 - Модуль GPS на U-blox NEO-M9N з інтерфейсною платою та керамічною антеною

Інтерфейс зв'язку з польотним контролером — UART зі стандартною швидкістю передачі 9600 бод за замовчуванням, з можливістю програмного перенастроювання на вищі швидкості (38400, 57600, 115200 бод) при потребі [36]. Прошивка ArduPilot автоматично розпізнає u-blox-сумісні приймачі за стандартним протоколом UBX і виконує автоконфігурування швидкості та активних повідомлень, тому додаткового конфігурування з боку розробника не потребує [37].

Альтернативами розглядалися приймачі попередніх поколінь — NEO-M8N (підтримує одночасний прийом лише двох сузір'їв, що знижує стійкість у міській забудові) і двочастотні модулі ZED-F9P (точність 1–2 см з RTK-поправками, але вартість у десятки разів вища та потребує постійного зв'язку з базовою станцією). Модель NEO-M9N обрано як оптимальний компроміс між точністю, ціною та простотою інтеграції.

2.2.2 Модуль оптичного потоку та далекоміра Matek 3901-L0X

На малих висотах (до кількох метрів від поверхні) точність GNSS суттєво погіршується через відбиття сигналів від оточуючих об'єктів — стін, паркану пункту видачі, дерев. Для стабілізації бокового зносу апарата у близькій до землі зоні використовується сенсор оптичного потоку, який вимірює відносний рух апарата щодо поверхні шляхом порівняння послідовних кадрів із низькороздільної камери, спрямованої вертикально

вниз. Доповненням до нього є лазерний далекомір (ToF), що забезпечує точне вимірювання висоти над поверхнею.

Обрано модуль Matek 3901-L0X, який поєднує два сенсори в одному корпусі: оптичний потік на базі мікросхеми PMW3901 та ToF-далекомір VL53L0X [38]. Модуль підключається до польотного контролера за єдиним інтерфейсом UART, що спрощує інтеграцію — обидва сенсори обмінюються даними з контролером через спільну шину (рис. 2.4).

Рисунок 2.4 – Модуль Matek 3901-L0X

Робочий діапазон оптичного потоку — від 80 мм над поверхнею і вище, ToF-далекомір вимірює відстань від 30 мм до 2 м, що повністю покриває фазу фінального приземлення на пункт видачі [39]. Прошивка ArduPilot з версії 4.0 має штатну підтримку цього модуля без потреби у додаткових драйверах: достатньо встановити параметри FLOW_TYPE = 6 для оптичного потоку та RNGFND1_TYPE = 16 для далекоміра.

Такий вибір продиктований кількома міркуваннями. По-перше, модуль є інтегрованим рішенням, тобто розробник позбавляється задачі окремого монтажу і калібрування двох сенсорів. По-друге, обидва сенсори вже підтримуються прошивкою ArduPilot. По-третє, наявний практичний досвід інтеграції цього модуля з польотним контролером сімейства Matek H743 спрощує налаштування системи.

2.2.3 Камера для детекції ArUco-маркера

Камера є ключовим сенсором системи точного приземлення: вона забезпечує детекцію друкованого ArUco-маркера на поверхні пункту видачі. Від характеристик камери залежать дальність впевненої детекції, точність

обчислення відносної позиції апарата та надійність роботи в умовах змінного освітлення.

З-поміж камер, сумісних із Raspberry Pi 5, обрано Raspberry Pi Camera Module 3. Модуль побудований на сенсорі Sony IMX708 із максимальною роздільною здатністю 11,9 мегапікселів та підтримує запис відео у роздільній здатності 1080p при 50 кадрах за секунду [40]. Підключення здійснюється стрічковим шлейфом до спеціалізованого CSI-порту Raspberry Pi (рис. 2.5)

Рисунок 2.5 – Модуль камери Raspberry Pi Camera Module 3

Модуль має автоматичне регулювання експозиції та балансу білого, що важливо для роботи в зовнішньому середовищі зі змінним освітленням. Для задачі ArUco-детекції достатньо роздільної здатності 1280×720 при 30 кадрах за секунду. Кадр такої роздільної здатності обробляється функцією cv2.aruco.detectMarkers() бібліотеки OpenCV приблизно за 10–15 мс на Raspberry Pi 5, що дає реальну робочу частоту понад 20 Гц.

Перед використанням камеру необхідно один раз геометрично відкалібрувати — визначити матрицю внутрішніх параметрів та коефіцієнти дисторсії об'єктива за допомогою друкованої шахівниці або ChArUco-плати [41]. Результати зберігаються у файлі, який підвантажується бортовою програмою при старті.

2.2.4 Ультразвукові сенсори

Для виявлення перешкод у горизонтальній площині використовуються чотири ультразвукові сенсори, розташовані по чотирьох сторонах апарата. Сенсори вимірюють відстань до найближчої перешкоди шляхом випромінювання звукової хвилі ультразвукового діапазону та фіксації часу повернення відбитого сигналу.

Обрано сенсори HC-SR04 з інтерфейсним модулем-перетворювачем GY-US42 на шині I2C [42]. Базовий сенсор HC-SR04 за паспортом має діапазон вимірювання від 2 см до 4 м, кут діаграми спрямованості близько 15° та точність порядку 3 мм. Підключення усіх чотирьох сенсорів реалізовано через шину I2C з різними адресами кожного модуля, що дозволяє опитувати їх польотним контролером без надмірних витрат GPIO-ліній (рис. 2.6)

Рисунок 2.6 – Ультразвуковий сенсор GY-US42

Обрання саме ультразвукових сенсорів обґрунтоване їх стійкістю до змінного освітлення (на відміну від оптичних рішень) та низькою вартістю порівняно з лазерними далекомірами. Обмеженням є робота тільки на коротких дистанціях (до 4 м) та чутливість до м'яких або похилих поверхонь, які можуть розсіювати звукову хвилю. Для задачі у безпосередній близькості до приймального пункту видачі цих характеристик достатньо. Прошивка ArduPilot підтримує до 10 далекомірів одночасно з можливістю призначення кожному з них напрямку орієнтації [43].

2.2.5 Польотний контролер, бортовий комп'ютер та комплектація БПЛА

Обрано польотний контролер Matek H743-WING на базі мікроконтролера STM32H743 з тактовою частотою 480 МГц. Контролер має вбудовані інерціальний модуль ICM-42688P, барометр DPS310 і компас QMC5883L, а також 9 UART-портів, шину I2C, виходи DShot для керування ESC та допоміжні PWM-виходи [44]. Такої кількості інтерфейсів достатньо для одночасного підключення усіх сенсорів системи без додаткових мультиплексорів.

Контролер сумісний із прошивкою ArduPilot Copter, що є основою архітектурного рішення цієї роботи. Усі необхідні функції підтримуються прошивкою штатно і налаштовуються параметрами без потреби модифікації коду. Наявний практичний досвід роботи з цим контролером був важливим фактором у виборі моделі.

Щодо бортового комп’ютреа, було обрано Raspberry Pi 5 з чотириядерним процесором Arm Cortex-A76 на частоті 2,4 ГГц та 8 ГБ оперативної пам'яті [45]. Така продуктивність забезпечує виконання детекції ArUco-маркера в реальному часі засобами OpenCV з частотою понад 20 Гц, паралельно зі зв'язком із сервером через LTE-модем та обміном телеметрією з польотним контролером.

Raspberry Pi 5 має спеціалізований CSI-порт для підключення камери, USB-порти для підключення LTE-модема, та апаратні UART-інтерфейси для обміну з польотним контролером. Лінії GPIO використовуються для зчитування стану мікроперемикача механізму утримання вантажу.

Бортова електроніка інтегрується у готову платформу класу 7-дюймового квадрокоптера, що включає карбонову раму, чотири безколекторні мотори з відповідними електронними регуляторами швидкості, літій-іонний акумулятор конфігурації 6S2P та USB LTE-модем для зв'язку з сервером. Проєктування рами, силової частини та системи живлення не входить до завдань кваліфікаційної роботи, компоненти підбираються з готових серійних рішень, сумісних із обраним польотним контролером.

2.2.6 Механізм утримання та скидання вантажу

Механізм утримання та скидання вантажу складається з двох компонентів: сервопривода, що утримує і відпускає вантаж, та мікроперемикача, що сигналізує про факт фізичної наявності вантажу під апаратом.

Обрано сервопривід MG996R з металевими шестернями, моментом утримання 9–11 кг·см та робочою напругою 4,8–7,2 В. Сервопривід керується ШІМ-сигналом стандарту 50 Гц з допоміжного виходу польотного контролера: довжина імпульсу 1000 мкс відповідає закритому положенню защіпки, 2000 мкс — відкритому. Запас моменту утримання вибраний з розрахунку на корисне навантаження до 1 кг із коефіцієнтом запасу, який враховує динамічні навантаження під час маневрів і поривів вітру.

Для контролю наявності вантажу використовується мініатюрний мікроперемикач у конфігурації «нормально розімкнутий» (NO). Перемикач механічно з'єднаний із посадковим місцем вантажу таким чином, що вага вантажу замикає його контакти. Сигнал перемикача зчитується одним з GPIO-входів Raspberry Pi 5 з програмною фільтрацією короткочасних спрацювань.

Сумісна робота двох компонентів забезпечує підтвердження факту скидання вантажу: після команди на відкриття сервопривода Raspberry Pi протягом заданого інтервалу очікує переходу сигналу мікроперемикача з «замкнуто» в «розімкнуто», що інтерпретується як успішне відділення вантажу. Якщо такий перехід не зафіксовано, формується подія помилки, яка передається серверу і ініціює повторну спробу скидання.

2.3 Розробка схеми електричної принципової

На основі обґрунтованого у підрозділі 2.2 переліку компонентів та зв'язків, представлених у схемі структурній, розроблено схему електричну принципову системи у відповідності до вимог ДСТУ 2.701 та ДСТУ 2.702. Схему виконано у системі автоматизованого проєктування Altium Designer

26.3.0 на форматі А1 (841×594 мм). Позиційні позначення елементів виконано літерами латинської абетки за ДСТУ 2.710. Головні компоненети схеми електричної принципової розроблюваної системи наведено на рис. 2.7.

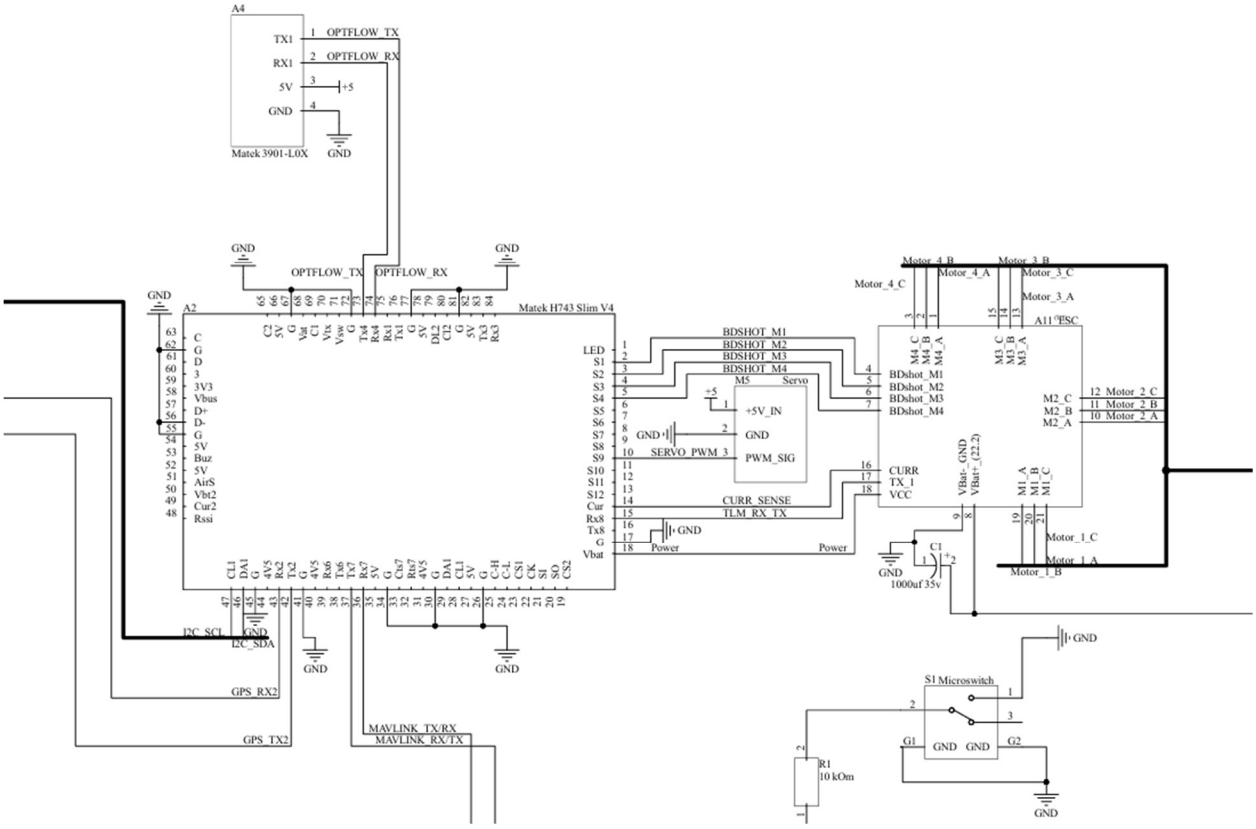

Рисунок 2.7 – Схема електрична принципова системи автономної доставки БПЛА

Центральним обчислювальним вузлом системи є бортовий комп'ютер Raspberry Pi 5 (A1), розташований у центрі схеми. До нього через CSI-інтерфейс підключено камеру Pi Camera Module 3 (A5), через USB — LTE-модем (A10), а через GPIO та підтягувальний резистор R1 — мікроперемикач S1 для контролю наявності вантажу. Зв'язок з польотним контролером Matek H743 (A2) реалізовано двома лініями UART за протоколом MAVLink (MAVLINK_TX, MAVLINK_RX).

До польотного контролера підключено навігаційні сенсори: GNSS-модуль u-blox NEO-M9N (A3) і модуль оптичного потоку Matek 3901-L0X (A4) — кожен через окремий UART-інтерфейс. Чотири ультразвукові сенсори GY-US42v2 (A6–A9), розташовані по чотирьох сторонах апарата, об'єднано

спільною шиною I2C. Унікальні I2C-адреси кожного сенсора задаються програмно при ініціалізації.

Електронний регулятор обертів формату 4-в-1 (A11) керує чотирма безколекторними моторами M1–M4 через трифазні виходи. Зв'язок з польотним контролером реалізовано чотирма двонаправленими лініями BDSHOT_M1–BDSHOT_M4 за протоколом Bi-directional DShot. Додатково ESC передає аналоговий сигнал струму споживання (CURR_SENSE) та цифрову телеметрію по лініях TLM_TX/TLM_RX. Сервопривід механізму скидання MG996R (M5) керується ШІМ-сигналом SERVO_PWM з допоміжного виходу польотного контролера.

Блок живлення побудовано наступним чином: літій-іонний акумулятор GB1 конфігурації 6S2P (24 В) через захисний запобіжник F1 (30 А) підключений до силового роз'єму XT60 (XS1). Силова лінія +24V розгалужується на ESC напряму і на знижувальний перетворювач BEC (A12 Matek 5V 9A), який формує стабілізовану лінію +5V для живлення польотного контролера, бортового комп'ютера, сенсорів і сервопривода. Електролітичний конденсатор C1 виконує функцію фільтрації пульсацій поблизу ESC та виходу BEC.

Розроблену схему перевірено вбудованим засобом електричних правил (ERC), критичних помилок не виявлено. Повний перелік елементів схеми наведено у Додатку В.

2.4 Обґрунтування вибору програмного забезпечення

Програмне забезпечення системи розподілене між двома обчислювальними вузлами — польотним контролером і бортовим комп'ютером, кожен з яких виконує функції, що відповідають його продуктивності та вимогам реального часу.

На польотному контролері Matek H743 виконується прошивка ArduPilot Copter версії 4.5. ArduPilot це одна з найпоширеніших відкритих прошивок для

безпілотних літальних апаратів, що активно розвивається з 2010 року спільнотою ArduPilot.org [46]. Прошивка реалізує повний набір функцій, необхідних для автономного польоту: стабілізацію апарата у просторі за даними інерціального вимірювального модуля, обробку даних GNSS, оптичного потоку та далекомірів, виконання маршрутних місій із заздалегідь заданими точками, режими повернення на базу (Return-to-Launch) та аварійного приземлення (Land), а також підтримку точного приземлення через MAVLink-повідомлення LANDING_TARGET [30]. Усі необхідні функції налаштовуються через параметри без модифікації вихідного коду. Конфігурування параметрів виконується через наземну станцію Mission Planner, що є офіційним програмним забезпечення проєкту ArduPilot для Windows, яке надає графічний інтерфейс для зміни параметрів, моніторингу телеметрії та планування місій [47].

Вибір ArduPilot обґрунтований кількома думками. По-перше, прошивка є відкритою (ліцензія GPL v3) і вільнодоступною, що відповідає вимогам, сформульованим у розділі 1. По-друге, ArduPilot має повну підтримку всіх обраних апаратних компонентів. По-третє, прошивка підтримує MAVLink — стандартизований протокол обміну з безпілотниками, що забезпечує сумісність бортового комп'ютера з польотним контролером без необхідності розробки власного інтерфейсу.

На бортовому комп'ютері Raspberry Pi 5 виконується операційна система Raspberry Pi OS (Debian-based) у версії на основі Linux ядра 6.16. Вибір саме цього дистрибутиву зумовлений офіційною підтримкою з боку виробника апаратної платформи, наявністю усіх необхідних драйверів для CSI-камери та GPIO, а також багатою екосистемою готових пакетів для задач комп'ютерного зору і мережевого зв'язку.

Прикладна частина програмного забезпечення бортового комп'ютера реалізована мовою Python 3.11. Вибір Python обумовлений швидкістю розробки, читабельністю коду, наявністю готових бібліотек для усіх задач системи та підтримкою всіма основними операційними системами. Хоча

Python поступається C++ за продуктивністю, для задачі детекції ArUco-маркера з частотою 20–30 Гц на Raspberry Pi 5 його продуктивності цілком достатньо, оскільки обчислювально критичні операції виконуються у скомпільованих C-розширеннях бібліотек.

Для розв'язання прикладних задач використовуються наступні бібліотеки:

OpenCV 4.13 — бібліотека комп'ютерного зору, що містить готову реалізацію детекції ArUco-маркерів у модулі cv2.aruco. Функція cv2.aruco.detectMarkers() забезпечує виявлення маркерів у кадрі, а cv2.solvePnP() обчислює положення та орієнтацію камери відносно маркера на основі його відомих геометричних розмірів [48].

pymavlink — Python-реалізація протоколу MAVLink, що забезпечує обмін повідомленнями з польотним контролером через UART. Бібліотека використовується для надсилання повідомлень LANDING_TARGET, отримання телеметрії та керування режимами польоту [49].

picamera2 — офіційна бібліотека для роботи з Raspberry Pi Camera Module 3, що забезпечує отримання відеопотоку з CSI-інтерфейсу з мінімальною затримкою [50].

Зв'язок з сервером реалізується через стандартну мережеву підсистему Linux: USB LTE-модем розпізнається ядром як мережевий інтерфейс після налаштування пакету ModemManager, після чого Python-скрипт здійснює HTTP-обмін з сервером через стандартну бібліотеку requests. Аутентифікація користувача мікроперемикача механізму утримання вантажу зчитується через бібліотеку gpiozero, що надає високорівневий доступ до GPIO-ліній Raspberry Pi.

Логіка обміну між бортовим комп'ютером і польотним контролером базується на протоколі MAVLink 2.0 [50]. Ключовим повідомленням для системи точного приземлення є LANDING_TARGET, що містить кут азимуту та елевації цілі відносно камери, відстань до цілі та її розміри у кадрі. Це повідомлення передається бортовим комп'ютером з частотою не нижче 1 Гц

під час фази підльоту, а польотний контролер інтегрує ці дані у свій розширений фільтр Калмана (EKF) для коригування траєкторії посадки.

2.5 Розробка діаграми взаємодії компонентів

Для відображення динаміки взаємодії компонентів системи у часі застосовано діаграму послідовності у нотації UML 2.0. На відміну від структурної схеми, діаграма послідовності описує порядок обміну повідомленнями між підсистемами протягом виконання сценарію доставки.

Діаграма побудована для основного сценарію автономної доставки — від моменту фізичного приєднання вантажу до БПЛА на складі до завершення місії та повернення апарата на базу. На діаграмі відображено п'ять акторів: склад товарів, серверну частину, бортовий комп'ютер Raspberry Pi 5, польотний контролер та механізм скиду вантажу. Користувач, що ініціює замовлення через мобільний застосунок, у діаграмі явно не показаний — припускається, що замовлення вже сформоване і відображене на сервері, а вантаж за цим замовленням підготовлено на складі.

Діаграма послідовності для сценарію доставки наведена на рис. 2.8.

Сценарій починається з підключення вантажу до БПЛА на складі (повідомлення 1). Це фізичне підключення, яке здійснюється автоматизованою системою завантаження. Сервер у відповідь надсилає бортовому комп'ютеру координати цільового пункту видачі разом із командою на старт місії (2).

Бортовий комп'ютер транслює отримане завдання у польотний контролер у вигляді послідовності команд: встановлення режиму автономного польоту, завантаження маршруту з проміжними точками та активація зльоту (3). Польотний контролер виконує політ за GPS-координатами і протягом усієї фази періодично надсилає телеметрію (4) на бортовий комп'ютер. Бортовий комп'ютер у свою чергу транслює поточні координати на сервер (5), що дозволяє користувачу відстежувати рух апарата у мобільному застосунку у

режимі реального часу. Цей обмін відбувається з частотою порядку 4 Гц і триває протягом усієї маршрутної фази польоту.

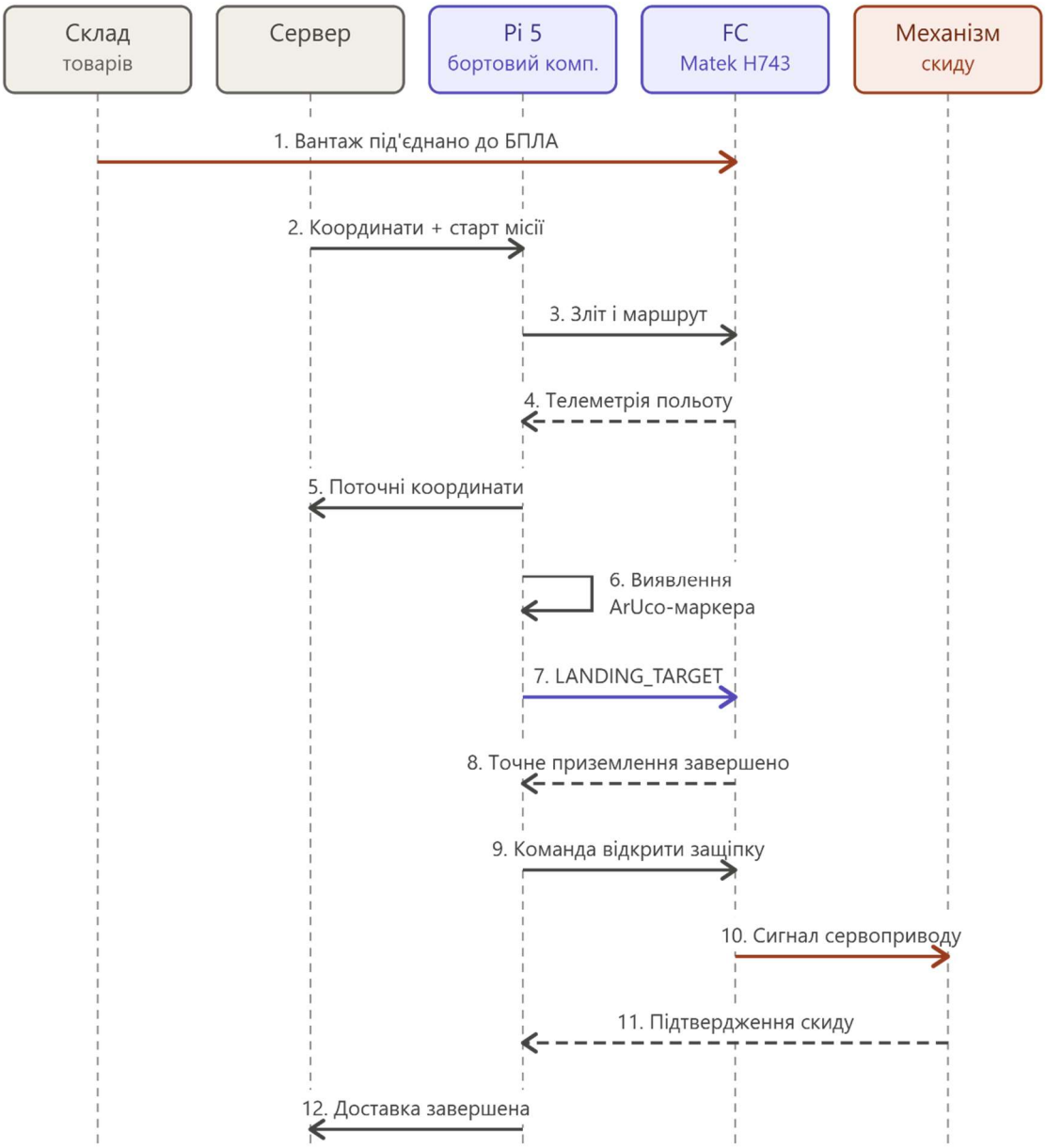

Рисунок 2.8 – Діаграма послідовності для сценарію автономної доставки

Коли апарат досягає зони над цільовим пунктом видачі, бортовий комп'ютер активує підсистему комп'ютерного зору і починає обробляти кадри з камери. Виявлення ArUco-маркера (6) є внутрішньою операцією Raspberry Pi 5, представленою на діаграмі як самопосилання — це обчислення відбувається локально на бортовому комп'ютері засобами OpenCV і не передбачає обміну з

іншими компонентами. Після успішного виявлення маркера обчислюються відносні координати апарата щодо цілі, які упаковуються у повідомлення LANDING_TARGET та передаються польотному контролеру (7) з частотою не нижче 2 Гц. Польотний контролер коригує траєкторію спуску за отриманими координатами і завершує приземлення, після чого надсилає бортовому комп'ютеру підтвердження завершення фази (8).

Після приземлення бортовий комп'ютер активує механізм скиду вантажу. Команда на відкриття защіпки передається польотному контролеру (9), який у свою чергу формує сигнал для сервопривода (10) через допоміжний вихід. Звільнений вантаж розмикає мікроперемикач, сигнал зміни стану якого надходить безпосередньо на Raspberry Pi (11) — це підтвердження факту скидання вантажу.

Завершальною дією сценарію є передача серверу повідомлення про успішну доставку (12), після чого сервер може ініціювати сповіщення для користувача та позначити замовлення як виконане. На цьому основний сценарій доставки завершується.

2.6 Розробка блок-схеми алгоритму роботи системи

Для формалізації логіки роботи бортового програмного забезпечення розроблено блок-схему алгоритму, яка описує внутрішню логіку прийняття рішень бортовим комп'ютером упродовж усього циклу доставки, включно з обробкою помилкових ситуацій та повторними спробами при невдачі.

Допоміжні сенсори бортової електроніки — модуль оптичного потоку з лазерним далекоміром та чотири ультразвукові сенсори на блок-схемі не показані окремими кроками. Вони ініціалізуються польотним контролером при увімкненні живлення і використовуються ним автоматично протягом усієї місії, бортовий комп'ютер не керує ними безпосередньо.

Блок-схема алгоритму наведена на рис. 2.9.

Алгоритм починається з фази підготовки апарата: вантаж фізично

під'єднується до защіпки сервопривода, після чого стан мікроперемикача підтверджує надійність закріплення. Після успішного підтвердження бортовий комп'ютер надсилає на сервер повідомлення про готовність і переходить у стан очікування команди.

Після отримання координат цільового пункту видачі виконується перевірка готовності компонентів — рівень заряду акумулятора, фіксація GPS, зв'язок з польотним контролером. У разі несправності алгоритм виконує до чотирьох повторних перевірок з інтервалом 30 секунд; при вичерпанні спроб сервер інформується про помилку готовності, і місія завершується без зльоту.

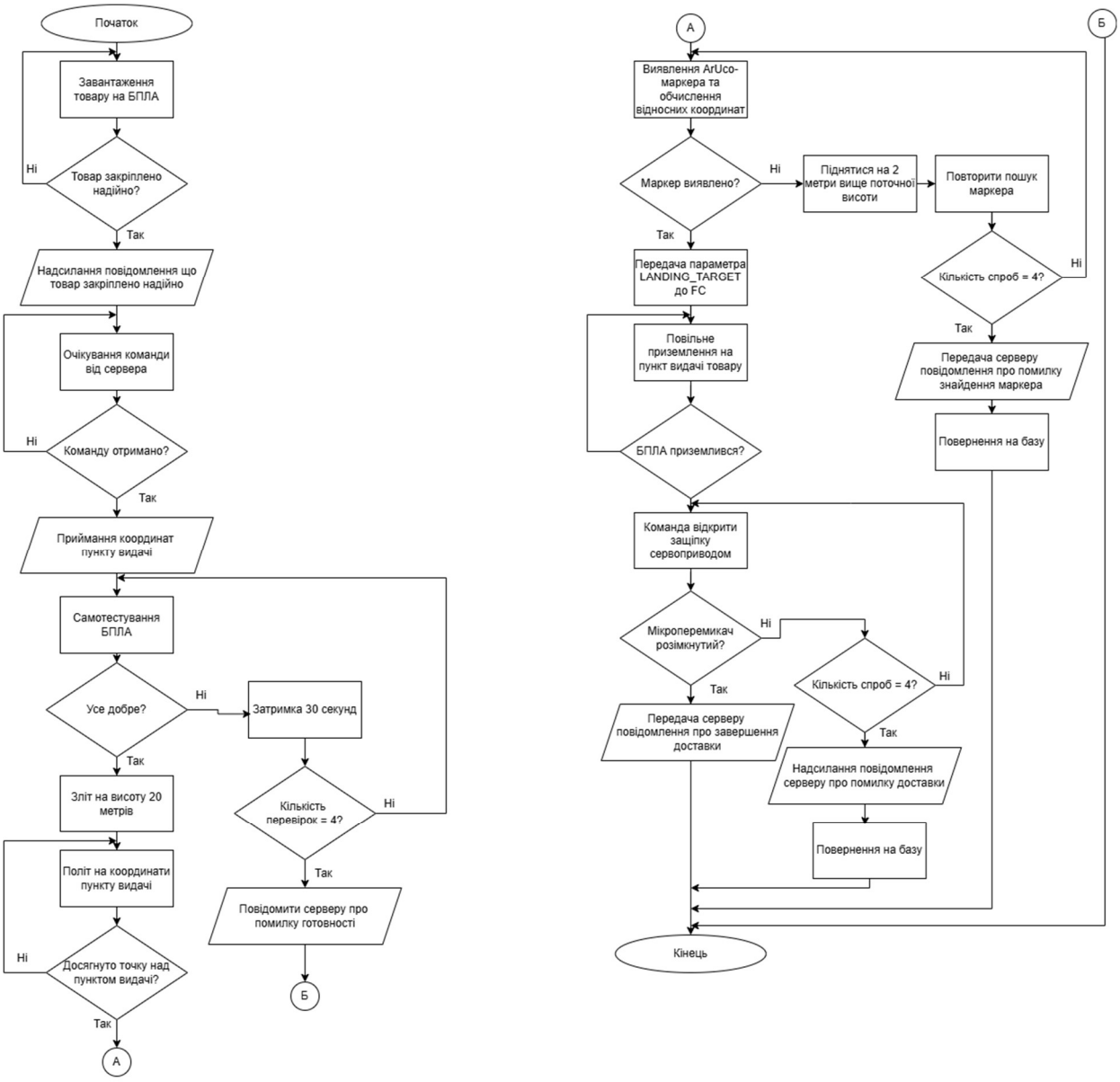

Рисунок 2.9 – Блок схема комп’ютерної системи доставки вантажу БПЛА

При успішній перевірці виконується зліт до висоти 20 метрів і автономний політ за GPS-координатами до точки пункту видачі. Над цільовою точкою активується підсистема комп'ютерного зору — обробка кадрів камери засобами OpenCV з пошуком ArUco-маркера. У разі невдачі апарат піднімається ще на 2 метри і пошук повторюється. Обмеження — чотири спроби. При вичерпанні спроб формується повідомлення про помилку, після чого апарат повертається на базу зі збереженим вантажем.

Після успішного виявлення маркера його координати передаються польотному контролеру у вигляді повідомлення LANDING_TARGET. Польотний контролер плавно знижується, постійно коригуючи позицію за оновлюваними координатами маркера, і завершує приземлення при висоті менш як 0,3 метра.

Після приземлення бортовий комп'ютер ініціює скид вантажу: подає команду на відкриття защіпки сервоприводом і очікує підтвердження від мікроперемикача. Команда скиду повторюється до чотирьох разів у разі неуспіху. При успішному скиданні сервер отримує повідомлення про завершення доставки, інакше — про помилку, після чого апарат повертається на базу.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Налаштування ArduPilot для точного приземлення

ArduPilot Copter містить штатну функціональність точного приземлення, яка приймає координати цілі від зовнішнього джерела через MAVLink-повідомлення LANDING_TARGET і коригує траєкторію посадки відповідно до них [30].

Усі параметри встановлюються через інтерфейс конфігурації Mission Planner (вкладка Config → Full Parameter List) або через утиліту MAVProxy у консольному режимі [47]. Налаштування виконуються один раз і зберігаються у незалежній пам'яті польотного контролера. Базові параметри, що визначають загальну поведінку режиму (таб. 3.1)

Таблиця 3.1 – Базові параметри поведінки приземлення

Параметр	Значення	Опис
PLND_ENABLED	1	Активація функції точного приземлення
PLND_TYPE	1	Тип цілі: MAVLink-повідомлення від companion computer
PLND_EST_TYPE	0	Тип оцінювача: Raw sensor (прямі координати)
PLND_YAW_ALIGN	0	Кут орієнтації камери відносно носа апарата, у сантиградусах
PLND_LAND_OFS_X	0	Зсув точки приземлення відносно центру маркера по X, см
PLND_LAND_OFS_Y	0	Зсув точки приземлення відносно центру маркера по Y, см

Параметр PLND_TYPE=1 вказує польотному контролеру очікувати координати цілі з повідомлень LANDING_TARGET, які надходять через MAVLink від бортового комп'ютера.

На фінальній фазі приземлення поведінка апарата керується додатковими параметрами (таб. 3.2).

Таблиця 3.2 – Додаткові параметри поведінки приземлення

Параметр	Значення	Опис
PLND_STRICT	1	Переривати приземлення при втраті цілі
LAND_SPEED	50	Швидкість зниження на фінальній фазі, см/с
LAND_SPEED_HIGH	200	Швидкість зниження на проміжній висоті, см/с
LAND_ALT_LOW	1000	Висота переходу на повільне зниження, см
WPNAV_SPEED_DN	200	Швидкість зниження у режимі AUTO, см/с

Значення LAND_SPEED=50 обрано як компроміс між тривалістю фінальної фази та плавністю приземлення на пункт видачі. При меншій швидкості час приземлення зростає непропорційно, при більшій — апарат може втратити маркер з поля зору камери.

Польотний контролер обмінюється з Raspberry Pi через UART-порт, призначений для протоколу MAVLink. У конфігурації Matek H743 [44] для цього використовується SERIAL2 (TELEM2). Праметри налаштування зображено в табл. 3.3.

Таблиця 3.3 – Параметри зв’язку між Pi та FC

Параметр	Значення	Опис
SERIAL2_PROTOCOL	2	Протокол: MAVLink 2
SERIAL2_BAUD	921	Швидкість UART, 921600 бод
SR2_POSITION	4	Частота відправки GLOBAL_POSITION, Гц
SR2_EXTRA1	4	Частота ATTITUDE, Гц
SR2_EXTRA2	4	Частота VFR_HUD, Гц
SR2_EXTRA3	2	Частота SYS_STATUS, Гц

Швидкість 921600 бод - для забезпечення стабільної передачі повідомлень LANDING_TARGET з частотою 5–10 Гц одночасно з телеметрією апарата у напрямку Raspberry Pi.

GNSS-модуль u-blox NEO-M9N [36] та модуль оптичного потоку Matek 3901-L0X [38] підключаються до окремих UART-портів польотного контролера (таб. 3.4)

Таблиця 3.4 – Параметри налаштування модуля Matek 3891-LOX

Параметр	Значення	Опис
SERIAL3_PROTOCOL	5	GPS
GPS_TYPE	1	u-blox
SERIAL4_PROTOCOL	7	OpticalFlow
FLOW_TYPE	6	Matek 3901-L0X
RNGFND1_TYPE	16	Matek 3901-L0X (Lidar)
RNGFND1_MIN_CM	5	Мінімальна вимірювана відстань, см
RNGFND1_MAX_CM	200	Максимальна вимірювана відстань, см
RNGFND1_ORIENT	25	Орієнтація: донизу

Чотири сенсори GY-US42v2 підключаються на спільну шину I2C з різними адресами і призначаються чотирьом напрямкам (таб. 3.5)

Таблиця 3.5 – Параметри налаштування ультразвукових сенсорів

Параметр	Значення	Опис
RNGFND2_TYPE	2	MaxbotixI2C (передній)
RNGFND2_ORIENT	0	Орієнтація: вперед
RNGFND2_ADDR	112	I2C-адреса 0x70
RNGFND3_TYPE	2	Правий
RNGFND3_ORIENT	6	Орієнтація: праворуч
RNGFND3_ADDR	113	0x71
RNGFND4_TYPE	2	Задній
RNGFND4_ORIENT	4	Орієнтація: назад
RNGFND4_ADDR	114	0x72
RNGFND5_TYPE	2	Лівий
RNGFND5_ORIENT	2	Орієнтація: ліворуч
RNGFND5_ADDR	115	0x73
AVOID_ENABLE	7	Активація уникнення перешкод
AVOID_MARGIN	2	Безпечна дистанція до перешкоди, м

Параметр AVOID_ENABLE=7 активує уникнення перешкод одночасно у режимах LOITER, RTL та AUTO, тобто протягом усієї автономної місії. Дистанція 2 метри забезпечує достатній запас для коригування траєкторії без різких маневрів, які могли б збурити стабільне утримання маркера у полі зору камери.

Сервопривід MG996R підключений до допоміжного виходу польотного контролера і керується параметрами (таб. 3.6).

Таблиця 3.6 – Параметри налаштуванян сервоприводу

Параметр	Значення	Опис
SERVO9_FUNCTION	7	RCPassThru (керування через канал RC9)
SERVO9_MIN	1000	ШІМ-імпульс закритого положення, мкс
SERVO9_MAX	2000	ШІМ-імпульс відкритого положення, мкс
SERVO9_TRIM	1000	Початкове положення (закрито)
SERVO9_REVERSED	0	Без інверсії

Команда відкрити защіпку від Raspberry Pi надсилається через MAVLink-повідомлення MAV_CMD_DO_SET_SERVO з параметром servo_number=9 та значенням ШІМ 2000 мкс. Команда закрити — те саме повідомлення зі значенням 1000 мкс.

Параметри для активації двонаправленого DShot-протоколу зобрежені в таб. 3.7.

Таблиця 3.7 – Параметри налаштування Bi-direction Dshot

Параметр	Значення	Опис
MOT_PWM_TYPE	6	DShot300
SERVO_BLH_AUTO	1	Авто-конфігурація BLHeli через DShot
SERVO_BLH_BDMASK	15	Активувати BDShot на каналах 1-4
SERVO_BLH_TRATE	10	Частота телеметрії, Гц

BDShot забезпечує отримання RPM з кожного мотора, що використовується для адаптивної фільтрації шумів через параметр INS_HNTCH_* (Harmonic Notch Filter).

Після введення всіх параметрів виконується команда Write Parameters в Mission Planner, після чого вся конфігурація зберігається у незалежній пам'яті польотного контролера. Резервна копія параметрів зберігається у форматі .param-файлу через функцію Save to file.

3.2 Розробка програмного забезпечення бортового комп'ютера

Програмне забезпечення бортового комп'ютера Raspberry Pi 5 реалізує функції координатора місії: отримання команд від сервера, керування процесом точного приземлення через детекцію ArUco-маркера, контроль скидання вантажу та повідомлення серверу про статус доставки. Програма виконана мовою Python 3.11 з використанням бібліотек OpenCV 4.13 [47], pymavlink [48], picamera2 [49] та gpiozero. Структурно вона побудована як набір функціональних модулів, що взаємодіють у межах головного циклу виконання місії.

Програма складається з чотирьох основних модулів, кожен з яких відповідає за окрему функціональну область:

– модуль зв'язку з польотним контролером, що інкапсулює обмін MAVLink-повідомленнями (mavlink_handler.py);

– модуль детекції ArUco-маркера, що обробляє кадри камери та обчислює відносні координати (aruco_detector.py);

– модуль керування механізмом скидання, що працює з мікроперемикачем через GPIO та надсилає команди сервоприводу (payload_release.py);

– головний модуль (main.py), який реалізує кінцевий автомат сценарію доставки відповідно до блок-схеми, наведеної у підрозділі 2.6.

Такий поділ для того, щоб відокремувати функції системи та незалежно їх тестувати, наприклад, перевірити детекцію маркера на статичних зображеннях без запуску всієї системи.

Ініціалізація системи. При старті програма послідовно ініціалізує усі підсистеми бортового комп'ютера: встановлює зв'язок з польотним контролером через UART, відкриває камеру, налаштовує GPIO-лінію мікроперемикача та читає файл з калібрувальними параметрами камери.

Лістинг фрагмент коду ініціалізації:

```
pythonimport cv2
import numpy as np
from picamera2 import Picamera2
from pymavlink import mavutil
from gpiozero import Button

# Підключення до польотного контролера
master = mavutil.mavlink_connection(
    '/dev/ttyAMA0',
    baud=921600
)
master.wait_heartbeat()
print(f"Підключено до системи {master.target_system}")

# Ініціалізація камери Pi Camera Module 3
camera = Picamera2()
config = camera.create_video_configuration(
    main={"size": (1280, 720), "format": "RGB888"}
)
camera.configure(config)
camera.start()

# Завантаження калібрувальних параметрів камери
calib_data = np.load('camera_calibration.npz')
camera_matrix = calib_data['camera_matrix']
dist_coeffs = calib_data['dist_coeffs']

# GPIO мікроперемикача (вантаж присутній = замкнуто)
load_switch = Button(17, pull_up=True)
```

Підключення до польотного контролера виконується функцією mavlink_connection() із вказанням послідовного порту /dev/ttyAMA0 та швидкості 921600 бод. Виклик wait_heartbeat() блокує виконання до отримання першого MAVLink heartbeat-пакета від польотного контролера, що підтверджує встановлення зв'язку. Калібрувальні параметри камери — матриця внутрішніх параметрів та коефіцієнти дисторсії — заздалегідь обчислюються процедурою калібровки за ChArUco-плати [41] та зберігаються у бінарному файлі camera_calibration.npz.

Підсистема комп'ютерного зору активується після того, як апарат досягає зони над пунктом видачі. Кожен кадр камери обробляється функцією cv2.aruco.detectMarkers(), яка повертає координати кутів виявлених маркерів та їх ідентифікатори. На основі координат кутів та відомих геометричних розмірів маркера обчислюється положення апарата відносно маркера за алгоритмом solvePnP [41].

Лістинг функції детекції:

```
    pythonARUCO_DICT=cv2.aruco.getPredefinedDictionary(cv2.aruco
.DICT_4X4_50)
    ARUCO_PARAMS = cv2.aruco.DetectorParameters()
    detector = cv2.aruco.ArucoDetector(ARUCO_DICT, ARUCO_PARAMS)
    MARKER_SIZE = 0.30   # розмір маркера, м
    TARGET_ID = 7        # очікуваний ID маркера пункту видачі

    def detect_marker(frame):
        """Виявлення ArUco-маркера у кадрі та обчислення
координат."""
        gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
        corners, ids, _ = detector.detectMarkers(gray)

        if ids is None or TARGET_ID not in ids:
            return None

        idx = list(ids.flatten()).index(TARGET_ID)
        target_corners = corners[idx]

        # Обчислення положення відносно маркера
        success, rvec, tvec = cv2.solvePnP(
            objectPoints=get_marker_3d_points(MARKER_SIZE),
            imagePoints=target_corners.reshape(-1, 2),
            cameraMatrix=camera_matrix,
            distCoeffs=dist_coeffs
```

```
        )

        if not success:
            return None

        # tvec містить координати маркера у системі координат
камери
        x, y, z = tvec.flatten()
        return {'x': x, 'y': y, 'z': z, 'rvec': rvec}
```

Використовується словник DICT_4X4_50 — 50 унікальних маркерів розміром 4×4 біти. Маркер пункту видачі має фіксований ідентифікатор TARGET_ID = 7, що дозволяє апарату відрізнити цільовий пункт видачі від інших можливих маркерів у полі зору камери. Геометричний розмір маркера задано константою MARKER_SIZE = 0.30 метра. Функція повертає координати у системі камери, де ось Z спрямована вздовж оптичної осі камери, X — праворуч, Y — донизу.

Обчислені координати маркера упаковуються у MAVLink-повідомлення LANDING_TARGET та передаються польотному контролеру [50]. ArduPilot інтегрує ці дані у свій розширений фільтр Калмана [6] і коригує траєкторію приземлення.

Лістинг функції передачі даних щодо преземлення:

```
pythondef send_landing_target(x_offset, y_offset, distance):
    """Передача координат цілі польотному контролеру."""
    # Перетворення лінійних зсувів у кути (радіани)
    angle_x = np.arctan2(x_offset, distance)
    angle_y = np.arctan2(y_offset, distance)

    master.mav.landing_target_send(
        time_usec=int(time.time() * 1e6),
        target_num=0,
        frame=mavutil.mavlink.MAV_FRAME_BODY_FRD,
        angle_x=angle_x,
        angle_y=angle_y,
        distance=distance,
        size_x=0,
        size_y=0
    )
```

Координати маркера, отримані з функції detect_marker(), перетворюються з лінійних метрів у кути (радіани) відносно осі камери.

Параметр frame=MAV_FRAME_BODY_FRD вказує, що координати наведені у системі апарата (Forward-Right-Down). Частота надсилання повідомлень становить 5 Гц, що достатньо для плавного коригування траєкторії приземлення без перевантаження каналу UART.

Після приземлення активується процедура скиду: бортовий комп'ютер передає польотному контролеру MAVLink-команду на перемикання сервопривода у відкрите положення, а потім протягом обмеженого часу очікує підтвердження від мікроперемикача — розмикання його контактів свідчить про те, що вантаж залишив апарат.

Лістинг функції скидання товару:

```
pythondef release_payload(max_attempts=4, timeout=3.0):
    """Скидання вантажу з контролем мікроперемикача."""
    for attempt in range(1, max_attempts + 1):
        # Команда відкрити защіпку (PWM 2000 мкс)
        master.mav.command_long_send(
            target_system=master.target_system,
            target_component=master.target_component,
            command=mavutil.mavlink.MAV_CMD_DO_SET_SERVO,
            confirmation=0,
            param1=9,          # номер сервоприводу (SERVO9)
            param2=2000,           #  ШІМ-імпульс  відкритого
положення
            param3=0, param4=0, param5=0, param6=0, param7=0
        )

        # Очікування підтвердження від мікроперемикача
        start_time = time.time()
        while time.time() - start_time < timeout:
            if not load_switch.is_pressed:
                # Контакт розімкнутий — вантаж скинуто
                return True
            time.sleep(0.1)

        print(f"Спроба {attempt} невдала, повторюємо...")

    return False  # Усі спроби вичерпано
```

Команда MAV_CMD_DO_SET_SERVO з параметрами param1=9 (номер сервоприводу) та param2=2000 (ШІМ-імпульс) надсилає польотному контролеру вказівку встановити сервопривід у відкрите положення. Польотний контролер генерує відповідний ШІМ-сигнал на виході AUX. Після

команди функція очікує не більше 3 секунд на розмикання мікроперемикача; якщо контакти не змінили стан, команда повторюється. Максимальна кількість спроб — 4, після чого формується помилка скиду, як визначено блок-схемою алгоритму.

Обмін з сервером реалізовано через HTTPS-протокол з використанням стандартної бібліотеки requests. Бортовий комп'ютер періодично надсилає поточні координати на сервер та отримує команди для виконання місії. Повідомлення серверу про статусні події (готовність, успіх, помилка) відправляються разово при відповідних подіях. Авторизація запитів виконується через токен JWT, що дозволяє ідентифікувати конкретний апарат у системі.

Модуль main.py реалізує кінцевий автомат сценарію доставки, у якому переходи між станами відповідають гілкам блок-схеми з підрозділу 2.6. Програма виконується у вигляді безперервного циклу, у якому послідовно перевіряються умови переходу між фазами згідно блок-схеми. Помилкові ситуації обробляються окремими гілками з обмеженням кількості повторних спроб, після чого формується відповідне повідомлення серверу.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Моделювання та прогнозування небезпечних ситуацій в системі автономної доставки БПЛА

Система автономної доставки вантажів за допомогою безпілотних літальних апаратів є складним багаторівневим технічним комплексом. Кожен із його компонентів є потенційним джерелом небезпечної ситуації, а взаємодія між ними породжує додаткові ризики системного характеру.

Принципова особливість досліджуваної системи полягає у її повній автономності — відсутність оператора в контурі керування під час польоту означає, що жодна людина не може втрутитись у критичний момент і зупинити апарат вручну. Вся відповідальність за безпеку перекладається виключно на алгоритм, що суттєво підвищує вимоги до його надійності та до якості попереднього моделювання небезпечних ситуацій. Тому таке моделювання є обов'язковою умовою безпечної експлуатації подібних систем відповідно до вимог НПАОП та чинного законодавства України у сфері охорони праці [51].

Перш ніж приступати до побудови моделей небезпек, необхідно чітко розуміти архітектуру системи та визначити найризикованіший етап її роботи. Користувач формує замовлення через мобільний застосунок, після чого запит надходить на сервер, де відбуваються перевірка даних, підтвердження оплати та координація з складом. БПЛА отримує завдання, завантажується вантажем — автоматизованою системою — і вирушає до точки доставки за GPS-координатами. Фінальним і найбільш відповідальним етапом є приземлення на посадкову платформу — пункт видачі. Саме цей етап є критичним, оскільки будь-яка похибка в навігації, розпізнаванні платформи чи стабілізації апарату на останніх метрах траєкторії може призвести до загрози для життя та здоров'я

людей у зоні посадки. Дані досліджень підтверджують, що близько 53,85% усіх аварій БПЛА припадає саме на три фінальні фази польоту — початковий захід, фінальний захід та посадку [52].

З урахуванням архітектури системи всі небезпечні ситуації можна поділити на три взаємопов'язані групи. До першої належать технічні небезпеки, обумовлені відмовою апаратного або програмного забезпечення. Втрата GPS-сигналу в умовах щільної міської забудови може спричинити похибку позиціонування понад допустимі 1–2 метри, що унеможливлює точне приземлення на пункт видачі без резервної системи навігації. Не менш критичною є відмова алгоритму комп'ютерного зору, відповідального за розпізнавання маркера посадкової платформи. Збій двигунів або контролера польоту під час зависання над пунктом видачі є найнебезпечнішим сценарієм, оскільки призводить до неконтрольованого падіння апарату з вантажем безпосередньо в зону, де може перебувати людина. Передчасний розряд акумулятора через недооцінку енергоспоживання в умовах зустрічного вітру також може спричинити вимушену аварійну посадку поза цільовою зоною.

Друга група охоплює зовнішні небезпеки, що визначаються умовами середовища експлуатації. Несприятливі метеорологічні умови — бічний вітер зі швидкістю понад 8–10 м/с, інтенсивні опади, туман або ожеледиця на поверхні посадкової платформи — суттєво ускладнюють стабілізацію апарату в повітрі. Електромагнітні завади від промислового обладнання, базових станцій мобільного зв'язку або навмисного глушіння сигналу можуть погіршити якість GPS-навігації та зв'язку між БПЛА і сервером. Третя група небезпек пов'язана з людським фактором на етапі підготовки: некоректне завантаження вантажу може призвести до зміщення центру мас БПЛА та нестабільного польоту, а помилки в серверній логіці — до відправлення апарата за неправильними координатами.

Центральним питанням безпеки життєдіяльності в досліджуваній системі є фізична небезпека для людини, що перебуває в зоні роботи

автономного БПЛА. БПЛА із вантажем загальною масою від 100 грам до 5 кілограмів є джерелом значної кінетичної енергії. Дослідження Campolettano et al., проведене у Virginia Polytechnic Institute та опубліковане в Annals of Biomedical Engineering, встановило, що при падінні БПЛА масою понад 1,2 кг ризик отримати важку травму рівня AIS 3+ перевищує 50%, причому ризик зростає пропорційно до маси апарату та висоти падіння [53]. Це означає, що навіть відносно невеликий БПЛА у разі неконтрольованого падіння здатний завдати людині травм, що потребують госпіталізації.

Окремою і медично задокументованою категорією ризику є травми від лопатей гвинтів БПЛА. Аналіз даних відділень невідкладної допомоги США за 2010–2017 роки, здійснений Johnson et al. та опублікований у American Journal of Preventive Medicine, засвідчує, що пошкодження гвинтами є провідним механізмом травмування серед усіх задокументованих інцидентів за участю дронів [54]. Клінічна картина охоплює порізи та садна в 42% випадків, забиття в 27%, а також значно тяжчі ушкодження, включаючи розриви очного яблука та відкриті переломи. Особливо уразливими є голова та обличчя — вони потерпають у 58% зафіксованих випадків [55]. Гвинти доставкового БПЛА обертаються зі швидкістю до 40 000 обертів на хвилину, а швидкість кінця лопаті може перевищувати 300 км/год, що перетворює їх на різальний елемент із надзвичайно високою травматичною здатністю.

Не менш важливим є психологічний вимір безпеки людини в умовах повністю автономної системи. Клієнт, що очікує посилку поблизу пункту видачі, або випадковий перехожий не отримують жодного попередження про наближення апарату. Відповідно до концепції безпеки людино-машинних систем, ситуація, коли людина не усвідомлює наближення небезпеки, є найбільш ризикованою — вона повністю виключає можливість захисної поведінки [56]. На відміну від систем із оператором, автономний дрон з'являється без будь-якого попередження з людського боку. Це висуває додаткові вимоги до архітектури системи: звукова та світлова сигналізація при

заході на посадку, автоматичне оголошення зони відчуження та примусове переривання посадки при виявленні людини в радіусі безпеки є не опціональними функціями, а обов'язковими елементами безпечного проєктування.

На основі проведеного моделювання формулюються конкретні технічні рішення. Поєднання GPS із системою комп'ютерного зору для розпізнавання маркера у пункті видачі забезпечує точність посадки до ±10 сантиметрів навіть за деградованого супутникового сигналу. Ультразвукові або лідарні сенсори виявляють присутність людини в зоні посадки на відстані до 5 метрів і автоматично ініціюють режим зависання з очікуванням. Інтеграція з метеорологічними сервісами блокує виліт у заздалегідь небезпечних погодних умовах. Протокол безпечного повернення на базу, що активується при будь-якій критичній відмові або падінні заряду акумулятора нижче встановленого порогу, гарантує, що апарат не залишиться без керування над населеним пунктом. Усі перелічені рішення разом формують багаторівневу систему захисту людини, що відповідає принципу глибокого захисту, прийнятому в сучасній інженерії безпеки [51].

4.2 Менеджмент охорони праці та безпеки життєдіяльності

Впровадження системи автономної доставки вантажів безпілотними літальними апаратами є принципово новим видом господарської діяльності, що не має усталених аналогів у традиційній охороні праці. Відповідальність за безпеку оточуючих людей і персоналу перекладається повністю на організаційну систему, яка має бути побудована ще до першого вильоту. Саме тому менеджмент охорони праці в такій системі є фундаментальною інженерною вимогою [57].

Міжнародною основою для побудови такої системи слугує концепція Safety Management System — SMS, викладена в керівництві ICAO Doc 9859 та

адаптована до операцій з безпілотними системами. SMS як структурований підхід до управління безпекою будується на чотирьох функціональних компонентах: політика безпеки, управління ризиками безпеки, забезпечення безпеки та просування культури безпеки [58].

Перший компонент - політика безпеки. Визначає загальні зобов'язання організації щодо охорони праці та встановлює розподіл відповідальності між учасниками системи. У розроблюваному проєкті це означає чітке закріплення відповідальності: розробник алгоритму посадки несе відповідальність за коректність роботи систем виявлення перешкод, оператор платформи — за технічний стан пункту видачі та дотримання регламентів обслуговування, а серверна система — за перевірку метеоумов перед кожним вильотом. Закон України «Про охорону праці» прямо встановлює, що власник або уповноважений ним орган зобов'язаний створити на кожному робочому місці умови праці відповідно до нормативно-правових актів, а також забезпечити безпеку технологічних процесів [57].

Другий компонент - управління ризиками. Реалізується через систематичну ідентифікацію небезпек та оцінку їхнього впливу на здоров'я і безпеку людей. У контексті автономної доставки особливої уваги потребують три категорії осіб: клієнт, який очікує посилку поблизу пункту видачі і може перебувати в зоні посадки в момент прибуття БПЛА; технічний персонал, що обслуговує платформу та проводить регламентні роботи; випадкові перехожі в радіусі можливого падіння апаранту. Для кожної з цих категорій встановлюється власний профіль ризику та визначаються відповідні захисні заходи. Ця інтеграція є обов'язковою вимогою для відповідності стандартам FAA та OSHA при експлуатації БПЛА. Аналогічні вимоги в українському правовому полі випливають із Кодексу цивільного захисту та НПАОП [59].

Третій компонент - забезпечення безпеки. Передбачає постійний моніторинг функціонування системи та верифікацію того, що вжиті заходи справді знижують ризик до прийнятного рівня. На практиці це реалізується

через ведення журналу польотних інцидентів, автоматичну фіксацію телеметрії кожного вильоту та регулярний аналіз накопичених даних. Принципи використання результатів внутрішніх аудитів і розслідувань інцидентів у режимі, наближеному до реального часу, та використання їх як можливість для вдосконалення системи дозволяє виявляти системні слабкі місця до того, як вони реалізуються у вигляді реального інциденту з травмуванням людини [60].

Четвертий компонент — просування культури безпеки. Набуває специфічного змісту в умовах автономної системи. Оскільки «оператора» в традиційному розумінні немає, носіями культури безпеки стають розробники алгоритмів, інженери з технічного обслуговування та адміністратори серверної частини. Кожен із них повинен розуміти, що його рішення безпосередньо впливає на фізичну безпеку людей у зоні роботи системи. Регулярні інструктажі, затверджені регламенти обслуговування платформ та чіткі процедури дій у позаштатних ситуаціях є обов'язковими елементами цього компонента відповідно до вимог Типового положення про порядок проведення навчання та перевірку знань з питань охорони праці [57].

Практичним інструментом реалізації менеджменту охорони праці в розроблюваній системі є операційні процедури безпеки. Зокрема, процедура планового обслуговування пункту видачі повинна передбачати автоматичне блокування вильотів на весь час присутності технічного персоналу в зоні платформи. Процедура обробки інциденту: будь-яке відхилення від штатної траєкторії, спрацювання датчиків виявлення перешкод або аварійне повернення на базу — має автоматично фіксуватися в системі та ініціювати перевірку перед наступним вильотом. Таким чином, менеджмент охорони праці в системі автономної доставки БПЛА реалізується не через контроль людини-оператора, а через проактивне проєктування безпеки на рівні архітектури всієї системи, що повністю відповідає сучасним міжнародним стандартам управління безпекою [58].

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра розроблено комп'ютеризовану систему автономної доставки вантажів безпілотним літальним апаратом із точним приземленням на стаціонарний пункт видачі. Систему орієнтовано на застосування у сфері останньої милі логістики, де доставка до фіксованої приймальної інфраструктури є альтернативою як ручній доставці, так і доставці на довільну адресу.

У процесі виконання роботи отримано такі основні результати.

У першому розділі проведено аналіз технічного завдання, оглянуто наявні комерційні рішення у сфері безпілотної доставки та академічні концепти інтеграції безпілотників з автоматизованими відділеннями. Виконано порівняльний аналіз методів автономного точного приземлення. Обґрунтовано, що оптимальним рішенням для розглядуваної задачі є використання ArUco-маркерів у поєднанні з оптичним потоком та лазерним далекоміром, оскільки такий підхід забезпечує сантиметрову точність при низькій вартості та відсутності пропрієтарного обладнання. На основі аналізу сформовано чіткий перелік функціональних, технічно-експлуатаційних, архітектурних та безпекових вимог до системи.

У другому розділі виконано проєктування системи на архітектурному рівні. Розроблено дворівневу структурну схему — на рівні взаємодії підсистем та на рівні бортової електроніки апарата. Обґрунтовано вибір апаратних компонентів, а також механізму утримання та скидання вантажу на базі сервопривода MG996R та мікроперемикача підтвердження. У середовищі Altium Designer розроблено електричну принципову схему системи у відповідності до вимог ЄСКД та ДСТУ. Обґрунтовано вибір програмного забезпечення. У нотації UML 2.0 розроблено діаграму послідовності, яка описує обмін повідомленнями між компонентами системи протягом сценарію доставки, а також блок-схему алгоритму роботи бортового програмного забезпечення з повним покриттям помилкових ситуацій.

У третьому розділі виконано практичну реалізацію системи. Складено детальний перелік параметрів прошивки ArduPilot, необхідних для активації режиму точного приземлення, налаштування UART-зв'язку з бортовим комп'ютером, інтеграції навігаційних сенсорів, ультразвукових далекомірів обходу перешкод, керування сервоприводом скидання та електронними регуляторами обертів з підтримкою двонаправленого DShot. Розроблено програмне забезпечення бортового комп'ютера з чотирьох основних модулів. Наведено лістинги ключових функцій програми мовою Python.

У четвертому розділі проведено аналіз безпеки життєдіяльності та основ охорони праці у контексті експлуатації автономної системи доставки безпілотним літальним апаратом, виявлено основні небезпечні ситуації та сформульовано вимоги до менеджменту охорони праці.

Можливими напрямами подальшого розвитку є проведення повного циклу льотних випробувань на фізичному прототипі апарата, розширення системи додатковими сенсорами (стереокамера, лазерний сканер) для роботи у складних погодних умовах, реалізація механізму планування маршруту з урахуванням динамічних перешкод та інтеграція з мережею пунктів видачі у межах міської логістичної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жаровський Р.О., Луцик Н.С., Осухівська Г.М., Паламар А.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024. 39 с.
2. Voloskyi V., Leshchyshyn Y., Romanyshyn N., Palamar A., Tarasenko L. Method and algorithm for efficient cell balancing in the lithium-ion battery control system. CEUR Workshop Proceedings, The 1st International Workshop on Bioinformatics and Applied Information Technologies (BAIT 2024), Zboriv, Ukraine, October 02-04, 2024. Vol. 3842. P. 258-267.
3. Лещишин Ю.З., Романишин Н.Р., Наконечний В.В., Паламарчук А.О. Розробка системи зв'язку як інтегрованого елементу роботизованих систем. Проблеми створення, розвитку та застосування високотехнологічних систем спеціального призначення з урахуванням досвіду антитерористичної операції. Збірник тез доповідей XXI Всеукраїнської науково-практичної конференції. Житомир, 2016. С. 102.
4. Лещишин Ю.З., Назаревич Т.О., Міська І.В. Створення вбудованих систем на базі структурно-параметричних моделей цифрових каналів зв'язку. VIII Науково-технічна конференція «Інформаційні моделі, системи та технології». Тернопіль, 2020. С. 127.
5. Leschyshyn Y., Scherbak L., Nazarevych O., Gotovych V., Tymkiv P., Shymchuk G. Multicomponent Model of the Heart Rate Variability Change-point. IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH). 2019. P. 110–113.
6. Tymkiv P., Leshchyshyn Y. Algorithm Reliability of Kalman Filter Coefficients Determination for Low-Intensity Electroretinosignal. IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM). 2019. P. 1-5.

7. Leschyshyn Y., Semchyshyn O. Periodically correlated heart rate variability detection by Neyman-Pearson criterion. 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics. 2007. P. 139–140.

8. Jazairy A., Persson E., Brho M., von Haartman R., Hilletofth P. Drones in last-mile delivery: a systematic literature review from a logistics management perspective. The International Journal of Logistics Management. 2025. Vol. 36, No. 7. P. 1–62. DOI: 10.1108/IJLM-04-2023-0149.

9. Eskandaripour H. et al. Unmanned Aerial Vehicles in Last-Mile Parcel Delivery: A State-of-the-Art Review. Drones. 2025. Vol. 9, No. 6. Art. 413. URL: https://www.mdpi.com/2504-446X/9/6/413 (дата звернення: 03.02.2026).

10. Drone Delivery Systems and Energy Management: A Review and Future Trends. arXiv preprint arXiv:2206.10765. 2022. URL: https://arxiv.org/pdf/2206.10765 (дата звернення: 03.02.2026).

11. UAV-Based Delivery Systems: A Systematic Review, Current Trends, and Research Challenges. ACM Journal on Autonomous Transportation Systems. 2024. DOI: 10.1145/3649224.

12. A Review of Last-Mile Delivery Optimization: Strategies, Technologies, Drone Integration, and Future Trends. Drones. 2025. Vol. 9, No. 3. Art. 158. URL: https://www.mdpi.com/2504-446X/9/3/158 (дата звернення: 04.02.2026).

13. Повітряний кодекс України: Закон України від 19.05.2011 № 3393-VI (зі змінами). URL: https://zakon.rada.gov.ua/laws/show/3393-17 (дата звернення: 05.02.2026).

14. Package Delivery by Drone (Part 135). Federal Aviation Administration. URL: https://www.faa.gov/uas/advanced_operations/package_delivery_drone (дата звернення: 05.02.2026).

15. Amazon Prime Air. Wikipedia. URL: https://en.wikipedia.org/wiki/Amazon_Prime_Air (дата звернення: 05.02.2026).

16. Public Involvement and Environmental Review for Drone Operations. Federal Aviation Administration. URL:

https://www.faa.gov/uas/advanced_operations/nepa_and_drones (дата звернення: 06.02.2026).

17. Amazon’s 100 drone deliveries puts Prime Air far behind Alphabet’s Wing and Walmart partner Zipline. CNBC. 18.05.2023. URL: https://www.cnbc.com/2023/05/18/amazons-100-drone-deliveries-puts-prime-air-behind-google-and-walmart.html (дата звернення: 07.02.2026).

18. Matternet Launches Secure Medical Drone Delivery Portal for Hospitals. HIT Consultant. 10.03.2020. URL: https://hitconsultant.net/2020/03/10/matternet-secure-medical-drone-delivery-portal-hospitals/ (дата звернення: 07.02.2026).

19. Matternet Launches World’s Longest Urban Drone Delivery Route Connecting Hospitals and Laboratories in Zurich, Switzerland. Business Wire. 12.12.2022. URL: https://www.businesswire.com/news/home/20221212005097/en/ (дата звернення: 07.02.2026).

20. Optimising last-mile delivery network through locker–drone logistics system design under uncertain demand. Transportation Research Part C. 2026. URL: https://www.sciencedirect.com/science/article/abs/pii/S0969699726000207 (дата звернення: 07.02.2026).

21. Smart Autonomous Drone Delivery Station Point. James Dyson Award. 2025. URL: https://www.jamesdysonaward.org/en-US/2025/project/smart-autonomous-drone-delivery-station-point (дата звернення: 09.02.2026).

22. Best Delivery Drones for Commercial Use in 2026: Payload, Range & Cost Guide. URL: https://jinghongdrone.com/best-delivery-drones-for-commercial-use (дата звернення: 09.02.2026).

23. GNSS PPK and RTK for Precision UAV Mapping. CHCNAV. URL: https://www.chcnav.com/about/news/2026/gnss-ppk-and-rtk-for-precision-uav-mapping (дата звернення: 09.02.2026).

24. Precision Landing and Loiter. ArduPilot Copter Documentation. URL: https://ardupilot.org/copter/docs/precision-landing-and-loiter.html (дата звернення: 10.02.2026).

25. Autonomous unmanned aircraft system. United States Patent No. 12168533. URL: https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/12168533 (дата звернення: 11.02.2026).

26. Springer J. et al. A Precision Drone Landing System using Visual and IR Fiducial Markers and a Multi-Payload Camera. arXiv preprint arXiv:2403.03806. 2024. URL: https://arxiv.org/pdf/2403.03806 (дата звернення: 11.02.2026).

27. Wubben J., Fabra F., Calafate C.T. et al. Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition. Electronics. 2019. Vol. 8, No. 12. Art. 1532. URL: https://www.mdpi.com/2079-9292/8/12/1532 (дата звернення: 11.02.2026).

28. Accurate Autonomous UAV Landing Using Vision-Based Detection of ArUco-Marker. Springer Nature Link. URL: https://link.springer.com/chapter/10.1007/978-3-030-60337-3_18 (дата звернення: 12.02.2026).

29. An Autonomous Tracking and Landing Method for Unmanned Aerial Vehicles Based on Visual Navigation. Drones. 2023. Vol. 7, No. 12. Art. 703. URL: https://www.mdpi.com/2504-446X/7/12/703 (дата звернення: 13.02.2026).

30. Precision Landing — Dev documentation. ArduPilot. URL: https://ardupilot.org/dev/docs/mavlink-precision-landing.html (дата звернення: 13.02.2026).

31. Optical Flow Sensors for Drones: Indoor Positioning Guide. Unmanned Tech. URL: https://www.unmannedtechshop.co.uk/blogs/knowledge-base/optical-flow-sensors-drones-autonomous-indoor-flight (дата звернення: 13.02.2026).

32. Precision Landing for Low-Maintenance Remote Operations with UAVs. Drones. 2021. Vol. 5, No. 4. Art. 103. URL: https://www.mdpi.com/2504-446X/5/4/103 (дата звернення: 14.02.2026).

33. Vision-Based Autonomous Ship Deck Landing of an Unmanned Aerial Vehicle Using Fractal ArUco Markers. AIAA SciTech Forum. 2025. URL: https://arc.aiaa.org/doi/10.2514/6.2025-2345 (дата звернення: 14.02.2026).

34. Long-Duration Fully Autonomous Operation of Rotorcraft Unmanned Aerial Systems for Remote-Sensing Data Acquisition. arXiv preprint arXiv:1908.06381. URL: https://arxiv.org/pdf/1908.06381 (дата звернення: 14.02.2026).

35. Companion Computers. ArduPilot Developer Documentation. URL: https://ardupilot.org/dev/docs/companion-computers.html (дата звернення: 20.04.2026).

36. Модуль GPS на U-blox NEO-M9N з інтерфейсною платою та антеною, для Arduino, трекерів, дронів. ROZETKA. URL: https://rozetka.com.ua/ua/599404687/p599404687/ (дата звернення: 22.04.2026).

37. GPS Auto Configuration. ArduPilot Documentation. URL: https://ardupilot.org/copter/docs/common-gps-auto-config.html (дата звернення: 23.04.2026).

38. Matek 3901-L0X Optical Flow & Lidar Sensor. MATEKSYS. URL: https://www.mateksys.com/?portfolio=3901-l0x (дата звернення: 25.04.2026).

39. STMicroelectronics VL53L0X World's smallest Time-of-Flight ranging and gesture detection sensor. Datasheet DS11555. 2018.

40. Raspberry Pi Camera Module 3. Raspberry Pi Ltd. URL: https://www.raspberrypi.com/products/camera-module-3/ (дата звернення: 25.04.2026).

41. Camera Calibration. OpenCV Documentation. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (дата звернення: 28.04.2026).

42. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ. 2019. 150 с.

43. Rangefinder. ArduPilot Documentation. URL: https://ardupilot.org/copter/docs/common-rangefinder-landingpage.html (дата звернення: 29.04.2026).

44. Matek H743-WING Flight Controller. MATEKSYS. URL: https://www.mateksys.com/?portfolio=h743-wing (дата звернення: 01.05.2026).

45. Raspberry Pi 5. Raspberry Pi Ltd. URL: https://www.raspberrypi.com/products/raspberry-pi-5/ (дата звернення: 02.05.2026).

46. Mission Planner Home. ArduPilot Documentation. URL: https://ardupilot.org/planner/ (дата звернення: 04.05.2026).

47. ArUco marker detection (aruco module). OpenCV Documentation. URL: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html (дата звернення: 05.05.2026).

48. pymavlink — Python MAVLink interface. GitHub. URL: https://github.com/ArduPilot/pymavlink (дата звернення: 07.05.2026).

49. picamera2 — Python library for Raspberry Pi cameras. GitHub. URL: https://github.com/raspberrypi/picamera2 (дата звернення: 08.05.2026).

50. MAVLink Developer Guide. URL: https://mavlink.io/en/ (дата звернення: 09.05.2026).

51. НПАОП 0.00-2.24-05 «Перелік робіт з підвищеною небезпекою». Київ, 2005.

52. Abujoub S. et al. Coarse Grained FLS-based Processor with Prognostic Malfunction Feature for UAM Drones using FPGA. arXiv preprint arXiv:2304.02099. 2023. URL: https://arxiv.org/pdf/2304.02099 (дата звернення: 25.05.2026).

53. Campolettano E.T. et al. Ranges of Injury Risk Associated with Impact from Unmanned Aircraft Systems. Annals of Biomedical Engineering. 2017. Vol. 45, No. 12. P. 2733–2741. DOI: 10.1007/s10439-017-1921-6.

54. Johnson C.Y. et al. Drone and Other Hobbyist Aircraft Injuries Seen in U.S. Emergency Departments, 2010–2017. American Journal of Preventive Medicine. 2019. Vol. 57, No. 6. P. 818–825. DOI: 10.1016/j.amepre.2019.07.008.

55. Gorucu S., Ampatzidis Y. Recreational Drone-Related Injuries in Children: A Review of NEISS Data. Pediatric Emergency Care. 2021. DOI: 10.1097/PEC.0000000000002444.

56. Запорожець О.І. Безпека життєдіяльності. 2-е вид. Київ: Центр учбової літератури, 2020. 448 с.

57. Закон України «Про охорону праці» від 21.11.2002 № 2694-XII (зі змінами та доповненнями). URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 27.05.2026).

58. ICAO Doc 9859. Safety Management Manual (SMM). Fourth Edition. Montréal: International Civil Aviation Organization, 2018. URL: https://www.icao.int/safety-management/SMM (дата звернення: 29.05.2026).

59. Гогіташвілі Г.Г., Лапін В.М. Основи охорони праці. 4-те вид. Київ: Знання, 2018. 302 с.

60. Bergesen E. et al. Beyond Compliance: How Industry Leaders Are Shaping SMS for Scalable and Safe Drone Operations. DRONELIFE. June 2025. URL: https://dronelife.com/2025/06/02/beyond-compliance-how-industry-leaders-are-shaping-sms-for-scalable-and-safe-drone-operations/ (дата звернення: 30.05.2026).

ДОДАТОК А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

«Затверджую»

завідувач кафедри КС

_____ Осухівська Г.М.

"___ "_____ 2026 р.

Комп'ютерна система доставки вантажу безпілотним літальним апаратом

ТЕХНІЧНЕ ЗАВДАННЯ

на __5__ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»

Керівник кваліфікаційної роботи

_____ к.т.н., доц. Лещишин Ю.З.

*«___»*_____ 2026 р.

«ВИКОНАВЕЦЬ»

Студент групи СІ-42

_____ Манько А.М.

*«___»*_____ 2026 р

Тернопіль 2026

1. Назва та підстава для виконання роботи.

1.1. Комп’ютерна система доставки вантажу безпілотним літальним апаратом.

1.2. Підставою для виконання кваліфікаційної роботи бакалавра (КРБ) є наказ по Університету (№ 4/9-189 від 24.04.2026р.).

2. Виконавець.

2.1. Студент групи СІ-42 кафедри КС
Тернопільського національного технічного університету ім. І. Пулюя
Манько Андрій Михайлович.

3. Мета роботи.

3.1. Метою роботи є розробка комп’ютерної системи автономної доставки вантажу безпілотним літальним апаратом із точним приземленням на пункт видачі з використанням комп’ютерного зору.

4. Склад виробу.

4.1. До складу системи повинні входити:

1) польотний контролер Matek H743;
2) бортовий комп’ютер Raspberry Pi 5;
3) GNSS-модуль u-blox NEO-M9N;
4) модуль оптичного потоку та LiDAR Matek 3901-L0X;
5) камера Raspberry Pi Camera Module 3;
6) ультразвукові сенсори GY-US42v2;
7) сервопривід MG996R та механізм скидання вантажу;

5. Технічні вимоги.

5.1. Вимоги по призначенню.

5.1.1. Вбудована система повинна мати наступні параметри:

1) Відсоток успішних доставок 99+ %

2) Відсоток втрат товару під час доставки менше ніж 1%

5.2. Вимоги до умов експлуатації:

5.2.1. По умовам експлуатації виріб повинен відповідати вимогам ГОСТ 15150 для УХЛ4.1

5.2.2. Температура експлуатації від -10 до +40^{o}C

5.2.3. Відносна вологість до 100% при t=25^{o}C

5.2.4. Швидкість відтру від 1 до 10 м/с

5.3. Конструктивні вимоги.

5.3.1. Конструювання корпусу приладу в КРБ не передбачено.

5.3.2. Для побудови системи мають бути використані сучасні компоненти з можливістю поверхневого монтажу друкованого вузла.

5.3.3. При побудові системи необхідно передбачити розміщення роз’ємів живлення і обміну даними.

5.3.4. Габаритні розміри при макетуванні, мм, не більше:

довжина - 800

ширина - 600

висота - 600

5.3.5. Маса макету не більше 3 кг.

5.3.6. Конструкція макету повинна забезпечувати доступ до всіх комплектуючих виробів при тестуванні.

5.4. Вимоги до надійності.

5.4.1. Система повинна відповідати вимогам ДСТУ 2862-94.

5.4.2. Наробка на відмову, не менше 5000 год.

5.5. Вимоги метрології.

5.5.1. Вимірювання параметрів системи при моделюванні повинно виконуватись на універсальних вимірювальних приладах.

6. Економічні показники.

6.1. Собівартість системи повинна бути не більше 40000 грн.

7. Вимоги до документації.

7.1. Конструкторська документація повинна відповідати вимогам ЄСКД, ДСТУ та ГОСТ.

7.2. До складу документації повинно входити:

1) Схема структурна С1
2) Схема електрична структурна Е1
3) Схема електрична принципова Е3
4) Блок-схема алгоритму роботи
5) ПЗ

8. Стадії та етапи розробки КРБ

8.1 Стадії та етапи виконання КРБ наведенні в таблиці 1.

Таблиця 1

№	Назва етапу	Строк виконання	
		початок	кінець
1	Технічне завдання	26.01	02.02
2	Розділ 1	03.02	15.02
3	Розділ 2	20.04	10.05
4	Розділ 3	11.05	24.05
5	Розділ охорона праці	23.05	31.05
6	Нормоконтроль	01.06	14.06
7	Попередній захист	15.06	21.06
8	Захист	з 22.06	

9. В дане ТЗ можуть вноситись зміни по узгодженню сторін.

ДОДАТОК Б

Перелік елементів

Позн.	Найменування		Примітка
	Модулі та компоненти		
A1	Raspberry Pi 5, 8 ГБ ОЗП	1	
A2	Польотний контролер Matek H743-SLIM v3	1	
A3	GNSS u-blox NEO-M9N з антеною	1	
A4	Matek 3901-L0X	1	
A5	Raspberry Pi Camera Module 3	1	
A6 - A9	Вимірювач відстані GY-US42v2	4	
A10	USB LTE-модем 4G Huawei E3372h-153	1	
A11	Регулятор обертів Matek PDB FCHUB-12S V2	1	
A12	BEC Matek BEC12S-PRO 9-55V	1	
C1	Конденсатор 1000 мкФ × 35 В, Low-ESR Panaconic	1	
	електролітичний		
F1	Запобіжник PolyFuse 30 A	1	
GB1	Батарея Li-Ion 6S2P, Козацька банка	1	
	Електромеханічні пристрої		
M1 - M4	Електродвигун безщітковий 2207, 1750 KV	4	
M5	Сервопривід MG996R	1	

Зм.	Арк.	№ документа	Підпис	Дата	КС КРБ 123.186.00.00 ПЕ		
Розробив		Манько А. М.			Комп'ютерна система доставки вантажу безпілотним літальним апаратом Перелік елементів	Літ	Аркуш 1 / Аркушів 2
Перевірив		Лещишин Ю. З.					
Рецензент		Михалик Д. М.				ТНТУ СІ-42 м. Тернопіль	
Н.контр.		Тиш Є. В.					
Зав. каф.		Осухівська Г.М.					

Позн.	Найменування		Примітка
R1	*Резистор 10 кОм ± 5 %, 0,125 Вт Bourns 0805*	1	
S1	*Мікроперемикач KW10-Z1P*	1	
XS1	*Роз'єм XT60H-F Amass Female*	1	

ДОДАТОК В

Лістинг головних блоків коду роботи системи

```
# ============================================================================
# БЛОК 1. КАМЕРА І ВИЯВЛЕННЯ ArUco
# ============================================================================

def init_camera():
    """Ініціалізує Picamera2 з потрібною роздільною здатністю та FPS."""
    cam = Picamera2()
    config = cam.create_video_configuration(
        main={"size": CAMERA_RESOLUTION, "format": "RGB888"},
        controls={"FrameRate": CAMERA_FPS},
    )
    cam.configure(config)
    cam.start()
    time.sleep(1.0)  # пауза на стабілізацію автоекспозиції
    log.info("Камеру ініціалізовано: %s @ %d fps",
             CAMERA_RESOLUTION, CAMERA_FPS)
    return cam


def init_aruco_detector():
    """Створює детектор ArUco для словника 4×4."""
    dictionary = cv2.aruco.getPredefinedDictionary(ARUCO_DICTIONARY)
    parameters = cv2.aruco.DetectorParameters()
    # Тонке налаштування для виявлення на відстані:
    parameters.adaptiveThreshWinSizeMin = 5
    parameters.adaptiveThreshWinSizeMax = 25
    parameters.adaptiveThreshWinSizeStep = 5
    parameters.minMarkerPerimeterRate = 0.02   # дозволяє маркери, що
віддалені
    parameters.cornerRefinementMethod = cv2.aruco.CORNER_REFINE_SUBPIX
    detector = cv2.aruco.ArucoDetector(dictionary, parameters)
    log.info("Детектор ArUco готовий (словник DICT_4X4_50)")
    return detector


def detect_target_marker(frame, detector, target_id):
    """
    Шукає маркер із заданим ID на кадрі.

    Повертає (corners, rvec, tvec) або None, якщо не знайдено.
    """
    gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    corners, ids, _ = detector.detectMarkers(gray)
    if ids is None:
        return None
    for i, marker_id in enumerate(ids.flatten()):
        if marker_id == target_id:
            # Оцінка пози маркера у системі координат камери:
            rvec, tvec, _ = cv2.aruco.estimatePoseSingleMarkers(
                [corners[i]], MARKER_SIZE_METERS,
                CAMERA_MATRIX, DISTORTION_COEFFS
            )
            return corners[i], rvec[0][0], tvec[0][0]
    return None



def compute_angular_offset(tvec):
    """
```

```
    Переводить положення маркера у системі камери (X, Y, Z у метрах)
    у кутові зсуви відносно оптичної осі. MAVLink LANDING_TARGET
    очікує саме кути в радіанах.
    """
    x, y, z = tvec
    if z < 0.01:
        return 0.0, 0.0
    angle_x = math.atan2(x, z)    # горизонтальний зсув (вправо +)
    angle_y = math.atan2(y, z)    # вертикальний зсув (вниз +)
    return angle_x, angle_y


#
================================================================================
# БЛОК 2. ЗВ'ЯЗОК З ПОЛЬОТНИМ КОНТРОЛЕРОМ ПО MAVLINK
#
================================================================================

def connect_to_fc():
    """Встановлює UART-з'єднання з ArduPilot на Matek H743."""
    log.info("З'єднання з польотним контролером: %s @ %d",
             MAVLINK_CONNECTION, MAVLINK_BAUDRATE)
    conn = mavutil.mavlink_connection(
        MAVLINK_CONNECTION,
        baud=MAVLINK_BAUDRATE,
        source_system=255,         # ID нашого бортового комп'ютера
        source_component=1,
    )
    log.info("Очікую HEARTBEAT від польотного контролера...")
    msg = conn.wait_heartbeat(timeout=10)
    if msg is None:
        raise RuntimeError("Польотний контролер не відповідає")
    log.info("Польотний контролер на зв'язку: system=%d, type=%d",
             conn.target_system, msg.type)
    return conn


def send_landing_target(conn, angle_x, angle_y, distance,
                        size_x=0.0, size_y=0.0):
    """
    Надсилає повідомлення LANDING_TARGET на польотний контролер.

    ArduPilot Copter (4.x) використовує його у режимах LAND і
PRECISION_LOITER
    разом із параметром PLND_ENABLED=1.
    """
    conn.mav.landing_target_send(
        time_usec=int(time.time() * 1e6),
        target_num=0,
        frame=mavutil.mavlink.MAV_FRAME_BODY_FRD,
        angle_x=angle_x,
        angle_y=angle_y,
        distance=distance,
        size_x=size_x,
        size_y=size_y,
    )


def request_data_streams(conn):
    """Просить FC надсилати телеметрію (висота, режим, статус)."""
    conn.mav.request_data_stream_send(
        TARGET_SYSTEM, TARGET_COMPONENT,
        mavutil.mavlink.MAV_DATA_STREAM_POSITION, 4, 1
    )
```

```
    conn.mav.request_data_stream_send(
        TARGET_SYSTEM, TARGET_COMPONENT,
        mavutil.mavlink.MAV_DATA_STREAM_EXTENDED_STATUS, 2, 1
    )


def check_landed_state(conn):
    """
    Перевіряє чи апарат приземлився. ArduPilot повідомляє статус приземлення
    через повідомлення EXTENDED_SYS_STATE (landed_state == ON_GROUND).
    """
    msg = conn.recv_match(type="EXTENDED_SYS_STATE", blocking=False)
    if msg is None:
        return None
    return msg.landed_state == mavutil.mavlink.MAV_LANDED_STATE_ON_GROUND


#
# ============================================================================
# БЛОК 3. GPIO — МІКРОПЕРЕМИКАЧ ТА СЕРВОПРИВІД СКИДАННЯ
#
# ============================================================================

def init_gpio():
    """Налаштовує GPIO-пристрої через gpiozero."""
    cargo_switch = Button(GPIO_CARGO_SWITCH, pull_up=True, bounce_time=0.05)
    servo = AngularServo(
        GPIO_SERVO_RELEASE,
        min_angle=0, max_angle=180,
        min_pulse_width=0.5 / 1000,    # 500 мкс
        max_pulse_width=2.5 / 1000,    # 2500 мкс
        initial_angle=SERVO_LOCKED_ANGLE,
    )

    def _on_cargo_attached():
        with state.lock:
            state.cargo_attached = True
        log.info("Вантаж під'єднано до БПЛА (мікроперемикач замкнено)")

    def _on_cargo_removed():
        with state.lock:
            state.cargo_attached = False
        log.info("Вантаж відсутній (мікроперемикач розімкнено)")

    cargo_switch.when_pressed = _on_cargo_attached
    cargo_switch.when_released = _on_cargo_removed

    # Початкова перевірка:
    with state.lock:
        state.cargo_attached = cargo_switch.is_pressed
    log.info("GPIO готовий. Стан вантажу: %s",
             "присутній" if state.cargo_attached else "відсутній")
    return cargo_switch, servo


def release_cargo(servo):
    """Відкриває защіпку, чекає скидання, повертає у вихідне положення."""
    log.info("Відкриваю защіпку механізму скидання...")
    servo.angle = SERVO_RELEASED_ANGLE
    time.sleep(2.0)   # чекаємо доки вантаж випаде

    # Перевіряємо що мікроперемикач справді розімкнувся:
    with state.lock:
        if state.cargo_attached:
```

```
            log.warning("Мікроперемикач все ще замкнено — спроба повторно")
            servo.angle = SERVO_LOCKED_ANGLE
            time.sleep(0.5)
            servo.angle = SERVO_RELEASED_ANGLE
            time.sleep(2.0)

    log.info("Повертаю защіпку у вихідне положення")
    servo.angle = SERVO_LOCKED_ANGLE
    time.sleep(0.5)

    with state.lock:
        state.cargo_released = not state.cargo_attached
    log.info("Скидання %s",
             "успішне" if state.cargo_released else "не підтверджено")


#
# ============================================================================
# БЛОК 4. ЗВ'ЯЗОК ІЗ СЕРВЕРОМ ЗАМОВЛЕНЬ
#
# ============================================================================

def fetch_mission():
    """
    Отримує параметри місії від сервера.
    Повертає словник {marker_id, address, ...} або None.

    У продакшені тут робиться HTTP GET. Для демонстрації прийнято заглушку,
    яка повертає першу адресу з PICKUP_POINT_MAP.
    """
    try:
        response = requests.get(SERVER_URL + "/assign",
                                timeout=SERVER_TIMEOUT)
        response.raise_for_status()
        mission = response.json()
        log.info("Місію отримано: маркер ID=%d, адреса '%s'",
                 mission["marker_id"], mission["address"])
        return mission
    except requests.RequestException as e:
        log.warning("Сервер недоступний (%s), використовую тестову місію", e)
        marker_id = 0
        return {
            "marker_id": marker_id,
            **PICKUP_POINT_MAP[marker_id],
        }


def telemetry_worker():
    """Окремий потік: періодично надсилає статус місії на сервер."""
    while state.running:
        snapshot = state.snapshot()
        try:
            requests.post(
                SERVER_URL + "/telemetry",
                json=snapshot,
                timeout=SERVER_TIMEOUT,
            )
        except requests.RequestException:
            pass  # втрата зв'язку не повинна валити політ
        time.sleep(TELEMETRY_INTERVAL)


def report_delivery_complete():
    """Надсилає фінальне підтвердження виконаної доставки."""
```

```
    snapshot = state.snapshot()
    snapshot["event"] = "delivery_complete"
    try:
        requests.post(SERVER_URL + "/complete",
                      json=snapshot, timeout=SERVER_TIMEOUT)
        log.info("Сервер повідомлено про завершення доставки")
    except requests.RequestException as e:
        log.warning("Не вдалося повідомити сервер: %s", e)
        # Зберігаємо повідомлення у файл для відкладеної відправки:
        offline_log = LOG_DIR / "pending_reports.jsonl"
        with offline_log.open("a", encoding="utf-8") as f:
            f.write(json.dumps(snapshot, ensure_ascii=False) + "\n")


#
# ============================================================================
# БЛОК 5. ГОЛОВНИЙ ЦИКЛ
#
# ============================================================================

def signal_handler(signum, frame):
    """Чисте завершення при SIGINT/SIGTERM."""
    log.info("Отримано сигнал %d, завершую роботу", signum)
    state.running = False


def main():
    signal.signal(signal.SIGINT, signal_handler)
    signal.signal(signal.SIGTERM, signal_handler)

    log.info("=" * 60)
    log.info("Запуск системи доставки БПЛА. Лог: %s", log_file)
    log.info("=" * 60)

    # 1. Отримуємо місію від сервера
    mission = fetch_mission()
    with state.lock:
        state.target_marker_id = mission["marker_id"]
        state.target_address = mission["address"]

    # 2. Ініціалізуємо обладнання
    camera = init_camera()
    detector = init_aruco_detector()
    cargo_switch, servo = init_gpio()
    fc = connect_to_fc()
    request_data_streams(fc)

    # 3. Запускаємо телеметрію у фоні
    telemetry_thread = threading.Thread(
        target=telemetry_worker, daemon=True, name="telemetry"
    )
    telemetry_thread.start()

    # 4. Перевірка передпольотних умов
    if not state.cargo_attached:
        log.error("ПЕРЕДПОЛЬОТНА ПЕРЕВІРКА: вантаж не під'єднано! Зупиняю.")
        return

    log.info("Усі підсистеми готові. Очікую виявлення маркера ID=%d",
             state.target_marker_id)

    # 5. Основний цикл обробки кадрів
    frame_count = 0
    detection_count = 0
```

```
    last_detection_time = 0
    target_id = state.target_marker_id

    try:
        while state.running:
            frame = camera.capture_array()
            frame_count += 1

            result = detect_target_marker(frame, detector, target_id)

            if result is not None:
                corners, rvec, tvec = result
                detection_count += 1
                last_detection_time = time.time()

                angle_x, angle_y = compute_angular_offset(tvec)
                distance = float(np.linalg.norm(tvec))

                with state.lock:
                    state.marker_visible = True
                    state.marker_offset_x = float(tvec[0])
                    state.marker_offset_y = float(tvec[1])
                    state.marker_distance = distance

                # Передаємо корекцію на польотний контролер:
                send_landing_target(fc, angle_x, angle_y, distance)

                # Лог раз на секунду (щоб не засмічувати):
                if frame_count % CAMERA_FPS == 0:
                    log.info("Маркер ID=%d: offset=(%.2f, %.2f) м, дист.=%.2f
м",
                             target_id, tvec[0], tvec[1], distance)
            else:
                # Маркер втратили з поля зору:
                if time.time() - last_detection_time > 0.5:
                    with state.lock:
                        state.marker_visible = False

            # Перевірка стану приземлення:
            landed = check_landed_state(fc)
            if landed:
                with state.lock:
                    if not state.landed:
                        state.landed = True
                        log.info("Польотний контролер повідомив про
приземлення")
                        # Виходимо з циклу детекції — переходимо до скидання
                        break

    finally:
        log.info("Кадрів оброблено: %d, успішних детекцій: %d",
                 frame_count, detection_count)

    # 6. Послідовність після приземлення
    if state.landed:
        time.sleep(1.0)            # пауза щоб апарат стабілізувався на землі
        release_cargo(servo)        # скидаємо вантаж
        report_delivery_complete()  # звітуємо серверу

    # 7. Завершення роботи
    state.running = False
    camera.stop()
    servo.angle = SERVO_LOCKED_ANGLE
    fc.close()
```

```
    log.info("Місію завершено. До побачення.")


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