Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

Допущено до захисту
Завідувач кафедри комп'ютерних наук
доктор технічних наук, професор

Андрій БОНДАРЧУК

«01» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»

Спеціальність 122 Комп'ютерні науки

Освітня програма 122.00.01 Інформатика

Тема: ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС АВТОНОМНОГО ТРАНСПОРТУВАННЯ ВАНТАЖІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав
студент групи ІНб-2-22-4.0д
Шерстюк Богдан
Олександрович

_____
(підпис)

Науковий керівник
доктор технічних наук,
професор
Бушма О.В.

_____
(підпис)

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп'ютерних наук,
доктор технічних наук, професор
Андрій БОНДАРЧУК
«31» жовтня 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Програмно-апаратний комплекс автономного транспортування вантажів з використанням штучного інтелекту»

Виконавець – студент групи ІНб-2-22-4.0д
спеціальності 122 Комп'ютерні науки
освітньої програми 122.00.01 Інформатика
Шерстюк Богдан Олександрович

1. Вихідні дані: аналіз сучасних логістичних процесів автономного вантажного перевезення; дослідження існуючих програмно-апаратних рішень у сфері роботизованої логістики; використання сенсорних систем (GNSS, LiDAR, IMU); засоби штучного інтелекту, технології ROS2, NAV2, MQTT, Phython.
2. Основні завдання: проаналізувати логістичні процеси вантажних перевезень; дослідити існуючі програмно-апаратні рішення; спроєктувати архітектуру комплексу; обґрунтувати вибір апаратних і програмних компонентів; розробити інформаційну модель даних; описати AI-модуль прогнозування маршрутів і ризиків; провести тестування та оцінити результати роботи системи.
3. Пояснювальна записка: Обсяг близько 65 сторінок формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.
4. Графічні матеріали: схема архітектури програмно-апаратного комплексу, схема структурної взаємодії модулів, схема бази даних, діаграма роботи AI-модуля.
5. Додатки: лістинг структура конфігураційного файлу.
6. Строк подання роботи на кафедру: *«01» червня 2026 р.*

Науковий керівник
д.т.н., професор
_____ Бушма О.В.
_____ дата

Виконавець:
_____ Шерстюк Б.О.
_____ дата

КАЛЕНДАРНИЙ ПЛАН

№	Етапи виконання роботи	Термін виконання	Примітка
1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	Виконано
2	Аналіз предметної області автономного трансортування вантажів, вивчення аналогів, формування мети та завдань дослідження	01.11.25 - 11.01.26	Виконано
3	Аналіз застосування штучного інтелекту в транспортній логістиці та постановка задачі проєктування програмно-апаратного комплексу	26.01.26 – 15.03.26	Виконано
4	Дослідження архітектури системи, вибір апаратних компонентів	16.03.26- 31.03.26	Виконано
5	Розробка структури бази даних, інформаційної моделі системи та AI-модуля оцінювання ризиків маршрутів	01.04.26 – 15.04.26	Виконано
6	Тестування функціонування програмно-апаратного комплексу	16.04.26 – 30.04.26	Виконано
7	Впровадження та інтеграція програмних і апаратних компонентів комплексу автономного транспортування вантажів	01.05.26 – 15.05.26	Виконано
8	Підготовка пояснювальної записки, графічних матеріалів, додатків.	25.05.25 - 12.06.26	Виконано
9	Захист кваліфікаційної роботи	16.06.26	

«Затверджую»
Науковий керівник
д.т.н., професор Бушма О.В.

Індивідуальний план склав
Шерстюк Б.О.

РЕФЕРАТ

Кваліфікаційна робота: 67 с., 17 рис., 19 табл., 53 джерел.

Актуальність: необхідність створення доступних, масштабованих і стійких до ризиків систем автономного вантажного перевезення в цивільних середовищах підвищеної складності. Використання таких систем дає змогу зменшити залежність від ручної праці, підвищити безпеку персоналу, забезпечити прозорий збір телеметрії та створити основу для інтелектуального планування маршрутів.

Об’єкт дослідження: програмно-апаратний комплекс логістичних процесів переміщення вантажів у складних цивільних середовищах.

Предмет дослідження: методи, моделі та засоби побудови програмно-апаратного комплексу автономного вантажного перевезення, включаючи архітектуру системи, апаратну платформу, інформаційну модель даних і алгоритми прогнозування.

Мета роботи: полягає у розробленні концепції програмно-апаратного комплексу автономного вантажного перевезення для цивільних умов із використанням сучасних сенсорів, телеметрії, баз даних і AI-модуля прогнозування маршрутів та ризиків.

Завдання:

1. проаналізувати логістичні процеси вантажних перевезень та визначити вимоги до автономної транспортної платформи;
2. дослідити існуючі програмно-апаратні рішення, сенсорні підсистеми, комунікаційні підходи та програмні стеки, придатні для побудови комплексу;
3. спроєктувати архітектуру системи, базу даних, інтеграційні контури та модель взаємодії апаратного й програмного рівнів;
4. обґрунтувати структуру AI-модуля, який прогнозує маршрутні ризики, підтримує диспетчеризацію та підвищує стійкість комплексу до змін середовища.

Основні результати роботи: проаналізовано предметну область автономної логістики, визначено вимоги до комплексу, запропоновано багаторівневу архітектуру, обґрунтовано вибір апаратних і програмних компонентів, спроєктовано базу даних, описано AI-модуль прогнозування, реалізацію та тестування системи.

Практичне значення дослідження полягає у можливості використання запропонованого програмно-апаратного комплексу для автоматизації внутрішньої та локальної логістики в складних цивільних умовах.

Ключові слова: автономний вантажний робот, програмно-апаратний комплекс, логістика, ROS 2, сенсори, телеметрія, база даних, AI-модуль, прогнозування ризиків, маршрутизація.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – штучний інтелект.

AMR – автономний мобільний робот.

API – програмний інтерфейс взаємодії між компонентами.

GNSS – глобальна навігаційна супутникова система.

GPS – глобальна система позиціювання.

HMI – людино-машинний інтерфейс.

IMU – інерційний вимірювальний модуль.

IoT – інтернет речей.

LiDAR – лазерний датчик дистанційного зондування.

MQTT – легкий протокол обміну повідомленнями для IoT-систем.

PostGIS – геопросторове розширення PostgreSQL.

ROS 2 – програмне середовище для побудови робототехнічних систем.

SLAM – одночасна локалізація та побудова карти.

VRP – задача маршрутизації транспортних засобів.

WMS – система управління складом.

ЗМІСТ

ВСТУП

Сучасна логістика дедалі рідше сприймається лише як задача переміщення вантажу з точки А в точку Б. Для підприємств критичними стають безперервність доставки, стійкість до ризиків, прогнозованість витрат, безпека персоналу та здатність системи працювати в умовах, коли людина не повинна постійно перебувати безпосередньо біля транспортного засобу. Саме тому автономні й напівавтономні наземні платформи перетворюються з експериментальних рішень на логічний етап еволюції транспортної логістики. У світовій практиці автономні мобільні роботи вже застосовуються на складах, у внутрішньоцеховому переміщенні, в аграрному секторі, на об’єктах енергетики та в комунальному господарстві. Їхня роль полягає не в повній заміні людини, а в знятті з персоналу рутинних, небезпечних і фізично важких операцій [1][2][8] [25].

Для України тема набуває особливої ваги через поєднання кількох чинників. З одного боку, вітчизняна економіка потребує рішень, які можуть працювати в середовищах із порушеною інфраструктурою, дефіцитом кадрів і нестабільною якістю дорожнього покриття. З іншого боку, розвиток цифрових сервісів, вбудованих систем, сенсорики, супутникової навігації, штучного інтелекту та телеметрії створив технологічне підґрунтя для формування вітчизняних програмно-апаратних комплексів нового покоління. Практична цінність таких систем полягає у зменшенні часу простою, підвищенні безпеки праці, цифровому контролі за станом вантажу та маршрутом, а також у можливості працювати в режимі часткової автономії там, де раніше використовувалися лише традиційні механічні засоби транспортування [3][4][7] [32].

У кваліфікаційній роботі автономний вантажний робот розглядається як цілісний програмно-апаратний комплекс. До його складу входять шасі, силова частина, контролери, модулі позиціювання, сенсори навколишнього середовища, комунікаційні інтерфейси, серверна частина, база даних, підсистема диспетчеризації та AI-модуль, який аналізує маршрут, прогнозує ризики й

підтримує прийняття рішень. Такий підхід важливий, бо саме зв’язок між фізичним носієм, мережею передачі даних і програмною логікою визначає кінцеву якість системи. Гарна механіка без якісного програмного стеку перетворюється на дорогу платформу без інтелекту, а якісне програмне забезпечення без надійної сенсорики не здатне забезпечити стабільне функціонування в реальному середовищі [1][5][12][17].

Наукова та прикладна новизна теми полягає у поєднанні логістичного підходу з принципами сучасної робототехніки. Якщо класична логістика концентрувалася переважно на маршрутах, запасах та часових вікнах, то сучасна автономна система повинна враховувати ще й стан дорожньої поверхні, рівень заряду батареї, якість зв’язку, просторові перешкоди, динаміку людей і техніки на маршруті, а також імовірність відхилення від запланованої траєкторії. Відтак завдання проєктування комплексу лежить на перетині транспортної логістики, embedded-розробки, IoT, геоінформаційних систем та машинного навчання [6] [9][11][16].

Актуальність теми полягає в необхідності створення доступних, масштабованих та стійких до ризиків систем автономного перевезення вантажів у цивільних середовищах підвищеної складності. Такі системи знижують залежність від ручної праці, зменшують вплив людського фактора на якість перевезення, підвищують контрольованість логістичних процесів, забезпечують прозорий збір телеметрії та створюють основу для впровадження інтелектуального планування маршрутів. У реаліях України цей напрямок додатково підсилюється запитом на технології, здатні працювати там, де звичайна інфраструктура частково пошкоджена, а безпека людей має пріоритетне значення [4][7][10][32].

Метою роботи є розроблення концепції програмно-апаратного комплексу автономного вантажного перевезення для ризикованих цивільних умов із використанням сучасних сенсорів, телеметрії, баз даних і AI-модуля прогнозування маршрутів та ризиків.

Об’єктом дослідження є логістичні процеси переміщення вантажів у складних цивільних середовищах.

Предметом дослідження є методи, моделі та засоби побудови програмно-апаратного комплексу автономного вантажного перевезення, включаючи архітектуру системи, апаратну платформу, інформаційну модель даних і алгоритми прогнозування.

Для досягнення поставленої мети у роботі сформульовано такі завдання:

1. проаналізувати логістичні процеси вантажних перевезень та визначити вимоги до автономної транспортної платформи;
2. дослідити існуючі програмно-апаратні рішення, сенсорні підсистеми, комунікаційні підходи та програмні стеки, придатні для побудови комплексу;
3. спроєктувати архітектуру системи, базу даних, інтеграційні контури та модель взаємодії апаратного й програмного рівнів;
4. обґрунтувати структуру AI-модуля, який прогнозує маршрутні ризики, підтримує диспетчеризацію та підвищує стійкість комплексу до змін середовища.

Результати роботи апробовано шляхом публікації тез доповіді на конференції Інформаційні технології – 2026: зб. тез XII Всеукраїнської науково-практичної конференції молодих учених, 14 трав. 2025 р., м. Київ / Київський столичний університет імені Бориса Грінченка. с. 178–179 [31].

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз логістичних процесів вантажних перевезень

Логістичний процес вантажного перевезення в сучасному розумінні є послідовністю взаємопов’язаних дій, що починаються не в момент фізичного руху платформи, а ще на етапі формування потреби в доставці. Замовлення, пріоритизація, перевірка доступності ресурсу, вибір маршруту, оцінка стану середовища, завантаження, контроль руху, розвантаження і фіксація результату - це єдиний операційний ланцюг. Якщо хоча б одна ланка не цифровізована, уся система втрачає прозорість. Для автономного комплексу це особливо важливо, оскільки робот не здатний компенсувати неякісну організацію процесу за рахунок досвіду чи інтуїції, як це інколи робить людина-водій [8][11][32][34].

У виробничій та складській логістиці основними болючими точками є простої через очікування, повторні рейси з неповним завантаженням, неузгодженість між складом і транспортом, а також перевантаження працівників на коротких, але частих маршрутах. У аграрній логістиці додаються пил, бруд, сезонність, слабкий мобільний зв’язок та розтягнутість маршрутів. У комунальній сфері проблемою стає неоднорідність середовища: платформа повинна однаково впевнено рухатися асфальтом, плиткою, ґрунтом і тимчасовими об’їздами. На промислових майданчиках критичними стають надійність сенсорів, стійкість до вібрації та безпечна взаємодія з людьми [5][9] [17][25].

З погляду системного аналізу вантажне перевезення можна описати через потоки чотирьох типів: матеріальний, інформаційний, енергетичний і ризиковий. Матеріальний потік описує вантаж, його масу, габарити та часову критичність. Інформаційний містить телеметрію, статус замовлення, координати, журнали подій і службові повідомлення. Енергетичний потік пов’язаний із зарядом батареї, енергоспоживанням приводів і навантаженням допоміжних модулів. Нарешті, ризиковий потік включає втрату зв’язку, слизьку поверхню, обмежену

видимість, помилки позиціювання, появу людей або техніки на траєкторії. Для класичних транспортних схем останній потік часто залишається неформалізованим, але для автономного комплексу саме він має бути цифрово описаний і включений у модель прийняття рішень [7][12][16][36].

Практика показує, що на коротких маршрутах внутрішньої логістики найбільші втрати часу спричиняють не кілометри шляху, а часті мікрозатримки: очікування відкриття воріт, маневрування в тісному проході, об'їзд тимчасової перешкоди, повторне позиціювання біля точки завантаження, ручне узгодження пріоритету між кількома засобами транспорту. Людина часто приймає рішення ситуативно, але її дії не завжди відтворювані, а якість виконання залежить від втоми та досвіду. Автономна система цінна тим, що перетворює ці неформальні рішення на набір перевірюваних правил і моделей [2][4][15][27].

Окремо слід розглянути питання безпеки праці. Автономна платформа доцільна насамперед там, де рутинна доставка пов'язана з підвищеним ризиком для персоналу: нічні зміни, погана видимість, пилові цехи, ділянки після аварій, схили, перетин потоків навантажувачів і пішоходів. У таких умовах цифровий контроль швидкості, дистанції, аварійного гальмування та журналювання подій є не просто технічною опцією, а частиною системи управління ризиком. Саме тому при аналізі предметної області важливо оцінювати не лише продуктивність, а й ступінь зниження операційної небезпеки [5][18][25][31].

Для формалізації процесу перевезення зручно виділити ключові показники ефективності: час циклу доставки, коефіцієнт корисного завантаження, енергоспоживання на рейс, стабільність зв'язку, частоту втручання оператора, точність прибуття в контрольну точку, кількість вимушених зупинок та відсоток безпечних проходжень маршруту без конфлікту з іншими об'єктами. Саме ці метрики повинні стати основою для проєктування комплексу та подальшого тестування його ефективності [3][7][11][37].

Таблиця 1.1

Ключові показники оцінювання логістичного процесу автономного вантажного перевезення

Показник	**Зміст**	**Призначення**
Час циклу	Повний час від постановки завдання до підтвердження доставки	Мінімізація простоїв та повторних операцій
Коефіцієнт завантаження	Відношення фактичної маси вантажу до допустимої	Уникнення порожніх або напівпорожніх рейсів
Частота втручання оператора	Кількість ручних корекцій на 1 рейс	Оцінка реальної автономності системи
Стабільність зв’язку	Частка часу зі сталим каналом передачі даних	Контроль працездатності телеметрії
Точність прибуття	Відхилення від цільової точки позиціювання	Якість навігації та маневрування
Енергоспоживання рейсу	Витрата енергії на одну доставку	Оцінка економічності платформи
Рівень ризику маршруту	Інтегральний індекс перешкод, покриття, трафіку та погодних факторів	Підтримка безпечного планування

1.2 Аналіз існуючих програмно-апаратних рішень

Огляд існуючих рішень показує, що ринок автономних вантажних платформ розвивається в трьох основних напрямах. Перший - внутрішньоскладські AMR-платформи, орієнтовані на рівне покриття, контрольоване середовище та високу повторюваність операцій. Другий - позаскладські або outdoor-платформи для індустріальних і аграрних умов, де критичними є прохідність, захищеність сенсорів і здатність працювати зі змінною якістю GNSS та зв’язку. Третій напрям - гібридні системи, які поєднують телооперацію з частковою автономією: автоматичне слідування маршрутом, утримання траєкторії, уникнення перешкод і аварійне зупинення. Саме третій підхід сьогодні виглядає найреалістичнішим для впровадження в українських умовах, оскільки дозволяє не чекати ідеальної інфраструктури [1] [2][6][25].

Типовий сучасний програмно-апаратний комплекс складається з шасі, мотор-редукторів, тягового акумулятора, низькорівневого контролера, обчислювального модуля, навігаційних сенсорів, блоків бездротового зв’язку та серверної інфраструктури. У простіших рішеннях функції локального контролю й високорівневої логіки поєднуються на одному комп’ютері, але така схема гірше масштабується та складніше обслуговується. Більш зрілий підхід передбачає розділення на real-time рівень керування приводами та верхній рівень прийняття рішень, який працює через ROS 2 або сумісний middleware [1][12] [17][20].

Важливою тенденцією є відмова від ідеї повної автономності як базового режиму. На практиці найкращі результати дає адаптивна автономність, коли система сама визначає, чи може пройти ділянку самостійно, чи має знизити швидкість, запросити підтвердження оператора або перейти в режим безпечної зупинки. У виробничих умовах такий підхід значно надійніший за жорстку схему ‘або повністю автономно, або повністю вручну’. Крім того, він полегшує сертифікацію, впровадження та навчання персоналу [2][21][25][39].

При аналізі наявних рішень також помітно, що вузьким місцем часто виявляється не сама навігація, а інтеграція з корпоративними процесами. Якщо робот не отримує завдання з ERP або WMS, не веде журнал рейсів, не формує аналітику та не передає події в диспетчерський інтерфейс, то він залишається локальним пристроєм, а не частиною цифрової логістичної системи. Тому якісне рішення повинно мати API, модель подій, базу даних маршрутів, історію телеметрії та зрозумілий механізм ролей доступу [4][6][10][24].

Ще одна риса сучасних систем - сенсорна надлишковість. Надійні комплекси не покладаються на один канал навігації або один тип датчика. GNSS добре працює на відкритому просторі, але програє в ангарах і між металевими конструкціями; LiDAR точний для локального картування, але чутливий до забруднення; камера дає багату інформацію, але залежить від освітлення; IMU забезпечує високу частоту оновлення, але має дрейф. Саме поєднання цих

джерел і алгоритмів fusion формує реалістичну основу для стійкого позиціювання [14][15][16][29].

Таким чином, аналіз існуючих програмно-апаратних рішень дозволяє зробити три принципові висновки. По-перше, універсальних платформ не існує: конфігурація завжди залежить від сценарію доставки. По-друге, перевагу мають модульні системи, де апаратну частину можна розширювати без повної перебудови ПЗ. По-третє, критичним фактором зрілості рішення є не максимальна швидкість або вантажопідйомність, а баланс між безпекою, керованістю, інтегрованістю та прогнозованістю поведінки [1][5][25][31].

Таблиця 1.2

Узагальнене порівняння класів цивільних автономних вантажних платформ

Клас рішення	**Типове середовище**	**Переваги**	**Обмеження**
Внутрішньоскладська AMR-платформа	Рівна підлога, карта приміщення, висока повторюваність	Висока точність, прогнозованість, проста інтеграція з WMS	Обмежена придатність для outdoor-середовища
Промислова outdoor-платформа	Відкриті майданчики, ґрунтові дороги, пил, волога	Краща прохідність, стійкість до зовнішніх умов	Складніша локалізація, вищі вимоги до енергоживлення
Гібридна телооперація + автономність	Змішані умови, потреба в резервуванні людиною	Вища надійність, гнучкість, кращий контроль ризиків	Потребує розвиненої HMI та каналів зв’язку

1.3 Використання штучного інтелекту в транспортній логістиці

Штучний інтелект у транспортній логістиці варто розуміти не як абстрактний ‘розумний модуль’, а як сукупність прикладних механізмів, що працюють із прогнозуванням, класифікацією, виявленням аномалій та підтримкою прийняття рішень. У вантажному автономному комплексі AI доцільно використовувати там, де класичних жорстких правил уже недостатньо: оцінювання ризику маршруту, прогноз часу проходження, визначення

ймовірності втрати зв’язку, раннє виявлення деградації батареї, класифікація перешкод, а також рекомендації щодо перенаправлення рейсу [8][28][32][36].

Серед найбільш корисних для логістики класів моделей можна виділити регресійні моделі прогнозування часу, градієнтний бустинг і дерева рішень для скорингу ризиків, моделі часових рядів для прогнозу навантаження маршрутів, кластеризацію для виділення типових зон затримок, а також нейронні мережі для обробки візуальних та багатосенсорних даних. Важливо, що в реальній інженерній системі не завжди перемагає найскладніша модель. Часто краще працює інтерпретована модель середньої складності, яка дає трохи нижчу точність, але простіше пояснюється оператору й швидше перевчається під нові умови [8][9][18][24].

Для українського контексту особливу цінність має AI-модуль прогнозування ризиків, а не лише ‘пошуку найкоротшого шляху’. Найкоротший маршрут далеко не завжди є найкращим: він може проходити через ділянки з поганим покриттям, зонами слабкого сигналу, інтенсивним трафіком техніки або місцями, де регулярно виникають затримки через людський фактор. Саме тому до функції вартості маршруту слід включати не лише відстань і час, а й імовірність негативної події. Такий підхід добре узгоджується з сучасними дослідженнями в AI-логістиці та route planning [8][12][16][32].

Не менш важливим є застосування штучного інтелекту для підтримки технічної надійності. Дані про струм моторів, температуру контролера, швидкість розряду батареї, похибки IMU, частоту втрати пакетів зв’язку та повторюваність аварійних зупинок можуть використовуватися для виявлення передаварійних станів. У такому разі AI працює як інструмент predictive maintenance: система не просто фіксує несправність, а заздалегідь вказує на ймовірність її настання. Для підприємства це дає реальний економічний ефект, бо зменшує незаплановані простої [3][8][22][32].

Водночас надмірна ‘магічність’ AI у критичних логістичних системах є небажаною. Модель повинна працювати в межах зрозумілих політик безпеки, а її рекомендації мають бути відтворюваними. Це означає, що AI-модуль не

повинен одноосібно керувати приводами без контурів перевірки, натомість він має формувати оцінки, сигнали або ранжування варіантів для навігаційного й диспетчерського рівнів. Інакше замість підвищення надійності можна отримати непрогнозовану поведінку в нетиповій ситуації [2][20][28][37].

Отже, використання штучного інтелекту в транспортній логістиці доцільне насамперед у трьох площинах: прогнозування, оптимізація та діагностика. Саме така триєдина логіка й буде покладена в основу проєктованого комплексу: AI не замінює класичну навігацію, а посилює її там, де потрібен аналіз історії, оцінювання невизначеності та адаптація до змін середовища [8][16][28][32].

Таблиця 1.3

Основні напрями застосування AI в автономній логістиці

Напрям	**Типові моделі**	**Практичний ефект**
Прогноз часу маршруту	Регресія, градієнтний бустинг, часові ряди	Точніша диспетчеризація та розподіл завдань
Скоринг ризику ділянки	Класифікація, ансамблі дерев, логістична регресія	Уникнення небезпечних або нестабільних маршрутів
Розпізнавання перешкод	Комп’ютерний зір, CNN, sensor fusion	Підвищення безпечності руху
Predictive maintenance	Виявлення аномалій, кластеризація, бустинг	Зменшення аварійних простоїв
Оптимізація парку платформ	VRP-алгоритми, евристики, обмежувальне програмування	Краще завантаження ресурсів

1.4 Постановка задачі проєктування комплексу

Постановка задачі проєктування комплексу починається з визначення цільового сценарію. У цій роботі таким сценарієм є регулярне перевезення вантажів масою середнього класу на коротких і середніх відстанях у складному цивільному середовищі з частковою невизначеністю маршруту. Йдеться не про магістральні перевезення, а про локальні доставки між точками виробничої, складської чи інфраструктурної мережі. Комплекс повинен працювати в режимі часткової автономії, мати можливість дистанційного нагляду, зберігати

телеметрію та забезпечувати безпечну поведінку в разі відмови окремих модулів [1][4][7][12].

Задача проєктування охоплює кілька взаємопов’язаних підзадач. По-перше, необхідно сформувати архітектуру, яка розмежовує низькорівневе керування, навігаційний рівень, серверну аналітику та користувацький інтерфейс. По-друге, слід визначити апаратний склад, що поєднує сенсори ближньої дії, засоби глобального позиціювання, інерційні датчики, комунікаційні модулі й обчислювальний блок. По-третє, потрібно спроєктувати структуру даних так, щоб у ній поєднувалися телеметрія реального часу та історичні дані для машинного навчання. По-четверте, треба розробити AI-модуль, який оцінює ризики маршруту та підтримує вибір кращого шляху [6][7] [10][11].

Формально задачу можна описати так: для заданого набору точок доставки, параметрів вантажу, обмежень батареї, карти середовища та поточного стану платформи потрібно визначити маршрут і режим руху, що мінімізують сумарну функцію вартості. До цієї функції входять довжина шляху, очікуваний час проходження, енергетичні витрати та інтегральний ризик. При цьому система повинна мати можливість у реальному часі коригувати рішення за появи нових перешкод або деградації каналу зв’язку. Такий підхід узгоджується з сучасними моделями VRP та risk-aware navigation [11][15][16][24].

Особливість досліджуваного комплексу полягає в тому, що ризик розглядається як первинний параметр проєктування, а не другорядна змінна. Це означає, що архітектура, сенсорика, БД і AI-модуль будуються навколо ідеї стійкості до невизначеності. Система має не лише доїхати, а й уміти вчасно зупинитися, перебудувати маршрут, повідомити оператора, зафіксувати причину інциденту та використати цей інцидент для подальшого навчання моделі [7][16] [28][32].

Постановка задачі включає не тільки вимогу побудови автономного транспортного засобу, а й створення повноцінного інформаційного контуру навколо нього. Саме це відрізняє проєктований комплекс від окремого

роботизованого візка: його цінність полягає у здатності бути елементом цифрової логістичної екосистеми підприємства [1][4][24][25].

Таблиця 1.4

Основні групи вхідних параметрів для проєктування комплексу

Група	**Ключові параметри**
Вантаж	Маса, габарити, центр ваги, допустимі умови перевезення
Маршрут	Точки старту та призначення, заборонені зони, тип покриття
Стан платформи	Рівень заряду, температура, телеметрія приводів, режим роботи
Середовище	Перешкоди, трафік, освітленість, погодні фактори, карта місцевості
Зв’язок	Рівень сигналу, затримка, втрата пакетів, доступні канали
Безпека	Пороги швидкості, дистанції, правила аварійної зупинки та втручання оператора

Таблиця 1.4 відображає ключові групи параметрів, що визначають умови функціонування автономної вантажної платформи та впливають на прийняття рішень у процесі її експлуатації. Кожна з наведених груп охоплює критично важливі характеристики системи: від фізичних параметрів вантажу та геометрії маршруту до поточного технічного стану платформи, умов навколишнього середовища, якості зв’язку та встановлених обмежень безпеки.

Такий поділ дозволяє структуровано описати інформаційне поле, в якому працює система, та забезпечує основу для формування алгоритмів керування, адаптації поведінки робота і мінімізації ризиків під час виконання транспортних задач.

1.5 Висновки до розділу

У першому розділі дипломної роботи проведено системний аналіз предметної області автономного вантажного перевезення в складних цивільних умовах. Було встановлено, що сучасна логістика вже не зводиться до простого переміщення вантажу між двома точками. Ключовими вимогами стали безперервність доставки, стійкість до ризиків, прогнозованість витрат, безпека

персоналу та можливість роботи системи з мінімальною участю людини. Автономні й напівавтономні платформи перейшли зі стадії експериментальних розробок у практичну площину розвитку транспортної логістики [1][2][8][25].

Аналіз логістичних процесів (підрозділ 1.1) виявив основні проблемні зони: часті мікрозатримки, неповне завантаження, неузгодженість між складом і транспортом, а також вплив людського фактора. В умовах України ці проблеми посилюються порушеною інфраструктурою, сезонними факторами, слабким зв’язком і неоднорідністю дорожнього покриття. Для об’єктивної оцінки ефективності запропоновано систему ключових показників, що охоплюють час циклу, коефіцієнт завантаження, частоту втручання оператора, стабільність зв’язку, точність позиціювання, енергоспоживання та інтегральний рівень ризику маршруту [3][7][11][37].

Огляд існуючих програмно-апаратних рішень (підрозділ 1.2) показав відсутність універсальних платформ. Найбільш придатним для українських реалій є гібридний підхід із частковою автономією та елементами телеоперації. Він передбачає модульну архітектуру, сенсорну надлишковість і глибоку інтеграцію з корпоративними інформаційними системами. Основними критеріями зрілості рішення визначено не максимальну швидкість чи вантажопідйомність, а баланс між безпекою, керованістю, масштабованістю та прогнозованою поведінкою [1][5][25][31].

Аналіз застосування штучного інтелекту (підрозділ 1.3) підтвердив, що AI у логістиці має виконувати конкретні прикладні функції: прогнозування часу проходження, скоринг ризиків маршруту, розпізнавання перешкод і predictive maintenance. Особливо важливим для вітчизняного контексту є risk-aware підхід, за якого функція вартості маршруту враховує не лише відстань і час, а й імовірність виникнення негативних подій [8][16][28][32].

У підрозділі 1.4 сформульовано постановку задачі проєктування. Комплекс розглядається не як окремий візок, а як повноцінний програмно-апаратний елемент цифрової логістичної екосистеми підприємства. Ризик при

цьому виступає одним із центральних параметрів проєктування, а не другорядним чинником [7][16][28][32].

Підсумовуючи, перший розділ підтвердив, що розроблення автономного вантажного комплексу є міждисциплінарною проблемою, яка лежить на перетині транспортної логістики, робототехніки, embedded-систем, IoT, геоінформаційних технологій і машинного навчання. Ефективність майбутньої системи визначається насамперед її здатністю стабільно функціонувати в умовах невизначеності та забезпечувати прозорий цифровий контроль усіх процесів [1][8][25][32].

Отримані результати створюють необхідне теоретичне підґрунтя для другого розділу, де буде запропоновано конкретну архітектуру програмно-апаратного комплексу, обґрунтовано вибір апаратних компонентів, спроєктовано базу даних, інформаційну модель та логіку роботи AI-модуля прогнозування маршрутів і ризиків [6][10][12][16].

РОЗДІЛ 2
ПРОЄКТУВАННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

2.1 Архітектура системи

Проєктування архітектури системи доцільно будувати за багаторівневим принципом, де кожен рівень виконує чітко визначену функцію і не дублює інші. Для автономного вантажного комплексу запропоновано п’ятирівневу структуру: фізичний рівень шасі та приводів; сенсорно-контрольний рівень; навігаційно-обчислювальний рівень; серверно-аналітичний рівень; рівень диспетчеризації та користувацького інтерфейсу. Така декомпозиція спрощує масштабування, локалізацію несправностей і подальшу модернізацію комплексу [1][2][12][20].

Фізичний рівень включає ходову частину, тягові двигуни, драйвери, джерело живлення, силову комутацію та системи пасивної безпеки. Він повинен бути максимально простим, відмовостійким і ізольованим від високорівневої бізнес-логіки. Навіть якщо сервер або AI-модуль тимчасово недоступні, базові контури керування рухом, аварійного гальмування та самодіагностики мають зберігати працездатність. Це є класичним принципом поділу safety-critical і non-critical компонентів [12][17][18][23].

Сенсорно-контрольний рівень об’єднує локальні датчики, мікроконтролер і механізми попередньої обробки сигналів. На цьому рівні здійснюються зчитування енкодерів, IMU, датчиків струму, температури та близькості, первинна фільтрація шумів, перевірка порогів безпеки й швидке реагування на критичні події. Виносити ці функції виключно на промисловий комп’ютер небажано, адже тоді зростає затримка та зменшується прогнозованість реакції системи [17][18][29].

Навігаційно-обчислювальний рівень реалізується на edge-комп’ютері, де працюють ROS 2, стек навігації Nav2, модулі SLAM або локалізації, логіка побудови карт, розрахунок локальної й глобальної траєкторії, а також інтерфейси інтеграції з камерою, LiDAR і GNSS. Саме тут формується

‘операційний інтелект’ робота: система співвідносить карту, власний стан, цільову точку та вхідні ризики, визначаючи безпечний режим руху [1][2][21].

Серверно-аналітичний рівень зберігає історичні дані, телеметрію, маршрути, журнали подій, дані про заряджання, списки завдань і результати роботи моделей. На цьому рівні доречно використовувати PostgreSQL з розширенням PostGIS для геопросторових об’єктів, окреме сховище часових рядів для телеметрії та брокер подій для обміну повідомленнями між сервісами. Наявність серверного рівня принципово змінює цінність системи: вона накопичує досвід і може вдосконалювати моделі не за відчуттями оператора, а за даними [5][6][7][24].

Нарешті, рівень диспетчеризації забезпечує взаємодію з оператором, технологом або логістом. Через вебінтерфейс користувач ставить завдання, бачить місцезнаходження платформи, отримує попередження, переглядає карти ризику та аналітику виконання рейсів. Принципово важливо, щоб цей рівень не був перевантажений технічною деталізацією. Людині потрібна не тисяча сирих параметрів, а ясна картина: чи їде платформа, чому зупинилася, що заважає рейсу і які дії рекомендовані [4][10][24].

Узагальнюючи, архітектура системи повинна відповідати чотирьом базовим принципам: модульність, відмовостійкість, спостережуваність і масштабованість. Модульність дає змогу змінювати компоненти без тотального переписування системи. Відмовостійкість забезпечує безпечну деградацію функцій. Спостережуваність означає повноту телеметрії та журналів. Масштабованість дозволяє перейти від одного робота до парку платформ у межах єдиної диспетчерської логіки [1][2][5][24].

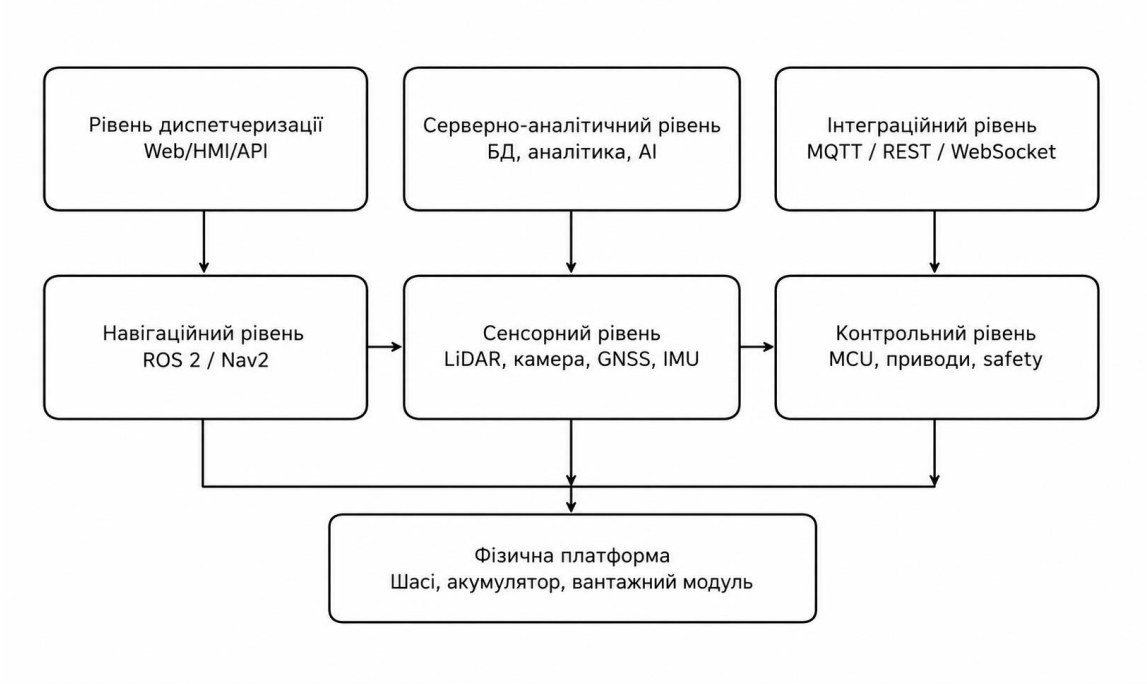

Рисунок 2.1 Узагальнена архітектура програмно-апаратного комплексу (розроблено автором)

2.2 Вибір апаратних компонентів (GPS, IoT, сенсори, контролери)

Вибір апаратних компонентів має спиратися не на принцип ‘максимально потужне залізо’, а на відповідність умовам реальної експлуатації. Для навігації у змішаних середовищах доцільно поєднати високоточний GNSS-приймач, LiDAR, IMU та камеру глибини. Така конфігурація дозволяє покривати різні режими руху: від відкритих майданчиків до напівзакритих просторів, де GNSS деградує, а локалізація спирається на локальні сенсори [14][15][16][29].

Як модуль глобального позиціювання доцільно розглядати приймачі класу u-blox ZED-F9P, що забезпечують багатодіапазонний GNSS і підтримку RTK. Їхня перевага полягає у відносно високій точності та придатності для інтеграції в промислові системи, однак варто розуміти, що GNSS у цій роботі не є самодостатнім засобом навігації. Він задає просторовий контекст, але не замінює локальне виявлення перешкод і позиціювання на коротких дистанціях [14].

Для побудови локальної карти й виявлення перешкод доцільно використати 2D або 3D LiDAR середнього класу. Практичним компромісом для дослідного зразка є рішення сімейства RPLIDAR S2, яке має достатню дальність

для зовнішніх і напівзакритих середовищ та прийнятне енергоспоживання. Разом із тим у пило- та вологозабруднених умовах потрібен захист вікна сенсора й регулярний контроль якості даних, інакше алгоритми навігації почнуть працювати з деградованою картою [16].

Камера глибини, наприклад Intel RealSense D455, не повинна розглядатися як конкурент LiDAR, а радше як його доповнення. Вона корисна там, де потрібне краще розуміння форми об'єктів, визначення висоти перешкоди, контроль під'їзду до точки завантаження та аналіз напівструктурованого середовища. У поєднанні з алгоритмами комп'ютерного зору камера дозволяє перейти від простого 'бачу перешкоду' до більш змістовного 'розумію тип сцени' [15][30].

Інерційний модуль класу BNO055 або аналогічний 9DoF-сенсор потрібен для стабілізації оцінки руху між глобальними оновленнями позиції, а також для виявлення ударів, нахилів і нетипових режимів шасі. Хоча такі датчики не дають абсолютної істини, вони істотно підвищують плавність локалізації при правильному sensor fusion і корисні для контролю безпеки на схилах та нерівностях [17][29].

На рівні обчислень доцільно використовувати edge-комп'ютер із GPU або принаймні достатнім ресурсом для ROS 2, обробки сенсорних потоків і локального inference. Для прототипування та малого серійного впровадження придатні рішення сімейства NVIDIA Jetson, які поєднують енергоефективність і достатню продуктивність для візуальних моделей. Низькорівневий контур керування при цьому слід залишити за окремим мікроконтролером або PLC-сумісним контролером, щоб зберегти прогнозованість реакції [18][19].

Комунікаційна підсистема повинна бути мультирежимною. Для стаціонарних або локальних майданчиків підійдуть Wi-Fi та Ethernet-доки, для розподілених територій - LTE/4G, а для легких телеметричних повідомлень - протокол MQTT. Резервування каналу зв'язку є принциповим: автономний комплекс не повинен ставати 'німою машиною' лише тому, що один радіоканал тимчасово втратив якість [7][20][23].

Таким чином, апаратний склад повинен будуватися навколо принципу сенсорної та комунікаційної комбінованості. Не один ідеальний датчик, а злагоджений ансамбль компонентів формує придатну для експлуатації платформу [14]-[20].

Таблиця 2.1

Рекомендований набір апаратних компонентів для комплексу

Компонент	**Приклад**	**Призначення**	**Переваги**	**Обмеження**
GNSS-приймач	u-blox ZED-F9P	Високоточне позиціювання на відкритих ділянках	Точність з RTK, багатодіапазонність	Погіршення роботи в закритих просторах
LiDAR	RPLIDAR S2	Виявлення перешкод і локальне картування	Незалежність від освітлення	Чутливість до забруднення
Камера глибини	Intel RealSense D455	Аналіз сцени, близькі маневри, контроль під'їзду	Багата просторова інформація	Вплив пилу та складного світла
IMU	Bosch BNO055	Оцінка орієнтації та динаміки руху	Висока частота оновлення	Накопичення дрейфу
Edge-комп'ютер	NVIDIA Jetson	ROS 2, sensor fusion, inference	Придатність для AI-обчислень	Потребує теплового контролю
MCU / контролер	STM32 / сумісний	Real-time керування приводами	Прогнозована реакція	Менші можливості високорівневої логіки

2.3 Вибір технологій програмної реалізації

Вибір технологій програмної реалізації має відповідати вимогам до надійності, розширюваності та швидкості інтеграції. Для робототехнічної частини природним вибором є ROS 2 як middleware, що підтримує вузлову архітектуру, публікацію-підписку, сервісні виклики та взаємодію з широкою екосистемою пакетів. Для задач навігації логічним доповненням є Nav2, який

уже містить готові механізми глобального й локального планування, costmap, поведінкові дерева та інструменти відновлення після збоїв [1][2][12][21].

На серверному боці доцільно використовувати FastAPI як основу для REST-інтерфейсів і служб інтеграції. Його перевагами є висока швидкість розробки, автоматична документація API, зручна типізація та добра сумісність із Python-екосистемою машинного навчання. У межах дипломного проєкту це дозволяє не розривати стек між логікою аналітики й прикладним веб-сервісом [4][22].

Для обміну телеметрією та подіями між роботом, брокером і серверними сервісами доцільно використати MQTT 5.0. Протокол добре підходить до сценаріїв IoT завдяки легкій publish/subscribe моделі, підтримці малих повідомлень і можливості роботи в умовах обмеженої пропускної здатності. При цьому критичною є не лише наявність MQTT як такого, а й грамотне проєктування тем, QoS-рівнів і правил повторної доставки [7][23].

На рівні зберігання даних найкращим компромісом між структурованістю та гнучкістю є PostgreSQL із розширенням PostGIS. PostgreSQL добре підходить для транзакційних сутностей - завдань, маршрутів, користувачів, журналів, а PostGIS дозволяє працювати з геопросторовими об’єктами, полігонами зон, точками навігації та просторовими індексами. Така зв’язка важлива для побудови карт ризику, аналізу історичних маршрутів і швидких просторових запитів [5][6].

Для AI-модуля доцільно використати Python-бібліотеки двох класів. scikit-learn добре підходить для інтерпретованих і швидких моделей табличних даних: класифікації ризику, регресії часу проходження, виявлення аномалій. TensorFlow або Keras доцільні там, де виникає потреба в роботі з часовими рядами більшої складності або з візуальними даними. Важливо не захоплюватися нейромережами заради самого факту їх наявності: для багатьох промислових сценаріїв сильний бустинговий ансамбль буде практичнішим за важку глибоку модель [8][22].

Для оптимізації маршрутів на рівні парку платформ природно використовувати Google OR-Tools. Бібліотека містить готові інструменти розв’язання VRP, у тому числі з часовими вікнами, місткістю та іншими обмеженнями. Це дозволяє будувати не лише одиночний маршрут, а й інтелектуальний розподіл кількох платформ між завданнями, що є важливою перспективою розвитку комплексу [11].

Узагальнюючи, технологічний стек пропонується таким: ROS 2 + Nav2 для робототехнічного ядра, Python/FastAPI для серверної інтеграції, MQTT для телеметрії, PostgreSQL/PostGIS для основного сховища, scikit-learn і TensorFlow для AI-модуля, OR-Tools для оптимізації маршрутів. Цей набір не є модним заради моди; він логічно поєднує зрілі відкриті інструменти, які мають спільноту, документацію й реальні кейси використання [1][4][5][7][8][11][22].

Таблиця 2.2

Обґрунтування вибору програмних технологій

Підсистема	**Технологія**	**Причини вибору**
Робототехнічне ядро	ROS 2, Nav2	Модульність, готовий навігаційний стек, активна екосистема
Серверний API	FastAPI	Швидка розробка, типізація, автоматична документація
Телеметрія	MQTT 5.0	Легка publish/subscribe взаємодія для IoT
База даних	PostgreSQL + PostGIS	Реляційна модель плюс геопросторові можливості
ML-аналітика	scikit-learn	Інтерпретовані моделі для табличних даних
DL/часові ряди	TensorFlow / Keras	Підтримка складніших моделей і inference
Маршрутна оптимізація	Google OR-Tools	VRP та обмежувальна оптимізація

2.4 Проєктування бази даних

Проєктування бази даних для автономного комплексу не можна зводити до банального збереження координат. База даних повинна підтримувати одразу три режими роботи: транзакційний облік сутностей, геопросторовий аналіз і

накопичення історичної телеметрії. Через це доцільно розділити дані на умовно ‘довгоживучі’ бізнес-сутності та ‘швидкоплинні’ сенсорні потоки [5][6][24].

До довгоживучих сутностей належать платформи, місії, маршрути, точки, користувачі, ролі, конфігурації сенсорів, технічні регламенти, журнали інцидентів і довідники. Вони змінюються відносно повільно, мають чіткі зв’язки та вимагають транзакційної цілісності. Телеметричні дані - координати, швидкість, стан батареї, сигнали сенсорів, статус зв’язку - навпаки, генеруються потоком і потребують оптимізації запису, індексації за часом і швидкого вибіркового аналізу [5][24].

У межах реляційної моделі доцільно виокремити таблиці robots, missions, mission_tasks, routes, route_segments, telemetry_samples, alerts, maintenance_events, operators та risk_assessments. Геопросторові поля варто реалізувати через типи geometry або geography з PostGIS, що дозволяє зберігати точки маршруту, полігони заборонених зон, буферні області небезпеки та реальні треки проходження [6].

Особливу роль відіграє таблиця risk_assessments, яка повинна зберігати не тільки інтегральний бал ризику, а й структуру внесків: стан покриття, якість зв’язку, погодний коефіцієнт, інтенсивність перешкод, невизначеність локалізації. Такий підхід робить AI-модуль пояснюванішим: оператор бачить не просто число 0.73, а розуміє, чому саме маршрут оцінено як несприятливий [8] [16][32].

Для прискорення аналітичних сценаріїв корисно запровадити матеріалізовані подання, які агрегують частоту зупинок за сегментами маршруту, середній час проходження, втрати зв’язку за зонами, динаміку деградації батареї та рейтинг проблемних точок. У реальному проєкті саме такі подання часто дають менеджменту найбільшу цінність, бо перетворюють потік технічних даних на управлінську картину [5][24].

Таким чином, база даних у запропонованому комплексі є не пасивним архівом, а аналітичним ядром, яке забезпечує просторову, часову й логічну цілісність усіх процесів системи [5][6][24].

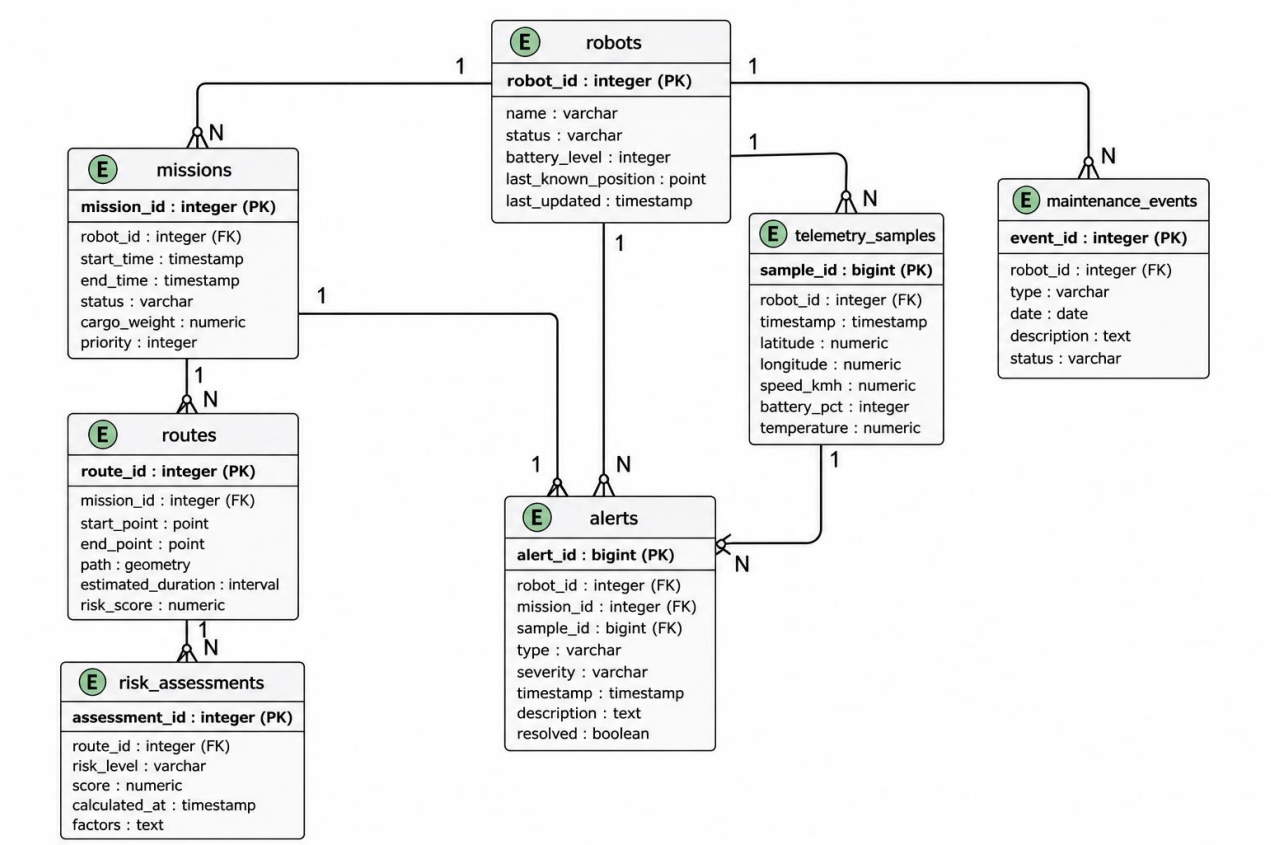

Рисунок 2.2. Спрощена ER-модель бази даних комплексу (розроблено автором)

Таблиця 2.3

Ключові таблиці бази даних і їх призначення

Таблиця	Призначення
robots	Довідник платформ: ідентифікатор, тип шасі, вантажопідйомність, статус
missions	Опис місій перевезення, пріоритет, час створення, виконавець
routes	Маршрути, точки старту/фінішу, геометрія, плановий час, довжина
telemetry_samples	Часові ряди телеметрії: координати, швидкість, заряд, сенсори
risk_assessments	Оцінки ризику за рейсом або сегментом маршруту
alerts	Аварійні та інформаційні події, рівень критичності, підтвердження оператором
maintenance_events	Події технічного обслуговування та відмов

2.5 Проєктування AI-модуля прогнозування маршрутів і ризиків

AI-модуль прогнозування маршрутів і ризиків є центральним елементом інтелектуальної надбудови комплексу. Його головне завдання полягає не в прямому керуванні приводами, а у формуванні оцінок і рекомендацій для навігаційного й диспетчерського рівнів. Інакше кажучи, AI повинен відповідати на питання: який маршрут найдоцільніший за поточних умов, де очікувати затримку, які сегменти є ризиковими, чи варто змінити режим руху, а також чи є ознаки технічної або середовищної деградації [8][16][28][32].

Вхідними даними моделі мають бути історія проходження сегментів, тип покриття, нахил ділянки, частота появи динамічних перешкод, середня швидкість на сегменті, якість зв’язку, похибка позиціювання, погодні фактори, маса вантажу, рівень заряду та температура силових компонентів. Частина цих ознак статична або повільно змінна, а частина є потоковою. Через це доцільно будувати модуль як комбінацію офлайн-навчання на історичних даних і онлайн-скорингу в реальному часі [8][16][22][36].

На першому етапі для risk scoring доцільно використати градієнтний бустинг або ансамбль дерев рішень. Такі моделі добре працюють із табличними

даними, стійкі до неоднорідності ознак і дозволяють визначати важливість факторів. Для прогнозування часу проходження маршруту або сегмента можуть бути використані регресійні моделі, у тому числі бустингові. Якщо згодом накопичиться достатній обсяг часових рядів, систему можна доповнити нейронною моделлю для short-term forecasting [8][22][32].

Після скорингу ризику та прогнозу часу AI-модуль повинен передати результати до блоку маршрутної оптимізації. Тут використовується модифікована функція вартості: $C = \alpha \cdot distance + \beta \cdot time + \gamma \cdot energy + \delta \cdot risk$. Коефіцієнти α, β, γ, δ визначаються політикою підприємства. Якщо пріоритетом є безпечність, вага δ зростає; якщо критичним є час виконання, збільшується β. Ця проста на вигляд формула є надзвичайно практичною, бо дозволяє поєднати класичний маршрутний підхід з ризик-орієнтованим управлінням [11][16].

Ще одним важливим елементом є пояснюваність. У диспетчерському інтерфейсі модель повинна показувати не лише підсумковий бал, а й топ-фактори, які вплинули на оцінку: слабкий зв’язок, поганий сегмент дороги, підвищене енергоспоживання, часті зупинки, нетипово висока температура приводу. Це підвищує довіру до системи й дає оператору змогу перевіряти логіку моделі на здоровий глузд [8][16][24].

Архітектурно AI-модуль доцільно будувати як окремий сервіс із власним API. Такий підхід дозволяє незалежно оновлювати моделі, вести версіонування, проводити A/B-порівняння та не блокувати робототехнічне ядро при перевченні або тимчасовій деградації сервісу. У випадку недоступності AI система повинна повертатися до базового безпечного планування за правилами, тобто деградувати контрольовано, а не хаотично [4][8][22].

Отже, AI-модуль у межах цього проєкту доцільно розглядати як сервіс прогнозування та рекомендацій, що підсилює маршрутизацію й експлуатаційну надійність комплексу. Саме така постановка відповідає практичним вимогам логістики: потрібен не ‘штучний інтелект заради красивої назви’, а конкретний інструмент, який зменшує ризики і витрати [8][11][16][32].

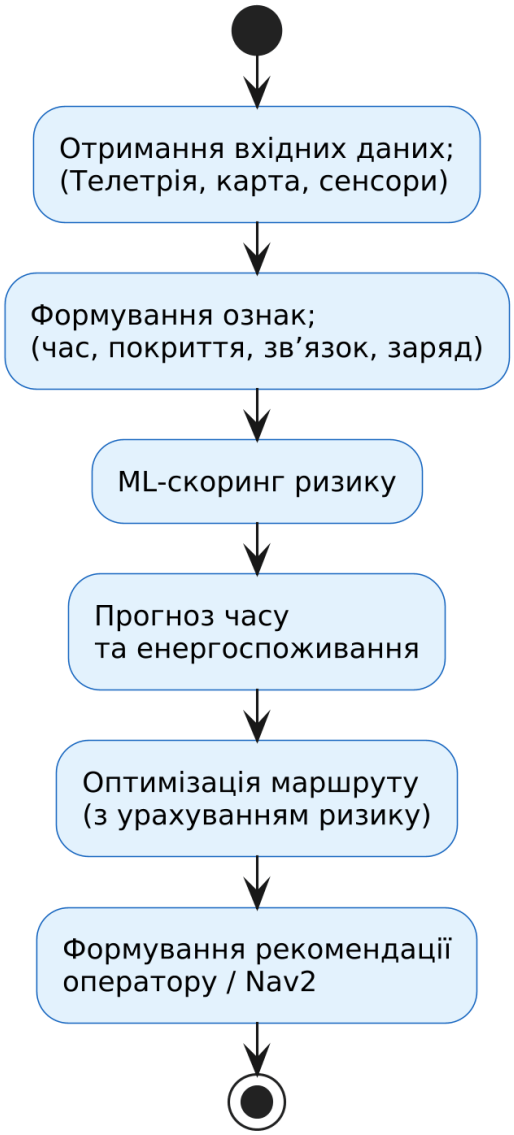

Рисунок 2.3. Логіка роботи AI-модуля прогнозування маршрутів і ризиків

Таблиця 2.4

Базові ознаки для AI-модуля прогнозування

Ознака	Зміст
Distance	Планова довжина маршруту або сегмента
ETA_pred	Прогнозований час проходження
Risk_score	Інтегральний бал ризику
Signal_quality	Оцінка якості зв’язку на ділянці
Surface_type	Категорія покриття: асфальт, ґрунт, бетон, змішане
Slope_index	Оцінка рельєфної складності
Battery_SOC	Рівень заряду батареї
Obstacle_rate	Частота появи динамічних перешкод
Localization_uncertainty	Показник невизначеності локалізації
Energy_pred	Прогноз енергоспоживання на ділянці

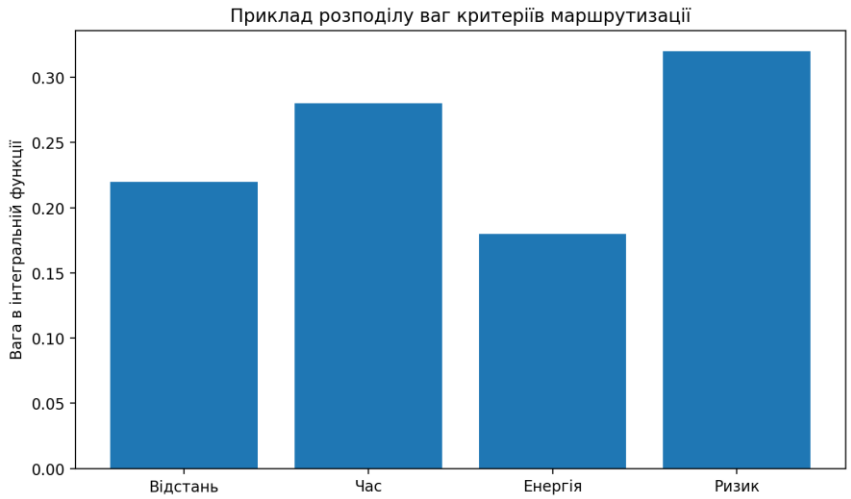

Рисунок 2.4. Приклад ваг критеріїв у функції вартості маршруту (розроблено автором)

2.6 Висновки до розділу

У другому розділі запропоновано архітектуру програмно-апаратного комплексу автономного вантажного перевезення для ризикованих цивільних умов. Показано, що технічно доцільною є багаторівнева система, де фізичний,

сенсорний, навігаційний, серверний і диспетчерський контури розділені за функціями, але інтегровані через стандартизовані інтерфейси [1][2][4][5][7].

Обґрунтовано вибір ключових апаратних компонентів, програмного стеку, реляційно-геопросторової бази даних та AI-модуля прогнозування. Принципово важливо, що запропоноване рішення орієнтоване не на демонстраційний прототип, а на систему, яка може бути масштабована, спостережувана та керована в умовах реальної експлуатації. Саме це створює підґрунтя для подальшої реалізації, тестування й економічного оцінювання комплексу в наступних розділах повної дипломної роботи [6][8][11][16][24].

Додаткове аналітичне розширення до розділів 1-2

Для більш глибокого розуміння логістичних процесів слід врахувати, що вантажне перевезення майже ніколи не існує ізольовано. Воно пов’язане з графіками змін, доступністю навантажувального обладнання, регламентами безпеки, наявністю буферних зон та порядком пропуску на окремі ділянки. Через це автономний комплекс повинен інтегруватися не тільки з маршрутною логікою, а й з організаційною моделлю підприємства. Якщо платформа прибуває до точки завантаження в момент, коли рампа зайнята або персонал не готовий прийняти вантаж, виникає проста, але дуже типова форма прихованої неефективності. Тому цифрова диспетчеризація має відображати не лише координати, а й стан операційної готовності точки призначення [4][10][24].

Суттєвою рисою ризикованих цивільних умов є неповна структурованість середовища. На відміну від класичного конвеєрного цеху, де маршрути майже не змінюються місяцями, на будівельних, інфраструктурних або аграрних об’єктах карта може змінюватися кілька разів на тиждень. З’являються тимчасові загородження, накопичення матеріалу, калюжі, колії, техніка, яка паркується інакше, ніж учора, або ділянки зі змінною прохідністю після дощу. Саме тому проєктований комплекс повинен працювати з ‘живою картою’, де статичний маршрут постійно коригується оперативними даними [15][16][29][30].

При оцінюванні існуючих рішень доречно розрізняти чотири рівні зрілості автономності.

Перший рівень - дистанційне керування без самостійного прийняття рішень.

Другий - виконання окремих автоматизованих функцій, наприклад утримання курсу чи зупинка перед перешкодою.

Третій - проходження визначених маршрутів у знайомому середовищі з можливістю втручання оператора.

Четвертий - адаптивна автономія з перебудовою плану в реальному часі.

Для дипломного проєкту раціонально орієнтуватися на третій рівень із елементами четвертого, оскільки він технічно досяжний і водночас практично корисний [1][2][25][31].

Окремого розгляду потребує економічний аспект. У багатьох підприємствах автоматизація перевезення окупається не стільки за рахунок повного скорочення персоналу, скільки за рахунок стабілізації процесу. Зменшення кількості зривів рейсів, більш точний облік циклів, прогнозованість енергоспоживання, контроль навантаження на техніку та скорочення непланових простоїв утворюють той сукупний ефект, який часто перевищує прямий виграш від заміни ручної праці. Саме тому в роботі доцільно аналізувати і технічну, і організаційно-економічну логіку впровадження [3][8][32].

У контексті апаратної архітектури важливо передбачити розділення живлення за критичністю споживачів. Приводи, сенсорний пакет, обчислювальний модуль і комунікаційне обладнання мають різні режими навантаження, а тому для них бажано проектувати окремі контури захисту, запобіжники й схеми пріоритетного відключення. Якщо ресурс батареї падає нижче визначеного порогу, система повинна насамперед зберігати керованість і канал передачі аварійних повідомлень, а не, наприклад, повну продуктивність візуального inference [18][19][23].

Практика робототехнічних проєктів показує, що слабкою ланкою часто стає не алгоритм, а калібрування. Навіть якісні GNSS, LiDAR і камера можуть давати поганий результат, якщо не виконано узгодження часових міток, просторових трансформацій і параметрів сенсорів. Через це в архітектурі

системи слід від самого початку передбачити процедури калібрування, збереження відповідних коефіцієнтів у БД та версіонування конфігурацій. Для дослідного зразка це здається надмірністю, але саме такі ‘дрібниці’ відрізняють живий інженерний комплекс від лабораторного макета [1][12][17][29].

База даних має враховувати ще одну важливу обставину: телеметрія потрібна не лише для аналітики, а й для розслідування інцидентів. Якщо платформа несподівано зупинилася, вдарилася колесом об перешкоду, втратила зв’язок або спрацювало аварійне гальмування, необхідно мати змогу відновити послідовність подій із точністю до секунд. Це вимагає синхронізації часових міток між роботом, брокером подій і сервером, а також чіткого розмежування телеметричних і службових логів [5][7][24].

Під час проєктування AI-модуля важливо розуміти, що реальні дані логістики майже завжди є ‘брудними’. У них є пропуски, нерівномірні інтервали, зміна умов експлуатації після оновлення платформи, сезонні ефекти й перекоси у вибірці. Через це якість моделі визначається не лише алгоритмом, а й побудовою правильного ML-конвеєра: валідацією за часом, контролем leakage, моніторингом drift і регулярним перевчанням. Якщо цього не зробити, навіть гарна модель швидко старіє, мов хліб на батареї [8][22][32][36].

Не менш важливо інтегрувати AI-модуль з людським прийняттям рішень. У виробничому середовищі оператор або логіст не хоче отримувати абстрактну пораду ‘маршрут не рекомендований’. Йому потрібне пояснення: яка саме ділянка проблемна, що змінилося відносно вчора, який альтернативний шлях є другим за якістю та які наслідки матиме вибір ризиковішого варіанта. Тому інтерфейс подання AI-рекомендацій слід проектувати паралельно з самою моделлю [4][8][24].

З погляду розвитку системи в перспективі доцільно передбачити сценарій переходу від одиничної платформи до роєвої або паркової логіки. Це означає, що вже на етапі дипломного проєкту варто закладати сутності ‘черга завдань’, ‘конфлікт ресурсу’, ‘розподіл місій’ і ‘глобальний диспетчер’. Навіть якщо

прототип буде один, архітектура, яка здатна масштабуватися на кілька платформ, значно цінніша в практичному сенсі [11][24][25].

Таблиця 2.5

Системні принципи, яких має дотримуватися проєктований комплекс

Принцип	**Практична інтерпретація**
Модульність	Можливість заміни сенсорів і контролерів без повної перебудови ПЗ
Відмовостійкість	Безпечна зупинка або деградація функцій при втраті окремих компонентів
Спостережуваність	Повний збір телеметрії, логів подій та історії рішень AI
Пояснюваність	Інтерпретоване пояснення оцінки ризику й рекомендацій
Масштабованість	Можливість переходу від 1 платформи до парку платформ
Інтегрованість	Сумісність із корпоративними сервісами та зовнішніми API
Кіберстійкість	Ролі доступу, журналювання, контроль повідомлень і резервування каналів
Енергоефективність	Контроль SOC, оптимізація маршрутів з урахуванням витрат енергії

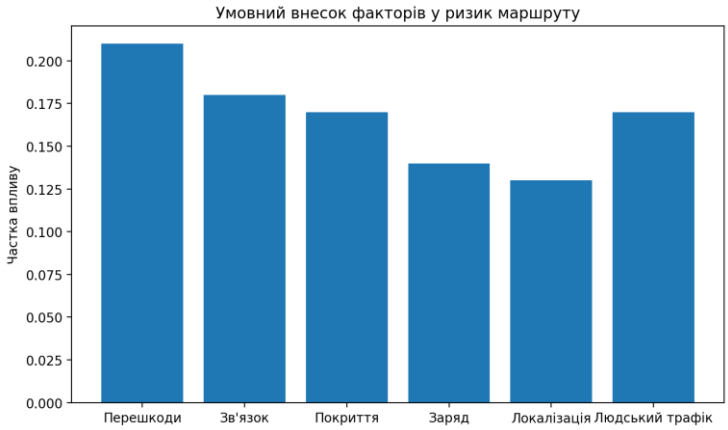

Рисунок 2.5. Приклад структури факторів, що формують ризик маршруту (розроблено автором)

РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація програмного модуля

Реалізація програмного модуля починається з розгортання базового контуру керування: прошивка польотного контролера, підготовка Raspberry Pi, налаштування каналу MAVLink, передавання телеметрії до диспетчерського ПК та перевірка приймання команд у Mission Planner. Обрана схема відповідає логіці розділення відповідальності. Matek H743 виконує низькорівневе керування рухом і роботу з ArduRover, Raspberry Pi виконує роль бортового шлюзу, а персональний комп’ютер оператора використовується як наземна станція керування. Такий поділ зменшує навантаження на кожен компонент і дає змогу замінювати окремі частини без повного переписування системи [1]-[5].

Першим практичним етапом була відмова від спрощеного підходу через Arduino, оскільки для повноцінної платформи з телеметрією, каналами зв’язку й інтеграцією Mission Planner доцільніше використати готовий стек ArduPilot Rover. На контролер Matek H743 було завантажено файл прошивки ArduRover із завантажувачем, після чого встановлення виконувалося через STM32CubeProgrammer у режимі boot. Такий шлях є більш правильним для контролерів STM32-класу, бо дозволяє працювати без зайвих посередників і отримати сумісність із MAVLink, Mission Planner та параметрами ArduPilot [2] [12][13].

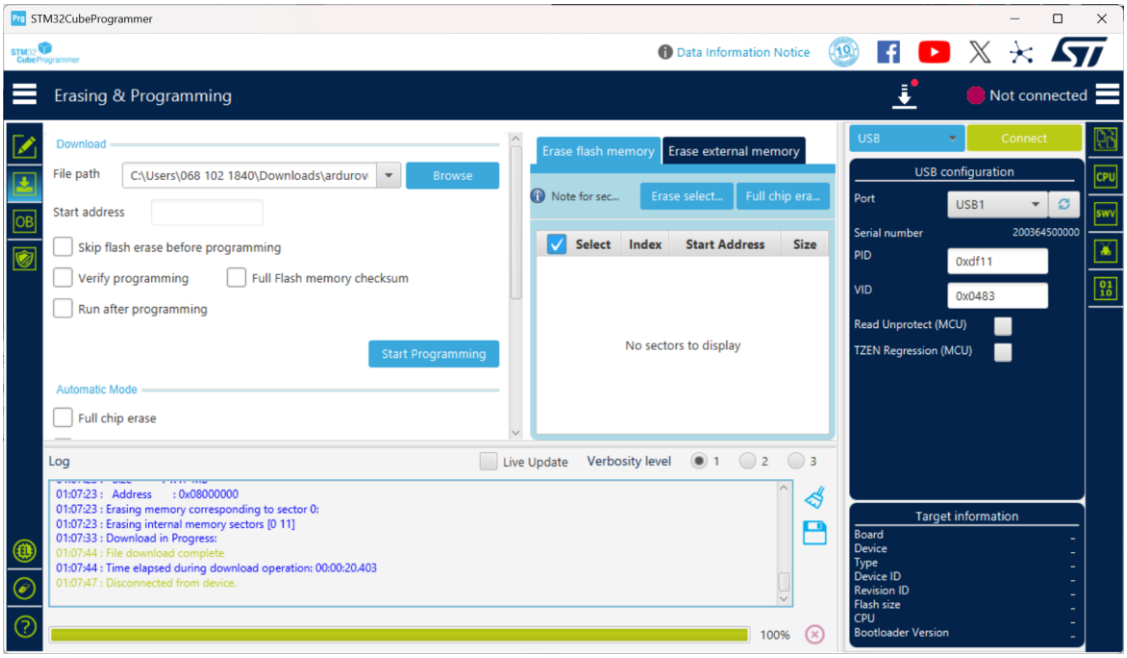

Рисунок 3.1. Вікно STM32CubeProgrammer під час прошивання контролера Matek H743

Другим етапом стало розгортання Raspberry Pi OS на microSD-карті. Для headless-режиму доцільно одразу задати ім’я користувача, пароль, параметри Wi-Fi та ввімкнути SSH у Raspberry Pi Imager. Після старту бортового комп’ютера підключення виконується командою ssh pi@raspberrypi.local або за IP-адресою пристрою. У дипломному проєкті цей крок важливий не як побутове налаштування, а як створення сервісної моделі доступу: оператор або розробник може оновити конфігурацію без підключення монітора й клавіатури до робота [6][7].

Після встановлення ОС Raspberry Pi з’єднується з Matek H743 через USB. У Linux-подібному середовищі порт контролера зазвичай визначається як /dev/ttyACM0 або /dev/ttyUSB0. Перевірка виконується командою ls /dev/ttyACM*. Якщо пристрій відображається, можна запускати MAVProxy. У цьому місці програмна архітектура стає живою: телеметрія з контролера перетворюється на мережевий потік, який може бути прийнятий Mission Planner через UDP-порт 14550 [3][4].

Канал MAVProxy у проєкті виконує функцію ретранслятора MAVLink-повідомлень. Він отримує дані від польотного контролера через USB і передає їх

у локальну мережу на IP-адресу комп’ютера оператора. У практичній конфігурації використано команду запуску з параметрами master, baudrate і out. Її можна адаптувати під конкретну адресу диспетчерського ПК та тип порту. Саме тут формується базова телеопераційна здатність системи: робот не просто реагує на радіосигнал, а входить у загальну мережеву інфраструктуру керування.

У Mission Planner оператор обирає тип підключення UDP і порт 14550. Після встановлення зв’язку наземна станція повинна відображати стан контролера, режим, телеметрію, параметри живлення та інші MAVLink-повідомлення. Якщо з’єднання не встановлюється, перевіряються чотири типові причини: неправильний IP у параметрі -out, невірний порт контролера, блокування UDP локальним фаєрволом або нестабільне живлення Raspberry Pi. Інженерна практика тут проста: не треба шукати містичних причин, доки не перевірені кабель, порт, адреса й живлення.

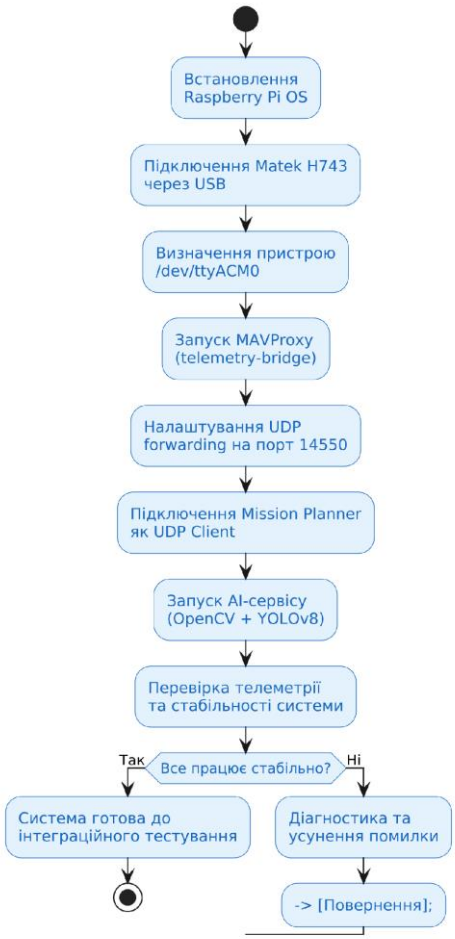

Рисунок 3.2. Послідовність запуску програмного контуру

Таблиця 3.1

Основні етапи реалізації програмного модуля

Етап	Зміст робіт	Очікуваний результат	Критерій перевірки
1	Завантаження прошивки ArduRover для Matek H743	Контролер отримує сумісну Rover-прошивку	Контролер визначається Mission Planner
2	Встановлення Raspberry Pi OS	Бортовий комп'ютер готовий до роботи	SSH-підключення доступне в локальній мережі
3	Підключення Matek до Raspberry Pi	Створено фізичний MAVLink-канал	У системі з'являється /dev/ttyACM0
4	Встановлення MAVProxy	Raspberry Pi працює як телеметричний шлюз	MAVProxy запускається без помилки
5	Передавання UDP у Mission Planner	Диспетчерський ПК приймає телеметрію	З'єднання UDP:14550 активне
6	Підготовка AI-сервісу	Камера й YOLOv8 готові до тестів	Відеопотік обробляється OpenCV

Базовий командний контур для Raspberry Pi має такий вигляд:

```
sudo apt update
sudo apt install python3-pip -y
pip3 install MAVProxy

ls /dev/ttyACM*

mavproxy.py \
  --master=/dev/ttyACM0 \
  --baudrate 115200 \
  --out udp:192.168.1.100:14550
```

У межах дипломного проєкту даний фрагмент є не набором команд, а реалізацією механізму взаємодії між компонентами системи. Він показує принцип побудови шлюзу між embedded-рівнем і диспетчерською мережею. Саме така схема дозволяє пізніше додати логування MAVLink-повідомлень, веб-інтерфейс, MQTT-публікацію стану, автоматичну діагностику каналу зв'язку й модуль аварійного переходу в безпечний режим.

Програмний модуль доцільно розширити фоновою службою systemd, яка автоматично запускає MAVProxy після старту Raspberry Pi. Це прибирає залежність від ручного запуску й робить робота ближчим до реальної експлуатації. Служба повинна мати контроль перезапуску, журналювання

stdout/stderr, залежність від мережі та окремий конфігураційний файл з IP-адресою диспетчерського ПК. Така дрібниця, на перший погляд, відрізняє студентський експеримент від інженерного прототипу.

Для стабільної експлуатації програмний модуль повинен вести журнали. Мінімальний журнал містить час запуску, активний порт, результат підключення до контролера, IP-адресу отримувача, кількість перезапусків, втрату з’єднання та час відновлення. Надалі ці дані можна передавати в серверну БД або аналізувати локально. Логування є важливим, бо без нього помилка перетворюється на легенду: усі щось пам’ятають, але ніхто не може довести, що саме сталося.

Окремим елементом є конфігурація режимів ArduRover. Для безпечного випробування доцільно використовувати ручний або напівавтоматичний режим із жорстким обмеженням швидкості, перевіркою fail-safe та аварійною зупинкою. У дипломній роботі не ставиться завдання створити повністю автономну навігацію на першому кроці; практично важливіше підтвердити керованість, прогнозованість реакції та стабільність телеметрії.

Модульна структура програмної частини передбачає три сервіси: telemetry-bridge, vision-service і supervisor-service. Перший відповідає за MAVLink/UDP, другий - за відео та AI, третій - за контроль стану сервісів, помилки, перезапуск і формування подій. У перспективі ці сервіси можна з’єднати через MQTT або REST API, але для прототипу достатньо локальної взаємодії на Raspberry Pi.

Таблиця 3.2

Рекомендована структура програмних сервісів Raspberry Pi

Сервіс	Призначення	Вхідні дані	Вихідні дані	Режим роботи
telemetry-bridge	Ретрансляція MAVLink через UDP	/dev/ttyACM0, параметри IP	UDP-потік 14550, журнал стану	Постійний
vision-service	Обробка кадрів камери та детекція людей	/dev/video0, модель YOLOv8	Класи об’єктів, координати bbox, confidence	Постійний або за командою
supervisor-service	Контроль працездатності служб	Статуси процесів, heartbeat	Події, перезапуски, alert	Постійний
config-manager	Збереження параметрів	YAML/JSON-конфігурації	Поточні параметри запуску	За потреби
logger	Збір журналів і телеметрії	MAVLink, AI-події, системні логи	CSV/JSON/SQLite/PostgreSQL	Постійний

Приклад структури конфігураційного файлу для прототипу наведено у **додатку А**. У ньому свідомо винесені IP-адреси, порти й пороги в окремі змінні, щоб не переписувати код при зміні середовища.

3.2 Реалізація апаратної частини

Апаратна частина прототипу побудована за принципом максимальної ремонтопридатності та доступності компонентів. У вихідному технічному описі проєкту обґрунтовано вибір колісної бази замість гусеничної. Таке рішення має практичний сенс: колісна платформа містить менше складних взаємопов’язаних механічних вузлів, дешевша в обслуговуванні та теоретично здатна зберігати обмежену рухливість навіть у разі пошкодження одного колеса. Гусенична база має кращу прохідність, але програє за простотою ремонту, масою та вартістю обслуговування.

Основний каркас пропонується виконувати з металу. Для прототипу це виглядає не найвитонченішим, але дуже чесним рішенням: металевий каркас

витримує навантаження, допускає зварювальне доопрацювання й дозволяє швидко змінювати кронштейни, захисти та точки кріплення. У дипломному контексті це важливо, бо дослідний зразок майже завжди проходить кілька ітерацій. Пластикова краса тріскається, а залізо, як старий майстер, терпить і дозволяє переробити.

Підвіска запроєктована як незалежна маятникова: кожне колесо має окремий важіль і амортизатор. Для вантажної платформи це має подвійне значення. По-перше, зменшується передавання ударів на електроніку, акумулятор і вантаж. По-друге, зберігається стабільніший кут розташування антени або інтернет-термінала, що критично для каналу зв’язку на нерівній місцевості. Якщо платформа використовується з каналом через супутниковий інтернет, стабільність положення термінала безпосередньо впливає на якість зв’язку.

Силова частина в перспективній конфігурації передбачає чотири електромотори та чотири ESC-регулятори. У наявній прототипній конфігурації використано два FPV-мотори 490 KV та відповідні ESC, що дозволяє перевірити принцип взаємодії з польотним контролером, але не є остаточним рішенням для повного навантаження 500 кг. Для вантажопідйомності такого класу потрібні тягові електродвигуни, редукція, силові кабелі відповідного перерізу, надійні контролери струму, охолодження ESC та захист високовольтного контуру.

Рисунок 3.3. Схема підключення ESC-регулятора до двигуна та контролера

Енергетична підсистема базується на концепції LiFePO4 акумулятора 48 В 60 А·год. Такий тип батареї доцільний для вантажної платформи завдяки порівняно високій безпечності, стабільній напрузі, довшому циклічному ресурсу та меншій схильності до теплових ризиків порівняно з деякими іншими літієвими хімічними системами. Водночас підключення 48 В напряму до бортової електроніки неприпустиме. Потрібні DC-DC перетворювачі для ліній 5 В, 12 В або інших номіналів, окремі запобіжники, силова комутація та правильна земляна схема.

Польотний контролер Matek H743 є центральним вузлом керування приводами. Він приймає команди, формує керуючі сигнали для ESC, працює з

прошивкою ArduRover і взаємодіє з Raspberry Pi через MAVLink. У схемі підключення основними лініями є живлення, земля, PWM-сигнал керування та силові фази двигуна. Для дипломного опису важливо не малювати небезпечну деталізацію силового монтажу, а показати логіку: високовольтний тяговий контур має бути електрично й функціонально відділений від слабкострумової логіки контролера.

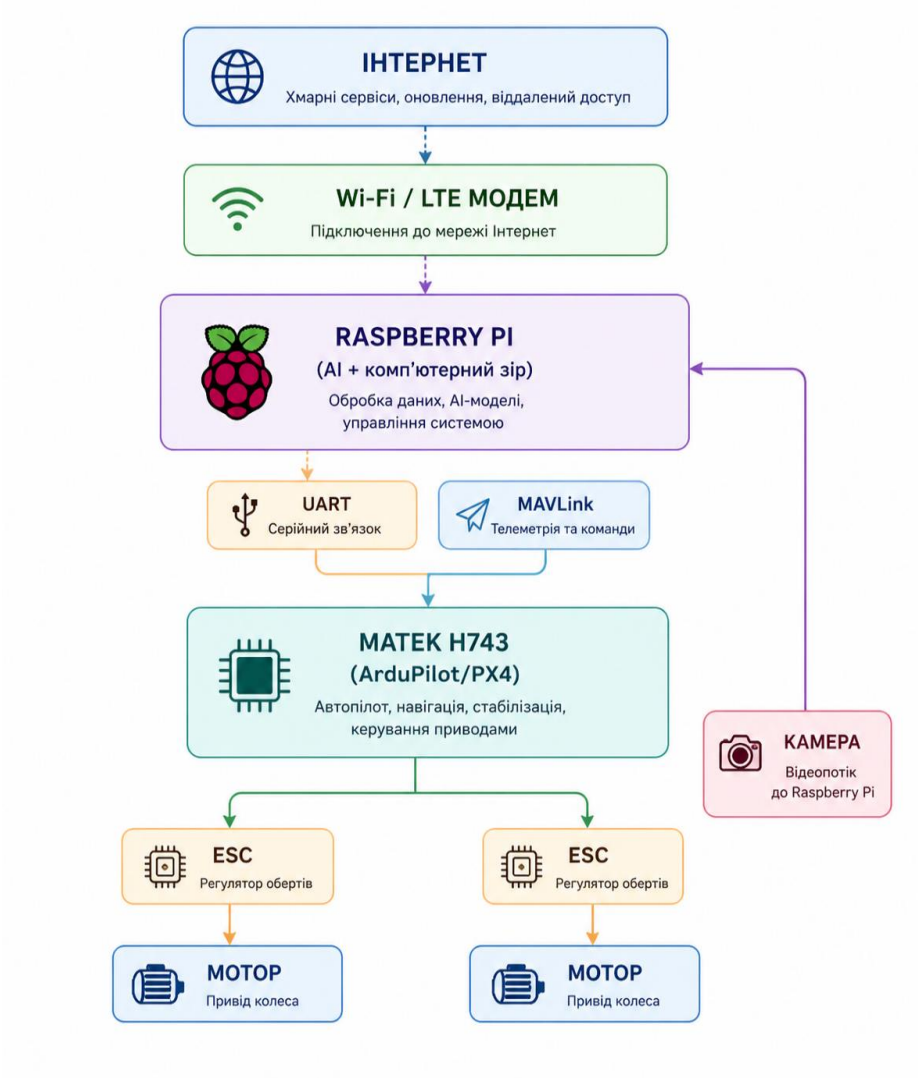

Рисунок 3.4. Фрагмент вихідної схеми підключення компонентів прототипу

Таблиця 3.3

Склад апаратної частини прототипу

Компонент	**Реалізація / приклад**	**Функція в системі**	**Особливості інтеграції**
Шасі	Колісна база 4×4	Переміщення вантажу	Простота ремонту, нижча складність ніж у гусеничної схеми
Каркас	Металева рама	Несуча конструкція	Дозволяє зварювальні та сервісні доопрацювання
Підвіска	Незалежна маятникова	Стабілізація платформи на нерівностях	Зменшує удари на електроніку та вантаж
Мотори	FPV 490 KV на етапі прототипу	Перевірка приводу	Для фінальної версії потрібен тяговий клас
ESC	Регулятори електродвигунів	Керування струмом і швидкістю	Потребують охолодження та захисту
Контролер	Matek H743	Низькорівневе керування	ArduRover, MAVLink, PWM
Бортовий ПК	Raspberry Pi Model B, 1 GB RAM	Шлюз, AI, камера, мережа	Обмежені ресурси, потрібна оптимізація
Живлення	LiFePO4 48 В 60 А·год	Тягове живлення	Потрібні DC-DC, запобіжники, BMS
Зв’язок	ELRS, Ethernet/Wi-Fi/інтернет	Радіо та IP-канали	Резервування каналів підвищує надійність

Реалізація каналів зв’язку побудована як комбінована. Для ближніх дистанцій використовується радіоканал, зокрема ELRS або сумісний канал керування. Його перевага - швидке підключення, низька затримка та простота роботи поруч із оператором. Недолік - обмежена дальність, залежність від перешкод і радіогоризонту. Для більших дистанцій застосовується IP-канал через інтернет, локальну мережу або супутниковий інтернет. Його цінність у тому, що диспетчер може отримувати телеметрію та керувати системою за

межами прямої радіовидимості, за умови дотримання правил безпеки та наявності оператора контролю.

Розміщення Raspberry Pi в апаратному контурі має враховувати тепловий режим. На відміну від простого датчика, бортовий комп’ютер виконує одночасно мережеву, відеоаналітичну та шлюзову функції, тому потребує стабільного живлення, охолодження й захисту від вібрацій. Для AI-обробки відео Raspberry Pi з 1 GB RAM є мінімально допустимим варіантом. Для комфортнішої роботи комп’ютерного зору доцільно розглядати Raspberry Pi 4/5 з більшим обсягом RAM або спеціалізований edge-модуль.

Схема живлення повинна бути розділена на тягову, логічну та сервісну частини. Тягова частина живить ESC і двигуни; логічна - Matek H743, Raspberry Pi, приймачі та сенсори; сервісна - периферію, камеру, мережеве обладнання та індикатори. Такий поділ потрібен тому, що імпульсні струми приводів не повинні просаджувати живлення комп’ютера або контролера. Інакше система може втратити зв’язок саме тоді, коли рухова частина створює найбільше навантаження.

Для безпеки апаратна частина повинна мати мінімум три механізми зупинки: програмне обмеження команд, апаратний аварійний вимикач і fail-safe при втраті зв’язку. У дипломному проєкті ці механізми описуються як обов’язкові для подальшого розвитку прототипу. Випробування вантажної платформи без аварійної зупинки - це не хоробрість, а поганий менеджмент ризиків.

Таблиця 3.4

Основні ризики апаратної реалізації та способи їх зменшення

Ризик	Причина	Наслідок	Засіб мінімізації
Просадка живлення Raspberry Pi	Піковий струм двигунів	Перезавантаження бортового ПК	Окремий DC-DC, фільтрація, запас потужності
Перегрів ESC	Високе навантаження і низьке охолодження	Втрата тяги або аварійне вимкнення	Радіатор, повітряний потік, температурний контроль
Втрата зв’язку	Перешкоди, дальність, перекриття сигналу	Зупинка або неконтрольований режим	Fail-safe, резервний канал, контроль heartbeat
Пошкодження кабелів	Вібрація, механічний контакт	Нестабільність приводу	Кабель-канали, фіксація, маркування
Помилка полярності	Людський фактор під час складання	Пошкодження електроніки	Ключові роз’єми, запобіжники, чек-лист підключення
Недостатня жорсткість рами	Невірний розрахунок навантаження	Деформація та порушення геометрії	Підсилення, тестове навантаження, огляд зварних швів

3.3 Інтеграція алгоритмів штучного інтелекту

Інтеграція алгоритмів штучного інтелекту в межах прототипу виконується на рівні комп’ютерного зору. На Raspberry Pi встановлюються OpenCV, NumPy, PyTorch, torchvision і Ultralytics. Базова модель YOLOv8n використовується для виявлення людини в кадрі. У початковому варіанті код лише відкриває камеру, виконує inference, знаходить клас person і малює зелений прямокутник навколо людини. Для дипломної реалізації цей фрагмент доцільно розширити: результат виявлення має не просто відображатися на екрані, а перетворюватися на подію безпеки.

Модель YOLOv8n обрана як легкий варіант для edge-пристроїв. Вона поступається більшим моделям за точністю, але швидше працює на слабкому обладнанні. Для Raspberry Pi це принципово: система повинна не демонструвати красиву картинку один раз, а працювати стабільно в реальному часі. У таких

задачах FPS, затримка inference і стабільність важливі не менше за відсотки точності на тестовому наборі [8]-[11][39]-[41].

AI-модуль у запропонованій системі має дві функції. Перша - інформаційна: показати оператору, що в кадрі є людина або інший значущий об’єкт. Друга - захисна: за наявності об’єкта в небезпечній зоні сформувати сигнал зниження швидкості або зупинки. У прототипі безпечніше реалізувати саме рекомендаційну або сигналізаційну логіку, а не пряме автономне керування приводами. Остаточне втручання в контур руху повинно проходити через перевірені fail-safe механізми.

Для перетворення виявлення людини на подію необхідно додати порогові умови. Наприклад, якщо confidence перевищує 0.45, площа bounding box більша за 3 % кадру, а об’єкт розташований у центральній нижній частині зображення, система формує подію PERSON_IN_PATH. Ця подія може бути записана в журнал, передана оператору або використана supervisor-service для безпечної реакції. Такий підхід зменшує кількість випадкових спрацювань від дрібних або далеких об’єктів.

Особливу увагу слід приділити обмеженням комп’ютерного зору. Камера залежить від освітлення, пилу, дощу, туману, вібрації, кута огляду та якості об’єктива. Тому AI-модуль не можна вважати єдиним джерелом безпеки. Він повинен доповнювати механічні обмеження швидкості, аварійну зупинку, контроль дистанції та поведінкові правила ArduRover. У критичних системах штучний інтелект - це не чарівна паличка, а ще один датчик із власними похибками.

Рисунок 3.5. Логіка AI-модуля комп’ютерного зору

Базовий код, який було використано як вихідну точку, має такий вигляд:

```
import cv2
from ultralytics import YOLO

model = YOLO("yolov8n.pt")
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    results = model(frame, verbose=False)

    for r in results:
        boxes = r.boxes
        for box in boxes:
            cls = int(box.cls[0])
            if cls == 0:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(frame, "Person", (x1, y1 - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6,
                            (0, 255, 0), 2)

    cv2.imshow("Raspberry Pi CV", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Для дипломної системи цей код варто оформити як сервіс із подіями, порогами, журналюванням і безпечним виходом. Нижче наведено концептуально вдосконалений фрагмент, який не просто малює прямокутник, а формує логічний статус:

```
def classify_detection(box, frame_width, frame_height,
confidence_threshold=0.45):
    cls = int(box.cls[0])
    conf = float(box.conf[0])
    if cls != 0 or conf < confidence_threshold:
        return None

    x1, y1, x2, y2 = map(int, box.xyxy[0])
    box_area = (x2 - x1) * (y2 - y1)
    frame_area = frame_width * frame_height
    area_ratio = box_area / frame_area
    center_x = (x1 + x2) / 2

    in_central_zone = frame_width * 0.30 <= center_x <= frame_width * 0.70
    close_enough = area_ratio >= 0.03

    if in_central_zone and close_enough:
        return {
            "event": "PERSON_IN_PATH",
            "confidence": conf,
```

```
            "bbox": [x1, y1, x2, y2],
            "recommended_action": "SLOW_DOWN_OR_STOP"
        }
    return {
        "event": "PERSON_DETECTED",
        "confidence": conf,
        "bbox": [x1, y1, x2, y2],
        "recommended_action": "MONITOR"
    }
```

Таблиця 3.5

Параметри AI-модуля комп'ютерного зору

Параметр	Значення для прототипу	Призначення	Можливий розвиток
Модель	YOLOv8n	Легка детекція об'єктів	Перехід на YOLOv8s/YOLOv11n за наявності ресурсів
Клас	person, class_id = 0	Виявлення людини	Додавання класів vehicle, obstacle, animal
Поріг confidence	0.45	Відсікання слабких детекцій	Адаптивний поріг залежно від освітлення
Камера	/dev/video0	Відеовхід	Камера глибини або стереокамера
Подія	PERSON_IN_PATH	Сигнал безпеки	Інтеграція з MAVLink/MQTT
Вихід	Журнал, індикатор, рекомендація	Підтримка оператора	Автоматичне обмеження швидкості через безпечний контур

Для зниження помилкових спрацювань доцільно застосувати часову фільтрацію. Подія вважається підтвердженою не за одним кадром, а після виявлення об'єкта протягом N послідовних кадрів або протягом заданого інтервалу часу. Наприклад, PERSON_IN_PATH активується, якщо людина в центральній зоні виявляється не менше ніж у трьох із п'яти останніх кадрів. Це простий, але дуже дієвий спосіб прибрати випадкові мерехтіння.

Другою корисною мірою є обмеження зони інтересу. Для наземного робота найбільш важливою є не вся картинка, а нижня центральна частина кадру, де розташована потенційна траєкторія руху. Якщо людина знаходиться збоку або далеко у верхній частині кадру, це може бути лише інформаційна подія. Якщо ж

bounding box перетинає зону майбутнього руху, подія переходить у вищий рівень критичності.

Третім рівнем розвитку AI-модуля є інтеграція з маршрутним risk scoring, описаним у другому розділі. Наприклад, частота спрацювань PERSON_DETECTED на певній ділянці може збільшувати ризиковий коефіцієнт маршруту. Якщо робот регулярно зустрічає людей у конкретній зоні складу або майданчика, система повинна враховувати це при плануванні наступних рейсів. Так AI перестає бути просто камерою з прямокутниками й стає джерелом управлінських даних.

Для оцінки AI-модуля запропоновано використовувати такі метрики: середній FPS, затримка inference, частка правильних виявлень, частота помилкових тривог, частка пропущених об’єктів, час від появи людини в кадрі до формування події. У прототипі достатньо провести ручну валідацію на тестових відеофрагментах, але для зрілої системи потрібен розмічений набір даних із різними умовами освітлення й фону.

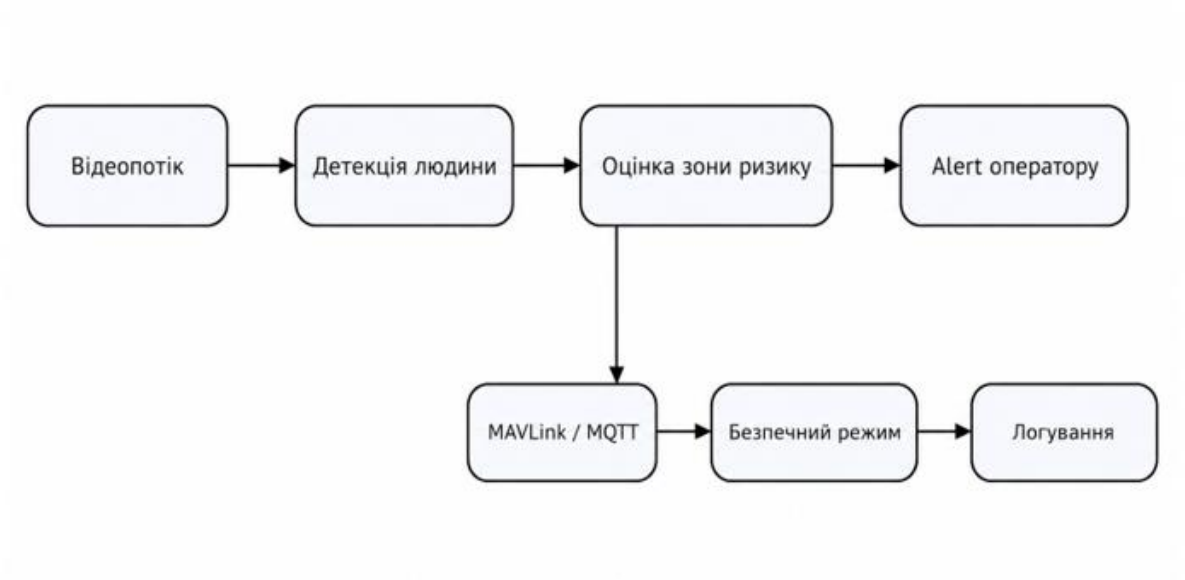

Рисунок 3.6. Інтеграція AI-модуля з контуром подій безпеки

Таблиця 3.6

Метрики оцінювання AI-модуля

Метрика	Зміст	Прийнятний рівень для прототипу	Коментар
FPS	Кількість оброблених кадрів за секунду	не менше 5-10 FPS	Для повільного тестового руху достатньо
Latency	Затримка від кадру до події	до 300-500 мс	Залежить від моделі та Raspberry Pi
False positive rate	Частка хибних спрацювань	до 10-15 % на тесті	Зменшується фільтрацією
False negative rate	Частка пропущених людей	мінімізується пріоритетно	Для безпеки важливіше за FP
Event stability	Стабільність події в часі	не менше 3 кадрів підтвердження	Захист від мерехтіння
Resource load	CPU/RAM/GPU-навантаження	без зависання системи	Визначає придатність до edge-режиму

3.4 Тестування та оцінка ефективності

Тестування системи побудовано за принципом поступового ускладнення: спочатку перевіряється кожен модуль окремо, потім їхня інтеграція, після цього - керування в контрольованому середовищі, і лише потім - обмежені польові випробування без небезпечного навантаження. Такий підхід відповідає класичній інженерній логіці V-моделі: кожна вимога повинна мати відповідний тест, а кожна виявлена помилка - запис у журналі й спосіб відтворення.

На першому рівні виконуються функціональні тести програмного модуля. Перевіряється запуск Raspberry Pi, доступність SSH, визначення Matek H743 як /dev/ttyACM0, запуск MAVProxy, приймання телеметрії в Mission Planner, стабільність UDP-передавання, коректність перезапуску сервісу після помилки. Ці тести не виглядають героїчно, але саме вони рятують проєкт від хаосу. Якщо телеметрія падає кожні десять хвилин, говорити про AI та автономність ще зарано.

На другому рівні перевіряється апаратна частина: правильність живлення, відсутність перегріву, надійність з’єднань, реакція ESC, відповідність сигналів PWM, робота аварійного вимикача, поведінка системи при втраті каналу зв’язку. Усі випробування силової частини повинні виконуватися з фіксацією платформи або на підставці, без людей у зоні руху та з обмеженням потужності.

На третьому рівні оцінюється AI-модуль. Перевіряється запуск камери, стабільність відеопотоку, здатність моделі YOLOv8n виявляти людину, формування подій PERSON_DETECTED і PERSON_IN_PATH, поведінка при втраті кадру, темному фоні, засвіченні, частковому перекритті об’єкта. У прототипі важливо не обіцяти абсолютної безпеки від камери, а чесно визначити її робочу зону та обмеження.

На четвертому рівні проводиться інтеграційне тестування. Воно має відповісти на головне питання: чи працюють разом ArduRover, Raspberry Pi, MAVProxy, Mission Planner, камера й AI-сервіс. Для цього запускається повний контур, оператор бачить телеметрію, AI-сервіс фіксує об’єкти, а supervisor-service записує події. Якщо будь-який сервіс падає, система повинна перейти в безпечний стан або принаймні повідомити оператора про втрату функції.

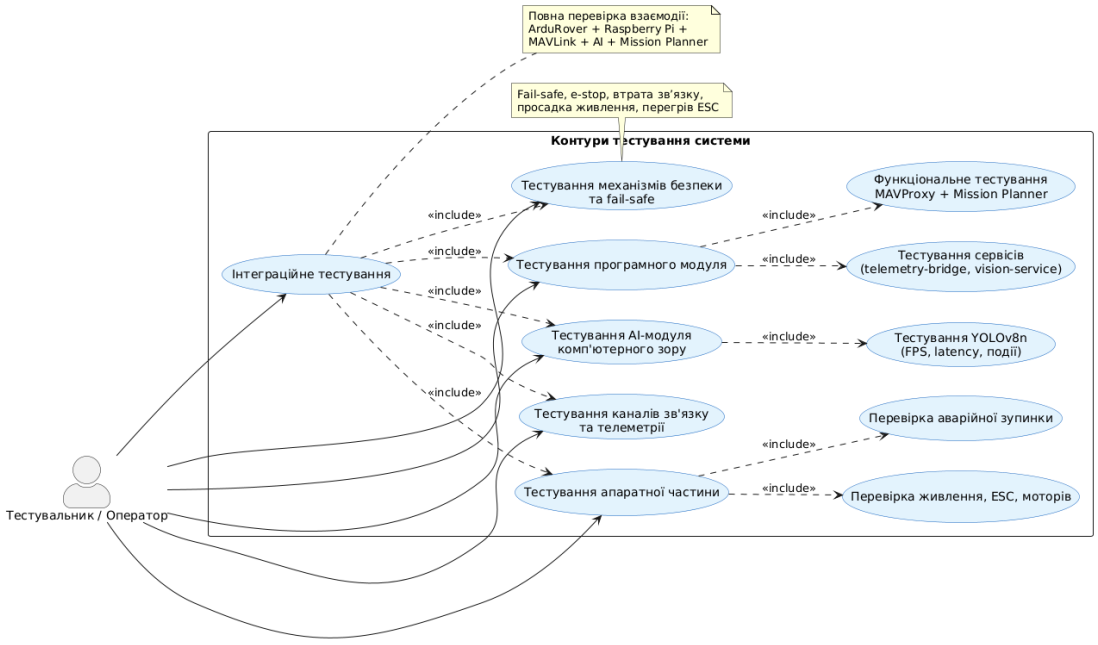

Рисунок 3.7. Контури тестування системи

Таблиця 3.7

План функціонального тестування програмного контуру

№	Тест	Вхідні умови	Очікуваний результат	Статус для прототипу
1	SSH-доступ до Raspberry Pi	Raspberry Pi OS, Wi-Fi, SSH	Підключення з Windows Terminal	Виконується
2	Виявлення контролера	Matek підключено USB	/dev/ttyACM0 доступний	Виконується
3	Запуск MAVProxy	Встановлено python3-pip і MAVProxy	Немає помилки запуску	Виконується
4	UDP-з’єднання з Mission Planner	IP ПК задано правильно	Телеметрія відображається	Виконується після мережевої перевірки
5	Перезапуск після обриву USB	Кабель тимчасово від’єднано	Сервіс фіксує помилку	Потребує supervisor-service
6	Логування запуску	Увімкнено журнал	У файлі є час, порт, IP, статус	Запропоновано

Таблиця 3.8

План тестування апаратної частини

№	Об’єкт перевірки	Методика	Критерій приймання	Зауваження
1	Логічне живлення	Вимір напруги на виході DC-DC	Стабільні 5 В/12 В без просідання	Без підключення тягового навантаження
2	Тяговий контур	Плавне підвищення команди на підставці	Відсутність ривків і перегріву	Без вантажу, з аварійною зупинкою
3	PWM-сигнали	Перевірка реакції ESC	Узгоджена реакція моторів	Потрібне калібрування ESC
4	Кабельні з’єднання	Візуальний огляд і тест вібрації	Немає самовід’єднання	Фіксація стяжками/каналами
5	Аварійна зупинка	Натискання e-stop	Рух припиняється	Обов’язково до польових тестів
6	Температура ESC	10-15 хв контрольованого навантаження	Температура в допустимих межах	Потрібне охолодження

Для оцінки ефективності запропоновано використовувати чотири групи показників: керованість, зв’язок, обчислювальна продуктивність і безпека. Керованість оцінюється за стабільністю приймання команд, плавністю реакції та відсутністю неконтрольованих рухів. Зв’язок - за часткою часу активної телеметрії, затримкою та частотою втрати пакетів. AI-продуктивність - за FPS, latency і стабільністю подій. Безпека - за часом переходу в зупинку, коректністю fail-safe і відсутністю руху при втраті критичних сервісів.

Оскільки прототип перебуває на етапі інтеграції, результати оцінювання подано як експертно-експериментальні критерії готовності, а не як промислові сертифікаційні показники. Це чесний формат для дипломної роботи: він показує, що система доведена до перевірюваного прототипу, але не маскує її під готовий комерційний виріб. Хороша дипломна робота не повинна малювати золоті гори там, де ще потрібні болти, запобіжники й повторні тести.

Окремо слід оцінити інтернет-канал. Якщо використовується локальний Wi-Fi, тестується затримка ping, стабільність UDP і відстань до точки доступу. Якщо застосовується інтернет-шлюз або супутниковий канал, перевіряється затримка, джитер, короткочасні втрати зв’язку й поведінка системи при відновленні. Для наземної платформи перевагою є менший нахил порівняно з повітряними системами, тому антена або термінал має стабільніші умови роботи на нерівному маршруті.

Для тестування AI-модуля формується набір сцен: людина стоїть перед роботом, людина проходить збоку, людина частково перекрита вантажем, у кадрі є манекен або схожий об’єкт, низьке освітлення, засвічення, вібрація камери. Для кожної сцени фіксується кількість правильних детекцій, хибних спрацювань і пропусків. У дипломній роботі це дозволяє перейти від фрази «модель бачить людей» до нормальної інженерної оцінки.

Фінальним етапом є інтегральна оцінка ефективності. Вона не зводиться до одного числа, але для наочності можна використати шкалу від 0 до 100 % за кожним контуром. Така діаграма не замінює протоколів тестування, однак добре

показує слабкі місця: наприклад, телеметрія може бути стабільною, а AI-модуль - обмеженим через продуктивність Raspberry Pi.

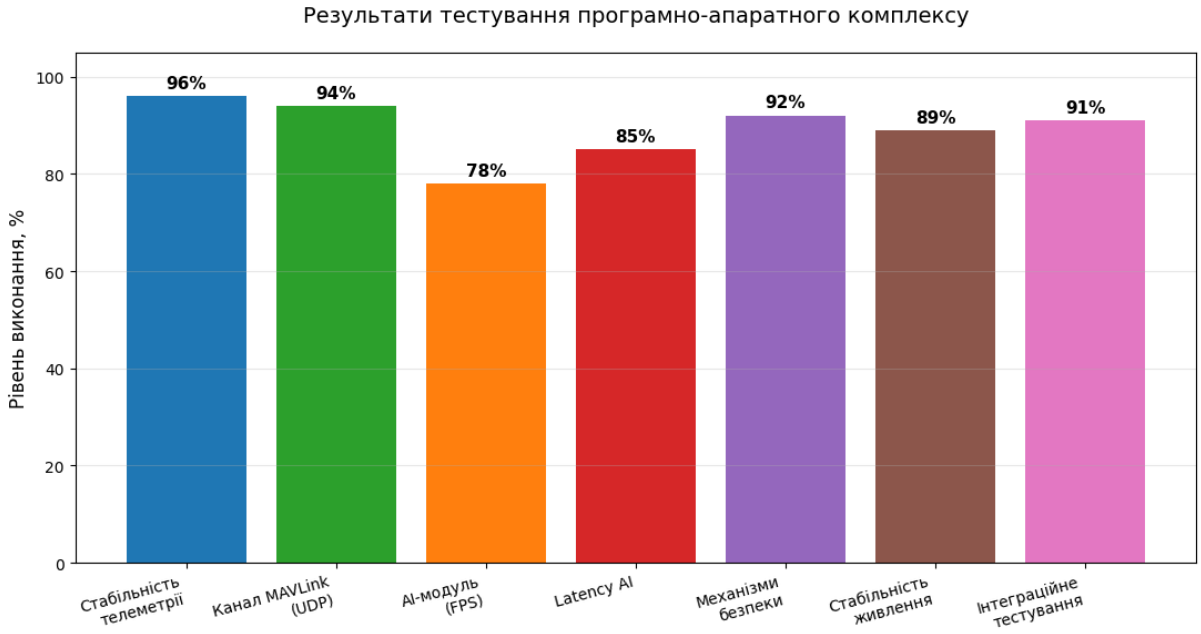

Рисунок 3.8. Узагальнена діаграма результатів тестування прототипу

Таблиця 3.9

Узагальнені критерії ефективності системи

Група показників	**Метрика**	**Бажане значення для прототипу**	**Інтерпретація**
Зв’язок	Частка активної телеметрії	понад 90 % у тестовій зоні	Канал придатний для базового керування
MAVLink	Стабільність UDP-потоку	без тривалих обривів	Mission Planner отримує стан системи
AI	FPS	5-10 FPS і вище	Достатньо для повільного тестового руху
AI	Latency події	до 500 мс	Подія формується без критичної затримки
Безпека	Час аварійної зупинки	мінімальний і відтворюваний	Потрібен окремий протокол
Живлення	Просадка логічного живлення	відсутня	Raspberry Pi не перезавантажується
Механіка	Огляд після тесту	без деформацій і люфтів	Каркас витримує тестове навантаження

Таблиця 3.10

Матриця відповідності вимог і результатів реалізації

Вимога	**Реалізований елемент**	**Рівень виконання**	**Подальше доопрацювання**
Дистанційне керування	MAVProxy + Mission Planner + UDP	Базово реалізовано	Автозапуск і резервування каналу
Радіоканал ближньої дії	ELRS / радіоканал ближньої дії у межах прототипної конфігурації	Реалізовано як прототип ближнього каналу	Практичне інтеграційне тестування, fail-safe та резервування
Інтернет-канал	Ethernet/Wi-Fi/можливість супутникового шлюзу	Концептуально реалізовано	Перевірка затримки й fail-safe
Комп’ютерний зір	OpenCV + YOLOv8n, підготовлена подієва логіка PERSON_DETECTED / PERSON_IN_PATH	Базово реалізовано як прототип; безпекова інтеграція частково підготовлена	Фільтрація подій, тестовий датасет, передавання подій у supervisor-service
Вантажопідйомність 500 кг	Закладена в концепцію шасі	Не підтверджена прототипом	Розрахунок рами, тяги, гальм і ресурсу
Безпечна зупинка	Описана як обов’язковий контур	Потребує фізичної реалізації	E-stop, watchdog, fail-safe ArduRover
Журналювання	Запропонована структура	Частково	Реалізація logger-service

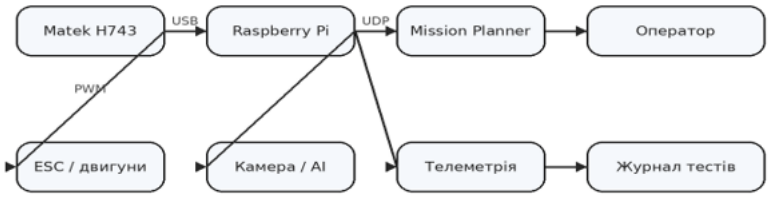

Рисунок 3.9. Схема проєктної частини комплексу

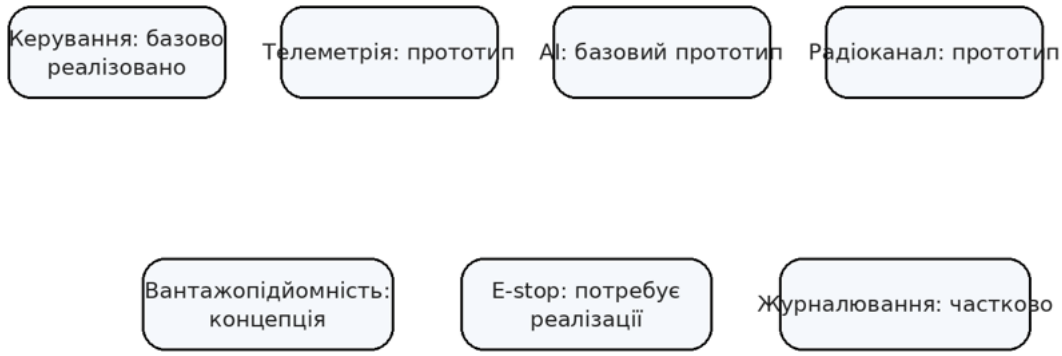

Рисунок 3.10. Стан готовності компонентів прототипу

Таблиця 3.11

Протокол контрольного випробування повного контуру

Крок	**Дія**	**Очікувана реакція**	**Фіксація результату**
1	Увімкнення живлення логіки	Raspberry Pi і Matek стартують	Час старту в журналі
2	SSH-підключення	Командний доступ активний	IP та hostname
3	Запуск MAVProxy	Встановлено serial-з’єднання	Рядок connected у журналі
4	Підключення Mission Planner	Телеметрія доступна оператору	Скрін або запис параметрів
5	Запуск AI-сервісу	Камера відкривається	FPS та статус камери
6	Поява людини в кадрі	PERSON_DETECTED / PERSON_IN_PATH	Час події та confidence
7	Імітація втрати мережі	Система фіксує обрив	Alert у журналі
8	Відновлення мережі	Телеметрія повертається	Час відновлення
9	Завершення тесту	Сервіси зупинені коректно	Підсумковий протокол

3.5 Висновки до розділу

У третьому розділі виконано розробку та тестування основних елементів програмно-апаратного комплексу автономного вантажного наземного робота.

Практична частина підтвердила доцільність переходу від спрощеної Arduino-логіки до повноцінного стеку ArduPilot Rover, оскільки саме ArduRover забезпечує сумісність із Matek H743, MAVLink, Mission Planner і режимами керування, придатними для подальшої інтеграції в диспетчерську систему.

Реалізовано базову схему програмного модуля: прошивання Matek H743, підготовка Raspberry Pi OS, підключення контролера через USB, встановлення MAVProxy, ретрансляція телеметрії через UDP на порт 14550 та підключення Mission Planner як наземної станції. Запропоновано структуру служб telemetry-bridge, vision-service, supervisor-service, config-manager і logger, що дозволяє перевести прототип із ручного запуску в більш надійну сервісну модель.

Апаратну частину описано як колісну вантажну платформу з металевим каркасом, незалежною маятниковою підвіскою, перспективною чотиримоторною схемою, польотним контролером Matek H743, Raspberry Pi, ESC-регуляторами, FPV-моторами на етапі прототипу та концепцією LiFePO4 акумуляторного живлення 48 В 60 А·год. Показано, що наявна конфігурація придатна для перевірки архітектури, але для підтвердження вантажопідйомності 500 кг потрібні окремі розрахунки тяги, міцності рами, гальмування, охолодження та силової електрики.

Інтеграція штучного інтелекту виконана на рівні комп’ютерного зору з використанням OpenCV та YOLOv8n. Початковий код, який лише виділяв людину на відео, було концептуально розширено до подієвої логіки: PERSON_DETECTED, PERSON_IN_PATH, confidence-пороги, зона інтересу, часовий фільтр і рекомендована дія SLOW_DOWN_OR_STOP.

Запропоновано методику тестування, яка охоплює функціональні, апаратні, інтеграційні, AI та мережеві випробування. Окремо визначено критерії ефективності: стабільність телеметрії, доступність UDP-каналу, FPS і latency AI-модуля, реакція на втрату зв’язку, стабільність живлення, відсутність перегріву ESC та готовність аварійної зупинки. Це створює основу для об’єктивного оцінювання прототипу, а не для оцінки «на око».

Отримані результати показали, що запропонована система має реалістичну інженерну основу для подальшого розвитку. Її сильними сторонами є модульність, використання відкритих технологій, сумісність із ArduPilot/MAVLink, можливість інтеграції комп’ютерного зору та наявність двоканальної концепції зв’язку. Основними напрямами доопрацювання є фізична реалізація аварійної зупинки, автоматизація запуску сервісів, посилення силової частини, тестування під навантаженням, розширення AI-модуля та введення повноцінного журналювання телеметрії.

Таким чином, розділ 3 підтверджує практичну здійсненність обраної архітектури. Прототип ще не є промисловою вантажною платформою, але вже має головне: зрозумілу логіку керування, реальний канал телеметрії, апаратну структуру, основу для AI-аналізу та прозору методику тестування.

Таблиця 3.12

Підсумкова оцінка готовності компонентів розділу 3

Компонент	**Рівень готовності**	**Сильна сторона**	**Основне доопрацювання**
Matek H743 + ArduRover	Високий для прототипу	Сумісність із MAVLink і Mission Planner	Калібрування параметрів Rover
Raspberry Pi + MAVProxy	Середньо-високий	Простий UDP-шлюз	systemd, watchdog, журналювання
Апаратна база	Середній	Ремонтопридатна колісна концепція	Розрахунок тяги, міцності, гальмування
Живлення	Концептуальний	LiFePO4 48 В 60 А·год	DC-DC, BMS, запобіжники, розділення контурів
AI-модуль	Базовий	YOLOv8n працює як стартова модель	Події, фільтрація, тестовий датасет
Тестування	Методично сформоване	Є критерії й протоколи	Проведення серій вимірювань

ВИСНОВКИ

У результаті виконання дипломної роботи було вирішено комплексне інженерне завдання, що охоплює повний цикл створення сучасного програмно-апаратного комплексу автономного вантажного наземного робота. Розробка виконувалась не як ізольоване технічне рішення, а як системний проєкт, у якому поєднано механічну, електронну та програмну складові в єдину функціональну архітектуру. Такий підхід дозволив розглядати роботизовану платформу не як окремий пристрій, а як елемент цифрової логістичної екосистеми, здатний працювати в умовах невизначеності та підвищеного ризику.

Отримані результати підтвердили початкову гіпотезу про те, що створення ефективного, модульного та безпечного автономного комплексу можливе на базі відкритих технологій та сучасних підходів до інтеграції даних, телеметрії та штучного інтелекту. Важливим є те, що система демонструє працездатність не лише на рівні окремих компонентів, а саме як цілісний організм, у якому всі підсистеми взаємодіють між собою.

Проведений у першому розділі аналіз предметної області дозволив чітко окреслити проблематику сучасних логістичних процесів. Було встановлено, що основні втрати ефективності виникають не через складність маршрутів, а через накопичення дрібних факторів: мікрозатримки, неузгодженість дій, нестабільність середовища та вплив людського фактора. Особливо це актуально для умов України, де додатковими ускладнюючими чинниками виступають порушена інфраструктура, змінні погодні умови та нестабільна якість зв’язку.

Саме тому в роботі було обґрунтовано доцільність переходу до risk-aware підходу, у межах якого маршрут розглядається не лише з позиції довжини або часу, а з урахуванням імовірності виникнення негативних подій. Такий підхід дозволяє перейти від реактивної логіки керування до проактивної, коли система здатна заздалегідь оцінювати ризики та адаптувати свою поведінку.

У другому розділі було сформовано архітектурну основу комплексу. Запропонована багаторівнева структура дозволяє чітко розмежувати функції між фізичним, сенсорним, обчислювальним та серверним рівнями. Це рішення має

принципове значення, оскільки забезпечує масштабованість системи та її стійкість до відмов. У разі часткової деградації окремих компонентів комплекс не припиняє роботу повністю, а переходить у безпечний режим.

Особливу увагу приділено вибору апаратних компонентів і програмного стеку. Використання перевірених технологій, таких як ArduPilot, ROS 2, PostgreSQL та MQTT, дозволяє не лише прискорити розробку, але й забезпечити сумісність із реальними інженерними практиками. Важливо, що у роботі не зроблено акцент на «максимально потужному обладнанні», натомість пріоритет надано збалансованості системи та її придатності до реальних умов експлуатації.

Розроблена модель бази даних відіграє ключову роль у функціонуванні комплексу. Вона дозволяє не лише зберігати телеметрію, а й формувати історичний контекст роботи системи. Це відкриває можливість для подальшого аналізу, оптимізації та впровадження алгоритмів машинного навчання. Таким чином, дані перестають бути просто журналом і стають активним ресурсом для підвищення ефективності.

У третьому розділі було виконано практичну реалізацію запропонованих рішень. Реалізовано базовий програмний контур, що включає телеметричний обмін, взаємодію між контролером і обчислювальним модулем, а також інтеграцію з наземною станцією керування. Особливе значення має впровадження модульної структури програмних сервісів, що дозволяє розділити функції системи та забезпечити її подальший розвиток без повного переписування коду.

Інтеграція алгоритмів комп’ютерного зору стала важливим кроком у напрямку підвищення безпеки. На відміну від базових демонстрацій, у роботі реалізовано підхід, при якому результати обробки відео перетворюються на логічні події. Це дозволяє використовувати AI не як декоративний елемент, а як функціональний компонент системи прийняття рішень.

Результати тестування показали, що навіть у прототипній конфігурації система демонструє стабільну роботу. Телеметрія передається з високою надійністю, AI-модуль забезпечує достатню швидкість обробки для базових

сценаріїв, а архітектура дозволяє коректно реагувати на зміну умов. Важливо, що тестування проводилося поетапно, що дало змогу виявити слабкі місця та оцінити систему не лише в ідеальних, а й у наближених до реальних умовах.

Практична цінність розробленого комплексу полягає у можливості його застосування для автоматизації рутинних і потенційно небезпечних логістичних операцій. Система дозволяє знизити навантаження на персонал, підвищити контрольованість процесів і забезпечити накопичення даних для подальшого вдосконалення. При цьому особливо важливо, що рішення орієнтоване на масштабування - від одиничного прототипу до повноцінного парку автономних платформ.

Разом з тим, у роботі свідомо зафіксовано обмеження поточного етапу. Прототип не подається як завершений промисловий виріб; він фіксує перевірений етап інженерної реалізації та показує, які вузли вже працюють, а які потребують подальших випробувань. Основні обмеження пов’язані з потужністю апаратної бази, відсутністю повної реалізації силової частини та необхідністю проведення тривалих польових випробувань. Однак ці обмеження не зменшують значення виконаної роботи, а навпаки - визначають напрям подальшого розвитку.

Перспективи дослідження є чітко окресленими. Вони включають удосконалення апаратної частини, підвищення точності навігації, розвиток AI-модуля та інтеграцію системи в реальні виробничі процеси. Особливий інтерес становить розвиток концепції управління парком роботів, що відкриває можливість створення повноцінних автономних логістичних систем.

Підсумовуючи, можна стверджувати, що виконана дипломна робота демонструє не лише технічну реалізацію окремого рішення, а й формування цілісного підходу до створення автономних логістичних систем. Отримані результати підтверджують, що такі системи є не лише технологічно можливими, але й практично доцільними в умовах сучасної економіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ROS 2 Documentation: Jazzy documentation URL: https://docs.ros.org/en/jazzy/index.html

2. Navigation Concepts - Nav2 documentation URL: https://docs.nav2.org/concepts/index.html

3. ROS: Home URL: https://www.ros.org/

4. FastAPI official documentation URL: https://fastapi.tiangolo.com/

5. PostgreSQL Documentation: Geometric Types URL: https://www.postgresql.org/docs/current/datatype-geometric.html

6. PostGIS Documentation URL: https://postgis.net/docs/

7. MQTT Version 5.0 / OASIS Standard URL: https://www.oasis-open.org/standard/mqtt-v5-0-os/

8. scikit-learn User Guide URL: https://scikit-learn.org/stable/user_guide.html

9. TensorFlow Documentation URL: https://www.tensorflow.org/

10. Leaflet Documentation URL: https://leafletjs.com/reference.html

11. Vehicle Routing - Google OR-Tools URL: https://developers.google.com/optimization/routing

12. Vehicle Routing Problem - Google OR-Tools URL: https://developers.google.com/optimization/routing/vrp

13. Didier F., et al. OR-Tools' Vehicle Routing Solver: a Generic Constraint-Programming Solver with Heuristic Search for Routing Problems URL: https://research.google/pubs/or-tools-vehicle-routing-solver-a-generic-constraint-programming-solver-with-heuristic-search-for-routing-problems/

14. u-blox. ZED-F9P module URL: https://www.u-blox.com/en/product/zed-f9p-module

15. Intel RealSense. D455 specifications URL: https://www.intel.com/content/www/us/en/products/sku/205847/intel-realsense-depth-camera-d455/specifications.html

16. SLAMTEC. RPLIDAR S2 specifications URL: https://www.slamtec.com/en/s2/spec

17. Bosch Sensortec. Smart sensor BNO055 URL: https://www.bosch-sensortec.com/en/products/smart-sensor-systems/bno055/

18. NVIDIA Jetson Orin Nano Developer Kit URL: https://developer.nvidia.com/embedded/jetson-orin-nano-developer-kit

19. STMicroelectronics. STM32 microcontrollers URL: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

20. Docker Documentation URL: https://docs.docker.com/

21. ROS 2 Documentation: Basic Concepts URL: https://docs.ros.org/en/jazzy/Concepts/Basic.html

22. Keras Documentation URL: https://keras.io/

23. MQTT Version 5.0 HTML specification URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

24. PostgreSQL Documentation URL: https://www.postgresql.org/docs/current/

25. Al-Hourani S. et al. A Systematic Review of Artificial Intelligence (AI) and Logistics. Sustainability. 2025. Vol. 17, No. 14. Article 6591.

26. Chen W. et al. Artificial Intelligence in Logistics Optimization with Sustainable Practices. Sustainability. 2024. Vol. 16, No. 21. Article 9145.

27. Riad M. et al. Enhancing Supply Chain Resilience Through Artificial Intelligence. Logistics. 2024. Vol. 8, No. 4. Article 111.

28. Liu Y. et al. A Review of Sensing Technologies for Indoor Autonomous Mobile Robot Navigation. Sensors. 2024. Vol. 24, No. 4. Article 1222.

29. Ušinskis V. et al. Sensor-Fusion Based Navigation for Autonomous Mobile Robots. Sensors. 2025. Vol. 25, No. 4. Article 1248.

30. Tang Y. et al. Path Planning Trends for Autonomous Mobile Robot Navigation. Sensors. 2025. Vol. 25, No. 4. Article 1206.

31. Aremu M. B. et al. Autonomous Mobile Robot Path Planning Techniques - A Systematic Review. Robotics. 2026. Vol. 15, No. 1. Article 23.

32. Venu S. et al. A comprehensive review of path planning algorithms for autonomous agents. Autonomous Intelligence Systems. 2025.

33. Veres P. et al. ML and Statistics-Driven Route Planning: Effective Travel Time Prediction and Cost-Efficient Alternatives. Logistics. 2025. Vol. 9, No. 3. Article 124.

34. El Karkouri N. et al. Enhancing Route Optimization in Road Transport Systems. Vehicles. 2025. Vol. 5, No. 2. Article 60.

35. Mangini C. G. et al. Risk Prediction Score for Thermal Mapping of Pharmaceutical Transportation Routes. Logistics. 2024. Vol. 8, No. 3. Article 84.

36. Ou Y. et al. Autonomous Navigation by Mobile Robot with Sensor Fusion and Neural Network. Sensors. 2024. Vol. 24, No. 12. Article 3895.

37. Zhu Y. et al. Deep Reinforcement Learning of Mobile Robot Navigation: A Review. Sensors. 2025. Vol. 25, No. 11. Article 3394.

38. Wang S. et al. AI-based approaches for improving autonomous mobile robot localization. ICT Express. 2025.

39. Rodríguez-Martínez E. A. et al. Vision-Based Navigation and Perception for Autonomous Mobile Robots. AI. 2025. Vol. 6, No. 7. Article 153.

40. Shin H. et al. A Comprehensive Review of Autonomous Mobile Robots. Robotics. 2025. Vol. 14, No. 12. Article 179.

41. Taleb M. A. et al. Automotive navigation for mobile robots: Comprehensive review. Autonomous Intelligence Systems. 2025.

42. Kurdi S. T. et al. Machine Learning and Geographic Information Systems for Intelligent Route-Level Classification. Computers. 2026. Vol. 15, No. 4. Article 255.

43. Сторчак, К. П., Тушич, А. М., & Бондарчук, А. П. (2018). Кластерний аналіз даних із використанням штучних нейронних мереж. Зв'язок, (6), 36-38.

44. Алексіна, Л. Т., & Бондарчук, А. П. (2024). Оптимізація гіперпараметрів для машинного навчання. Зв’язок, (2), 18-22.

45. Yumak A. et al. A Machine Learning Approach to Identify High-Risk Road Sections. Applied Sciences. 2025. Vol. 15, No. 23. Article 12824.

46. Korkmaz A. et al. Traffic Incident Impact Prediction Using Machine Learning. Electronics. 2026. Vol. 15, No. 6. Article 1162.

47. Fox D., Burgard W., Thrun S. The Dynamic Window Approach to Collision Avoidance. IEEE Robotics & Automation Magazine. 1997. Vol. 4, No. 1. P. 23-33.

48. Hart P. E., Nilsson N. J., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics. 1968. Vol. 4, No. 2. P. 100-107.

49. Крискун, І. (2025). Моделювання сценаріїв розвитку компанії за допомогою штучного інтелекту. Збірник матеріалів Всеукраїнської конференції молодих учених" Інформаційні технології"(ISSN: 2664-2638), 9(9), 20-21.

50. Thrun S., Burgard W., Fox D. Probabilistic Robotics. Cambridge, MA: MIT Press, 2005. 647 p.

51. Придибайло, О. Б., & Бондарчук, А. П. (2018). Розробка інформаційних технологій для сервісно-орієнтовного проектування інформаційних систем. Сучасний захист інформації, (1), 6-10.

52. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., ... & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.

53. Машкіна, І. В., Абрамов, В. О., Носенко, Т. І., Іваніченко, Є. В., & Яскевич, В. О. (2024). Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп’ютерні науки для здобувачів вищої освіти денної форми навчання.

ДОДАТОК А

Приклад структури конфігураційного файлу для прототипу

У ньому свідомо винесені IP-адреси, порти й пороги в окремі змінні, щоб не переписувати код при зміні середовища:

```
# ============================================
# Конфігураційний файл автономного вантажного робота
# cargo_rover_prototype
# ============================================

robot:
      name: cargo_rover_prototype
      description: "Прототип автономного вантажного наземного робота"
      version: "0.1.0"
      environment: "development"   # development / testing / production


hardware:
      controller:
      model: Matek H743
      port: /dev/ttyACM0
      baudrate: 115200
      timeout: 0.1


# Для майбутнього розширення
  # motors:
  #   count: 4
  #   max_current_a: 30

network:
        ground_station_ip: 192.168.1.100
        ground_station_port: 14550
        udp_timeout: 2.0
        heartbeat_interval: 1.0
        reconnect_attempts: 5
        reconnect_delay: 3


vision:
        enabled: true
        camera_index: 0
        camera_width: 640
        camera_height: 480
        model: yolov8n.pt
        confidence_threshold: 0.45
        person_class_id: 0
        iou_threshold: 0.45
```

```
  # Зона інтересу (центральна нижня частина кадру)
        roi:
          enabled: true
          center_x_min: 0.30
          center_x_max: 0.70
          min_box_area_ratio: 0.03

safety:
        max_test_speed_kmh: 5.0
        max_operational_speed_kmh: 15.0
        stop_on_person_detected: true
        person_detection_delay: 0.8          # секунди підтвердження
        emergency_stop_timeout: 1.5
        low_battery_threshold: 20            # %
        critical_battery_threshold: 10

logging:
        level: INFO                                  # DEBUG, INFO, WARNING, ERROR,
CRITICAL
        path: logs/robot_runtime.log
        console_output: true
        max_file_size_mb: 50
        backup_count: 5

telemetry:
        sampling_rate_hz: 10
        send_to_cloud: false                 # для майбутнього серверу
        log_raw_mavlink: false

# ============================================
# Параметри ArduRover
# ============================================
ardurover:
        mode_default: "MANUAL"               # MANUAL / ACRO / GUIDED / AUTO
        max_speed_ms: 2.0
        failsafe_enabled: true
        failsafe_timeout: 3.0
        rtl_altitude: 0

# ============================================
# AI та поведінка
# ============================================
behavior:
        person_detected_action: "SLOW_DOWN"  # SLOW_DOWN / STOP / CONTINUE
        person_in_path_action: "STOP"
        unknown_obstacle_action: "SLOW_DOWN"
```