НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК 004.415.2

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Інженерія програмного забезпечення комп'ютерних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система управління рухомою платформою на базі .NET Core і Angular»

Виконав:
студент 6 курсу, групи ІТ-91мн
Альперт Максим Іоганович _____

Науковий керівник:
доцент кафедри АУТС, к.т.н., доцент
Катін Павло Юрійович _____

Рецензент:
Доцент кафедри ОТ, к.т.н., доцент
Волокита Артем Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський)

Спеціальність – *121* «Інженерія програмного забезпечення»

Освітньо-наукова програма – «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2021 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Альперту Максиму Іогановичу

1. Тема дисертації «Система управління рухомою платформою на базі .NET Core і Angular», науковий керівник дисертації Катін Павло Юрійович, доцент кафедри АУТС, к.т.н., затверджені наказом по університету від «12» березня 2021 р. № 809-с

2. Термін подання студентом дисертації 11.05.2021

3. Об'єкт дослідження рухома платформа

4. Предмет дослідження система управління рухомою платформою на базі .NET Core і Angular

5. Перелік завдань, які потрібно розробити визначення основних вимог до системи, визначення сценаріїв використання системи, вибір технологій для розробки, реалізація багатошарової архітектури, розробка інтерфейсу користувача, математичне моделювання надійності системи.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу діаграма розгортання, структурна схема, функціональна схема, діаграма прецедентів, діаграма класів, діаграма компонентів, діаграма послідовності, ER-діаграма, ілюстрація макету.

7. Орієнтовний перелік публікацій Програмно-апаратна інфраструктура наземної автономної платформи з елементами штучного інтелекту, Підвищення ефективності обміну даними сутностей у реляційному представленні та їх обробки

9. Дата видачі завдання 01.02.2021

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Порівняльний аналіз існуючих рішень	02.02.2021-10.02.2021	
2	Визначення основних вимог до системи	11.02.2021-15.02.2021	
3	Визначення сценаріїв використання системи	16.02.2021-22.02.2021	
4	Вибір технологій для розробки	23.02.2021-28.02.2021	
5	Реалізація багатошарової архітектури	01.03.2021-28.03.2021	
6	Розробка інтерфейсу користувача	29.03.2021-05.04.2021	
7	Аналіз виконаної роботи	06.04.2021-08.04.2021	
8	Оформлення текстового матеріалу	09.04.2021-19.04.2021	
9	Оформлення графічного матеріалу	20.04.2021-04.05.2021	
10	Передзахист магістерської дисертації	05.05.2021	
11	Доопрацювання пояснювальної записки та підготовка презентації	06.05.2021-11.05.2021	
12	Захист магістерської дисертації	18.05.2021	

Студент Максим АЛЬПЕРТ

Науковий керівник Павло КАТІН

РЕФЕРАТ

Магістерська дисертація на тему «Система управління рухомою платформою на базі .NET Core і Angular» містить 115 аркушів пояснювальної записки, 47 таблиць, 40 рисунки, 10 додатків та 31 посилань на використані джерела.

Актуальність теми: полягає у потребі розробки рухомих автоматизованих і автоматичних платформ. Розроблена система дозволяє отримати прототип рухомого наземного пристрою, що надає можливість людині працювати у несприятливих умовах, а саме: робота на глибині, де вміст кисню в повітрі є мінімальним; робота в умовах підвищеного радіаційного фону, хімічного забруднення; проведення рятувальних операцій, а також для мінімізації людських контактів під час пандемії.

Мета роботи: покращення показників надійності рухомої платформи шляхом визначення потрібної кількості каналів зв'язку.

Об'єкт дослідження: рухома платформа.

Предмет дослідження: система управління рухомою платформою на базі .NET Core і Angular.

Методи дослідження: математичне моделювання з використанням марковських процесів і трансферне навчання нейронної мережі для класифікації зображень.

У магістерській дисертації виконано порівняльний аналіз існуючих рішень щодо наземних рухомих платформ і запропоновано новий програмно-апаратний продукт.

Досліджено надійність системи управління розробленої рухомої платформи. Проаналізовано надійність каналів зв'язку, проведено моделювання з використанням марковських процесів, визначено точність вимірювань нейронної мережі.

Ключові слова: система управління, мікрокомп'ютер Raspberry Pi, .NET Core, Angular, операційна система, надійність, нейронна мережа, трансферне навчання.

SUMMARY

Master’s dissertation on the topic “Movable platform control system based on .NET Core and Angular” contains 115 pages of explanatory note, 47 tables, 40 figures, 10 appendices and 31 references to used information sources.

Relevance of the topic: lies in the need to develop movable automated and automatic platforms. The developed system allows to obtain a prototype of a movable ground device that allows a person to work in adverse conditions, namely: work at depth, where the oxygen content in the air is minimal; work in the conditions of the increased radiation background, chemical pollution; rescue operations, as well as to minimize human contact during a pandemic.

Objective: to improve the reliability of the movable platform by determining the required number of communication channels.

Object of research: movable platform.

Subject of research: movable platform control system based on .NET Core and Angular.

Research methods: mathematical modeling using Markov processes, transfer training of a neural network for image classification.

A comparative analysis of existing solutions for ground-based movable platforms performs in the master's dissertation. A new software and hardware product is proposed.

The reliability of the control system of the developed movable platform is investigated. The reliability of communication channels is analyzed, the reliability of failures and recovery is modeled using Markov processes and the accuracy of neural network measurements is determined.

Keywords: control system, Raspberry Pi microcomputer, .NET Core, Angular, operating system, reliability, neural network, transfer learning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

.NET Core — модульна платформа для розробки програмного забезпечення з відкритим вихідним кодом

Angular — відкрита платформа для розробки веб-додатків, яка написана на мові TypeScript

MANET (англ. Mobile Ad hoc Network) — бездротова, децентралізована, мобільна IP-мережа, що здатна до самоорганізації та забезпечує встановлення з'єднань між довільними вузлами

LiDAR (англ. Light Identification, Detection and Ranging) — технологія отримання та обробки інформації про віддалені об'єкти за допомогою активних оптичних систем

UART (англ. universal asynchronous receiver/transmitter — універсальний асинхронний приймач/передавач) — тип асинхронного приймача-передавача, компонентів комп'ютерів та периферійних пристроїв

SPI (англ. Serial Peripheral Interface, SPI bus — послідовний периферійний інтерфейс, шина SPI) — фактичний послідовний синхронний повнодуплексний стандарт передачі даних, розроблений фірмою Motorola для забезпечення простого сполучення мікроконтролерів та периферії

I2C — послідовна шина даних для зв'язку інтегральних схем, розроблена фірмою Philips на початку 1980-х як проста шина внутрішнього зв'язку для створення керуючої електроніки

GPS (англ. Global Positioning System) — сукупність радіоелектронних засобів, що дозволяє визначати положення та швидкість руху об'єкта на поверхні Землі або в атмосфері

Wi-Fi — технологія бездротової локальної мережі з пристроями на основі стандартів IEEE 802.11

Full HD (Full High Definition) — роздільна здатність 1920×1080 точок (пікселей) і частотою кадрів не менше 24/сек

USB (англ. Universal Serial Bus) — універсальна послідовна шина, яка призначена для з'єднання комп'ютерів і периферійних пристроїв

HDMI (англ. High Definition Multimedia Interface) — інтерфейс та кабель для передачі цифрових відео та аудіо даних, є альтернативою аналогових інтерфейсів

GPIO (англ. General-purpose input/output) — інтерфейс для зв’язку між компонентами комп’ютерної системи, наприклад, мікропроцесором і різними периферійними пристроями

IoT (англ. Internet of Things) — концепція мережі передачі даних між фізичними об’єктами («речами»), оснащеними вбудованими засобами і технологіями для взаємодії один з одним або з зовнішнім середовищем

UML (англ. Unified Modeling Language, уніфікована мова моделювання) — мова графічного опису для об’єктного моделювання в області розробки програмного забезпечення, для моделювання бізнес-процесів, системного проектування та відображення організаційних структур

TensorFlow — відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для вирішення завдань побудови і тренування нейронної мережі

Front-end — це практика перетворення даних у графічний інтерфейс за допомогою HTML, CSS та JavaScript, щоб користувачі могли переглядати та взаємодіяти з цими даними

PostgreSQL — об’єктно-реляційна система управління базами даних

SSH (англ. Secure SHell) — мережевий протокол рівня застосунків, що дозволяє проводити віддалене управління комп’ютером і тунелювання TCP-з’єднань

ОС — операційна система

БНРП — безпілотна наземна рухома платформа

ПСРП — програмна система рухомої платформи

Кілометри за годину (км/год) — позасистемна одиниця вимірювання швидкості

ПМК — програмований мікрокомп’ютер

СУБД — система управління базами даних

Архітектура ARM (англ. Advanced RISC Machine) — система команд і сімейство описів і готових топологій 32-бітних і 64-бітових мікропроцесорних ядер, що розробляються компанією ARM Limited

ВСТУП

Актуальність магістерської дисертації полягає у потребі розробки рухомих автоматизованих і автоматичних платформ. Розроблена система дозволяє отримати прототип рухомого наземного пристрою, що надає можливість мінімізувати контакти в небезпечних умовах. Застосування автоматизованої платформи дозволяє людині працювати у несприятливих умовах, а саме: робота на глибині, де вміст кисню в повітрі є мінімальним; робота в умовах підвищеного радіаційного фону, хімічного забруднення; проведення рятувальних операцій; в умовах пандемії, коли розповсюдження вірусу відбувається повітряно-крапельним шляхом, і як наслідок, виникає необхідність людям взаємодіяти дистанційно.

Метою роботи є покращення показників надійності рухомої платформи шляхом визначення потрібної кількості каналів зв’язку.

Для реалізації поставленої мети були сформульовані такі завдання:

– розробка апаратної і програмної частини системи управління рухомою платформою на базі .NET Core і Angular;

– дослідження надійності каналів зв’язку системи управління рухомою платформою;

– розробка програмної складової нейронної мережі;

– дослідження ефективності застосування нейронної мережі.

Об’єктом дослідження є рухома платформа.

Предметом дослідження є система управління рухомою платформою на базі .NET Core і Angular.

Застосовано наступні методи дослідження: математичне моделювання з використанням марковських процесів та трансферне навчання нейронної мережі для класифікації зображень.

У магістерській дисертації виконано порівняльний аналіз існуючих рішень щодо наземних рухомих платформ та розроблено новий прототип системи управління рухомою платформою у вигляді програмно-апаратного рішення на базі .NET Core і Angular.

Наукова новизна одержаних результатів полягає у побудові моделі системи управління рухомою платформою:

– виявленні факторів, що впливають на точність розпізнавання необхідних образів при навчанні нейронної мережі;

– отриманні показників надійності системи управління рухомою платформою з використанням марковських процесів.

За цими результатами розроблено програмну інфраструктуру для системи управління рухомою платформою на базі .NET Core і Angular.

Апробація наукових досліджень: Друга міжнародна конференція «Advanced Trends in Information Theory», 25 листопада – 27 листопада 2020. стр. 395-398, doi: 10.1109/ATIT50783.2020.9349318.

Публікація результатів наукових досліджень: стаття «Програмно-апаратна інфраструктура наземної автономної платформи з елементами штучного інтелекту» в науковому журналі «Математичні машини і системи» у 2021 році у м. Києві; стаття «Підвищення ефективності обміну даними сутностей у реляційному представленні та їх обробки» в науковому журналі «Технічні науки та технології» у 2021 році у м. Чернігів.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд типів безпілотних наземних рухомих платформ

Безпілотні наземні рухомі платформи (БНРП) – це роботизовані системи, що працюють на суші під управлінням людини-оператора або автономно. Вони використовуються для широкого спектру цивільних завдань, особливо в небезпечних або несприятливих для людини умовах, а також для завдань, що є складними або мають високий ризик для життя.

БНРП можуть управлятися дистанційно за допомогою портативної чи стаціонарної станції управління або працювати автономно. Автономні БНРП можуть переміщатися між заздалегідь визначеними точками або у просторі для виконання своєї задачі. БНРП, що працюють в непозначених на карті мінливих середовищах, можуть збирати інформацію і будувати карту свого оточення, використовуючи такі методи, як одночасна локалізація і картографування. Штучний інтелект і машинне навчання також можуть допомогти їм адаптуватися до навколишнього середовища.

Два основних способи пересування БНРП – це колеса і гусениці. Колеса – енергоефективні і дозволяють розвивати високу швидкість на рівній місцевості, але не підходять для пересування по бездоріжжю, тому що вони можуть застрягти або зануритися в землю через низьку площу контактної поверхні. Гусениці – кращий варіант на пересіченій місцевості, але вони повільніші, менш ефективні, більш технічно складні.

Невеликі БНРП живляться від електричних акумуляторів. Більші можуть використовувати бензиновий або дизельний двигун або гібридну систему. Ця система використовує двигун внутрішнього згоряння для приводу електричного генератора. Розробляються також системи живлення БНРП на водневих паливних елементах.

Бездротовий зв’язок необхідний для віддаленої експлуатації БНРП, а також для передачі відеозаписів та інших даних датчиків. Зазвичай це робиться за допомогою радіочастотного зв’язку, супутникових ліній зв’язку або бездротової оптично-волоконної мережі. Технологія мобільної спеціальної мережі (MANET) часто

використовується для того, щоб допомогти БНРП підтримувати зв’язок навіть у несприятливих умовах.

БНРП можуть бути оснащені різними датчиками. Так під час роботи у важко доступних місцях, де немає доступу до супутникової системи навігації, БНРП можуть використовувати датчики LiDAR у поєднанні з інерційними навігаційними системами для більш точної навігації.

Зазвичай під програмно-апаратними рухомими платформами мають на увазі наземні дрони з ручним керуванням або автоматичні роботи. Такі рухомі платформи в основному мають доступ до інтернету, тому ними можна віддалено керувати і отримувати додаткову статистику. Оскільки проект передбачає побудову саме програмно-апаратної інфраструктури управління системами рухомої платформи, то у якості аналогів варто розглядати не тільки автоматичних роботів, а також системи з дистанційним керуванням. Прикладами таких роботизованих систем є: Amazon Scout, REV-1, Turtle Rover, тощо.

1.2 Amazon Scout

Amazon Scout – це автономний шестиколісний робот. Він живиться від акумуляторів і рухається зі швидкістю 24 км/год. Головною задачею Amazon Scout є доставка товарів. На рисунку 1.1 зображено робота [1, 2].

Рисунок 1.1 – Робот для доставки Amazon Scout [1, 2]

Компанія Amazon розробила реалістичне відтворення приміських районів і використовує їх для імітації тисяч поставок протягом однієї ночі. Ці симуляції створюють базу знань, яка дозволить роботові доставки аналізувати ситуації, з якими він може зіткнутися в реальному світі. Наприклад, якщо автомобіль починає виїжджати з дороги, коли Amazon Scout їде по тротуару, він буде знати, що потрібно зупинитися, тому що він бачив цю ситуацію в симуляції.

Для коректного функціонування Amazon Scout та швидкої обробки даних, він використовує такі загальні комунікаційні інтерфейси: UART, SPI, I2C, CAN. Операційна система реального часу дозволяє набагато швидше обробляти дані, оскільки опрацювання даних відбувається по мірі їх надходження без затримок у буфері.

Amazon Scout використовує нейронні мережі, щоб безпечно і ефективно переміщатися навколо домашніх тварин, пішоходів та інших перешкод на своєму шляху.

1.3 Rev-1

REV-1 – це автономний триколісний робот. При максимальній швидкості 19 км/год REV-1 важить 45 кг (рисунок 1.2) [3]. Головною задачею REV-1 є доставка товарів.

Рисунок 1.2 – Робот для доставки Rev-1 [3]

Робот рухається по місту та уникає перешкод автономно, використовуючи комбінацію GPS, технології LiDAR, радару та 12 оптичних камер. Частина встановлених камер має широке поле зору близько 200 градусів, а інша частина має більш вузьке поле зору від 90 до 100 градусів. Такий розподіл камер дає REV-1 повний 360-градусний огляд і дозволяє сприймати глибину оточення. Оскільки робот рухається відносно повільно, він також може використовувати ультразвукові датчики.

1.4 Turtle Rover

Turtle Rover, рухома платформа, що може працювати до чотирьох годин безперервно та має радіус дії Wi-Fi до 200 м. Ровер має апаратне та програмне забезпечення з відкритим вихідним кодом, водонепроникний корпус і камеру Full HD [4, 5].

Turtle Rover здатний працювати на пересічній місцевості. Це робот на радіокеруванні, який дозволяє встановити додаткове обладнання і вести запис в таких місцях, як печери, водойми і руїни.

Також даний ровер має підтримку додаткових модулів:

– можливість під’єднати GoPro;

– можливість під’єднати 360 градусну камеру або LiDAR для картографування місцевості;

– можливість під’єднати роботизований маніпулятор.

Основні особливості та характеристики Turtle Rover:

– до 4 годин роботи від акумулятора;

– вбудований Wi-Fi з діапазоном до 200 м;

– відкритий вихідний код;

– мікрокомп’ютер Raspberry Pi як «головний мозок»;

– захист від вологи та пилу;

– Full HD камера для запису та передачі відео.

Рисунок 1.3 – Наземний дрон Turtle Rover [4, 5]

Корпус роверу повністю герметичний, включаючи додатковий маніпулятор. Це дозволяє їздити роверу в будь-яку погоду та під водою.

Ровер веде пряму трансляцію з HD-камери в режимі реального часу. Програмний додаток працює на будь-якому пристрої, який має доступ до веб-браузера.

1.5 Програмно-апаратна інфраструктура наземного робота для дослідження території

В основі наземного робота лежить мікрокомп’ютер Orange Pi Zero Plus. При побудові апаратної інфраструктури було вирішено відокремити живлення двигунів від мікрокомп’ютера. Щоб привести дану платформу до робочого стану було використано такі компоненти [6]:

– ПМК Orange Pi Zero Plus;

– 2 двигуни з напругою по 6В;

– H-Bridge L298N для керування двигунами;

– 4 АА батерейок або акумуляторів для живлення двигунів;

– microUSB веб-камера Gear Head WC740i-CP10 для отримання зображення або потокового відео;

– акумулятор для живлення Orange Pi.

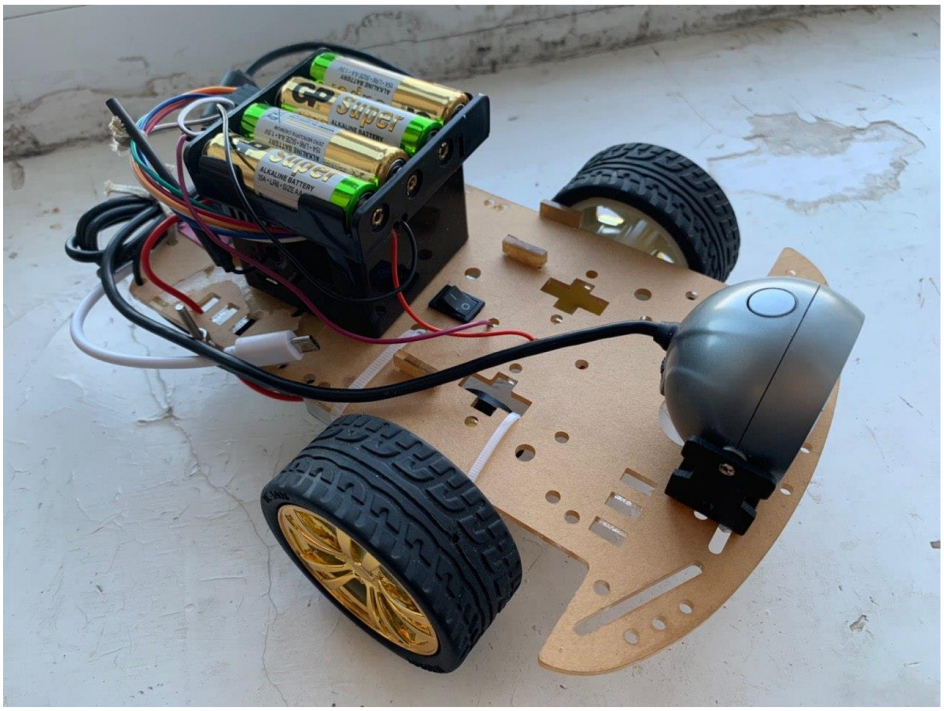

Рисунок 1.4 – Наземний робот для дослідження території [6]

На основі розглянутих компонентів було створено повноцінний прототип системи (рисунок 1.4). Для того, щоб проводити різні маніпуляції з даним роботом, були проведені різні тестування системи передачі команд, передачі потокового відео та збереження даних.

Для розробки програмного додатку було вирішено використовувати такий набір програмного забезпечення: Spring Framework, Spring Boot, Apache Tomcat та JPA Hibernate. Для серверної частини використано найлегшу СУБД SQLite.

1.6 Висновки до розділу

Надано огляд та проаналізовано існуючі рішення щодо наземних рухомих платформ. На підставі проведеного аналізу отримані такі особливості, що притаманні сучасним пристроям даного класу. Серед них:

– мікрокомп’ютер, який буде керувати рухомою платформою та обробляти дані, отримані від зовнішніх датчиків та периферії;

– програмний додаток для керування рухомою платформою.

Деякі існуючі рішення мають ряд важливих недоліків:

– висока ціна рухомої платформи;

– перевантаженість зайвими датчиками;

– відомі наземні рухомі платформи мають програмну частину, що є окремим рішенням, в більшості випадків. Отже, зазначена програмна частина не може стати прототипом узагальненої системи для розвитку аналогічних рішень.

Для розв’язання цих проблем вирішено розробити власну систему управління рухомою платформою, що побудована на базі типових фреймворків і може в подальшому швидко розвиватися і зручно супроводжуватися. Отже, систему управління буде створено у вигляді веб-додатку, що дає змогу використовувати будь-яку мікропроцесорну систему, яка підтримує браузери, у якості пульта управління.

2 ВИБІР ТА ОБГРУНТУВАННЯ СКЛАДОВИХ ЕЛЕМЕНТІВ ПРОГРАМНО-АПАРАТНОЇ ІНФРАСТРУКТУРИ НАЗЕМНОЇ ПЛАТФОРМИ

2.1 Узагальнена програмно-апаратна інфраструктура

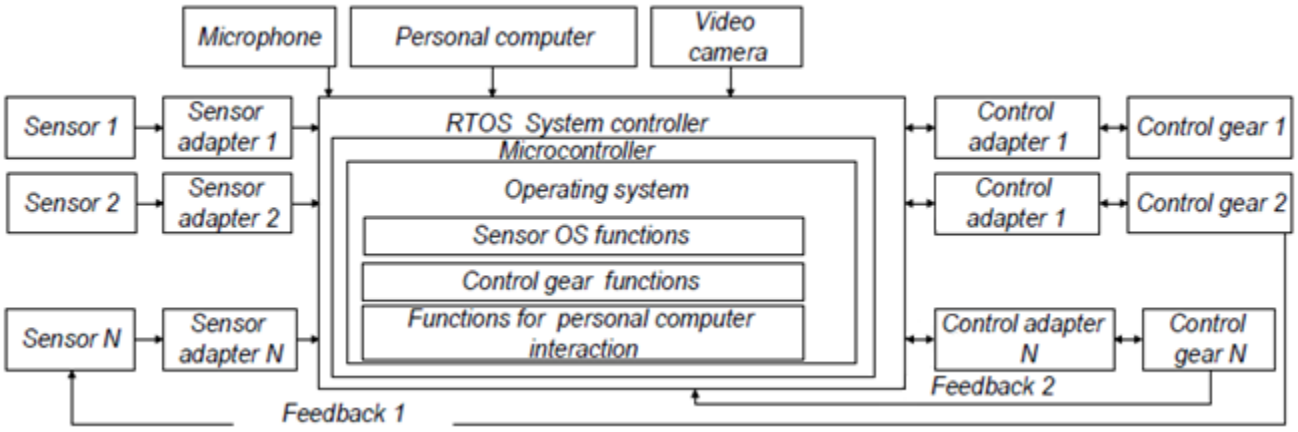

Рисунок 2.1 – Узагальнена схема програмно-апаратної інфраструктури [7]

На теперішній час існують узагальнені підходи для побудови програмно-апаратної інфраструктури системи управління наземними платформами і операційними системами реального часу. Вона може бути реалізована на основі мікрокомп’ютера будь-якої архітектури і потужності, в залежності від завдань. При цьому може бути використана складна система контролю, яка вимагає потужності мікрокомп’ютера на основі Cortex-A, наприклад для управління відеокамерою.

Приклад узагальненої схеми розроблено в статті [7] і відображено на рисунку 2.1. Дана система містить датчики, виконавчі механізми, забезпечує зв’язок з персональним комп’ютером і може стати основою БНРП.

На основі узагальненої схеми розробимо діаграму розгортання наземної платформи.

На рисунку 2.2 представлено діаграму розгортання БНРП на основі мікрокомп’ютера. Основним елементом керування БНРП, що виконує всі операції обчислення та операції керування, є мікрокомп’ютер Raspberry Pi. На діаграмі розгортання на рисунку 2.2 зображено виконавчі механізми БНРП, які підключені через інтерфейс вводу-виведення мікрокомп’ютера. На рисунку 2.2 загалом відображено три контури управління БНРП, які проходять через мікрокомп’ютер.

Мікрокомп’ютер керує запуском двигуна, сервомоторами і забезпечує передачу зображення у режимі онлайн. Окремо виділено контур передачі візуальної інформації через USB-камеру, що забезпечує передавання відео до мікрокомп’ютера БНРП.

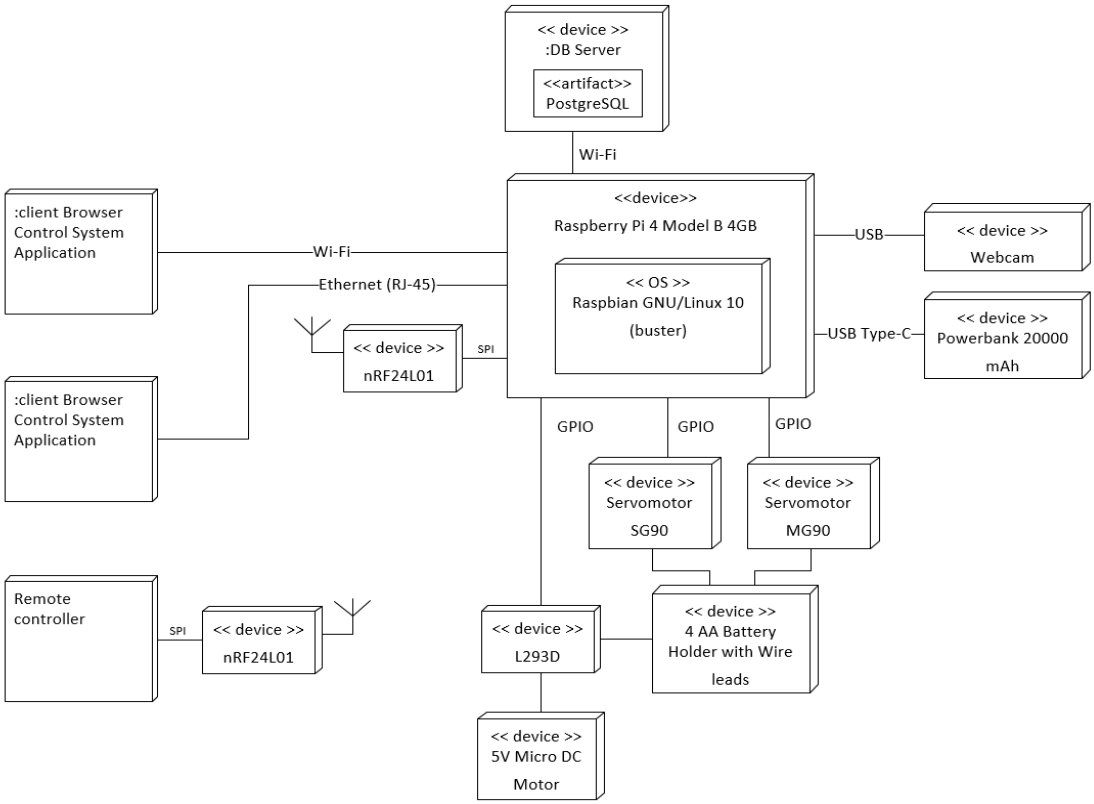

Рисунок 2.2 – Діаграма розгортання

Загалом система складається з підсистем управління двигуном, двома сервомоторами і відеокамери.

Діаграму розгортання також представлено у додатку А.

2.2 Автоматизована система управління

Існує велика кількість об’єктів, управління якими необхідно здійснювати спільними діями людини і технічних пристроїв, тобто використовувати людино-машинну систему управління. Системи управління, в контурі управління яких для реалізації мети системи спільно функціонують людина (людські колективи) і технічні засоби переробки інформації, називають автоматизованими системами управління. В якості основного технічного засобу переробки інформації в автоматизованій системі

управління використовується комп’ютер. Розподіл функцій з управління об’єктом між людиною і комп’ютером може здійснюватися різними способами. На комп’ютер в автоматизованій системі управління покладаються функції підготовки інформації, необхідної для вибору раціонального керуючого впливу. Прийняття рішення, як правило, залишається в автоматизованій системі управління за людиною.

Роль людини в автоматизованій системі управління. Людина може виконувати різні функції в автоматизованій системі управління або може бути декількома способами включена в контур управління. Комп’ютер отримує інформацію про стан об’єкта управління, обробляє її і передає людині-оператору у вигляді, зручному для прийняття рішення. Людина в цій системі виконує функції прийняття рішення [8].

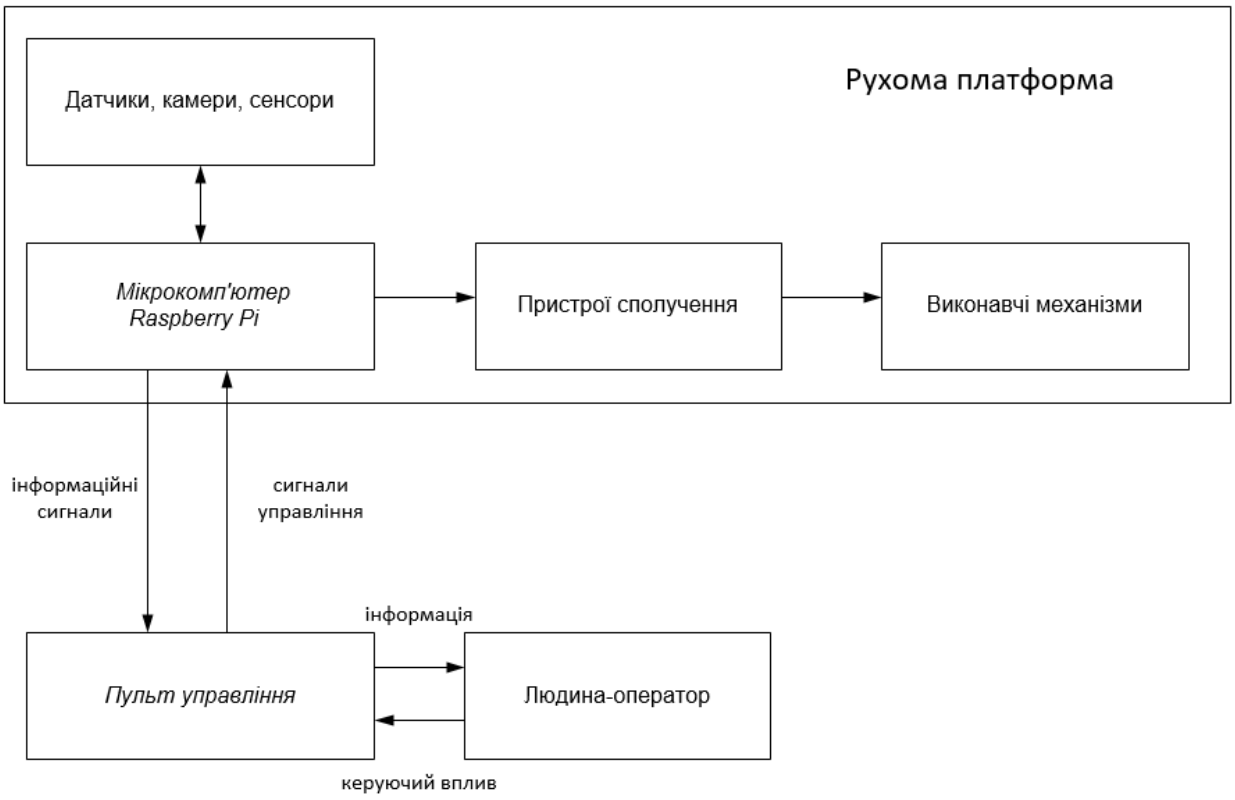

Рисунок 2.3 – Узагальнена схема системи управління БНРП

На рисунку 2.3 показана узагальнена схема БНРП, що керується людиною-оператором через пульт дистанційного управління. Опишемо призначення кожного елемента схеми БНРП:

– пульт управління БНРП використовується для передачі інформації, аудіо, відео людині-оператору;

– мікрокомп’ютер БНРП використовується для обробки інформації, що отримується з датчиків, камери і сенсорів;

– пристрій сполучення БНРП використовується для сполучення з виконавчими механізмами;

– виконавчі механізми БНРП використовуються для зміни положення БНРП у просторі;

– датчики, камери, сенсори БНРП використовуються для отримання інформації з зовнішнього середовища.

2.3 Вибір мікрокомп’ютера

Проведемо вибір мікрокомп’ютера для БНРП. Мікрокомп’ютери – це одноплатні пристрої, які мають процесор, графічний модуль, а також мережеві інтерфейси (як провідні, так і бездротові), порти USB, відеовиходи, накопичувачі для зберігання даних або порти для їх підключення. Електроживлення мікрокомп’ютерів здійснюється різними способами, як правило, з використанням порта USB.

Одноплатні мікрокомп’ютери можуть застосовуватися в робототехніці та різноманітних проектах, що вимагають програмного управління електронікою.

На сьогоднішній день на ринку представлений великий вибір мікрокомп’ютерів: Raspberry Pi, Orange Pi, Arduino та інші.

Raspberry Pi – це мініатюрний одноплатний мікрокомп’ютер. Він характеризується високою продуктивність при компактних габаритах (85.6 мм х 56.5 мм х 17 мм).

Вся лінійка Raspberry Pi має процесори з ARM-архітектурою, які є недорогими. На даний час найбільш підходящою операційною системою (ОС) для Raspberry Pi є Raspberry Pi Desktop. В її основі лежить образ Raspbian 10 Buster. Перевагою даної ОС є те, що вона 32-бітна. Зараз розробляється 64-бітна версія системи, яка буде

виконувати різні обчислювальні операції набагато швидше. Raspberry Pi має спеціальний роз'єм GPIO з 40-пінами для підключення зовнішніх пристроїв [9, 10].

Orange Pi – це група одноплатних мікрокомп'ютерів на базі Linux, що має підтримку таких операційних систем як Android, Ubuntu, Debian, Raspbian Image, Armbian. Orange Pi має потужний 4-х ядерний процесор AllWinner H5 або H6 64 біта, 1 або 2 Гб оперативної пам'яті, виділену відеокарту Mali T720, чотири USB-порта 2.0 і 3.0, виходи HDMI для підключення монітора або телевізора, аудіовихід, роз'єм для підключення камери, роз'єм для карт пам'яті, а також спеціальний роз'єм GPIO з 40-пінами для підключення будь-яких інших пристроїв. Orange Pi було створено в першу чергу як дешеву копію Raspberry Pi [11].

Arduino – це мікроконтролер для створення автоматизованих систем [12]. Мікроконтролер може одночасно виконувати одну задачу, а одноплатні мікрокомп'ютери виконують програми в рамках операційної системи (найчастіше Linux). Також одноплатні мікрокомп'ютери характеризуються більшою продуктивністю.

До недоліків даних мікроконтролерів можна віднести:

– низьку продуктивність (одноядерний процесор);

– для роботи з інтернетом необхідні додаткові модулі та глибоке знання протоколів;

– відсутність можливості для роботи з відео, комп'ютерним зором із-за низької потужності.

Однією з вимог до системи управління рухомою платформою є підтримка апаратних відеокодеків, HDMI-виходу. І дивлячись на можливості перерахованих мікрокомп'ютерів та мікроконтролерів, Raspberry Pi є найкращим кандидатом для системи управління рухомою платформою.

Як основу для системи управління БНРП було обрано Raspberry Pi, бо він має наступні характеристики, а саме:

– відносну низьку вартість;

– багатозадачність;

– універсальність по напрямкам застосування;

– велику кількість бібліотек, що дають можливість пришвидшити розробку;

– підтримку будь-яких мов програмування, які доступні в операційних системах.

– можливість захоплення відеопотоку через веб-камеру.

2.4 Вибір операційної системи

Проведемо вибір операційної системи для БНРП. Для використання функціональних можливостей Raspberry Pi, необхідно застосовувати операційну систему (ОС). ОС виконує роль мосту між користувачем і обладнанням Raspberry Pi. Різні ОС мають підтримку різного програмного забезпечення. Операційні системи для Raspberry Pi дозволяють програмному забезпеченню взаємодіяти з апаратним забезпеченням для керування зовнішньою периферією. При цьому Raspbian є офіційною ОС Raspberry Pi. Слід зазначити, що існують і інші альтернативні операційні системи, доступні для запуску на Raspberry Pi.

2.4.1 Raspberry Pi Desktop (Raspbian OS)

Raspberry Pi Desktop – це офіційна ОС, яку можна використовувати на всіх моделях Raspberry [10]. Ця операційна система відома як модифікована версія ОС Debian.

Особливості Raspberry Pi Desktop:

– Raspbian підходить для будь-якої моделі Raspberry Pi, що дозволяє оптимізувати програмні додатки до відповідної моделі;

– наявність програмного інструменту, що дозволяє завантажувати будь-яке програмне забезпечення при підключенні до мережі інтернет.

Розробники Raspbian створюють 64-розрядну версію своєї ОС, яка проходить бета-тестування.

2.4.2 Операційна система DietPi

DietPi – це легка версія операційної системи (ОС) Debian [13]. Вона займає в 3 рази менше простору ніж інші доступні ОС Raspberry Pi. Установка ОС автоматизована.

Особливості DietPi:

– Diet Processing Tool, що поставляється разом з ОС, допомагає визначити рівень пріоритету встановлених програм і управляти планувальниками задач;

– можливість налаштування продуктивності апаратного та програмного забезпечення;

– ОС використовує мінімальний простір в оперативній пам’яті, що допомагає працювати одноплатному комп’ютеру з максимальною продуктивністю. Мінімальний розмір образу ОС починається з 400 МБ.

2.4.3 Windows 10 IoT Core

Windows 10 IoT Core – це операційна система, яка походить від повноцінної ОС Windows 10 [14].

Особливості Windows 10 IoT Core:

– ця операційна система вирішує задачі, пов’язані з безпекою, підключенням, розробкою проектів та інтеграцією з хмарними технологіями;

– ОС поставляється з Microsoft Visual Studio, і є єдиним середовищем розробки, яке дає будь-які команди або може працювати з іншими програмними додатками;

– версія для вбудованих систем не має тих самих функцій, що є в повноцінній Windows 10.

Для розробки власного проекту вибір в основному стояв між Raspbian та Windows 10 IoT Core. Далі приведено аналіз відмінностей між цими операційними системами (таблиця 2.1):

Таблиця 2.1 – Порівняння Raspberry Pi Desktop та Windows 10 IoT Core

	Raspberry Pi Desktop	Windows 10 IoT Core
Підтримка різних плат	працює в основному для плат Raspberry Pi	працює на різних одноплатних мікрокомп'ютерах
Відкритість операційної системи	відкрите програмне забезпечення побудоване на Debian	закрита, власна операційна система від Microsoft
Можливості операційної системи	повноцінна операційна система	операційна система з обмеженим функціоналом
Багатозадачність	може працювати багато додатків можуть	може працювати тільки один додаток
Підтримка сторонніх розробників	має велику підтримку спільноти	відсутність великої підтримки із-за закритості ОС

За аналізом можливостей операційних систем, які були наведені у таблиці 2.1 можна зробити висновок, що операційна система для розробки системи управління БНРП, яка підходить для реалізації необхідних задач, є Raspberry Pi Desktop.

2.5 Вибір окремих апаратних компонентів системи

2.5.1 Вибір сервомотору рухомої платформи

Сервомотор – це мотор, вал якого може встати в задане положення або підтримувати задану швидкість обертання. Іншими словами, валом сервомотору можна управляти, наприклад, задаючи йому положення в градусах або певну частоту обертання. Сервомотори використовуються в різноманітних областях, наприклад, в

робототехніці вони допомагають моделювати різні рухи роботів. Сервомотори – ефективне рішення для переміщення механізмів у просторі [15].

Сервомотори бувають цифрові і аналогові. За зовнішнім виглядом вони майже не відрізняються один від одного. Основна відмінність полягає в принципі управління мотором. У аналогових сервомоторів управління відбувається за допомогою спеціальної мікросхеми, а цифрові сервомотори мають мікропроцесор. Мікросхема і мікропроцесор здатні приймати і аналізувати керуючі імпульси. Тільки на мікросхему вони зазвичай надходять з частотою 50 Гц, а на мікропроцесор – з частотою 200 Гц і більше. В результаті цього цифровий сервомотор мобільніше і чіткіше реагує на керуючий сигнал.

На теперішній час з’явився новий вид виконавчих механізмів до який можна віднести цифрові сервомотори. До переваг цифрових сервомоторів відносяться: висока точність позиціонування, можливість більш швидкого управління мотором, можливість підтримки постійного крутного моменту. Але в даній роботі буде використано аналоговий сервомотор із-за обмеженого бюджету.

Для розробки БНРП на базі Raspberry Pi може бути підключений сервомотор. Підключення здійснюється через кабелі, які виходять з сервомотору. Зазвичай це три кабелі: червоний; коричневий або чорний; жовтий, помаранчевий або білий.

Підключення сервомотору до плати Raspberry Pi проводиться через ШІМ-виводи (широтно-імпульсна модуляція).

Для системи управління БНРП було вирішено обрати два аналогових сервомотори SG90 і MG90. Різниця між ними полягає в тому, що SG90 має пластикові шестерні, а MG90 – металеві. SG90 буде використано для повороту камери, у той час як MG90 буде повертати передні колеса рухомої платформи.

2.5.2 Вибір двигуна рухомої платформи

Для побудови БНРП необхідно обрати двигун постійного струму. Двигуни постійного струму – це електромагнітні пристрої, які використовують взаємодію магнітних полів і провідників для перетворення електричної енергії в механічну

енергію обертання. На ринку існує безліч типів двигунів постійного струму. Щіткові і безщіткові двигуни є найбільш поширеними двигунами постійного струму.

Двигуни постійного струму живляться від джерела постійного струму, підключеного до роторного вузла через вугільні щітки [16].

Кроковий двигун працює аналогічно безщітковим двигунам постійного струму, за винятком того, що він рухається набагато меншими кроками. Його єдиною рухомою частиною є також ротор, який містить магніти. Полярність кожної котушки регулюється змінним струмом. При зміні полярності кожна котушка отримує поштовх.

Кроковий двигун – це обертовий двигун з дискретними кутовими переміщеннями ротора, який здійснюється за рахунок імпульсів сигналу управління [18].

В даній роботі було обрано двигун постійного струму, тому що необхідність підрахунку кроків обертання відсутня.

2.6 Вибір фреймворку для серверної частини

Під час розробки серверної частини БНРП необхідно використати фреймворк, який зможе задовільнити всі необхідні вимоги системи. Різноманітні сьогоднішні веб-фреймворки орієнтовано мають схожий функціонал і загалом вибір залежить лише від мови програмування [18]. У даній роботі розглядається розробка серверної частини з використанням мови програмування C#.

ASP.NET Core – це міжплатформена, високопродуктивна платформа з відкритим вихідним кодом для створення сучасних хмарних додатків, підключених до інтернету.

ASP.NET Core забезпечує наступні переваги [18]:

– єдиний підхід у створенні веб-інтерфейсу і веб-API;

– спроектований для простого тестування;

– можливість розробки і запуску в Windows, macOS і Linux;

– з відкритим вихідним кодом і орієнтований на спільноту;

– інтеграція сучасних клієнтських фреймворків і робочих процесів розробки;

– готова до роботи у хмарі система конфігурації.

NuGet – це менеджер пакетів для .NET. Клієнтські інструменти NuGet надають можливість створювати і споживати пакети. Колекція NuGet – це центральне сховище пакетів, яке використовується всіма авторами та споживачами пакетів. NuGet надає доступ до таких пакетів, які використовуються для управління апаратними частинами системи: System.Device.Gpio, Iot.Device.Bindings, Unosquare.Raspberry.IO.

2.7 Вибір фреймворку для клієнтської частини

Програми на Angular створюються за допомогою мови TypeScript, надрядкового коду для JavaScript, який забезпечує більш високу безпеку, оскільки підтримує типи (примітиви, інтерфейси тощо). Це допомагає виявляти і усувати помилки на ранній стадії при написанні коду або виконанні завдань обслуговування [19].

До основних переваг варто віднести:

– підходить для програмних додатків з динамічним контентом;

– структура і архітектура, спеціально створені для великої масштабованості проекту;

– MVVM (Model-View-ViewModel) – можливість працювати окремо над одним розділом програми, використовуючи один і той же набір даних;

– Angular-language-service – автозаповнення шаблону HTML-компонента.

Angular має доступ до менеджера пакетів npm. Менеджер пакетів дозволяє отримати доступ до всіх необхідних бібліотек, які буде використано при розробці системи управління рухомою платформою.

При розробці клієнтської частини БНРП було вирішено обрати Angular версії 11.

2.8 Розробка узагальненої схеми каналів передачі даних для безпілотних наземної рухомої платформи

Wi-Fi є основною системою зв’язку для системи управління рухомою платформою. Wi-Fi застосовується для завантаження розгорнутого веб-сайту системи управління.

Ethernet – це резервний дротовий канал зв’язку. Резервний канал зв’язку використовується у разі відсутності зв’язку Wi-Fi.

Радіоканал – це резервний бездротовий канал зв’язку на базі модулю nRF24L01. До мікрокомп’ютера Raspberry Pi підключено модуль радіоканалу через інтерфейс SPI (Serial Peripheral Interface). Щоб керувати рухомою платформою через радіоканал використовується пульт дистанційного управління.

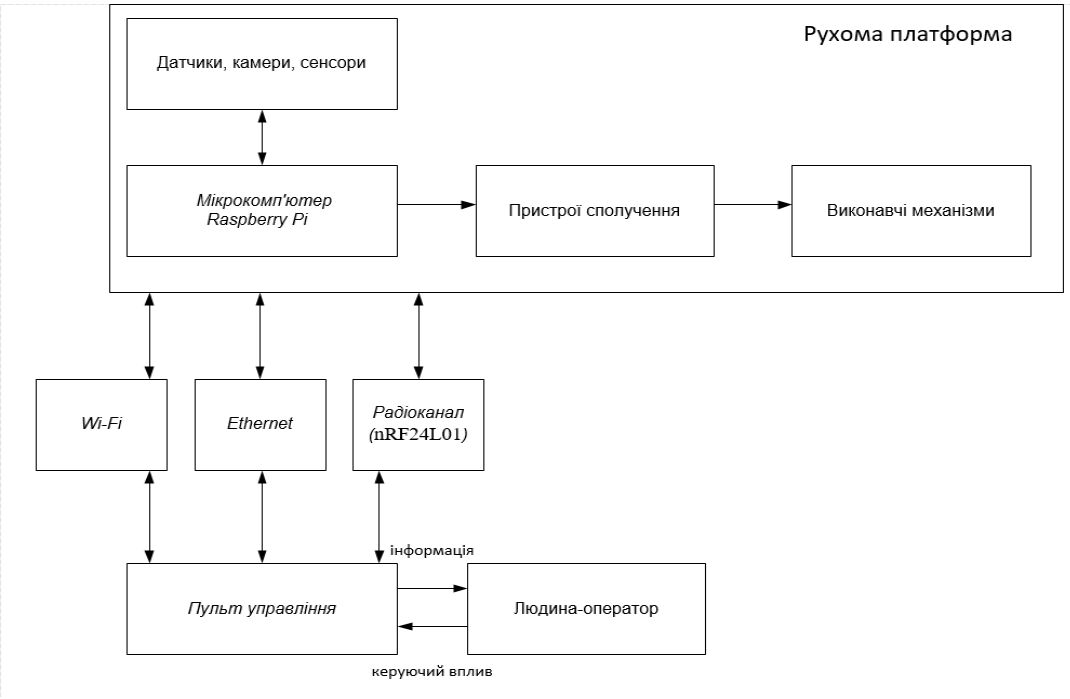

Рисунок 2.4 – Узагальнена схема каналів зв’язку БНРП

На рисунку 2.4 показана узагальнена схема каналів зв’язку для управління БНРП, що керується людиною-оператором через пульт дистанційного управління. Опишемо особливості та призначення кожного каналу БНРП:

– канал Wi-Fi використовується для зв’язку між пультом управління і мікрокомп’ютером;

– канал Ethernet використовується для зв’язку між пультом управління і мікрокомп’ютером;

– радіоканал (nRF24L01) використовується для зв’язку між пультом управління і мікрокомп’ютером.

2.9 Висновки до розділу

У цьому розділі обґрунтовано програмно-апаратну інфраструктуру наземної платформи:

– представлено діаграму розгортання БНРП (безпілотної наземної рухомої платформи) на основі мікрокомп’ютера;

– розглянуто різні мікрокомп’ютери і в якості основного обрано мікрокомп’ютер Raspberry Pi;

– проведено порівняльний аналіз операційних систем, які можна встановити на Raspberry Pi і прийнято рішення, що операційна система для розробки системи управління, яка підходить для реалізації необхідних задач, є Raspberry Pi Desktop;

– проаналізовано окремі апаратні компоненти рухомої платформи, а саме: двигуна та сервомоторів;

– обґрунтовано вибір фреймворків для серверної та клієнтської частини системи управління рухомої платформи;

обрано системи зв’язку для управління БНРП.

3 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ СИСТЕМИ УПРАВЛІННЯ РУХОМОЮ ПЛАТФОРМОЮ

3.1 Розробка структурної схеми апаратної частини системи управління рухомої платформи

Структурна схема системи управління БНРП показана рисунку 3.1 і складається з наступних компонентів:

- мікрокомп’ютер Raspberry Pi;
- модуль радіоканалу на базі nRF24L01;
- сервомотори: для камери, для повороту передніх коліс;
- двигун постійного струму;
- міст L293D для керування напрямком і швидкістю двигуна;
- акумулятор для двигуна і сервомоторів;
- акумулятор великої ємності для Raspberry Pi;
- USB-камера.

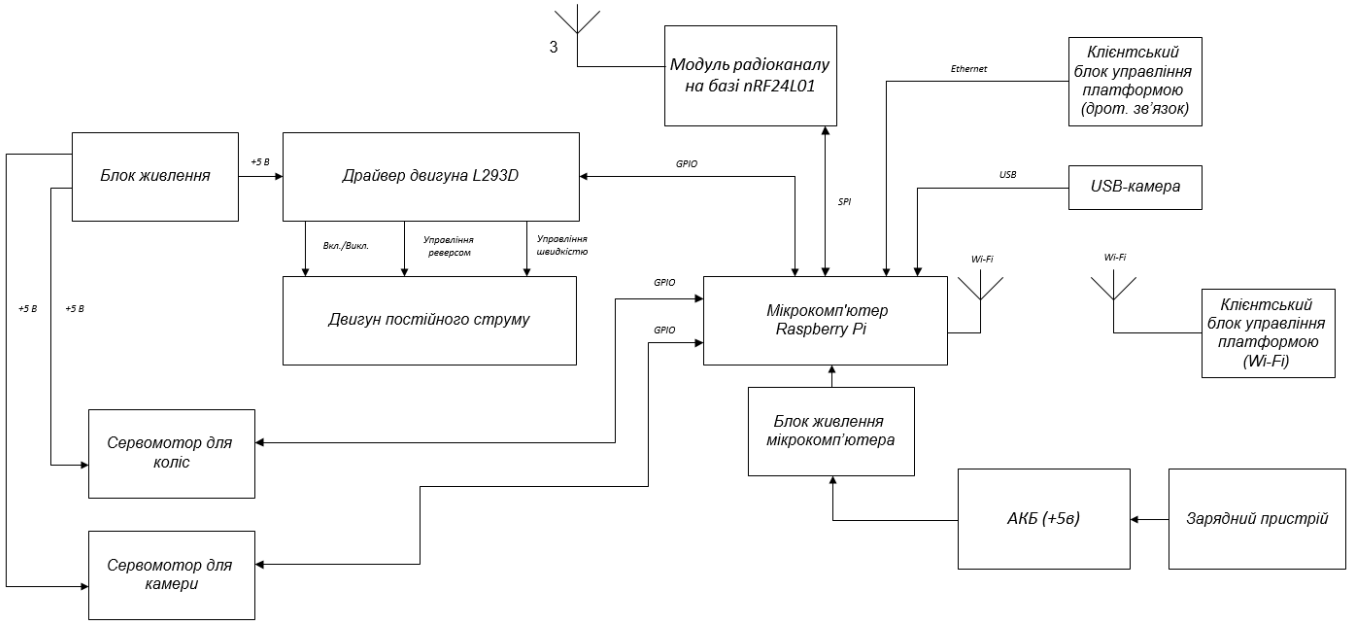

Рисунок 3.1 – Структурна схема системи управління БНРП

До клієнтських блоків управління рухомою платформою відносяться наступні пристрої:

– персональний комп’ютер;

– смартфон;

– пульт дистанційного управління.

Клієнтські блоки управління платформою мають наступні канали зв’язку:

– Wi-Fi;

– дротовий зв’язок через Ethernet;

– радіоканал (nRF24L01).

До мікрокомп’ютера Raspberry Pi підключено модуль радіоканалу на базі nRF24L01 через інтерфейс SPI. Даний модуль встановлює зв’язок між мікрокомп’ютером Raspberry Pi і пультом дистанційного управління.

Пульт дистанційного управління включає наступні компоненти:

– мікроконтролер, який має субблоки GPIO та SPI;

– джойстик та клавіатуру, які підключені до субблоку GPIO;

– блок зв’язку RF діапазону, який підключено до субблоку SPI.

Розробка БНРП вимагає використання виконавчих механізмів.

До виконавчих механізмів відносяться:

– сервомотори, які підключені до мікрокомп’ютера Raspberry Pi. Сервомотори призначені для повороту камери та зміни напрямку руху БНРП, підключаються до портів GPIO та отримують управляючі сигнали за допомогою бібліотек для вбудованих систем;

– двигун постійного струму, який підключено до мікрокомп’ютера Raspberry Pi. Для управління двигуном було використано драйвер двигуна L293D. Драйвер дозволяє управляти увімкненням і вимиканням двигуна, керувати напрямком і швидкістю обертання двигуна.

До виконавчих механізмів підключено блок живлення з напругою не менше 5В.

Мікрокомп’ютер Raspberry Pi також підключено до окремого блоку живлення.

До БНРП підключена камера для отримання візуальної інформації.

Повну структурну схему БНРП можна знайти у додатку Б.

3.2 Розробка функціональної схеми апаратної частини системи управління рухомої платформи

Функціональну схему БНРП показано на рисунку 3.2 і розробляється на основі структурної схеми для кожного блоку, в результаті з окремих функціональних елементів складається загальна функціональна схема об’єкту.

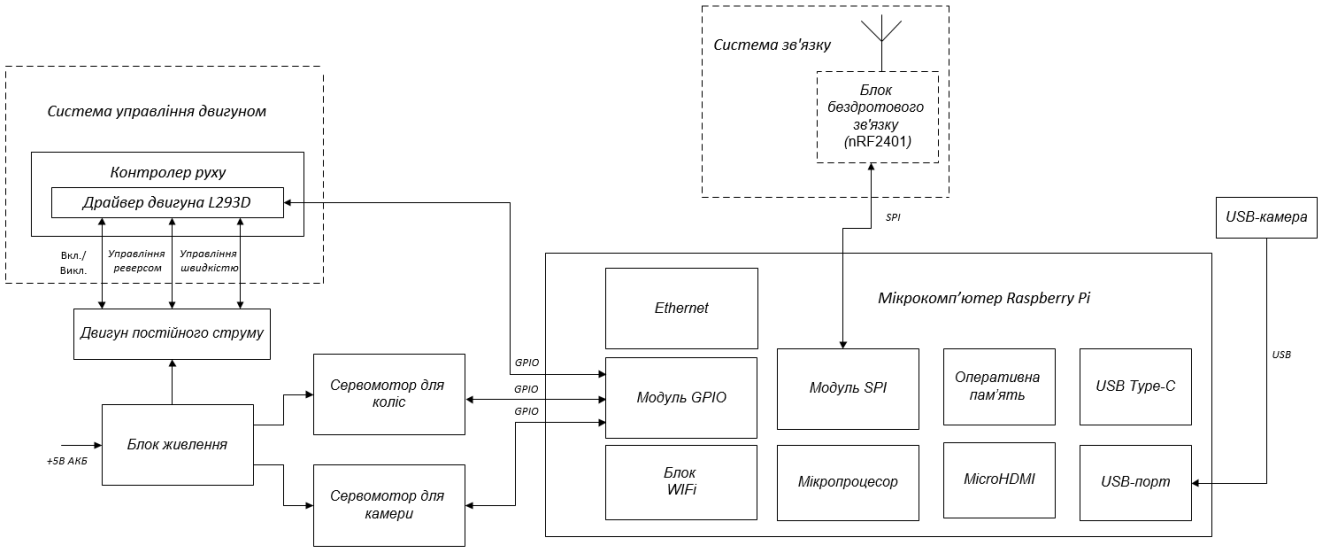

Рисунок 3.2 – Функціональна схема системи управління БНРП

Мікрокомп’ютер Raspberry Pi складається з наступних компонентів:

– мікропроцесор;

– оперативна пам’ять на 4 ГБ;

– 4 USB порти;

– USB Type-C для живлення;

– microHDMI для підключення до дисплею;

– блок Wi-Fi;

– модуль GPIO;

– модуль SPI.

До системи зв’язку відносяться:

– блок дротового зв’язку, який підключено через Ethernet до мікрокомп’ютера Raspberry Pi;

– блок бездротового зв’язку на базі модуля nRF24L01, який підключено через інтерфейс SPI до мікрокомп’ютера Raspberry Pi.

Клієнтський блок управління з використанням Wi-Fi складається з:

– персонального комп’ютера;

– клавіатури;

– дисплею;

– блоку бездротового зв’язку Wi-Fi.

Клієнтський блок управління з використанням Ethernet складається з:

– персонального комп’ютера;

– клавіатури;

– дисплею;

– блоку дротового зв’язку Ethernet.

Клієнтський блок управління з використанням радіоканалу на базі nRF24L01, який є пультом дистанційного управління складається з:

– мікроконтролеру, який має субблоки GPIO та SPI;

– джойстику, який підключено до субблоку GPIO;

– клавіатури, яка підключена до субблоку GPIO;

– блоку зв’язку RF діапазону, який підключено до субблоку SPI.

До БНРП підключена камера для отримання візуальної інформації.

Система управління двигуном має контролер руху. Контролер руху за допомогою драйвера L293D виконує наступні функції:

– керувати увімкненням і вимиканням двигуна;

– керувати реверсом двигуна;

– керувати швидкістю двигуна.

Сервомотори призначені для повороту камери та повороту напрямку руху БНРП, підключаються до портів GPIO та отримують управляючі сигнали за допомогою бібліотек для вбудованих систем.

Для живлення виконавчих механізмів використано акумуляторну батарею з напругою не менше 5В.

Повну функціональну схему БНРП можна знайти у додатку В.

Задумана система складається з апаратної частини та програмної частини. Апаратна частина – пристрій, який виконує функцію рухомої платформи. На рисунку 3.3 з зображено макет пристрою.

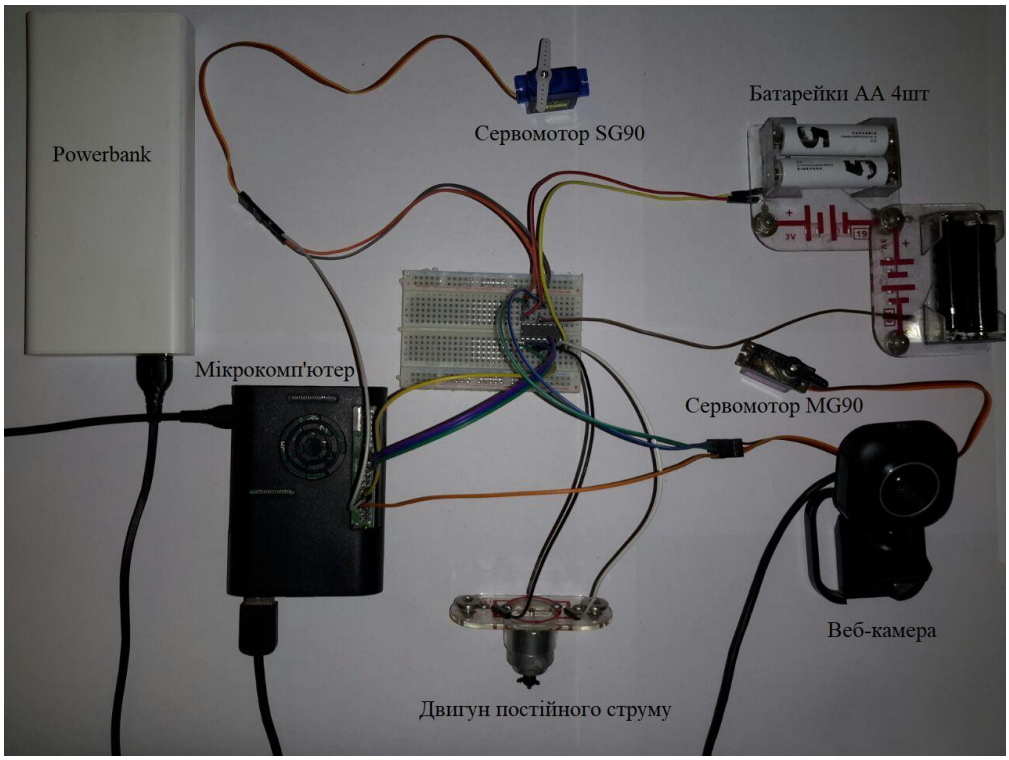

Рисунок 3.3 – Макет пристрою

Макет представляє собою частини з яких можна зібрати рухому платформу. Камера потрібна, щоб передавати зображення, акумулятори необхідні для живлення сервомоторів та двигуна.

Схему з’єднання пристроїв макету показано у додатку Г.

3.3 Висновки до розділу

У даному розділі розроблено апаратну частину системи управління БНРП, у тому числі, структурну і функціональну схеми. Структурна схема відображає принцип дії рухомої платформи у найзагальнішому вигляді. Функціональну схему створено на основі структурної. Макет БНРП зібрано на основі структурної та функціональної схем.

4 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ УПРАВЛІННЯ РУХОМОЮ ПЛАТФОРМОЮ НА БАЗІ .NET CORE І ANGULAR

4.1 Розробка UML-діаграми прецедентів системи управління рухомої платформи

Апаратна інфраструктура призначена для реалізації програмної системи рухомої платформи (ПСРП). Рухома платформа може пересуватись як вперед, так і назад. А сервомотор для передніх коліс дозволяє робити повороти вліво або вправо. Щоб контролювати рухому платформу необхідно передавати команди з достатньою швидкістю. У разі втрати зв’язку важливо вчасно зупинити рухому платформу, щоб уникнути додаткових пошкоджень.

Побудуємо таблиці для опису прецедентів ПСРП з наступними полями: найменування діяльності, унікальний ідентифікатор, загальний опис, акторів, тригери реагування, передумови для виконання, післяумови для виконання, основні й альтернативні події розвитку, можливі помилки.

Діаграму прецедентів ПСРП для користувача наведено у додатку Д. Діаграму прецедентів для ПСРП для зареєстрованого користувача наведено у додатку Е.

Таблиця 4.1 – Опис прецеденту активації камери

Назва	Включити камеру
ID	C_01
Опис	Після початку зв’язку з сервером, користувач може увімкнути камеру для отримання зображення у браузері.
Актори	Зареєстрований користувач системи
Тригери	Користувач натискає кнопку на клієнті.
Передумови	Сервер успішно запущено і він здатний приймати команди.

Продовження таблиці 4.1

Назва	Включити камеру
Післяумови	Відеокамера готова для передачі потокового відео на клієнт.
Основний потік розвитку	– користувач вмикає сервер; – клієнт встановлює зв’язок з сервером; – користувач натискає кнопку включення камери; – клієнт отримує поточне зображення.
Можливі помилки	Доступ до камери заборонено у браузері.

Таблиця 4.2 – Опис прецеденту управління поворотами камери

Назва	Повернути камеру
ID	C_02
Опис	Після початку сесії керування та підтвердження зв’язку з сервером, користувач може подати команду на поворот сервомотору вліво або вправо. Користувач обирає положення камери, використовуючи віртуальні кнопки для управління сервомотором.
Актори	Зареєстрований користувач системи
Тригери	Користувач натискає кнопку на клієнті або на фізичній клавіатурі для повороту камери.
Передумови	Користувач розпочав сесію керування та отримав підтвердження дієздатності ПСРП.

Продовження таблиці 4.2

Назва	Повернути камеру
Післяумови	Сервомотор одноразово змінив положення та переходить в режим очікування для подальшої зміни положення.
Основний потік розвитку	– користувач встановлює положення камери за допомогою кнопок; – клієнт відсилає запит на сервер; – сервер визначає отриману команду; – сервер посилає команду до програми керування сервомотором; – програма керування сервомотором вмикає сервомотор; – сервер очікує отримання наступної команди.
Можливі помилки	Відсутність зв’язку з програмою керування сервомотором. Сервер перестає посилати команди, відсилає повідомлення користувачу і намагається перезапустити програму керування сервомотором.

Таблиця 4.3 – Опис прецеденту збереження зображення з камери

Назва	Зберегти зображення
ID	C_03
Опис	Після початку зв’язку з сервером, користувач може увімкнути камеру для отримання зображення на клієнт і потім зберегти поточне зображення з камери на сервері.

Продовження таблиці 4.3

Назва	Зберегти зображення
Актори	Зареєстрований користувач системи
Тригери	Користувач має авторизуватися. Користувач натискає кнопку «Зберегти фото».
Передумови	Користувач успішно підключився до камери та отримує поточне зображення.
Післяумови	Після збереження фото, користувач отримує повідомлення про успішне збереження, та може знову зберігати фото.
Основний потік розвитку	– користувач отримує поточне зображення з камери; – користувач у клієнті натискає на кнопку збереження зображення; – сервер визначає отриману команду; – сервер посилає команду до програми збереження зображення; – програма конвертує зображення у формат base64 та записує у базу; – сервер очікує отримання наступної команди.
Можливі помилки	Відсутність відеопотоку з камери. Сервер перестає посилати команди, відсилає повідомлення користувачу.

Таблиця 4.4 – Опис прецеденту демонстрації збережених зображень з камери

Назва	Переглянути зображення
ID	C_04
Опис	Після початку зв’язку з сервером, користувач може перейти на вкладку, де збережено всі зроблені зображення.
Актори	Зареєстрований користувач системи
Тригери	Користувач переходить на вкладку, де містяться збережені зображення з камери.
Передумови	Користувач має авторизуватися. Користувач повинен мати збережені зображення.
Післяумови	Збережені зображення можна переглядати.
Основний потік розвитку	– користувач переходить на вкладку зі збереженими фото; – користувач бачить список зображень з міні-прев’ю.
Можливі помилки	– відсутність зв’язку з базою даних; – відсутність зв’язку з сервером.

Таблиця 4.5 – Опис прецеденту видалення збережених зображень з камери

Назва	Видалити зображення
ID	C_05
Опис	Після початку зв’язку з сервером, користувач може перейти на вкладку, де збережено всі зроблені зображення і видалити будь-яке.
Актори	Зареєстрований користувач системи

Продовження таблиці 4.5

Назва	Видалити зображення
Тригери	Користувач переходить на вкладку, де містяться збережені зображення з камери і натискає кнопку «Видалити».
Передумови	Користувач має авторизуватися. У користувача мають бути зображення.
Післяумови	Після видалення зображення, воно зникає зі списку, і запис у базі даних також видаляється.
Основний потік розвитку	– користувач переходить на вкладку зі збереженими фото; – користувач бачить список зображень з міні-прев'ю; – користувач може натиснути на одне з прев'ю і видалити необхідне зображення.
Можливі помилки	– відсутність зв'язку з базою даних; – відсутність зв'язку з сервером.

Таблиця 4.6 – Опис прецеденту збереження журналу дій

Назва	Збереження журналу
ID	C_06
Опис	Після початку зв'язку з сервером, користувач може записати дії, які він проводить з ПСРП: – які команди виконував двигун;

Продовження таблиці 4.6

Назва	Збереження журналу
Опис	– які команди виконував сервомотор для передніх коліс; – які команди виконував сервомотор для камери.
Актори	Зареєстрований користувач системи
Тригери	Користувач виконує дії управління
Передумови	Користувач має авторизуватися. Користувач запускає систему управління БНРП на Raspberry Pi і передає команди управління.
Післяумови	Щоб завершити запис, користувач має вийти з системи. Після наступної авторизації автоматично буде створено новий журнал для запису.
Основний потік розвитку	– користувач авторизується у систему; – сервер визначає отриману команду; – сервер посилає команду до програми збереження записів журналу дій; – програма зберігає записи журналу дій у базу.
Можливі помилки	Сервер перестає посилати команди, відсилає повідомлення користувачу.

Таблиця 4.7 – Опис прецеденту зчитування журналу дій

Назва	Зчитати журнал
ID	C_07

Продовження таблиці 4.7

Назва	Зчитати журнал
Опис	Після початку зв’язку з сервером, користувач може перейти на вкладку, де збережено всі журнали та зчитати їх.
Актори	Зареєстрований користувач системи
Тригери	Користувач переходить на вкладку, де містяться збережені записи журналу дій.
Передумови	Користувач має авторизуватися. Має бути хоча б один журнал. Вкладка має доступ до бази даних зі збереженими записами журналу дій.
Післяумови	Збережені записи журналу можна переглядати.
Основний потік розвитку	– користувач переходить на вкладку зі збереженими журналами; – користувач бачить список журналів; – запис журналу містить команду, яку виконував пристрій; – користувач може зчитати журнал дій.
Можливі помилки	– відсутність зв’язку з базою даних; – відсутність зв’язку з сервером.

Таблиця 4.8 – Опис прецеденту видалення журналу дій

Назва	Видалити журнал
ID	C_08

Продовження таблиці 4.8

Назва	Видалити журнал
Опис	Після початку зв’язку з сервером, користувач може перейти на вкладку, де збережено всі журнали і видалити будь-який.
Актори	Зареєстрований користувач системи
Тригери	Користувач переходить на вкладку, де містяться збережені журнали і натискає кнопку «Видалити».
Передумови	Користувач має авторизуватися.
Післяумови	Після видалення журналу, воно зникає зі списку, і запис у базі даних також видаляється.
Основний потік розвитку	– користувач переходить на вкладку зі збереженими записами журналу; – користувач бачить список зі збережених записів журналу дій; – користувач може видалити будь-який запис журналу.
Можливі помилки	– відсутність зв’язку з базою даних; – відсутність зв’язку з сервером.

Таблиця 4.9 – Опис прецеденту реєстрації користувача

Назва	Реєстрація користувача
ID	C_09
Опис	Користувач може зареєструватися, щоб мати повний доступ до ПСРП.

Продовження таблиці 4.9

Назва	Реєстрація користувача
Актори	Користувач системи
Тригери	Користувач вводить дані у форму реєстрації і натискає кнопку підтвердження.
Передумови	Користувач має перейти на форму реєстрації користувача.
Післяумови	Користувач може авторизуватися у систему для отримання повного доступу до ПСРП.
Основний потік розвитку	– користувач вводить дані у форму; – користувач натискає кнопку підтвердження реєстрації.
Можливі помилки	Не вдалося зареєструвати користувача: повідомити користувача, пристрій працює з обмеженим функціоналом.

Таблиця 4.10 – Опис прецеденту авторизації користувача

Назва	Авторизація користувача
ID	C_10
Опис	Користувач може авторизуватися і отримати розширені права для ПСРП.
Актори	Користувач системи
Тригери	Користувач вводить дані у форму і підтверджує вхід в систему.
Передумови	Користувач має бути зареєстрованим.

Продовження таблиці 4.10

Назва	Авторизація користувача
Післяумови	Користувач отримує повний доступ до ПСРП. Користувач після завершення роботи може вийти з системи.
Основний потік розвитку	– користувач вводить дані у форму; – користувач натискає кнопку підтвердження авторизації; – перехід на сторінку управління БНРП.
Можливі помилки	Не вдалося авторизувати користувача: повідомити користувача про невірний пароль або логін, пристрій працює з обмеженим функціоналом.

Таблиця 4.11 – Опис прецеденту управління поворотами колес

Назва	Повернути колеса
ID	C_11
Опис	Після початку сесії керування та підтвердження зв’язку з сервером, користувач може подати команду на поворот сервомотору вліво або вправо. Користувач обирає положення колес, використовуючи віртуальний джойстик для управління сервомотором.
Актори	Зареєстрований користувач системи
Тригери	Користувач натискає кнопку на клієнті або на фізичній клавіатурі для повороту платформи.

Продовження таблиці 4.11

Назва	Повернути колеса
Передумови	Користувач розпочав сесію керування та отримав підтвердження дієздатності ПСРП.
Післяумови	Сервомотор одноразово змінив положення та переходить в режим очікування для подальшої зміни положення.
Основний потік розвитку	– користувач встановлює положення віртуального джойстика; – клієнт відсилає запит на сервер; – сервер визначає отриману команду; – сервер посилає команду до програми керування сервомотором; – програма керування сервомотором вмикає сервомотор; – сервер очікує отримання наступної команди.
Можливі помилки	Відсутність зв’язку з програмою керування сервомотором. Сервер перестає посилати команди, відсилає повідомлення користувачу і намагається перезапустити програму керування сервомотором.

Таблиця 4.12 – Опис прецеденту управління двигуном

Назва	Управління двигуном
ID	C_12

Продовження таблиці 4.12

Назва	Управління двигуном
Опис	Після початку сесії керування та підтвердження зв’язку з сервером, користувач може подати команду на початок руху вперед або назад. Користувач пересуває повзунок від середини (0.0) обираючи швидкість руху платформи (від -1.0 або до 1.0).
Актори	Зареєстрований користувач системи
Тригери	Користувач пересуває повзунок для руху вперед або назад.
Передумови	Користувач розпочав сесію керування та отримав підтвердження дієздатності мобільної платформи.
Післяумови	Клієнт продовжує посилати команду поки повзунок здвинуто в тому чи іншому напрямку. Сервер вмикає двигун і утримує його у стані руху поки повзунок не зрушать в тому чи іншому напрямку.
Основний потік розвитку	– користувач пересуває повзунок в якомусь напрямку; – клієнт відсилає запит на сервер; – сервер визначає отриману команду; – сервер посилає команду до програми керування двигуном; – програма керування двигуном вмикає двигун.

Продовження таблиці 4.12

Назва	Управління двигуном
Можливі помилки	– відсутність зв’язку. Клієнт спробує відновити зв’язок. Якщо зв’язок не буде встановлено, то клієнт повідомить про втрату сигналу; – сервер перестає відправляти команди, у разі відсутності сигналу.

Таблиця 4.13 – Опис прецеденту розпізнавання вибоїн

Назва	Розпізнавання вибоїн
ID	C_13
Опис	Після початку сесії керування та підтвердження зв’язку з сервером, користувач може включити камеру та почати розпізнавати вибоїни.
Актори	Зареєстрований користувач системи
Тригери	Зареєстрований користувач натискає кнопку на екрані для включення розпізнавання об’єкту.
Передумови	Користувач має авторизуватися. Користувач має увімкнути камеру.
Післяумови	У місці демонстрації зображення з камери з’являються прямокутники, які виділяють розпізнані вибоїни.
Основний потік розвитку	– користувач вмикає камеру; – у браузері потрібно надати сайту дозвіл до користування камерою;

Продовження таблиці 4.13

Назва	Розпізнавання вибоїн
Основний потік розвитку	– користувач натискає кнопку з розпізнаванням вибоїн.
Можливі помилки	– камера не підключена; – було розпізнано помилковий об’єкт із-за низької освітленості.

Таблиця 4.14 – Опис прецеденту розпізнавання загальних об’єктів

Назва	Розпізнавання загальних об’єктів
ID	C_14
Опис	Після початку сесії керування та підтвердження зв’язку з сервером, користувач може включити камеру та почати розпізнавати загальні об’єкти.
Актори	Зареєстрований користувач системи
Тригери	Зареєстрований користувач натискає кнопку на екрані для включення розпізнавання загальних об’єктів.
Передумови	Користувач має авторизуватися. Користувач має увімкнути камеру.
Післяумови	У місці демонстрації зображення з камери з’являються прямокутники, які виділяють розпізнані об’єкти.
Основний потік розвитку	– користувач вмикає камеру; – у браузері потрібно надати сайту дозвіл до користування камерою;

Продовження таблиці 4.14

Назва	Розпізнавання загальних об’єктів
Основний потік розвитку	– користувач натискає кнопку з розпізнаванням загальних об’єктів.
Можливі помилки	– камера не підключена; – було розпізнано помилковий об’єкт із-за низької освітленості.

4.2 Вибір багатошарової архітектури програмної частини системи управління рухомої платформи

Розглянемо структуру програмного рішення ПСРП на базі .NET Core і Angular на рисунку 4.1.

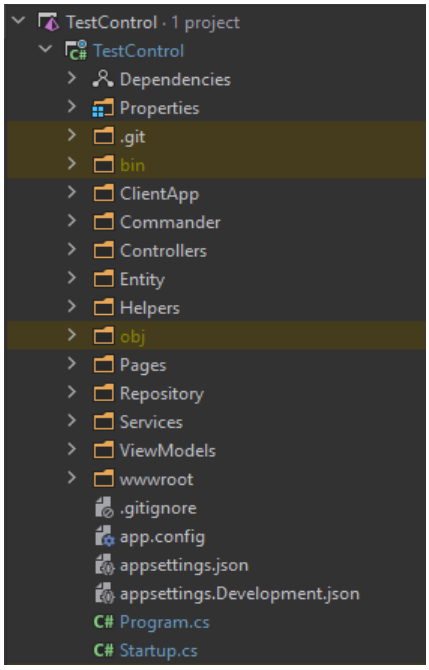

Рисунок 4.1 – Структура програмного рішення

Директорія Entity – містить POCO (Plain Old CLR Object) класи (Code first). Ця директорія являє собою рівень доменних сутностей цибулевої архітектури. Ці класи використовуються для створення таблиць бази даних. Це основна і центральна частина програми.

Директорія Repository – містить універсальні класи репозиторію з реалізацією їх інтерфейсів. Ця директорія являє собою рівень репозиторію цибулевої архітектури.

Директорія Services – містить бізнес-логіку та інтерфейси. Ці інтерфейси взаємодіють між інтерфейсом користувача та логікою доступу до даних. Оскільки цей рівень взаємодіє через інтерфейси, то така взаємодія є слабко зв’язаною. Ця директорія являє собою сервісний рівень цибулевої архітектури.

Директорія Controllers – не містить інтерфейсу користувача і є посередником між бізнес-логікою та інтерфейсом користувача. Кожен контролер має конструктор, через який за допомогою механізму dependency injection передаються сервіси в залежності від призначення.

Директорія ClientApp – програмний додаток, створений на Angular, який містить інтерфейс користувача. Щоб звернутися до контролера компонент Angular має через свої сервіси викликати методи, які знаходяться у контролері.

Використання багатошарової архітектури має наступні переваги:

– зменшення часу розробки;

– покращення якості програмних продуктів.

Головна перевага архітектури полягає у тому, що кожен рівень може розроблятися одночасно окремими командами розробників, виконуватися на власній інфраструктурі і бути оновлений чи масштабований за необхідності, не впливаючи на інші рівні.

4.3 Розробка діаграми класів системи управління рухомою платформою

Виділимо загальний інтерфейс IBaseEntity для всіх сутностей. Кожна сутність буде мати ідентифікатор (таблиця 4.15).

Таблиця 4.15 – Інтерфейс IBaseEntity

Назва	Тип даних	Призначення
Id	TKey	Ідентифікатор

Клас User представляє користувача і успадковується від класу IdentityUser, який переймає всі його властивості (таблиця 4.16).

Таблиця 4.16 – Клас User

Назва	Тип даних	Призначення
FirstName	string	Ім’я користувача
LastName	string	Прізвище користувача

Класи у таблицях 4.17-4.19 описують структуру кожної сутності.

Таблиця 4.17 – Клас Journal

Назва	Тип даних	Призначення
Id	int	Ідентифікатор журналу
UserId	int	Ключ до таблиці User

Клас JournalItems описує таблицю операцій, які виконуються для управління ПСРП (таблиця 4.18).

Таблиця 4.18 – Клас JournalItems

Назва	Тип даних	Призначення
Id	int	Ідентифікатор запису журналу
JournalId	int	Ключ до таблиці Journal
OperationName	string	Назва операції
OperationDate	DateTime	Час виконання операції

Таблиця 4.19 – Клас PhotoCamera

Назва	Тип даних	Призначення
Id	int	Ідентифікатор фотографії

Продовження таблиці 4.19

Назва	Тип даних	Призначення
UserId	int	Ключ до таблиці User
Photo	string	Збережене фото
PhotoDate	DateTime	Дата збереження фото

Реалізуємо патерн «Repository» для роботи через Entity Framework Core. Створимо інтерфейс репозиторію (таблиця 4.20).

Таблиця 4.20 – Інтерфейс IBaseRepository

Назва	Тип даних
Insert(TEntity entity)	Task<TKey>
Update(TEntity entity)	Task<bool>
Patch(TEntity entity)	Task<bool>
GetById(TKey id)	Task<TEntity>
Delete(TKey id)	Task<bool>
GetAll()	Task<IList<TEntity>>

Реалізуємо клас GlobalRepository, який може працювати з різними сутностями. Базова реалізація для репозиторію, використовує Entity Framework Core (таблиця 4.21).

Таблиця 4.21 – Клас GlobalRepository

Назва	Тип даних	Призначення
Insert(TEntity entity)	Task<TKey>	Додати запис до таблиці
Update(TEntity entity)	Task<bool>	Оновити запис у таблиці
Patch(TEntity entity)	Task<bool>	Частково оновити запис у таблиці
GetById(TKey id)	Task<TEntity>	Дістати запис по Id запису з таблиці
Delete(TKey id)	Task<bool>	Видалити запис з таблиці
GetAll()	Task<IList<TEntity>>	Отримати всі записи з таблиці

Репозиторій зберігає посилання на контекст і набір DbSet для роботи з поточною сутністю. Всі методи репозиторію фактично викликають методи DbSet і контексту даних.

Підхід Entity Framework Code First вимагає створення класу контексту доступу до даних, успадкованого від класу IdentityDbContext. Створимо клас контексту даних MyAppContext. При використанні Identity клас контексту даних буде успадковуватися не від DbContext, а від IdentityDbContext (таблиця 4.22).

Таблиця 4.22 – Клас MyAppContext

Назва	Тип даних	Призначення
Journal	DbSet<Journal>	Збереження сутності Journal у базі даних
JournalItems	DbSet<JournalItems>	Збереження сутності JournalItems у базі даних
PhotoCamera	DbSet<PhotoCamera>	Збереження сутності PhotoCamera у базі даних
OnModelCreating(ModelBuilder modelBuilder)		Можна використовувати для створення сутностей

Фактично клас MyAppContext представляє реалізацію UnitOfWork - він містить ряд репозиторіїв. Кожен репозиторій представлений об’єктом DbSet, за допомогою функціональності якого можна отримувати, створювати та видаляти дані.

Таблиця 4.23 – Клас JournalRepository

Назва	Тип даних	Призначення
GetByAppUserId(int userId)	Task<Journal>	Метод для отримання журналу по користувачу

Таблиця 4.24 – Клас JournalItemsRepository

Назва	Тип даних	Призначення
GetByJournalId(int journalId)	Task<IList<JournalItems>>	Метод для отримання запису журналу по ідентифікатору
DeleteByJournalId(int journalId)		Видалення запису журналу по ідентифікатору журналу

Таблиця 4.25 – Клас JournalRepository

Назва	Тип даних	Призначення
GetByAppUserId(int userId)	Task<PhotoCamera>	Метод для отримання фото по користувачу

Реалізацію інтерфейсів відповідних репозиторіїв оголошено на рівні Domain Services (таблиці 4.26-4.28).

Таблиця 4.26 – Клас JournalService

Назва	Тип даних	Призначення
_journalRepo	IJournalRepository	Реалізація інтерфейсу репозиторію журналу
GetById(int id)	Task<Journal>	Дістати журнал по Id запису з таблиці
GetAll()	Task<IList<Journal>>	Дістати всі журнали
GetByAppUserId(int id)	Task<Journal>	Метод для отримання журналу по користувачу
Delete(int id)	Task	Видалити журнал з таблиці
Create(Journal value)	Task<Journal>	Створити журнал у таблиці

Таблиця 4.27 – Клас JournalItemsService

Назва	Тип даних	Призначення
_journalItemsRepo	IJournalItemsRepository	Реалізація інтерфейсу запису журналу
GetById(int id)	Task< JournalItems >	Дістати запис журнал по Id запису з таблиці
GetAll()	Task<IList<JournalItems>>	Дістати всі записи журналів
GetByJournalId(int id)	Task<IList<JournalItems>>	Метод для отримання записів журналу по Id журналу
GetByAppUserId(int id)	Task< JournalItems >	Метод для отримання записів журналу по Id користувачу
Delete(int id)	Task	Видалити запис журналу у таблиці
DeleteByJournalId(int id)		Видалити запис журналу по Id журналу
Create(JournalItems value)	Task<JournalItems>	Створити запис журналу

Таблиця 4.28 – Клас PhotoCameraService

Назва	Тип даних	Призначення
_photoCameraRepo	IPhotoCameraRepository	Реалізація інтерфейсу фото
GetById(int id)	Task<PhotoCamera>	Дістати фото по Id запису з таблиці
GetAll()	Task<IList<PhotoCamera>>	Дістати всі фото
GetByAppUserId(int id)	Task<PhotoCamera>	Метод для отримання фото по користувачу

Продовження таблиці 4.28

Назва	Тип даних	Призначення
Delete(int id)	Task	Видалити фото з таблиці
Create(PhotoCameravalue)	Task<PhotoCamera>	Створити фото у таблиці

Реалізація інтерфейсів сервісів міститься у директорії Controllers.

Окремо розглянемо клас для ПСРП.

Таблиця 4.29 – Клас PiCommander

Назва	Тип даних	Призначення
InitGpio()		Ініціалізація GPIO
RotateWheels(int position)		Повернути колеса
RotateCamera(int position)		Повернути камеру
RotateServoMotor(int position, GpioPin pinNumber)		Загальний метод повороту сервомотору

Клас AccountController містить методи реєстрації та авторизації користувача, бо тільки зареєстрований користувач може керувати ПСРП.

Клас Program містить метод CreateDbIfNotExists, який створює базу даних по описаному контексту MyAppContext.

Повна діаграма класів ПСРП відображена у додатку Ж.

4.4 Розробка діаграми компонентів серверної частини

Основа серверної частини системи – це веб-сервер, реалізований на технологіях ASP.NET Core і Angular. З використанням цих технологій здійснюється обмін даними між кінцевим користувачем та розробленим пристроєм.

Діаграму компонентів ПСРП можна умовно поділити на дві складові: зовнішню і внутрішню.

Зовнішня частина для підключення до ПСРП складається з наступних компонентів:

– через браузер за допомогою Wi-Fi;

– через браузер за допомогою Ethernet;

– через джойстик.

Зовнішні пристрої, які підключаються до Raspberry Pi:

– двигун постійного струму;

– сервомотор для колес;

– сервомотор для веб-камери;

– веб-камера.

Для зберігання даних сервер використовує реляційне сховище даних PostgreSQL, яке знаходиться на віддаленому сторонньому сервері. У наступних розділах буде розроблено відповідні таблиці, схему бази даних для потреб системи. Для зберігання тимчасових даних, наприклад, даних сесії, що йдуть від користувача пристрою, використовується локальний кеш у браузері.

Внутрішня частина містить розгорнутий сервер на базі технологій ASP.NET Core і Angular.

ASP.NET Core містить директорії з контролерами, сервісами, репозиторіями і сутностями. Компонент «Commander» використовує бібліотеки, які необхідні для контролювання зовнішніх пристроїв, які підключаються до Raspberry Pi.

Angular надає інтерфейс користувача. Angular містить сервіси і компоненти, які необхідні для зв’язку з бекендом.

Для роботи нейронної мережі використовується бібліотека TensorFlowJS, яка встановлена у клієнтському додатку Angular. Для коректної роботи бібліотеки використовується файл model.json для розпізнавання об’єктів.

Веб-камера підключена до Raspberry Pi. Для коректної роботи камери було створено компонент, який може з нею взаємодіяти за допомого html-тегу <video>.

Всі описані зв’язки відображені на детальній діаграмі компонентів серверної частини у додатку И.

4.5 Розробка діаграми послідовності для системи управління рухомою платформою

Спочатку користувач вводить у строку браузера ім’я хосту і порт підключення до розгорнутого сервера на Raspberry Pi. Якщо сервер успішно розгорнуто, то користувачу необхідно авторизуватися, щоб побачити вікно, де демонструється зображення з камери та тригери керування.

Розглянемо процес передачі команди «повороту коліс». Користувач встановлює положення віртуального джойстика і клієнт надсилає запит на сервер. Виклик методу повороту коліс проходить через багатошарову архітектуру. Сервер визначає отриману команду. Сервер посилає команду до програми керування сервомотором. Програма керування сервомотором вмикає сервомотор. Даний процес показано на діаграмі послідовності у додатку К.

4.6 Розробка ER-діаграми

Для позначення обмежень використовуються наступні скорочення:

– PK – Primary Key (первісний ключ);

– FK – Foreign Key (зовнішній ключ);

– NN – Not Null (обов’язкова колонка);

– UN – Unique (унікальне значення).

Таблиця 4.30 – AspNetUsers

Назва поля	Тип даних	Обмеження
Id	integer	Pk, Unique, Not null
UserName	varchar(256)	
NormalizedUserName	varchar(256)	Unique
Email	varchar(256)	
NormalizedEmail	varchar(256)	Unique
EmailConfirmed	boolean	Not null

Продовження таблиці 4.30

Назва поля	Тип даних	Обмеження
PasswordHash	text	
SecurityStamp	text	
ConcurrencyStamp	text	
PhoneNumber	text	
PhoneNumberConfirmed	boolean	Not null
TwoFactorEnabled	boolean	Not null
LockoutEnd	timestamp with time zone	
LockoutEnabled	boolean	Not null
AccessFailedCount	integer	Not null
FirstName	text	
LastName	text	

Таблиця 4.31 – AspNetRoles

Назва поля	Тип даних	Обмеження
Id	integer	Pk, Unique, Not null
Name	varchar(256)	
NormalizedName	varchar(256)	Unique
ConcurrencyStamp	text	

Таблиця 4.32 – AspNetUserRoles

Назва поля	Тип даних	Обмеження
UserId	integer	Not null, Fk(AspNetUsers)
RoleId	integer	Not null, Fk(AspNetRoles)

Таблиця 4.33 – Journal

Назва поля	Тип даних	Обмеження
Id	integer	Pk, Unique, Not null

Продовження таблиці 4.33

Назва поля	Тип даних	Обмеження
UserId	integer	Not null, Fk(AspNetUsers)

Таблиця 4.34 – JournalItems

Назва поля	Тип даних	Обмеження
Id	integer	Pk, Unique, Not null
JournalId	integer	Not null, Fk(Journal)
OperationName	text	
OperationDate	timestamp	

Таблиця 4.35 – PhotoCamera

Назва поля	Тип даних	Обмеження
Id	integer	Pk, Unique, Not null
UserId	integer	Not null, Fk(AspNetUsers)
Photo	text	
PhotoDate	timestamp	

Інфологічна модель бази даних зображена на ER-діаграмі у додатку Л.

4.7 Розробка інтерфейсу користувача

Під час розробки інтерфейсу користувача для ПСРП було використано передові інструменти у області веб-програмування, а саме: Angular та Bootstrap. Також було задіяно додаткову бібліотеку Tensorflow.js, за допомогою якої можна навчати і розгортати моделі машинного навчання. Tensorflow.js можна використовувати з різними front-end фреймворками, у тому числі з Angular.

Всі бібліотеки було завантажено за допомогою менеджера пакетів npm.

На рисунку 4.2 зображено інтерфейс для ПСРП. Інтерфейс управління складається з двох окремих Angular компонентів. Перший компонент відповідає за управління камерою і він включає наступний функціонал:

– демонстрація зображення з камери у прямому ефірі;

– кнопка «Toggle Webcam» може відкривати/приховувати вікно демонстрації зображення з камери;

– кнопка «Detect pit» включає розпізнавання об’єкту «вибоїна»;

– кнопка «Detect default object» спрямована на локалізацію та ідентифікацію декількох об’єктів на одному зображенні;

– кнопка зі значком камери робить фото з камери та зберігає його у базу даних.

Другий компонент складається з трьох об’єктів:

– джойстик може пересувати платформу вперед або назад, вліво або вправо;

– дві кнопки справа від джойстику обертають камеру вліво або вправо.

Джойстик належить до бібліотеки ngx-joystick і дозволяє швидко, майже без затримок, керувати рухомою платформою.

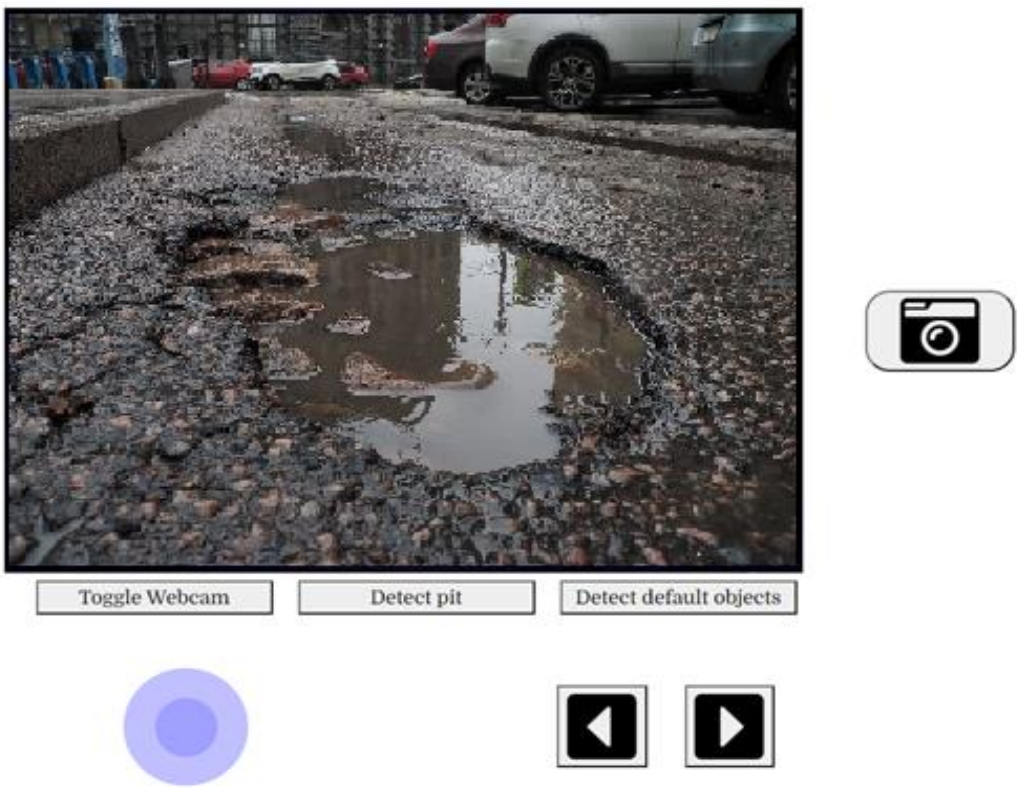

Рисунок 4.2 – Інтерфейс управління рухомою платформою

Щоб відобразити ці два компоненти на одній сторінці було вирішено застосувати селектор. Селектор повідомляє Angular про створення та вставку екземпляра компонента скрізь, де він знаходить відповідний тег у шаблоні HTML.

4.8 Висновки до розділу

Для розробки ПСРП (програмної системи рухомої платформи) застосовано багатошарову архітектуру.

В магістерській дисертації для опису ПСРП побудовано декілька видів діаграм:

– діаграму класів, яка ілюструє структуру ПСРП, описує класи, їх атрибути, методи і відносини між об’єктами;

– діаграму компонентів для візуалізації організації ПСРП і залежностей між ними;

– діаграму послідовності для відображення процесу передачі команди «Повернути колесо»;

– ER-діаграму для проектування реляційної бази даних.

Розроблено інтерфейс користувача для управління БНРП.

5 НЕЙРОННА МЕРЕЖА В СИСТЕМІ УПРАВЛІННЯ РУХОМОЮ ПЛАТФОРМОЮ

У статті [20] було розроблено модифіковану діаграму компонентів, яка містить інфраструктурні елементи для забезпечення роботи нейронної мережі. Для реалізації було обрано Keras API. Keras – це провідна open-source бібліотека, яка призначена для створення нейронних мереж і проектів машинного навчання.

Підхід роботи і взаємодії з нейронною мережею, який описано у статті [20] є не дуже вдалим. Взаємодія з камерою відбувається через скрипт у файлі server.py за допомогою бібліотеки OpenCV, який виконується через протокол SSH (Secure Shell). Такий підхід не є безпечним та продуктивність такого рішення є достатньо низькою. Так розпізнавання об’єкту з Raspberry Pi займає багато ресурсів і в результаті на виході, отримуємо не більше 1-2 кадрів/секунду.

У статті [21] розглядається побудова програмної системи на базі малопотужного мікроконтролера, але в нашому випадку це рішення може бути використано як допоміжне для сполучення виконавчого механізму та базового контролера, що містить систему нейронної мережі.

У статті [22] опубліковане рішення використання мікрокомп’ютера для системи управління технологічним процесом, але в цьому рішенні відсутній образ програмної системи, що може розпізнавати образи.

У статті [23] запропоновано використати метод SSD (Single Shot Multibox Detector). Цей метод дозволяє виявляти об’єкти на зображеннях з використанням однієї глибокої нейронної мережі. SSD дискретизує вихідний простір обмежуючих блоків в набір блоків для різних форматів зображення і масштабів. Під час прогнозування мережа генерує оцінки наявності кожної категорії об’єктів в кожному блоці за замовчуванням і виробляє коригування для блоку, щоб краще відповідати формі об’єкта. Крім того, мережа об’єднує прогнози з декількох карт характеристик з різними роздільними здатностями для природної обробки об’єктів різних розмірів. Експериментальні результати для наборів даних POSCAL VOC, COCO і ILSVRC підтверджують, що SSD має конкурентоспроможну точність по відношенню до

методів, які використовують додатковий крок пропозиції об’єкта і набагато швидше, забезпечуючи при цьому уніфіковану структуру для навчання і виведення.

Завдання виявлення, локалізації та класифікації об’єктів виконуються за один прямий прохід мережі.

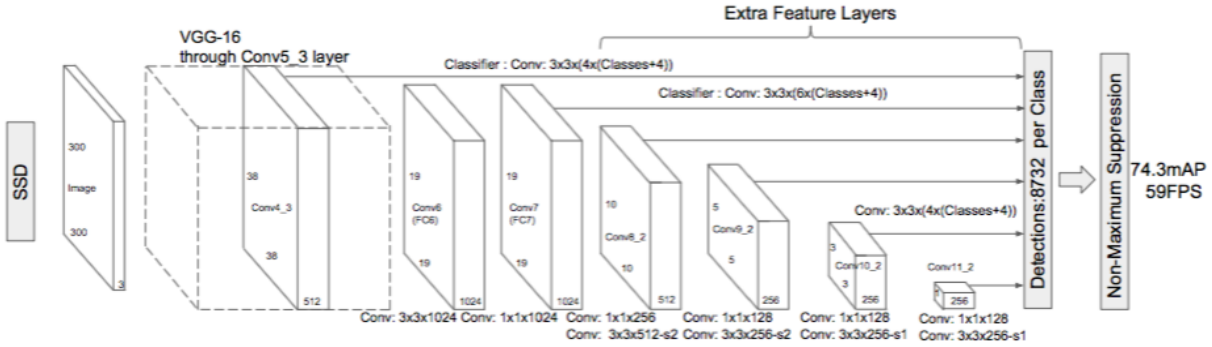

Рисунок 5.1 – Архітектура SSD [23]

5.1 Технологія навчання й адаптації нейронної мережі для системи управління рухомою платформою

В ПСРП для допомоги оператору у процесі автоматизованого управління рухомою платформою включений модуль розпізнавання об’єктів на основі нейронної мережі. Загальна структура БНРП з нейронною мережею показана на рисунку 5.2. Опишемо призначення кожного елемента схеми БНРП:

– пульт управління БНРП використовується для передачі інформації, аудіо, відео людині-оператору;

– мікрокомп’ютер БНРП використовується для обробки інформації, що отримується з датчиків, камери і сенсорів;

– пристрій сполучення БНРП використовується для сполучення з виконавчими механізмами;

– виконавчі механізми БНРП використовуються для зміни положення БНРП у просторі;

– програмний модуль захоплення зображення використовується для аналізу зображення з відеокамери;

– нейронна мережа використовується для розпізнавання об’єктів з програмного модуля захоплення зображення;

– програмні елементи для обробки сигналів і інформації оператора використовуються для отримання результатів розпізнавання для людини-оператора.

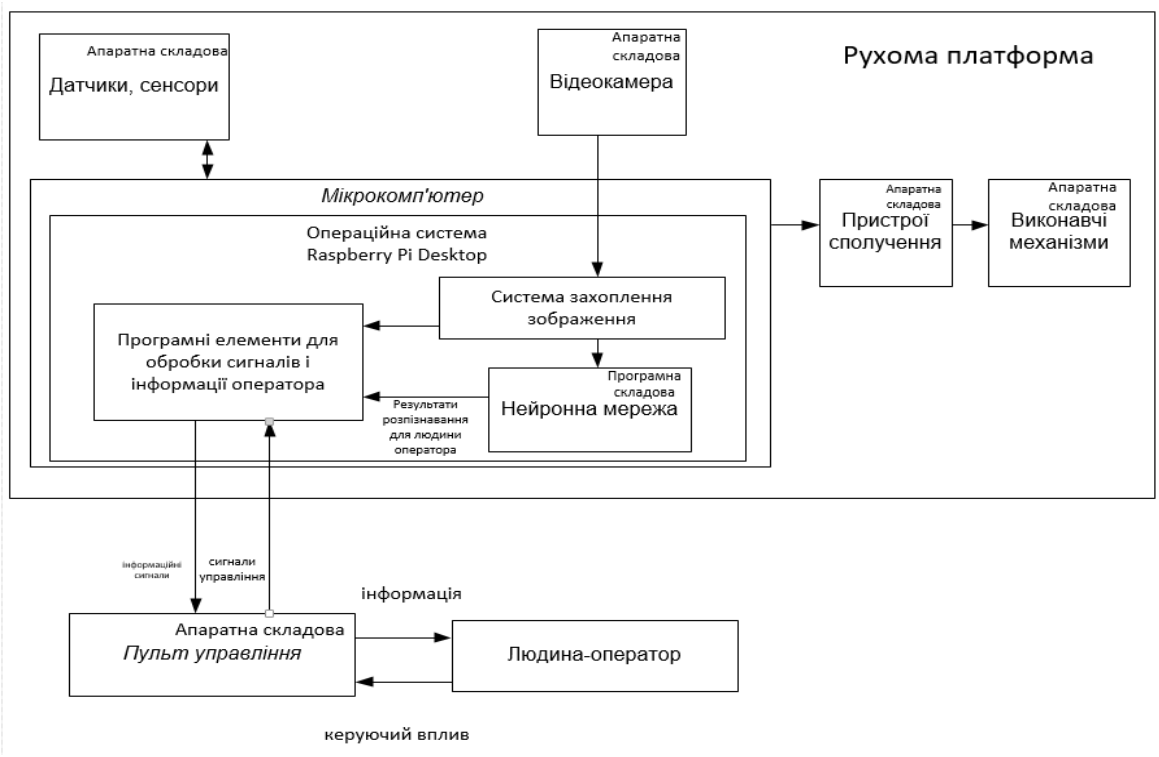

Рисунок 5.2 – Загальна структура рухомої платформи з нейронною мережею

Розглянемо практичну реалізацію нейронної мережі для управління рухом платформи. В якості прикладу будуть обрані нерівності, що зустрічаються на шляху рухомої платформи.

Для вирішення цієї задачі було зібрано близько 200 зображень вибоїн. Для тестування було використано близько 50 зображень, а інші – для частини навчання. Наступним кроком є анотування отриманих зображень. Для цього було застосовано програму LabelImg, LabelImg – це інструмент для анотування графічних зображень. Отримані анотації зберігаються у вигляді XML-файлів у форматі PASCAL VOC. Далі представлені зразки вибоїн (рисунок 5.3), фото яких було зроблено у місті Києві. На рисунку 5.4 зображено анотоване зображення за допомогою LabelImg.

Рисунок 5.3 – Приклад фото до підготовки

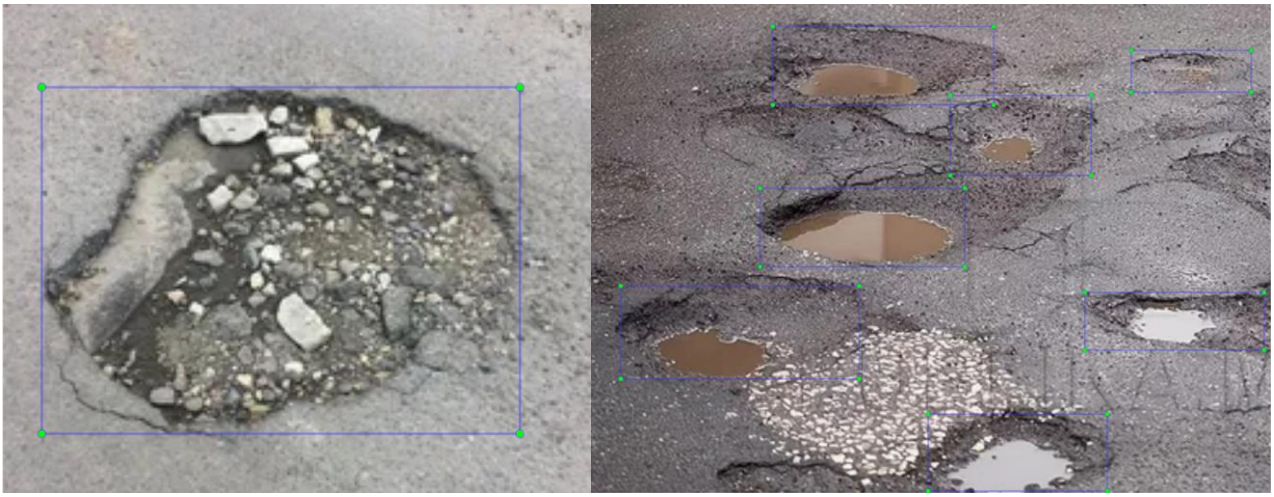

Рисунок 5.4 – Приклад фото після підготовки

При роботі з великими наборами даних, використання двійкового формату файлів для зберігання даних може мати істотний вплив на продуктивність конвеєру імпорту (зображень) і, як наслідок, на час навчання моделі. Двійкові дані займають менше місця на диску, копіюють менше часу і можуть зчитуватися з диску набагато ефективніше, для цих задач підійде TFRecord. Однак чиста продуктивність – не єдина перевага формату файлу TFRecord. TFRecord оптимізований для використання з TensorFlow у декількох варіантах. Формат TFRecord дозволяє легко об’єднувати кілька наборів даних і легко інтегрується з функціями імпорту та попередньої обробки даних, що надаються бібліотекою. Зокрема для наборів даних, які занадто великі для того, щоб їх можна було повністю зберегти в пам’яті, це є перевагою,

оскільки тільки дані, які потрібні в даний момент, завантажуються з диска і потім обробляються. Сконвертуємо XML-файли, які було отримані після обведення в LabelImg, у формат TFRecord.

Щоб перетворити XML-файли в TFRecord, спочатку перетворимо їх в CSV за допомогою скрипта Python xml_to_csv.py, який можна знайти за посиланням [24].

XML-файли в папках “images/training”, а також “images/test” перетворюються в два файли CSV, один для тренування та один для тестування моделі.

З репозиторію [24] використаємо скрипт generate_tfrecord.py. За допомогою цього скрипту можна згенерувати TFRecord файли. Для цього необхідно внести корективи у ньому. У метод class_text_to_int(row_label) необхідно додати назву об’єкту, який досліджуємо та id (if row_label == ‘pit’: return 1).

Запустимо наступні скрипти у консолі:

- python generate_tfrecord.py --csv_input=data/train_labels.csv --output_path=data/train.record
- python generate_tfrecord.py --csv_input=data/test_labels.csv --output_path=data/test.record

Щоб отримати детектор вибоїн, можна використовувати попередньо навчену модель, і потім використовувати трансферне навчання для вивчення нового об’єкта, або можна вивчати нові об’єкти повністю з нуля. Перевага трансферного навчання полягає в тому, що навчання може бути набагато швидше, а необхідних даних, які можуть знадобитися, набагато менше. З цієї причини було обрано проводити трансферне навчання. TensorFlow має досить багато попередньо навчених моделей з доступними файлами контрольних точок, а також файлами конфігурації.

Для цієї задачі було використано файл конфігурації ssd_mobilenet_v1_pets.config [25], який буде скориговано відповідно до моделі. У файлі конфігурації необхідно встановити кількість досліджуваних об’єктів (num_classes), у fine_tune_checkpoint вказуємо шлях до папки, де лежить попередньо навчена модель. Також необхідно створити мапу міток, яка в основному представляє собою словник, який містить ідентифікатор та ім’я класів, які хочемо виявити. Такий файл буде мати розширення pbtxt.

Перейдемо безпосередньо до навчання моделі. Попередньо необхідно завантажити репозиторій [25] і з папки legacy скопіюємо всі файли у папку object_detection і перейдемо до цієї папки. Наступний скрипт запустить навчання моделі:

python train.py --train_dir=training/ --pipeline_config_path=training/ssd_mobilenet_v1_pets.config --logtostderr

Перевіримо на скільки добре працює модель. Для цього потрібно експортувати граф виводу. У “models/object_detection”, є скрипт “export_inference_graph.py”.

python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/ssd_mobilenet_v1_pets.config --trained_checkpoint_prefix training/model.ckpt-**** --output_directory new_graph, де **** номер контрольної точки.

Було створено власну модель, яка може виявляти необхідний об’єкт. Така модель має вихідний формат *.pb.

Формат *.pb не підійде для Angular, тому необхідно сконвертувати цю модель у формат, який підтримує Angular. Для цього необхідно завантажити бібліотеку tensorflowjs (pip install tensorflowjs). TensorFlow.js - це бібліотека для машинного навчання в JavaScript, але вона підійде і для TypeScript. Сконвертуємо модель на Python у формат, який підтримує TypeScript;

tensorflowjs_converter
--input_format=tf_saved_model
--
output_node_names=‘detection_boxes,detection_classes,detection_features,detection_multiclass_scores,detection_scores,num_detections,raw_detection_boxes,raw_detection_scores’ --output_format=tfjs_graph_model
--signature_name=serving_default new_graph/saved_model new_graph/converted

На виході отримаємо файл у форматі json та декілька bin файлів. Головним файлом є json і з ним можна працювати на Angular.

Для того, щоб провести порівняння процесу навчання нейронної мережі було вирішено використати сервіс Google Colab. Цей сервіс дозволяє писати і виконувати

код Python у браузері. Також у Google Colab можна легко змінювати середовище виконання і проводити навчання на віддаленому GPU.

Для локального навчання власної моделі було використано власний CPU Intel Core i7 4770.

Процес навчання проводився на віддаленому GPU і локальному CPU. Далі надано порівняльну характеристику процесу навчання нейронної мережі.

5.2 Результати порівняння процесу навчання нейронної мережі

TensorBoard надає візуалізацію та інструменти, необхідні для експериментів з машинним навчанням:

– відстеження та візуалізація таких показників, як втрати і точність;

– візуалізація графа моделі (операції і шари);

– перегляд гістограм ваг, зсувів або інших тензорів по мірі їх зміни в часі;

– проектування вкладень в простір меншої розмірності;

– відображення зображень, тексту та аудіоданих;

– профілювання програм TensorFlow.

Далі на рисунках 5.5 і 5.6 наведено показники швидкості навчання нейронної мережі. Швидкість навчання показує як швидко навчається нейронна мережа розпізнавати об’єкт на CPU, де кількість кроків складає 2313 за 1 годину 53 хвилини, а на GPU 3628 за 23 хвилини 44 секунди. Швидкість навчання на GPU в 4.4 рази швидше ніж на CPU.

Рисунок 5.5 – Швидкість навчання на CPU

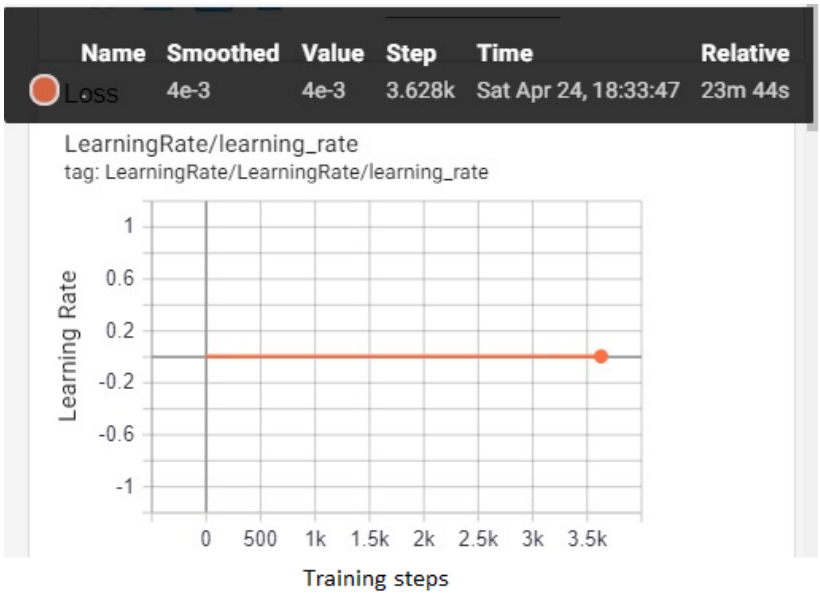

Рисунок 5.6 – Швидкість навчання на GPU

Втрата класифікації описує наскільки результат для класифікації об’єкту, отриманий нейронною мережею, відрізняється від очікуваного результату, тобто вказує на величину помилки, яку модель зробила при прогнозі. При аналізі зображення втрата класифікації вказує, чи збігається клас обмежувальної рамки з прогнозованим класом. Втрату класифікації відображено на рисунках 5.7 і 5.8.

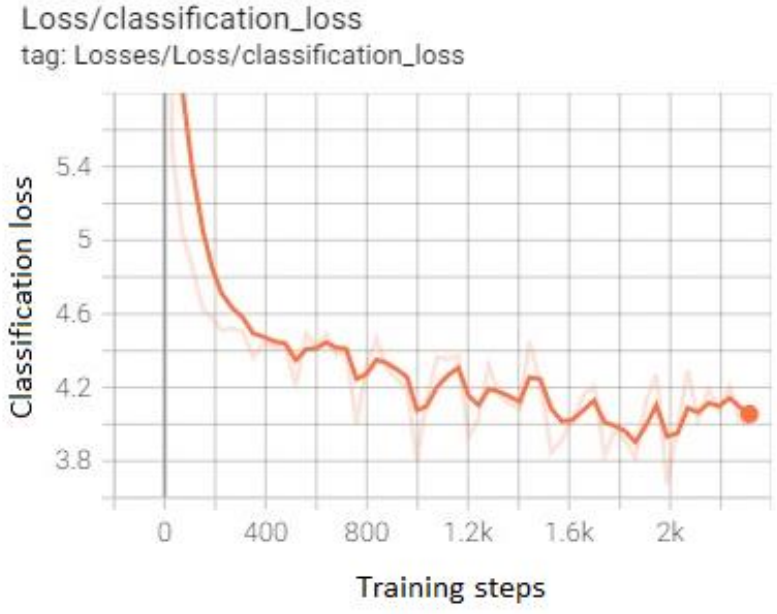

Рисунок 5.7 – Втрата класифікації на CPU

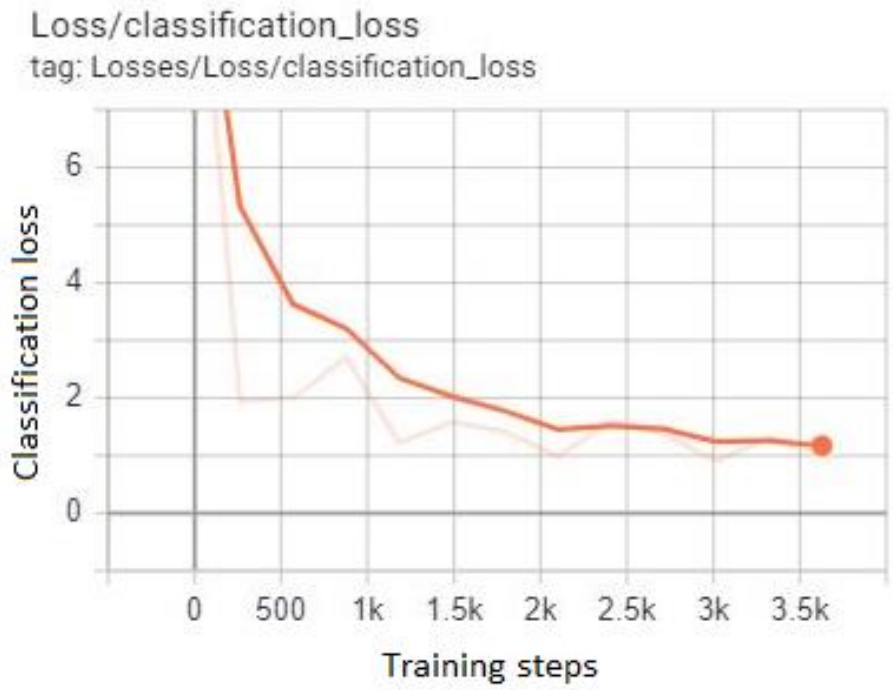

Рисунок 5.8 – Втрата класифікації на GPU

На рисунках 5.9 і 5.10 відображено показники втрати локалізації, які відповідають за передбачення зміщення обмежувальної рамки.

Рисунок 5.9 – Втрати локалізації на CPU

Рисунок 5.10 – Втрати локалізації на GPU

На рисунках 5.11 і 5.12 відображено показники регуляризації, які відносяться до набору різних методів, що знижують складність моделі нейронної мережі під час навчання. Таким чином, нейронна мережа запобігає перенавчанню і використанню зайвих ресурсів.

Рисунок 5.11 – Втрати регуляризації на CPU

Рисунок 5.12 – Втрати регуляризації на GPU

На рисунках 5.13 і 5.14 відображено показники загальної суми втрат, що коливається по мірі руху моделі. Кожна міні-партія прикладів зображень оновлює параметри моделі, щоб підштовхнути її до кращої відповідності тим прикладам, які не завжди є репрезентативними для всього набору даних.

Рисунок 5.13 – Загальні втрати на CPU

Рисунок 5.14 – Загальні втрати на GPU

На рисунках 5.15 і 5.16 відображено показник ефективності Global_step/sec, який показує, скільки партій (зображень) було оброблено за секунду під час підготовки моделі.

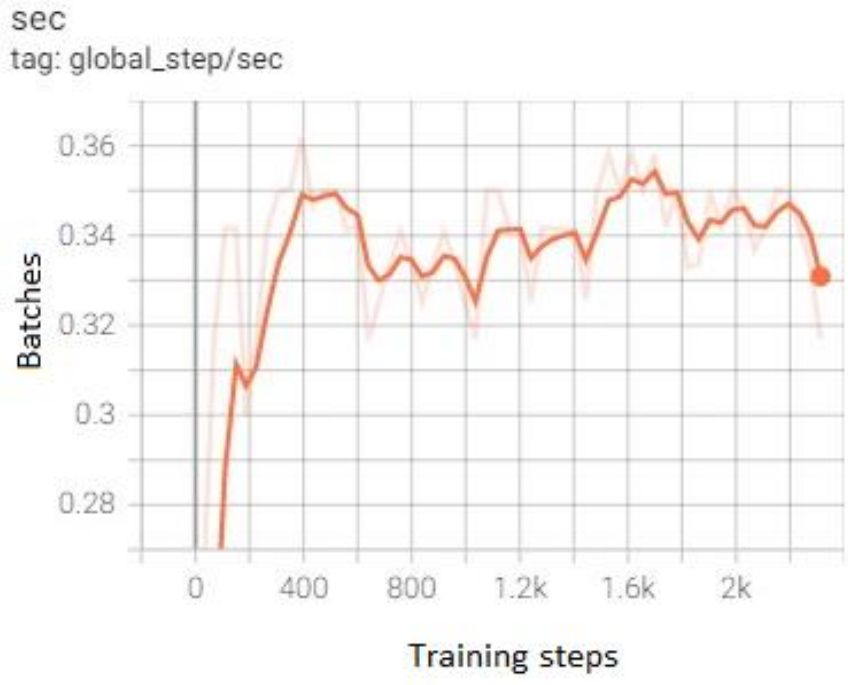

Рисунок 5.15 – Показник ефективності на CPU

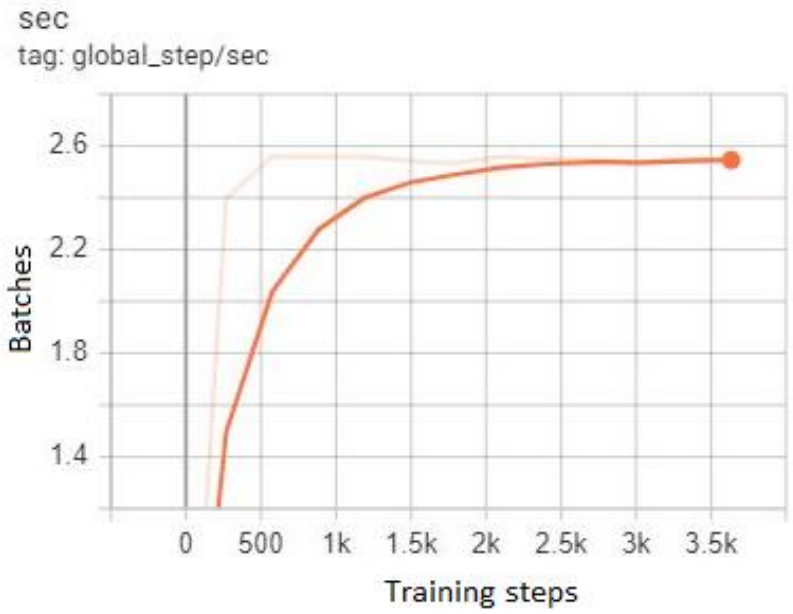

Рисунок 5.16 – Показник ефективності на GPU

5.3 Експеримент по тестуванню нейронної мережі

Умови експерименту.

Фотографії вибоїн на дорогах 12 шт. Камера системи має роздільну здатність 640*480. Кольорові фото, кількість вибоїн на фото від 1 до 13.

Опис експерименту.

Демонстрація зображень відбувається з планшету через камеру, яка підключена до Raspberry Pi. Для початку експерименту необхідно запустити веб-сервер, який розміщено за адресою raspberrypi:5000. Наступним кроком активуємо камеру для розпізнавання вибоїн. Після розпізнавання зображення камерою, запишемо результати до таблиць 5.1 і 5.2.

Результати експерименту.

Таблиця 5.1 – Результати розпізнавання нейронною мережею

Номер експерименту	Номер фото, короткий опис фото	Результати розпізнавання, характеристики
1	i_051.jpg, наявність 1 вибоїни	Розпізнано 1 вибоїну

Продовження таблиці 5.1

Номер експерименту	Номер фото, короткий опис фото	Результати розпізнавання, характеристики
2	i_052.jpg, наявність 2 вибоїн	Розпізнано 2 вибоїни
3	i_054.jpg, наявність 1 вибоїни	Розпізнано 1 вибоїну
4	i_060.jpg, наявність 6 вибоїн	Розпізнано 4 вибоїни
5	i_066.jpg, наявність 5 вибоїн	Розпізнано 4 вибоїни
6	i_067.jpg, наявність 1 вибоїни	Розпізнано 1 вибоїну
7	i_070.jpg, наявність 2 вибоїн	Розпізнано 2 вибоїни
8	i_072.jpg, наявність 1 вибоїни	Розпізнано 1 вибоїну
9	i_073.jpg, наявність 2 вибоїн	Розпізнано 2 вибоїни
10	i_075.jpg, наявність 13 вибоїн	Розпізнано 5 вибоїн
11	i_084.jpg, наявність 2 вибоїн	Розпізнано 2 вибоїни
12	i_162.jpg, наявність 4 вибоїн	Розпізнано 2 вибоїни

Кількість експериментів залежить від складності розпізнавання вибоїн. Якщо на зображенні знаходиться більше 2 вибоїн, експеримент для цього зображення буде проведено повторно.

Таблиця 5.2 – Результати розпізнавання нейронною мережею

Кількість експериментів	Відсоток розпізнавання	Усереднені характеристики розпізнавання
1	i_051.jpg, 88-92%	90%
2	i_052.jpg, 80-90%	85%
3	i_053.jpg, 97-99%	98%
3	i_060.jpg, 0-90%	60%
4	i_066.jpg, 0-94%	75,2%
1	i_067.jpg, 93-99%	96%

Продовження таблиці 5.2

Кількість експериментів	Відсоток розпізнавання	Усереднені характеристики розпізнавання
2	i_070.jpg, 83-91%	87%
1	i_072.jpg, 97-99%	98%
2	i_073.jpg, 84-88%	86%
6	i_075.jpg, 0-88%	33,85%
3	i_084.jpg, 88-94%	91%
4	i_162.jpg, 0-96%	48%

Узагальнення результатів обробки зведено до графіків, що показані на рисунках 5.17 і 5.18.

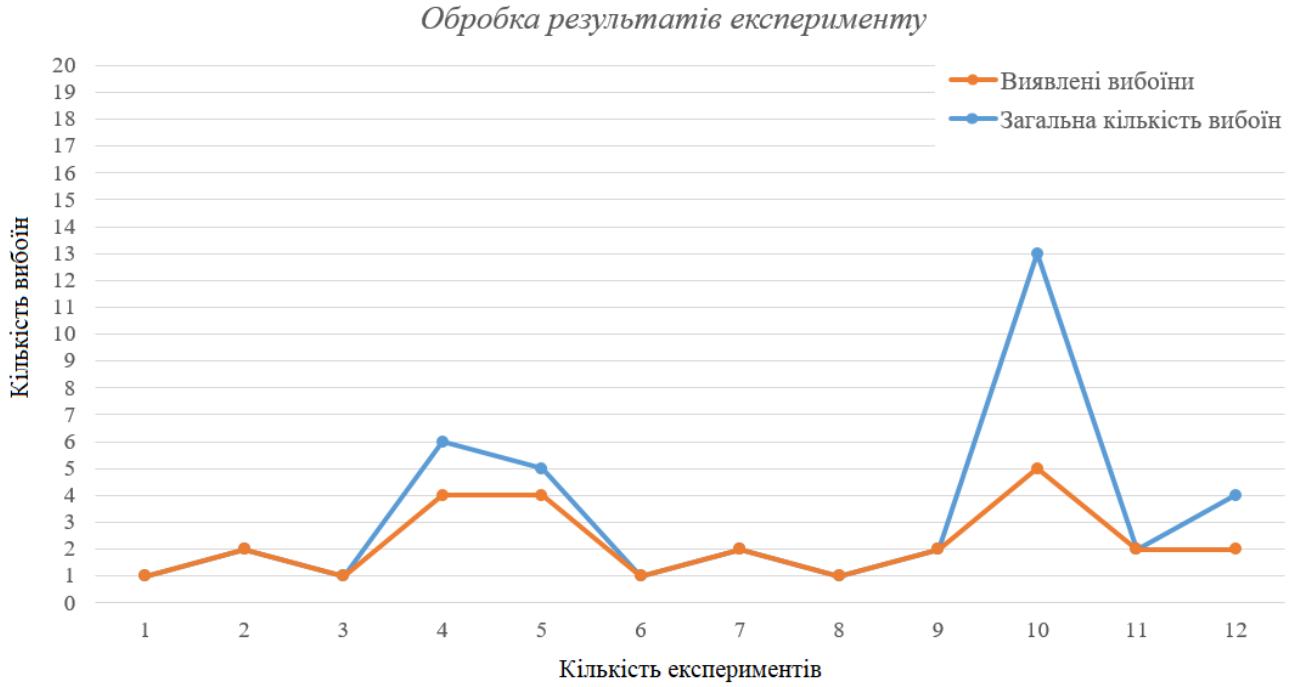

Рисунок 5.17 – Оброблені результати експерименту

На рисунку 5.17 синім кольором позначено загальну кількість вибоїн. Помаранчевим кольором позначено виявлені вибоїни. Якщо на зображенні загальна кількість вибоїн не перевищує 4, то нейронна мережа розпізнає всі вибоїни. Якщо

загальна кількість вибоїн на зображенні більше 4, то точність розпізнавання нейронною мережею спадає, що продемонстровано у таблиці 5.2.

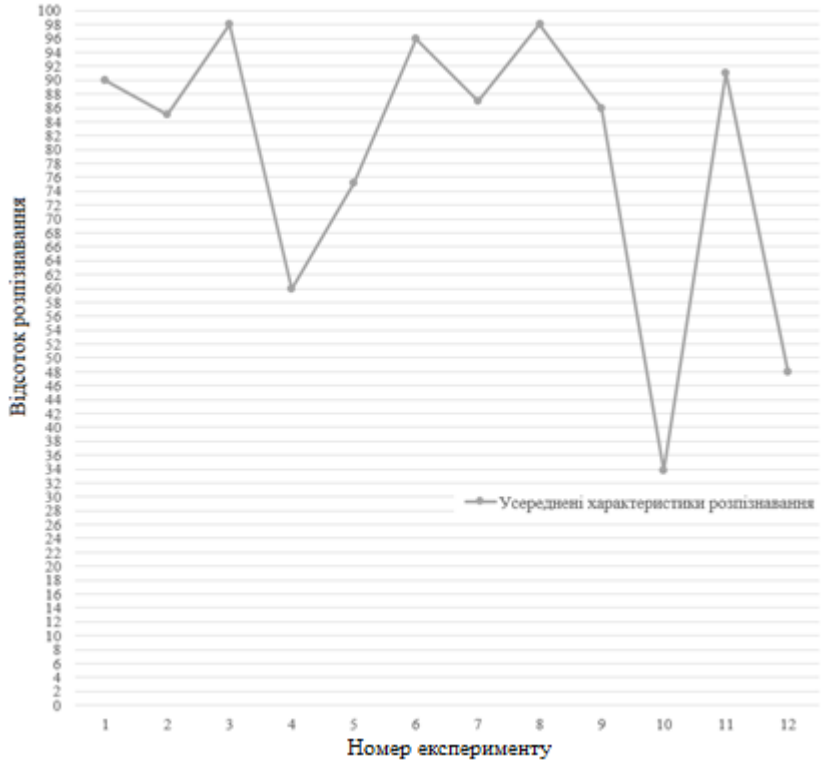

Рисунок 5.18 – Відсоток розпізнавання вибоїн

На рисунку 5.18 наведено усереднені характеристики розпізнавання вибоїн. Якщо в процесі проведення експерименту загальна кількість вибоїн зростає, то відсоток розпізнавання зменшується. Необхідність збільшення кількості експериментів для розпізнавання зображень зумовлено кількістю вибоїн на зображенні. Збільшення кількості експериментів покращує результати розпізнавання.

5.4 Висновки до розділу

У результаті проведеного дослідження отримана програмна складова нейронної мережі.

У даному розділі застосовано процес навчання й адаптації нейронної мережі. Процес підготовки до навчання нейронної мережі містить наступні пункти:

– використання програмних інструментів з репозиторіїв, які наведено у підрозділі 5.1;

– анотування близько 200 зображень вибоїн;

– приведення анотацій зображень (координат).

Для класифікації зображень застосовано метод трансферного навчання нейронної мережі.

Проведено порівняльний аналіз продуктивності процесу навчання нейронної мережі на CPU і GPU за різними критеріями, які наведено у підрозділі 5.2.

Нейронна мережа була протестована через ПСРП. За результатами експериментів були отримані такі висновки:

– якщо загальна кількість вибоїн на зображенні більше 4, то точність розпізнавання нейронною мережею спадає;

– якщо в процесі проведення експерименту загальна кількість вибоїн зростає, то відсоток розпізнавання зменшується.. Кількість експериментів для розпізнавання зображень залежить від кількості вибоїн на зображенні. Збільшення кількості експериментів покращує результати розпізнавання.

На точність результатів розпізнавання образів також впливає освітленість навколишнього середовища, де було проведено експерименти.

6 ВИЗНАЧЕННЯ ПОТРІБНОЇ КІЛЬКОСТІ КАНАЛІВ ЗВ'ЯЗКУ ДЛЯ ПОКРАЩЕННЯ ПОКАЗНИКІВ НАДІЙНОСТІ РУХОМОЇ ПЛАТФОРМИ

Показники надійності – це кількісні характеристики одного або декількох властивостей, складових надійності об'єкта. Якщо показник надійності характеризує одну з властивостей надійності, то він називається одиничним, якщо ж кілька властивостей — комплексним показником надійності [26].

Під ймовірністю безвідмовної роботи об'єкта мається на увазі ймовірність того, що в межах заданого напрацювання відмова об'єкта не виникне. Ймовірність безвідмовної роботи є основною кількісною характеристикою безвідмовності об'єкта на заданому часовому інтервалі. Зазвичай T – час безперервної справної роботи об'єкта від початку роботи до першої відмови, а через t – час, за який необхідно визначити ймовірність безвідмовної роботи.

Середнє напрацювання до відмови – це функція розподілу (інтегральна функція або щільність), яка повністю характеризує випадкову величину. Однак для вирішення деяких завдань досить знати тільки кілька моментів випадкової величини. Момент першого порядку (математичне очікування) напрацювання до першої відмови позначають $T_{сер}$ і називають середнім напрацюванням до відмови (або середнім часом безвідмовної роботи).

Інтенсивність відмов – це відношення числа відмовлення об'єктів в одиницю часу до середнього числа об'єктів, що продовжують справно працювати в даний інтервал часу. Зазвичай інтенсивність відмов позначають λ.

6.1 Вибір моделі надійності

Найчастіше визначити конкретний закон розподілу часу до відмови елемента або системи дуже складно або неможливо. У цих випадках мають на увазі, що випадкова величина – час до відмови – розподілена за відомим законом, тобто що закон розподілу відомий попередньо. В якості таких законів може бути використано будь-який розподіл, визначений на позитивній основі часу (або комбінація

розподілів). Найбільш часто в теорії надійності використовуються експоненціальний закон розподілу, розподілу Вейбула і Релея, логнормальний закон розподілу та інші. Вибравши конкретний закон розподілу, кажуть, що для елемента (системи) справедлива відповідна модель надійності.

6.1.1 Експоненціальна модель надійності (ЕМН)

ЕМН має на увазі, що інтенсивність відмов об’єкта (системи) постійна [26]:

$$\lambda(t) = \lambda = const, \quad \lambda > 0, \tag{6.1}$$

а час до відмови є безперервною випадковою величиною, розподіленою експоненціально, тобто функція ймовірності відмови приймає вигляд [26]

$$Q(t) = 1 - e^{-\lambda t}. \tag{6.2}$$

Далі наведено формули ймовірності безвідмовної роботи (6.3), щільності відмов (6.4), а також середнього часу до відмови системи (6.5) [26]:

$$P(t) = e^{-\lambda t}; \tag{6.3}$$

$$f(t) = \lambda e^{-\lambda t}; \tag{6.4}$$

$$T_{\text{сер}} = \frac{1}{\lambda}. \tag{6.5}$$

Безсумнівною перевагою ЕМН є простота залежностей між показниками надійності, а також простота розрахунку надійності для складних систем. Однак ЕМН не може похизуватися акуратністю. Експоненціальна модель адекватна тільки в період нормальної експлуатації об’єкта (системи), тому що ігнорує періоди приробітку і зносу.

6.1.2 Модель надійності Релея (МНР)

МНР має на увазі, що час до відмови є безперервною випадковою величиною, розподіленою за законом Релея [26]:

$$Q(t) = 1 - \exp\left(-\frac{t^2}{2\sigma^2}\right), \sigma > 0, \tag{6.6}$$

де σ – параметр розподілу, що має розмірність часу. Тоді, інші показники надійності будуть виражатися наступними залежностями [26]:

$$P(t) = \exp\left(-\frac{t^2}{2\sigma^2}\right); \quad (6.7)$$

$$f(t) = \frac{t}{\sigma^2}\exp\left(-\frac{t^2}{2\sigma^2}\right); \quad (6.8)$$

$$\lambda(t) = \frac{f(t)}{P(t)} = \frac{t}{\sigma^2}; \quad (6.9)$$

$$T_{\text{сер}} = \sigma\sqrt{\frac{\pi}{2}}. \quad (6.10)$$

Форма функції інтенсивності (6.9) визначає обмеження щодо використання цієї моделі-відмови механічних систем (де в силу тертя інтенсивність постійно зростає) або моделювання процесів зносу.

6.1.3 Модель надійності Вейбула (МНВ)

МНВ має на увазі, що час до відмови є безперервною випадковою величиною, розподіленою за законом Вейбула [26]:

$$Q(t) = 1 - exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right],\ \eta, \beta > 0, \quad (6.11)$$

де β – безрозмірний параметр форми, η – параметр масштабу, який вимірюється в одиницях часу (годинах). Інші показники надійності будуть надаватися наступними залежностями [26]:

$$P(t) = exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right]; \quad (6.12)$$

$$f(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1} exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right]; \quad (6.13)$$

$$\lambda(t) = \frac{f(t)}{P(t)} = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1}; \quad (6.14)$$

$$T_{\text{сер}} = \eta \cdot \Gamma\left(1 + \frac{1}{\beta}\right), \quad (6.15)$$

де $\Gamma(x) = \int_0^{\infty} t^{x-1}e^{-t}dt$ – гамма-функція, що задається зазвичай в табличному вигляді.

Параметр форми β значно впливає на форму функцій щільності та інтенсивності відмов: при $\beta > 1$ функція інтенсивності монотонно зростає, а функція щільності має характерний «горб»; при $\beta < 1$ функція інтенсивності монотонно спадає, також як і функція щільності відмов.

МНВ широко використовується для аналізу і розрахунку надійності технічних систем.

Для розрахунку надійності БНРП було вирішено обрати експоненціальну модель надійності, тому що інтенсивність відмов є постійною.

6.2 Розрахунок надійності по резервуванню каналів зв'язку

При вирішенні завдання теорії надійності представлення блок-схеми надійності є зручним у використанні і відображає вплив відмов елементів на працездатність системи в цілому [26].

У разі використання ЕМН інтенсивності відмов елементів постійні: $\lambda_i(t) = \lambda_i$. Тоді

$$P_i(t) = e^{-\lambda_i t}. \quad (6.16)$$

При цьому загальна інтенсивність відмов для системи для послідовних елементів:

$$\lambda_S(t) = \sum_{i=1}^{N} \lambda_i(t). \quad (6.17)$$

Загальна ймовірність безвідмовної роботи системи для послідовних елементів:

$$P_S = \prod_{i=1}^{N} P_i. \quad (6.18)$$

Таким чином, можна стверджувати, що якщо для всіх елементів послідовної системи справедлива ЕМН, то вона справедлива і для системи в цілому. Іншими словами, таку систему можна представити як новий елемент, для якого справедлива експоненціальна модель надійності з параметром λ_S.

Побудовану блок-схему надійності БНРП показано на рисунку 6.1:

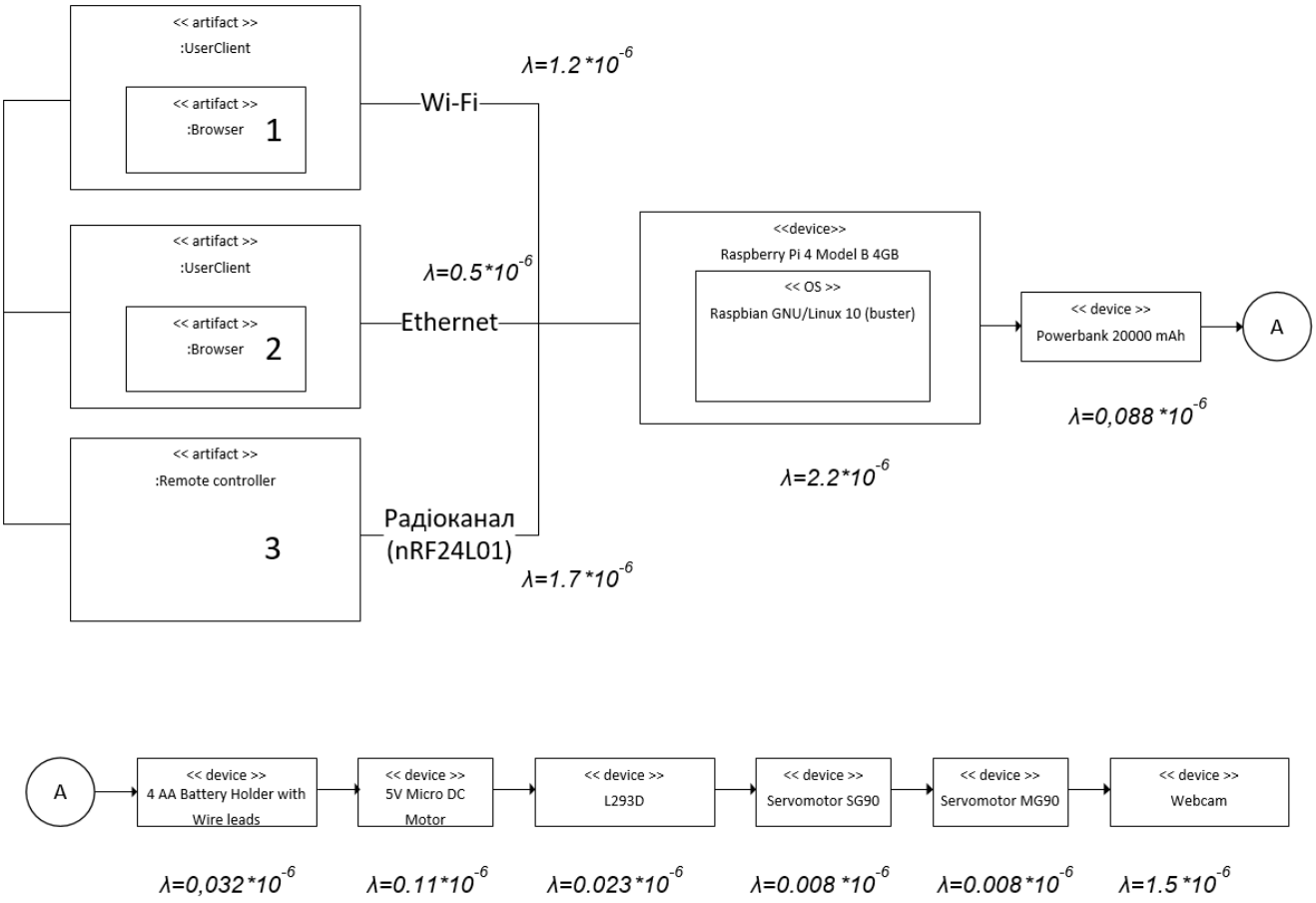

Рисунок 6.1 – Блок-схема надійності рухомої платформи

Інтенсивність відмов є постійною для кожного елемента рухомої платформи і їх можна знайти за спеціалізованими таблицями інтенсивності відмов [27, 28].

Розрахуємо загальну інтенсивність відмов послідовної частини системи:

$$\lambda_{sum} = (2.2+0.088+0.032+0.11+0.023+0.008+0.008+1.5) * 10^{-6} = 3{,}96 * 10^{-6}$$

Припустимо, що система відпрацювала t = 58997 годин.

Тоді ймовірність безвідмовної роботи послідовної частини системи:

$$P_{Послідовні\ ел}(t) = e^{(-\lambda*t)} = e^{(-3.96*(10^{-6})*58997)} \approx 0{,}791$$

Розрахуємо ймовірність безвідмовної роботи каналів зв’язку, які мають паралельне об’єднання:

$$P_{Wi\text{-}Fi}(t) = e^{(-\lambda*t)} = e^{(-1.2*(10^{-6})*58997)} \approx 0{,}932$$

$$P_{Ethernet}(t) = e^{(-\lambda*t)} = e^{(-0.5*(10^{-6})*58997)} \approx 0{,}971$$

$$P_{радіоканал}(t) = e^{(-\lambda*t)} = e^{(-1.7*(10^{-6})*58997)} \approx 0{,}904$$

Для паралельного з’єднання елементів при резервуванні з постійним включенням елементів система зберігає працездатність при справності хоча б одного з елементів. Ймовірність безвідмовної роботи для *n* паралельно з’єднаних елементів (кратність резервування ((n - 1)/1) обчислюється як

$$P(t) = 1 - Q(t) = 1 - \prod_{i=1}^{n}[1 - p_i(t)]$$

Відповідно ймовірність безвідмовної роботи для системи резервування зв’язку:

$$P_{\text{паралельні ел}}(t) = 1 - ((1 - P_{\text{Wi-Fi}}(t))*(1 - P_{\text{Ethernet}}(t))*(1 - P_{\text{радіоканал}}(t))) =$$

$$= 1 - ((1 - 0.932)*(1 - 0.971)*(1 - 0.904)) \approx 0{,}999$$

Виходячи з формули (6.18), розрахуємо ймовірність відмови всієї системи:

$$P_{\text{Вся система}}(t) = P_{\text{Послідовні ел}}(t) * P_{\text{Паралельнії ел}}(t) = 0.791 * 0.999 = 0{,}79$$

6.3 Загальна оцінка надійності

Розрахуємо ймовірність безвідмовної роботи кожного елементу БНРП.

Ймовірність безвідмовної роботи для Raspberry Pi:

$$P_{\text{RaspberryPi}}(t) = e^{(-\lambda*t)} = e^{(-2.2*(10\text{^-}6)*58997)} \approx 0{,}879$$

Ймовірність безвідмовної роботи для PowerBank:

$$P_{\text{PowerBank}}(t) = e^{(-\lambda*t)} = e^{(-0.088*(10\text{^-}6)*58997)} \approx 0{,}995$$

Ймовірність безвідмовної роботи для 4 батарейок типу AA:

$$P_{\text{4AABatteries}}(t) = e^{(-\lambda*t)} = e^{(-0.032*(10\text{^-}6)*58997)} \approx 0{,}998$$

Ймовірність безвідмовної роботи для двигуна постійного струму:

$$P_{\text{DCMotor}}(t) = e^{(-\lambda*t)} = e^{(-0.11*(10\text{^-}6)*58997)} \approx 0{,}993$$

Ймовірність безвідмовної роботи для драйвера двигуна L293D:

$$P_{\text{L293D}}(t) = e^{(-\lambda*t)} = e^{(-0.023*(10\text{^-}6)*58997)} \approx 0{,}999$$

Ймовірність безвідмовної роботи для двох сервомоторів:

$$P_{\text{2ServoMotors}}(t) = e^{(-\lambda*t)} = e^{(-0.016*(10\text{^-}6)*58997)} \approx 0{,}999$$

Ймовірність безвідмовної роботи для веб-камери:

$$P_{\text{WebCam}}(t) = e^{(-\lambda*t)} = e^{(-1.5*(10\text{^-}6)*58997)} \approx 0{,}915$$

Щоб отримати загальну оцінку надійності БНРП необхідно додатково розглянути всі можливі стани працездатності системи. Всього є 9 станів працездатної системи при наявності лінії резервування і 1 стан – непрацездатної системи. Розглянемо ці стани на рисунку 6.2.

Рисунок 6.2 – Стани працездатності системи

З рисунку 6.2 розрахуємо ймовірність безвідмовної роботи працездатної системи за умови, якщо працездатними залишаться такі лінії зв'язку: Wi-Fi (1) і Ethernet (2).

$$P_{1,2}(t) = 1 - ((1 - P_{Wi\text{-}Fi}(t))*(1 - P_{Ethernet}(t)) = 1 - (1\text{-}0.932)*(1\text{-}0.971) = 0.998$$

Відповідно розрахуємо ймовірність безвідмовної роботи працездатної системи з іншими лініями зв'язку Wi-Fi (1) і радіоканал (3).

$$P_{1,3}(t) = 1 - ((1 - P_{Wi\text{-}Fi}(t))*(1 - P_{радіоканал}(t)) = 1 - (1\text{-}0.932)*(1\text{-}0.904) = 0.993$$

Також розрахуємо ймовірність безвідмовної роботи працездатної системи з такими лініями зв'язку Ethernet (2) і радіоканалу (3).

$$P_{2,3}(t) = 1 - ((1 - P_{Ethernet}(t))*(1 - P_{радіоканал}(t)) = 1 - (1\text{-}0.971)*(1\text{-}0.904) = 0.997$$

Розрахуємо загальну оцінку надійності всієї системи за різних умов працездатності ліній зв'язку відповідно.

$$P_{Вся\ система\ 1,2}(t) = P_{Послідовні\ ел}(t) * P_{1,2}(t) = 0.791 * 0.998 = 0{,}789$$

$$P_{Вся\ система\ 1,3}(t) = P_{Послідовні\ ел}(t) * P_{1,3}(t) = 0.791 * 0.993 = 0{,}785$$

$$P_{Вся\ система\ 2,3}(t) = P_{Послідовні\ ел}(t) * P_{2,3}(t) = 0.791 * 0.997 = 0{,}788$$

На загальну оцінку надійності всієї системи впливає показник безвідмовної роботи всіх послідовних елементів.

6.4 Моделювання відмов і відновлень за допомогою марковських процесів

У попередньому розділі було визначено ймовірність безвідмовної роботи системи з використанням Wi-Fi, провідного каналу, радіоканалу на базі nRF24L01. Інтенсивності відмов були отримані з таблиць [27, 28].

Припущення і обмеження. Припустимо, що технічні і надійнісні характеристики радіоканалу та каналу Wi-Fi мають однакові чисельні значення. Під час відмови одного з каналів автоматично підключається резервний. Час підключення значно менший ніж час роботи системи управління. Тобто часом переключення між каналами зв'язку будемо зневажати.

Видалимо із системи управління провідний канал і додамо один резервний радіоканал, отже маємо два радіоканали, що працюють у гарячому резерві і Wi-Fi канал.

Припустимо, що показники надійності інших елементів системи значно кращі надійності каналів зв'язку для управління.

Постановка задачі.

Виходячи з розділу 6.3, інтенсивність відмов λ для радіоканалу має середнє значення $1{,}23 * 10^{-6}$ (1/год) при ідеальних умовах. Цей показник підходить тільки для ідеальних умов, тому приймемо інтенсивність відмов для радіоканалу в залежності від часу безвідмовної роботи радіоканалу. Припустимо, що радіоканал може відмовити через 1 рік = 8760 годин, тоді інтенсивність відмов одного радіоканалу становить $\lambda \approx 1{,}23 * 10^{-4}$ (1/год). Зважаючи з приведених припущень, дана система представляє собою 3 паралельних канали зв'язку. Один з даних каналів є основним інші 2 у гарячому резерві. Таким чином, система може перебувати у чотирьох станах:

1. S_0 – система справна, працездатна має 2 канали зв'язку у гарячому резерві та 1 основний;

2. S_1 – система працездатна, один резервний канал несправний, інші всі – справні, 1 канал – у гарячому резерві, 1– основний;

3. S_2 – система працездатна, два резервних канали несправні, 1 – основний канал;

4. S_3 – система повністю несправна і відновлюється.

З урахуванням припущень отримаємо надійнісні характеристики технічної системи. Постановка задачі передбачає що для її рішення можна використовувати схему гибелі та розмноження [29, 30].

Для розрахунку необхідне значення μ, де μ – це інтенсивність відновлення. Використаємо інше визначення: μ – величина зворотна часу відновлення елемента. Звідси виходить, що:

$$\mu = \frac{1}{\tau},$$

де τ – час відновлення елементу.

Розглянемо безперервний марковський процес з дискретними станами, в якому система може переходити за малий проміжок часу Δt тільки в сусідні два стани, і існують крайні стани. Граф цього процесу представлений на рисунку 6.3.

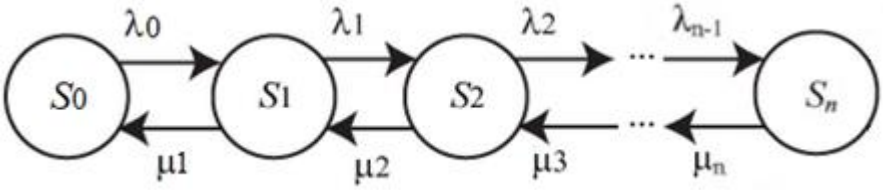

Рисунок 6.3 – Граф станів процесу загибелі і розмноження [29, 30]

Такий випадковий процес має назву процес загибелі і розмноження, і цей термін запозичений з математичної біології, а також використовується в інших областях. Для таких моделей важливо визначити граничні ймовірності станів. Для цього знайдемо стаціонарний розподіл.

Сформуємо матрицю інтенсивностей переходів по графу для $(n+1)$ станів:

$$\Lambda = \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 & 0 & \dots & 0 \\ \mu_1 & -(\lambda_1 + \mu_1) & \lambda_1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & \mu_n & -\mu_n \end{pmatrix}$$

Складемо систему алгебраїчних рівнянь для знаходження стаціонарних ймовірностей $p_0, \dots, p_n$:

$$\begin{gathered}\lambda_0 p_0 = \mu_1 p_1; \\ (\lambda_1 + \mu_1) p_1 = \lambda_0 p_0 + \mu_2 p_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \lambda_{n-1} p_{n-1} = \mu_n p_n\end{gathered} \tag{6.19}$$

За цими алгебраїчними рівняннями виконується умова нормування (6.20)

$$p_0 + p_1 + p_2 + \cdots + p_n = 1 \tag{6.20}$$

Виразимо із цієї системи (6.19) $p_1, \ldots, p_n$ через p_0 і підставимо в умову нормування (6.20) Після нескладних перетворень отримаємо кінцевий результат.

$$p_0 = \left(1 + \frac{\lambda_0}{\mu_1} + \frac{\lambda_0 \lambda_1}{\mu_1 \mu_2} + \cdots + \frac{\lambda_0 \ldots \lambda_{n-1}}{\mu_1 \ldots \mu_n}\right)^{-1}, \tag{6.21}$$

$$p_1 = \frac{\lambda_0}{\mu_1} p_0,\ p_2 = \frac{\lambda_0 \lambda_1}{\mu_1 \mu_2} p_0, \ldots,\ p_n = \frac{\lambda_0 \ldots \lambda_{n-1}}{\mu_1 \ldots \mu_n} p_0. \tag{6.22}$$

Формули (6.21) і (6.22) є розрахунковими формулами для визначення характеристик.

У формулах (6.22) для $p_0, \ldots, p_n$ коефіцієнти при p_0 − це доданки, які стоять після одиниці у формулі (6.21). Чисельники цих коефіцієнтів являють собою добуток інтенсивностей відмов, які стоять біля стрілок зліва направо зі стану S_0 до S_n. А знаменник – це добуток інтенсивностей відновлення, які стоять біля стрілок справа наліво зі стану S_n до S_0.

У нашому випадку система може знаходитись у 4 станах, що показано на рисунку 6.4.

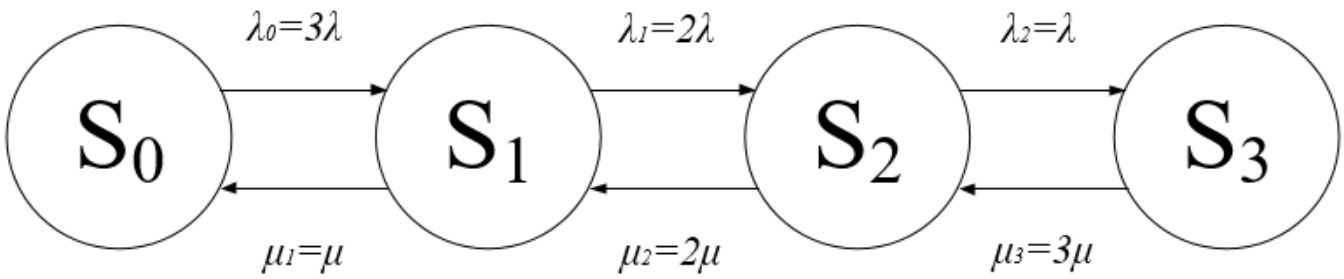

Рисунок 6.4 – Граф станів процесу загибелі і розмноження для 3 (трьох) каналів зв’язку

У стані S_0 працюють всі канали зв'язку, тоді:

$$\lambda_0 = 3\lambda = 3.69 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_1 працюють 2 радіоканали, тоді:

$$\lambda_1 = 2\lambda = 2.46 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_2 працює 1 радіоканал:

$$\lambda_2 = \lambda = 1.23 * 10^{-4} \frac{1}{\text{год}}$$

Відновлення елементу каналу зв'язку може зайняти $\tau = 25$ годин, оскільки даний елемент дуже складно відновити.

$$\mu = \frac{1}{25} = 0.04 \frac{1}{\text{год}}$$

У стані S_1 відновлюється канал зв'язку Wi-Fi, тоді:

$$\mu_1 = \mu = 0.04 \frac{1}{\text{год}}$$

У стані S_2 відновлюються канали зв'язку Wi-Fi і один радіоканал, тоді:

$$\mu_2 = 2\mu = 0.08 \frac{1}{\text{год}}$$

У стані S_3 відновлюються 3 канали зв'язку:

$$\mu_3 = 3\mu = 0.12 \frac{1}{\text{год}}$$

Розрахуємо ймовірність стану при якому всі канали зв'язку справні:

$$p_0 = \left(1 + \frac{3.69 * 10^{-4}}{0.04} + \frac{(3.69 * 2.46) * 10^{-4}}{0.04 * 0.08} + \frac{(3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12}\right)^{-1}$$

$$\approx 0.9909$$

Розрахуємо ймовірність стану при якому канал зв'язку Wi-Fi вийшов з ладу, а 2 (два) радіоканали – справні:

$$p_1 = \frac{3.69 * 10^{-4}}{0.04} * 0.9909 \approx 9{,}14 * 10^{-3}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 1 (один) радіоканал вийшли з ладу, і 1 (один) радіоканал залишився справним.

$$p_2 = \frac{(3.69 * 2.46) * 10^{-4}}{0.04 * 0.08} * 0.9909 \approx 8.64 * 10^{-8}$$

Розрахуємо ймовірність стану при якому всі канали зв’язку є несправними.

$$p_3 = \frac{(3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12} * 0.9909 \approx 0$$

Один канал зв’язку має вартість ≈ 150 грн.

Розрахуємо Sum_3 – вартість всієї системи при N = 3, де N – кількість каналів зв’язку.

$Sum_3 = 1800 + 150 + 20 + 1500 + 400 + 50 + 180 + 150 * 3 = 4550$ грн.

У нашому випадку система може знаходитись у 5 станах, що показано на рисунку 6.5.

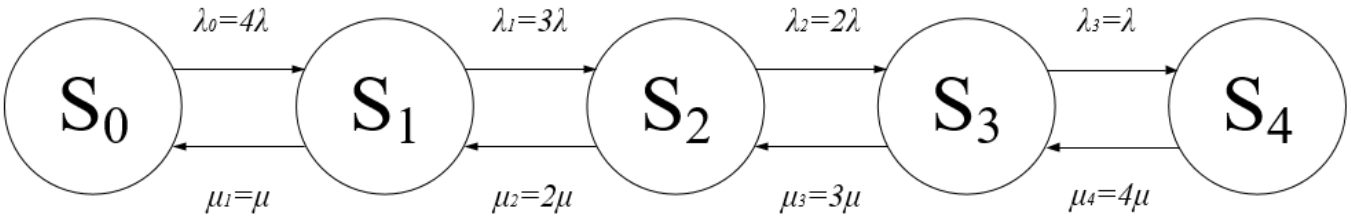

Рисунок 6.5 – Граф станів процесу загибелі і розмноження для 4 (чотирьох) каналів зв’язку

У стані S_0 працюють всі канали зв’язку:

$$\lambda_0 = 4\lambda = 4.92 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_1 працюють 3 (три) радіоканали, тоді:

$$\lambda_1 = 3\lambda = 3.69 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_2 працюють 2 (два) радіоканали, тоді:

$$\lambda_2 = 2\lambda = 2.46 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_3 працює 1 (один) радіоканал:

$$\lambda_3 = \lambda = 1.23 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_1 відновлюється канал зв'язку Wi-Fi, тоді:

$$\mu_1 = \mu = 0.04 \frac{1}{\text{год}}$$

У стані S_2 відновлюються канали зв'язку Wi-Fi і 1 (один) радіоканал, тоді:

$$\mu_2 = 2\mu = 0.08 \frac{1}{\text{год}}$$

У стані S_3 відновлюються канали зв'язку Wi-Fi і 2 (два) радіоканали, тоді:

$$\mu_3 = 3\mu = 0.12 \frac{1}{\text{год}}$$

У стані S_4 відновлюються 4 (чотири) канали зв'язку:

$$\mu_4 = 4\mu = 0.16 \frac{1}{\text{год}}$$

Розрахуємо ймовірність стану при якому всі канали зв'язку справні:

$$p_0 = \left(1 + \frac{4.92 * 10^{-4}}{0.04} + \frac{(4.92 * 3.69) * 10^{-4}}{0.04 * 0.08} + \frac{(4.92 * 3.69 * 2.46) * 10^{-4}}{0.04 * 0.08 * 0.12} + \frac{(4.92 * 3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16}\right)^{-1} \approx 0.9878$$

Розрахуємо ймовірність стану при якому канал зв'язку Wi-Fi вийшов з ладу, а 3 (три) радіоканали – справні:

$$p_1 = \frac{4.92 * 10^{-4}}{0.04} * 0.9878 \approx 1.21 * 10^{-2}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 1 (один) радіоканал вийшли з ладу, і 2 (два) радіоканали залишилися справними.

$$p_2 = \frac{(4.92 * 3.69) * 10^{-4}}{0.04 * 0.08} * 0.9878 \approx 3.447 * 10^{-7}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 2 (два) радіоканали вийшли з ладу, і 1 (один) радіоканал залишився справним.

$$p_3 = \frac{(4.92 * 3.69 * 2.46) * 10^{-4}}{0.04 * 0.08 * 0.12} * 0.9878 \approx 0$$

Розрахуємо ймовірність стану при якому всі канали зв'язку є несправними.

$$p_4 = \frac{(4.92 * 3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16} * 0.9878 \approx 0$$

Один канал зв’язку має вартість ≈ 150 грн.

Розрахуємо Sum_4 – вартість всієї системи при N = 4, де N – кількість каналів зв’язку.

$Sum_4 = 1800 + 150 + 20 + 1500 + 400 + 50 + 180 + 150 * 4 = 4700$ грн.

У нашому випадку система може знаходитись у 6 станах, що показано на рисунку 6.6.

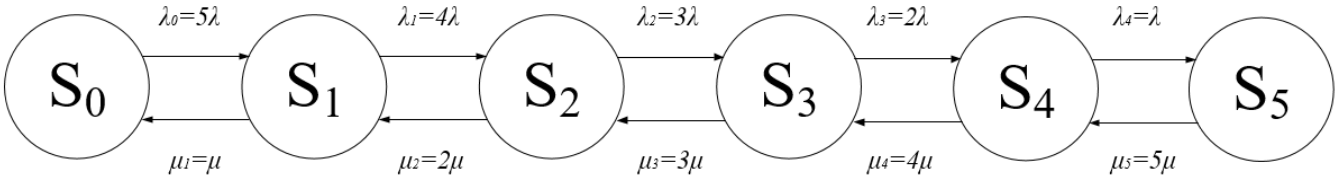

Рисунок 6.6 – Граф станів процесу загибелі і розмноження для 5 (п’яти) каналів зв’язку

У стані S_0 працюють всі канали зв’язку:

$$\lambda_0 = 5\lambda = 6.15 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_1 працюють 4 (чотири) радіоканали:

$$\lambda_1 = 4\lambda = 4.92 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_2 працюють 3 (три) радіоканали, тоді:

$$\lambda_2 = 3\lambda = 3.69 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_3 працюють 2 (два) радіоканали, тоді:

$$\lambda_3 = 2\lambda = 2.46 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_4 працює 1 (один) радіоканал:

$$\lambda_4 = \lambda = 1.23 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_1 відновлюється канал зв’язку Wi-Fi, тоді:

$$\mu_1 = \mu = 0.04 \frac{1}{\text{год}}$$

У стані S_2 відновлюються канали зв’язку Wi-Fi і 1 (один) радіоканал, тоді:

$$\mu_2 = 2\mu = 0.08\frac{1}{\text{год}}$$

У стані S_3 відновлюються канали зв'язку Wi-Fi і 2 радіоканали, тоді:

$$\mu_3 = 3\mu = 0.12\frac{1}{\text{год}}$$

У стані S_4 відновлюються 4 (чотири) канали зв'язку:

$$\mu_4 = 4\mu = 0.16\frac{1}{\text{год}}$$

У стані S_5 відновлюються 5 (п'ять) каналів зв'язку:

$$\mu_5 = 5\mu = 0.2\frac{1}{\text{год}}$$

Розрахуємо ймовірність стану при якому всі канали зв'язку справні:

$$p_0 = \left(1 + \frac{6.15 * 10^{-4}}{0.04} + \frac{(6.15 * 4.92) * 10^{-4}}{0.04 * 0.08} + \frac{(6.15 * 4.92 * 3.69) * 10^{-4}}{0.04 * 0.08 * 0.12} + \frac{(6.15 * 4.92 * 3.69 * 2.46) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16} + \frac{(6.15 * 4.92 * 3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16 * 0.2}\right)^{-1} \approx 0.985$$

Розрахуємо ймовірність стану при якому канал зв'язку Wi-Fi вийшов з ладу, а 4 (чотири) радіоканали – справні:

$$p_1 = \frac{6.15 * 10^{-4}}{0.04} * 0.985 \approx 1.51 * 10^{-2}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 1 (один) радіоканал вийшли з ладу, і 3 (три) радіоканали залишилися справними.

$$p_2 = \frac{(6.15 * 4.92) * 10^{-4}}{0.04 * 0.08} * 0.985 \approx 8.59 * 10^{-7}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 2 (два) радіоканали вийшли з ладу, і 2 (два) радіоканали залишилися справними.

$$p_3 = \frac{(6.15 * 4.92 * 3.69) * 10^{-4}}{0.04 * 0.08 * 0.12} * 0.985 \approx 0$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 2 (два) радіоканали вийшли з ладу, і 1 (один) радіоканал залишився справним.

$$p_4 = \frac{(6.15 * 4.92 * 3.69 * 2.46) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16} * 0.985 \approx 0$$

Розрахуємо ймовірність стану при якому всі канали зв’язку є несправними.

$$p_5 = \frac{(6.15 * 4.92 * 3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16 * 0.2} * 0.985 \approx 0$$

Один канал зв’язку має вартість ≈ 150 грн.

Розрахуємо Sum_5 – вартість всієї системи при N = 5, де N – кількість каналів зв’язку.

$Sum_5 = 1800 + 150 + 20 + 1500 + 400 + 50 + 180 + 150 * 5 = 4850$ грн.

У нашому випадку система може знаходитись у 7 станах, що показано на рисунку 6.7.

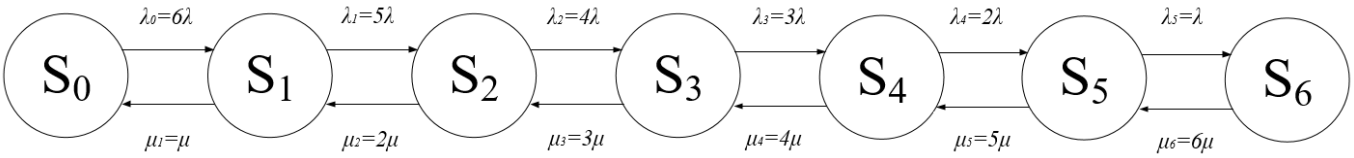

Рисунок 6.7 – Граф станів процесу загибелі і розмноження для 6 (шести) каналів зв’язку

У стані S_0 працюють всі канали зв’язку:

$$\lambda_0 = 6\lambda = 7.38 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_1 працюють 5 (п’ять) радіоканалів:

$$\lambda_0 = 5\lambda = 6.15 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_2 працюють 4 (чотири) радіоканали:

$$\lambda_1 = 4\lambda = 4.92 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_3 працюють 3 (три) радіоканали, тоді:

$$\lambda_2 = 3\lambda = 3.69 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_4 працюють 2 (два) радіоканали, тоді:

$$\lambda_3 = 2\lambda = 2.46 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_5 працює 1 (один) радіоканал:

$$\lambda_4 = \lambda = 1.23 * 10^{-4} \frac{1}{\text{год}}$$

У стані S_1 відновлюється канал зв'язку Wi-Fi, тоді:

$$\mu_1 = \mu = 0.04 \frac{1}{\text{год}}$$

У стані S_2 відновлюються канали зв'язку Wi-Fi і 1 (один) радіоканал, тоді:

$$\mu_2 = 2\mu = 0.08 \frac{1}{\text{год}}$$

У стані S_3 відновлюються канали зв'язку Wi-Fi і 2 (два) радіоканали, тоді:

$$\mu_3 = 3\mu = 0.12 \frac{1}{\text{год}}$$

У стані S_4 відновлюються 4 (чотири) канали зв'язку:

$$\mu_4 = 4\mu = 0.16 \frac{1}{\text{год}}$$

У стані S_5 відновлюються 5 (п'ять) каналів зв'язку:

$$\mu_5 = 5\mu = 0.2 \frac{1}{\text{год}}$$

У стані S_6 відновлюються 6 (шість) каналів зв'язку:

$$\mu_6 = 6\mu = 0.24 \frac{1}{\text{год}}$$

Розрахуємо ймовірність стану при якому всі канали зв'язку справні:

$$p_0 = \left(1 + \frac{7.38 * 10^{-4}}{0.04} + \frac{(7.38 * 6.15) * 10^{-4}}{0.04 * 0.08} + \frac{(7.38 * 6.15 * 4.92) * 10^{-4}}{0.04 * 0.08 * 0.12} + \frac{(7.38 * 6.15 * 4.92 * 3.69) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16} + \frac{(7.38 * 6.15 * 4.92 * 3.69 * 2.46) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16 * 0.2} + \frac{(7.38 * 6.15 * 4.92 * 3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16 * 0.2 * 0.24}\right)^{-1} \approx 0.982$$

Розрахуємо ймовірність стану при якому канал зв'язку Wi-Fi вийшов з ладу, а інші 5 (п'ять) радіоканалів – справні:

$$p_1 = \frac{7.38 * 10^{-4}}{0.04} * 0.982 \approx 1.81 * 10^{-2}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 1 (один) радіоканал вийшли з ладу, а інші 4 (чотири) радіоканали залишилися справними.

$$p_2 = \frac{(7.38 * 6.15) * 10^{-4}}{0.04 * 0.08} * 0.982 \approx 1.71 * 10^{-6}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 2 (два) радіоканали вийшли з ладу, а інші 3 (три) радіоканал залишився справним.

$$p_3 = \frac{(7.38 * 6.15 * 4.92) * 10^{-4}}{0.04 * 0.08 * 0.12} * 0.982 \approx 4.98 * 10^{-14}$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 3 (три) радіоканали вийшли з ладу, а інші 2 (два) радіоканали залишилися справними.

$$p_4 = \frac{(7.38 * 6.15 * 4.92 * 3.69) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16} * 0.982 \approx 0$$

Розрахуємо ймовірність стану при якому канали зв'язку Wi-Fi і 4 (чотири) радіоканали вийшли з ладу, а 1 (один) радіоканал залишився справними.

$$p_5 = \frac{(7.38 * 6.15 * 4.92 * 3.69 * 2.46) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16 * 0.2} * 0.982 \approx 0$$

Розрахуємо ймовірність стану при якому всі канали зв'язку є несправними.

$$p_6 = \frac{(7.38 * 6.15 * 4.92 * 3.69 * 2.46 * 1.23) * 10^{-4}}{0.04 * 0.08 * 0.12 * 0.16 * 0.2 * 0.24} * 0.982 \approx 0$$

Один канал зв'язку має вартість ≈ 150 грн.

Розрахуємо Sum_6 – вартість всієї системи при N = 6, де N – кількість каналів зв'язку.

$Sum_6 = 1800 + 150 + 20 + 1500 + 400 + 50 + 180 + 150 * 6 = 5000$ грн.

6.5 Розрахунок характеристик надійності у системі без відновлення

Розглянемо варіант без відновлення системи з 3 (трьома) каналами зв'язку. Для розрахунку ймовірності чистої загибелі використаємо наступні формули [31]:

$$p_{n-i}(t) = \frac{n!}{(n-i)!}\sum_{k=0}^{i}\frac{e^{-(n-k)\mu t}}{\sum_{l=0}^{i}\prod_{h=0}^{i}\frac{(k-h)}{(k-l)}} \tag{6.23}$$

$$\prod_{h=0}^{i}\frac{(k-h)}{(k-l)} = \prod_{h=0}^{l-1}(k-h)\prod_{m=l+1}^{l}(k-m) \tag{6.24}$$

$$p_0(t) = 1 - \sum_{i=0}^{n-1} p_{n-i}(t) \tag{6.25}$$

$t = 72$ год – приблизний час через, який система може відмовити.

Розрахуємо ймовірність знаходження у стані S_3 при якому всі канали зв'язку є несправними:

$p_3(t) = 1.769 * 10^{-4}$

Розрахуємо ймовірність знаходження у стані S_2 при якому 2 (два) канали зв'язку є несправними:

$p_2(t) = 4.726 * 10^{-3}$

Розрахуємо ймовірність знаходження у стані S_1 при якому 1 (один) канал зв'язку є несправним:

$p_1(t) = 6.736 * 10^{-2}$

Розрахуємо ймовірність знаходження у стані S_0 при якому всі канали зв'язку є справними:

$p_0(t) = 0.9277$

Аналогічні розрахунки виконано для систем з чотирма, п'ятьма, шістьма каналами зв'язку і розміщено у таблиці 6.1.

Результати процесу чистої загибелі, процесу загибелі і розмноження розміщено у відповідних таблицях 6.1 і 6.2.

Таблиця 6.1 – Результати розрахунку ймовірностей відмов без відновлення

	Ймовірність повної відмови	Ймовірності справного і працездатного станів	Вартість резервування
n = 1	$P_1 = 5{,}613476*10^{-2}$	$P_0 = 0{,}944$	0
n = 2	$P_2 = 3{,}151112*10^{-3}$	$P_1 = 5{,}613*10^{-2}$ $P_0 = 0{,}941$	150
n = 3	$P_3 = 1{,}768869*10^{-4}$	$P_2 = 4{,}727*10^{-3}$ $P_1 = 6{,}736*10^{-2}$ $P_0 = 0{,}928$	300
n = 4	$P_4 = 9{,}929504*10^{-6}$	$P_3 = 3{,}538*10^{-4}$ $P_2 = 7{,}563*10^{-3}$ $P_1 = 8{,}42*10^{-2}$ $P_0 = 0{,}908$	450
n = 5	$P_5 = 5{,}573904*10^{-7}$	$P_4 = 2{,}482*10^{-5}$ $P_3 = 7{,}0754*10^{-4}$ $P_2 = 1{,}182*10^{-2}$ $P_1 = 1{,}036*10^{-1}$ $P_0 = 0{,}884$	600
n = 6	$P_6 = 3{,}128898*10^{-8}$	$P_5 = 1{,}672*10^{-6}$ $P_4 = 5{,}958*10^{-5}$ $P_3 = 1{,}327*10^{-3}$ $P_2 = 1{,}745*10^{-2}$ $P_1 = 1{,}24*10^{-1}$ $P_0 = 0{,}857$	750

Таблиця 6.2 – Результати розрахунку ймовірностей відмов з відновленням

	Ймовірність повної відмови	Ймовірності справного і працездатного станів	Вартість резервування
n = 1	$p_1 = 3{,}065573*10^{-3}$	$p_0 = 0{,}997$	0

Продовження таблиці 6.2

	Ймовірність повної відмови	Ймовірності справного і працездатного станів	Вартість резервування
n = 2	$p_2 = 0$	$p_0 = 0,994$ $p_1 = 6,112*10^{-3}$	150
n = 3	$p_3 = 0$	$p_0 = 0,991$ $p_1 = 9,141*10^{-3}$ $p_2 = 8,643*10^{-8}$	300
n = 4	$p_4 = 0$	$p_0 = 0,988$ $p_1 = 1,215*10^{-2}$ $p_2 = 3,447*10^{-7}$ $p_3 = 0$	450
n = 5	$p_5 = 0$	$p_0 = 0,985$ $p_1 = 1,514*10^{-2}$ $p_2 = 8,591*10^{-7}$ $p_3 = 0$ $p_4 = 0$	600
n = 6	$p_6 = 0$	$p_0 = 0,982$ $p_1 = 1,812*10^{-2}$ $p_2 = 1,713*10^{-6}$ $p_3 = 4,981*10^{-14}$ $p_4 = 0$ $p_5 = 0$	750

За результатами наведеними у таблиці 6.1 та 6.2 можна зробити висновок, що достатньо використовувати подвійне резервування каналів зв’язку. Ймовірність справного і працездатного станів за наявністю двох каналів зв’язку є високою, а придбання додаткових каналів зв’язку не є обґрунтованим.

6.6 Висновки до розділу

В даному розділі проведено розрахунок показників надійності каналів зв’язку рухомої платформи. Визначено оптимальну кількість каналів зв’язку для покращення надійності рухомої платформи.

В результаті виконаних розрахунків та їх аналізу отримано висновок, що збільшення кількості каналів зв’язку (більше двох) не є обґрунтованим, оскільки зазначене збільшення каналів зв’язку суттєво не впливає на якість роботи системи управління рухомою платформою, але вартість її значно зростає.

Для розрахунку надійності системи управління рухомою платформою обрано експоненціальну модель надійності, оскільки інтенсивність відмов є постійною.

Крім того, в магістерській дисертації:

– визначено ймовірності безвідмовної роботи для кожної апаратної частини системи управління рухомої платформи;

– проаналізовано 10 станів працездатності системи.

Зазначимо, що на точність вимірювань вплинуло кілька факторів:

– обрана модель надійності;

– інтенсивність відмов об’єкта;

– похибка при вимірюванні ймовірності безвідмовної роботи.

Отже, при моделюванні надійності відмов і відновлення за допомогою марковських процесів було побудовано граф станів процесу загибелі і розмноження системи зв’язку рухомої платформи. Виконано розрахунки ймовірностей у різних станах системи. Розглянуто доцільність резервування каналів зв’язку, якщо їх кількість більше двох. Окремо досліджено варіант без відновлення каналів зв’язку.

7 РОЗРОБКА СТАРТАП-ПРОЕКТУ

7.1 Опис ідеї проекту

Таблиця 7.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	Доставка товарів	Отримання товарів
	Розпізнавання різних об’єктів	Збір додаткової інформації для аналізу

Основними техніко-економічними характеристиками є:

– автоматизована система управління;

– віддалене керування пристроєм;

– доступ до системи онлайн як через ПК так і смартфон;

– можливість зробити фото місцевості;

– перегляд журналу дій;

– застосування у різних галузях.

Є рухомі платформи різного класу серед конкурентів (див. розділ 1). Для визначення сильних, нейтральних та слабких характеристик системи, було здійснено збір та аналіз інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку. Таблиця 7.2 містить порівняння даної системи із конкурентами, в ній стовпчик W – слабка сторона, N – нейтральна сторона, S – сильна сторона.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів				W	N	S
		Мій проект	Amazon Scout	Turtle Rover	DJI Robomaster S1			
1	Автоматизована система управління	Має	Має	Має	Має	-	+	-
2	Віддалене керування пристроєм	Має	Має	Має	Має	-	-	+
3	Доступ до системи онлайн як через ПК так і смартфон	Має	Немає	Має	Немає	-	+	-
4	Можливість зробити фото місцевості	Має	Немає	Має	Має	-	+	-
5	Перегляд журналу дій	Має	Має	Немає	Має	-	+	-
6	Застосування у різних галузях	Має	Немає	Немає	Має	-	-	+

7.2 Розподіл стартап-проекту між умовними учасниками

Для будь-якого стартап-проекту необхідна команда професіоналів, яка відповідає за ту чи іншу сферу діяльності.

У команду даного стартапу входять такі працівники:

1. ІТ-спеціаліст – генератор ідей, є головним у команді стартап-проекту, відповідальний за розробку програмного забезпечення.

2. Спеціаліст відповідальний за апаратне забезпечення – є допоміжним до ІТ-спеціаліста, виконує всі зобов’язання з налагодження і підтримки апаратне забезпечення.

3. Менеджер стартапу – відповідальний за юридичні питання, пошук інвесторів та аналітику.

Тепер необхідно визначити завдання, які будуть виконувати команда стартап-проекту.

Таблиця 7.3 – Кроки виконання завдання стартап-проекту

	Необхідне завдання	Кількість місяців
1	Дослідження методів реалізації	0,5
2	Видача ТЗ	1
3	Розподіл обов’язків	0,5
4	Розробка ПЗ	2
5	Тестування ПЗ	1
6	Розробка конструкції	0,5
7	Оцінка і налагодження	0,5
8	Пошук інвесторів	2
9	Юридичне забезпечення	1,5
10	Рекламна кампанія	
11	Аналітика	

Команда – це в першу чергу постійна співпраця і взаємодія між учасниками проекту. На рис 7.1 показано взаємодію між ключовими учасниками проекту. За вхідними і вихідними зв’язками можна визначити відношення між учасниками проекту. Розрахуємо цю взаємодію:

1 Гі = Вх/Вих = 10/10 = 1

2 С = Вх/Вих = 8/7 = 1,14

3 Д = Вх/Вих = 6/7 = 0,86

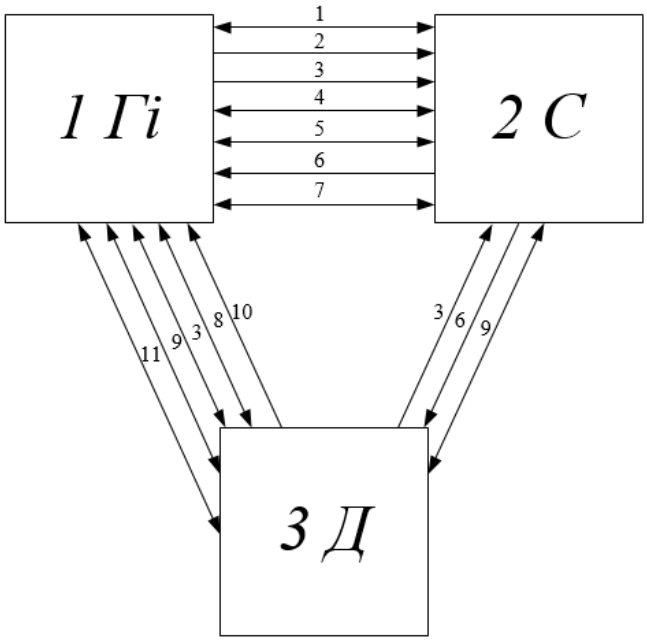

Рисунок 7.1 – Схема взаємодії між учасниками проекту

Кожен учасник стартап-проекту буде виконувати свою частину роботи певний проміжок часу. Побудуємо загальну схему виконаних завдань по відношенню до затраченого часу на виконання (рис 7.2).

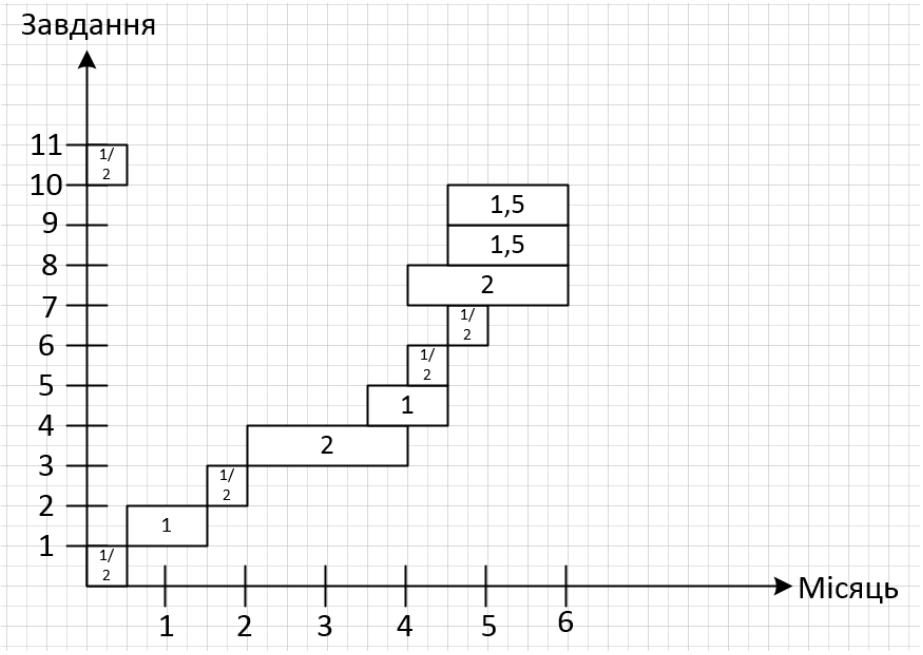

Рисунок 7.2 – Схема виконаних завдань по відношенню до затраченого часу

Розрахуємо скільки часу витратив кожен учасник на виконання свого завдання:

1 Гі = 0,25 + 1 + 0,25 + 1,5 + 0,25 + 0,25 + 0,5 + 0,25 + 0,5 = 4,75 (місяців)

2 С = 0,25 + 0,5 + 0,75 + 0,5 + 0,25 + 0,25 = 2,5 (місяців)

3 Д = 0,25 + 1,5 + 1 + 1,5 + 0,5 + 0,5 + 0,5 = 5,75 (місяців)

Розрахуємо загальний відсоток кожного учасника виконаного завдання від затраченого часу:

1 Гі = 4,75 / 6 = 0,79%

2 С = 2,5 / 6 = 0,4%

3 Д = 5,75 / 6 = 0,96%

Розрахуємо завантаженість кожного учасника стартап-проекту:

1 Гі = 1 / 0,79 = 1,26

2 С = 1,14 / 0,4 = 2,85

3 Д = 0,86 / 0,96 = 0,89

Визначимо основні елементи вкладу у створення стартапу. За Демлером [32] запропоновано такі основні елементи: ідея, підготовка бізнес-плану, компетентність в сфері діяльності компанії, залученість і ризики, обов’язки.

Ідея – компанії не існувало б, якщо не було первісної ідеї.

Підготовка бізнес-плану – опрацювання деталей, оцінка слабких місць проекту, складання стратегії просування.

Компетентність – досвід роботи, зв’язки, розуміння технологій. Ключовим фактором успіху в практично будь-якому бізнесі.

Залученість і ризики – скільки часу учасники стартапу приділяють реалізації проекту.

Обов’язки – розподіл відповідальності між учасниками та виконавцями.

Для кожної компанії співвідношення важливості кожного фактору буде різним: ресторану, наприклад, ідея не така важлива, як грамотне виконання, тобто хороша кухня і налагоджений сервіс. У той час як IT-стартапу ніколи не стати новою величезною компанією без принципово нової технології. Отже, оцінимо важливість кожного фактору для нашої компанії за шкалою від 0 до 10 (таблиця 7.4).

Таблиця 7.4 – Визначення важливості факторів щодо їх вкладу у створення та реалізацію стартапу

Фактор	Вага (важливість)
Ідея	5
Підготовка бізнес плану	8
Компетентність	7
Залученість і ризики	7
Обов’язки	8

Необхідно визначити і оцінити внесок кожного співробітника в загальну справу. Кожен з учасників отримає власну оцінку за різні етапи створення стартапу.

Таблиця 7.5 – Оцінювання особистого внеску кожного партнера у створення та реалізацію стартапу

Фактор	Генератор ідей	Спеціаліст	Дипломат
Ідея	10	3	2
Підготовка бізнес плану	1	1	10
Компетентність	5	7	2
Залученість і ризики	4	4	5
Обов’язки	5	5	7

Об’єднаємо таблиці 7.4 і 7.5

Таблиця 7.6 – Оцінювання важливості кожного фактору і внеску кожного учасника

Фактор	Вага	Генератор ідей	Спеціаліст	Дипломат
Ідея	5	10	3	2
Підготовка бізнес плану	8	1	1	10
Компетентність	7	5	7	2
Залученість і ризики	7	4	4	5
Обов’язки	9	5	5	7

Визначимо ступінь важливості фактору. Складемо показники кожного партнера, помножимо на вагу і визначимо процентне співвідношення.

Таблиця 7.7 – Визначення дольової участі у стартап-проекті кожного учасника

Фактор	Генератор ідей	Спеціаліст	Дипломат	
Ідея	10	3	2	
Підготовка бізнес плану	1	1	10	
Компетентність	5	7	2	
Залученість і ризики	4	4	5	
Обов’язки	5	5	7	
Разом	166	145	202	513
Відсоток	32,36%	28,26%	39,38%	100,0%

7.3 Висновки до розділу

Проект є автоматизованою системою управління рухомою платформою на базі мікрокомп’ютера. Розплановано реалізацію проекту, а також за його підсумками розроблено і затверджено чіткий план реалізації проекту, прив’язаний до календарного плану. Сформовано умовну команду стартап-проекту. Проаналізовано вклад кожного учасника і розподілено прибуток. Визначено слабкі та сильні характеристики проекту.

ВИСНОВКИ

В процесі роботи над магістерською дисертацію виконано розробку апаратної і програмної частини системи управління рухомою платформою на базі .NET Core і Angular. Досягнуто покращення показників надійності рухомої платформи шляхом визначення потрібної кількості каналів зв'язку.

В магістерській дисертації надано огляд та проаналізовано існуючі рішення щодо наземних рухомих платформ. На підставі проведеного аналізу отримані такі особливості, що притаманні сучасним пристроям даного класу. Серед них:

– мікрокомп'ютер, який керує рухомою платформою та обробляє дані, отримані від зовнішніх датчиків та периферії;

– програмний додаток для керування рухомою платформою.

Деякі існуючі рішення мають ряд важливих недоліків:

– висока ціна рухомої платформи;

– перевантаженість зайвими датчиками;

– відомі наземні рухомі платформи мають програмну частину, що є окремим рішенням в більшості випадків. Отже, зазначена програмна частина не може стати прототипом узагальненої системи для розвитку аналогічних рішень.

Для вирішення цих проблем розроблено власну систему управління рухомою платформою, що побудована на базі типових фреймворків і може в подальшому швидко розвиватися і зручно супроводжуватися. Отже, систему управління створено у вигляді веб-додатку, що дає змогу використовувати будь-яку мікропроцесорну систему, яка підтримує браузери, у якості пульта управління.

В магістерській дисертації обґрунтовано програмно-апаратну інфраструктуру наземної платформи:

– представлено діаграму розгортання БНРП (безпілотної наземної рухомої платформи) на основі мікрокомп'ютера;

– розглянуто різні мікрокомп'ютери і в якості основного обрано мікрокомп'ютер Raspberry Pi;

– проведено порівняльний аналіз операційних систем, які можна встановити на Raspberry Pi і прийнято рішення, що операційна система для розробки системи управління, яка підходить для реалізації необхідних задач, є Raspberry Pi Desktop;

– проаналізовано окремі апаратні компоненти рухомої платформи, а саме: двигуна та сервомоторів;

– обґрунтовано вибір фреймворків для серверної та клієнтської частини системи управління рухомої платформи;

– обрано системи зв’язку для управління БНРП.

Розроблено апаратну частину системи управління БНРП, у тому числі, структурну і функціональну схеми. Структурна схема відображає принцип дії рухомої платформи у найзагальнішому вигляді. Функціональну схему створено на основі структурної. Макет БНРП зібрано на основі структурної та функціональної схем.

Для розробки ПСРП (програмної системи рухомої платформи) застосовано багатошарову архітектуру.

В магістерській дисертації для опису ПСРП побудовано декілька видів діаграм:

– діаграму класів, яка ілюструє структуру ПСРП, описує класи, їх атрибути, методи і відносини між об’єктами;

– діаграму компонентів для візуалізації організації ПСРП і залежностей між ними;

– діаграму послідовності для відображення процесу передачі команди «Повернути колесо»;

– ER-діаграму для проектування реляційної бази даних.

Розроблено інтерфейс користувача для управління БНРП.

У результаті проведеного дослідження отримана програмна складова нейронної мережі.

В магістерській дисертації застосовано процес навчання й адаптації нейронної мережі.

Процес підготовки до навчання нейронної мережі містить наступні пункти:

– використання програмних інструментів з репозиторіїв;

– анотування близько 200 зображень вибоїн;

– приведення анотацій зображень (координат).

Для класифікації зображень застосовано метод трансферного навчання нейронної мережі.

Проведено порівняльний аналіз продуктивності процесу навчання нейронної мережі на CPU і GPU за різними критеріями.

Нейронна мережа протестована через ПСРП. За результатами експериментів отримані такі висновки:

– якщо загальна кількість вибоїн на зображенні більше 4, то точність розпізнавання нейронною мережею спадає;

– якщо в процесі проведення експерименту загальна кількість вибоїн зростає, то відсоток розпізнавання зменшується.. Кількість експериментів для розпізнавання зображень залежить від кількості вибоїн на зображенні. Збільшення кількості експериментів покращує результати розпізнавання.

На точність результатів розпізнавання образів також впливає освітленість навколишнього середовища.

В магістерській дисертації проведено розрахунок показників надійності каналів зв’язку рухомої платформи. Визначено оптимальну кількість каналів зв’язку для покращення надійності рухомої платформи.

В результаті виконаних розрахунків та їх аналізу отримано висновок, що збільшення кількості каналів зв’язку (більше двох) не є обґрунтованим, оскільки зазначене збільшення каналів зв’язку суттєво не впливає на якість роботи системи управління рухомою платформою, але вартість її значно зростає.

Для розрахунку надійності системи управління рухомою платформою обрано експоненціальну модель надійності, оскільки інтенсивність відмов є постійною.

Крім того, в магістерській дисертації:

– визначено ймовірності безвідмовної роботи для кожної апаратної частини системи управління рухомої платформи;

– проаналізовано 10 станів працездатності системи.

Зазначимо, що на точність вимірювань вплинуло кілька факторів:

– обрана модель надійності;

– інтенсивність відмов об’єкта;

– похибка при вимірюванні ймовірності безвідмовної роботи.

Отже, при моделюванні надійності відмов і відновлення за допомогою марковських процесів було побудовано граф станів процесу загибелі і розмноження системи зв’язку рухомої платформи. Виконано розрахунки ймовірностей у різних станах системи. Розглянуто доцільність резервування каналів зв’язку, якщо їх кількість більше двох. Окремо досліджено варіант без відновлення каналів зв’язку.

Результатом магістерської дисертації є побудова автоматизованої системи управління рухомою платформою на базі мікрокомп’ютера. Розроблено стартап-проект, зокрема, визначено слабкі та сильні характеристики проекту; розплановано реалізацію проекту, а також за його підсумками розроблено і затверджено чіткий план реалізації проекту, прив’язаний до календарного плану; сформовано умовну команду стартап-проекту; проаналізовано вклад кожного учасника і розподілено прибуток.

СПИСОК ЛІТЕРАТУРИ

1. Scott S. https://www.aboutamazon.com/news/transportation/meet-scout [Електронний ресурс] / Sean Scott. – 2019. – Режим доступу до ресурсу: https://www.aboutamazon.com/news/transportation/meet-scout.

2. Ходаковский К. Роботы-доставщики Amazon Scout появились ещё в двух штатах США [Електронний ресурс] / Константин Ходаковский. – 2020. – Режим доступу до ресурсу: https://3dnews.ru/1016332.

3. Coxworth B. Three-wheeled delivery robot sees increased use in Michigan [Електронний ресурс] / Ben Coxworth. – 2020. – Режим доступу до ресурсу: https://newatlas.com/robotics/rev-1-delivery-robot/.

4. Наземный дрон Turtle Rover – личный дрон для путешественников и исследователей [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://drone-expo.ru/ru/article/nazemniy-dron-turtle-rover-lichniy-dron-dlya-puteshestvennikov-i-issledovateley-73625.

5. Turtle Rover - Outdoor Robotics Development Kit [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.kickstarter.com/projects/1174032727/turtle-rover-worlds-first-earth-rover/description.

6. Бессмертний Р. С. Програмно-апаратна інфраструктура наземного робота для дослідження території [Електронний ресурс] / Роман Сергійович Бессмертний. – 2019. – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/30129/1/Bessmertnyi_bakalavr.pdf.

7. Катин П. Ю. Разработка типовых программных паттернов «состояние» для микроконтроллеров cortex-м в режиме реального времени [Електронний ресурс] / П. Ю. Катин, В. О. Чмелев, В. Н. Шемаев. – 2020. – Режим доступу до ресурсу: http://journals.uran.ua/eejet/article/download/205377/210012.

8. Шишмарев В. Ю. Основы автоматического управления / В. Ю. Шишмарев. – М.: Издательский центр «Академия», 2008. – 352 с.

9. Обзор плат Raspberry Pi [Електронний ресурс] – Режим доступу до ресурсу: https://3d-diy.ru/wiki/arduino-platy/obzor-plat-raspberry-pi/.

10. Raspberry Pi [Електронний ресурс] – Режим доступу до ресурсу: https://www.raspberrypi.org/.

11. Microcomputers. Armbian OS. [Електронний ресурс] – Режим доступу до ресурсу: https://www.armbian.com/download/.

12. Микроконтроллеры Ардуино для чайников [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://arduinoplus.ru/mikrokontrolleri-arduino-dlya-chainikov/.

13. Hasan M. Top 20 Best Raspberry Pi OS Available to Use [Електронний ресурс] / Mehedi Hasan. – 2021. – Режим доступу до ресурсу: https://www.ubuntupit.com/best-raspberry-pi-os-available/.

14. Windows для Интернета вещей [Електронний ресурс] – Режим доступу до ресурсу: https://developer.microsoft.com/ru-ru/windows/iot/.

15. Что такое сервопривод (сервомотор) и как им управлять [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://supereyes.ru/articles/other/chto_takoe_servoprivod_servomotor_i_kak_im_upravlyat/.

16. Электродвигатели постоянного тока. Устройство и Работа. Виды [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://masterpotoku.ru/full/elektrodvigatel-postoannogo-toka-osobennosti-primenenia-i-remonta-110-foto.html.

17. Что такое шаговый двигатель и как им управлять [Електронний ресурс] – Режим доступу до ресурсу: https://supereyes.ru/articles/other/chto_takoe_shagovyy_dvigatel_i_kak_im_upravlyat/.

18. Introduction to ASP.NET Core [Електроний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.0.

19. Sakshi A. 8 Proven Reasons You Need Angular for Your Next Development Project [Електронний ресурс] / Arora Sakshi. – 2018. – Режим доступу до ресурсу: https://www.grazitti.com/blog/8-proven-reasons-you-need-angular-for-your-next-development-project/.Singh S. Onion Architecture In ASP.NET Core [Електронний

ресурс] / Sandeep Singh. – 2020. – Режим доступу до ресурсу: https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/.

20. Програмно-апаратна інфраструктура наземної автономної платформи з елементами штучного інтелекту / С. І. Альперт, М. І. Альперт, П. Ю. Катін, Н. О. Літвінова. // Математичні машини і системи. – 2021. – №1. – С. 24–31.

21. Katin, P. (2017). Development of variant of software architecture implementation for low-power general purpose microcontrollers by finite state machines. EUREKA: Physics and Engineering, 3, 49–54. doi: https://doi.org/10.21303/2461-4262.2017.00361

22. Катін П. Бессмертний Р. / ВИКОРИСТАННЯ ВИСОКОПРОДУКТИВНИХ МІКРОКОНТРОЛЕРІВ ДЛЯ ПІДВИЩЕННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВИРОБНИЦТВА ДЖЕМУ / Стандартизація, сертифікація, якість – №3(115) ,2019. – с.69-76.

23. Liu W. et al. (2016) SSD: Single Shot MultiBox Detector. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2

24. Tran D. Raccoon Detector Dataset [Електронний ресурс] / Dat Tran. – 2018. – Режим доступу до ресурсу: https://github.com/datitran/raccoon_dataset.

25. Tensorflow [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/tensorflow/models/tree/master/research/object_detection/samples/configs.

26. Ефремов А. А. Теория надежности / Александр Александрович Ефремов. – Томск: Издательство Томского политехнического университета, 2015. – 83 с.

27. ГЛАЗАЧЕВ А. Интенсивность отказов элементов справочник [Електронний ресурс] / АЛЕКСЕЙ ГЛАЗАЧЕВ. – 2018. – Режим доступу до ресурсу: https://areliability.com/intensivnost-otkazov-elementov-spravochnik/.

28. Деменкова Т. А., Александров А. И. Аппаратно-программный комплекс мониторинга метеоусловий в системах умного дома // Радиопромышленность. 2018. № 2. С. 80–89.

29. Кошуняева Н. В. Процессы гибели и размножения / Н. В. Кошуняева, Н. Н. Патронова // Теория массового обслуживания / Н. В. Кошуняева, Н. Н. Патронова. – Архангельск, 2013. – (САФУ). – С. 37–40.

30. Схема марковской модели гибели и размножения [Електронний ресурс] – Режим доступу до ресурсу: http://bourabai.kz/cm/dead_n_birth.htm.

31. Вентцель Е. С. Теория случайных процессов и ее инженерные приложения / Е. С. Вентцель, Л. А. Овчаров. – Москва: Высшая школа, 2000. – 383 с.

32. Demmler F. THE FOUNDERS' PIE CALCULATOR [Електронний ресурс] / Frank Demmler – Режим доступу до ресурсу: http://www.andrew.cmu.edu/user/fd0n/35%20Founders%27%20Pie%20Calculator.htm.