Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

ІНТЕЛЕКТУАЛЬНА СИСТЕМА НАВІГАЦІЇ МОБІЛЬНОГО РОБОТА НА БАЗІ IOT ДЛЯ АВТОНОМНОГО МОНІТОРИНГУ СЕРЕДОВИЩА

INTELLIGENT IOT-BASED MOBILE ROBOT NAVIGATION SYSTEM FOR AUTONOMOUS ENVIRONMENTAL MONITORING

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Базилюк Сергій Андрійович

(підпис)

Керівник:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т.ТЕРЛЕЦЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Базилюку Сергію Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Інтелектуальна система навігації мобільного робота на базі IoT для автономного моніторингу середовища*

Керівник роботи *к.т.н., доцент Гринюк Сергій Васильович*

затверджені наказом закладу вищої освіти від «17» червня 2025 року № 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Огляд та аналіз сучасних роботизованих систем моніторингу
Проектування архітектури та розробка математичного забезпечення системи
Експериментальне дослідження системи у середовищі симуляції
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Загальна архітектура взаємодії компонентів у системі мобільного моніторингу
Структурна схема алгоритму локалізації на базі EKF для об'єднання даних одометрії та інерціального модуля
Порівняння ефективності алгоритмів пошуку шляху
Візуалізація роботи алгоритму DWA: генерація пучка можливих траєкторій та вибір оптимальної

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Огляд та аналіз сучасних роботизованих систем моніторингу	*Гринюк С.В., доцент*		
Проектування архітектури та розробка математичного забезпечення системи	*Гринюк С.В., доцент*		
Експериментальне дослідження системи у середовищі симуляції	*Гринюк С.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Гринюк С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., ст.викладач*		

7. Дата видачі завдання 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 01.08.2025 р.	
2.	*Огляд та аналіз сучасних роботизованих систем моніторингу*	До 20.08.2025 р.	
3.	*Проектування архітектури та розробка математичного забезпечення системи*	До 25.09.2025 р.	
4.	*Експериментальне дослідження системи у середовищі симуляції*	До 20.10.2025 р.	
5.	*Висновки та пропозиції*	До 25.10.2025 р	
6.	*Формування списку використаних джерел*	До 27.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 05.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 11.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Базилюк С.А.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Гринюк С.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Базилюк С. А. Інтелектуальна система навігації мобільного робота на базі IoT для автономного моніторингу середовища. Рукопис.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

У першому розділі роботи проведено системний аналіз існуючих аналогів мобільних роботів, який виявив потребу у створенні бюджетної платформи з відкритою архітектурою. Здійснено порівняння методів одночасної навігації та картування (SLAM), за результатами якого обґрунтовано вибір лідарного методу як найбільш ефективного для закритих приміщень.

Другий розділ присвячено проектуванню апаратних та програмних засобів системи. Розроблено структурно-функціональну схему та принципову електричну схему робота, що включає підсистеми живлення, керування приводами та сенсорами. Описано математичні моделі кінематики диференціального приводу та імовірнісної локалізації. Реалізовано програмне забезпечення в середовищі ROS 2.

У третьому розділі наведено результати експериментальних досліджень системи. Виконано порівняльний аналіз точності алгоритмів SLAM, досліджено ефективність планування маршруту в умовах статичних та динамічних перешкод, а також проведено навантажувальне тестування обчислювальної системи. Підтверджено здатність розробленого комплексу забезпечувати стабільну навігацію та передачу даних телеметрії в реальному часі.

Ключові слова: мобільний робот, slam, ROS2, IoT, mqtt, Gazebo, навігація, моніторинг, Raspberry PI, Lidar.

ANNOTATION

Bazyliuk S. Intelligent navigation system for a mobile robot based on IoT for autonomous environmental monitoring. Manuscript.

Master's thesis. Educational Program «Computer Engineering», Specialty 123 «Computer Engineering». Lutsk National Technical University. Lutsk, 2025.

The thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter presents a systematic analysis of existing mobile robot analogs, which revealed the need to create a low-cost platform with an open architecture. A comparison of Simultaneous Localization and Mapping (SLAM) methods was carried out, based on the results of which the choice of the Lidar-based method was substantiated as the most effective for indoor environments.

The second chapter is devoted to the design of the system's hardware and software. The structural-functional diagram and the schematic circuit diagram of the robot were developed, including subsystems for power supply, actuator control, and sensors. Mathematical models of differential drive kinematics and probabilistic localization are described. The software implementation was performed in the ROS 2 environment.

The third chapter presents the results of experimental research on the system. A comparative analysis of the accuracy of SLAM algorithms was performed, path planning efficiency under conditions of static and dynamic obstacles was investigated, and load testing of the computing system was conducted. The ability of the developed complex to ensure stable navigation and real-time telemetry data transmission was confirmed.

Keywords: mobile robot, SLAM, ROS 2, IoT, MQTT, Gazebo, navigation, monitoring, Raspberry Pi, Lidar.

ЗМІСТ

ВСТУП

Сучасний етап розвитку промисловості та урбаністики, що проходить під егідою концепцій «Індустрія 4.0» та «Smart City», висуває нові вимоги до систем контролю безпеки життєдіяльності. Традиційні стаціонарні системи екологічного моніторингу, незважаючи на свою поширеність, мають суттєвий недолік – локальність вимірювань, що залишає значні «сліпі зони» у великих виробничих приміщеннях або складних офісних будівлях. В умовах потенційних витоків шкідливих газів, змін температурного режиму або необхідності інспекції аварійних ділянок, використання людини-оператора пов’язане з невиправданим ризиком для здоров’я.

Вирішенням цієї проблеми є інтеграція мобільної робототехніки з технологіями Інтернету речей (IoT). Автономні мобільні роботи, оснащені мультисенсорними системами та засобами навігації, здатні динамічно змінювати точки контролю, створюючи детальні карти забруднення середовища. Однак, існуючі промислові рішення часто є надмірно коштовними та мають закриту архітектуру, що ускладнює їх адаптацію під специфічні задачі. У зв’язку з цим, розробка бюджетної мобільної платформи з відкритою архітектурою (Open Source), здатної до автономної навігації в умовах невизначеності та надійної передачі телеметрії в хмару, є актуальним науково-технічним завданням.

Метою роботи є підвищення ефективності та безпеки процесу моніторингу навколишнього середовища у закритих приміщеннях шляхом розробки мобільної роботизованої IoT-системи, здатної до автономного картування, навігації та передачі даних у реальному часі.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) провести аналіз існуючих методів навігації (SLAM) та апаратних платформ для мобільної робототехніки;

2) обґрунтувати вибір апаратного забезпечення (Raspberry Pi, лідар, сенсори) та розробити структурно-функціональну схему системи;

3) розробити математичні моделі руху робота та алгоритми пошуку оптимального маршруту з урахуванням динамічних перешкод;

4) спроектувати програмну архітектуру системи в середовищі ROS 2 та реалізувати шлюз для IoT-взаємодії через протокол MQTT;

5) створити віртуальний полігон у середовищі Gazebo та провести експериментальне дослідження ефективності розроблених алгоритмів та стабільності передачі даних.

Об’єктом дослідження є процес автоматизованого моніторингу параметрів навколишнього середовища з використанням рухомих носіїв вимірювальної апаратури.

Предметом дослідження є методи та алгоритми автономної навігації (SLAM, A*, DWA) та архітектура інформаційної взаємодії в роботизованій IoT-системі.

Наукова новизна одержаних результатів полягає в удосконаленні методу автономної навігації мобільного робота шляхом інтеграції алгоритму SLAM Toolbox з адаптивним налаштуванням локального планувальника DWA, що дозволило підвищити точність проходження траєкторії в динамічному середовищі. Крім того, набула подальшого розвитку архітектура «Edge-Fog-Cloud» для мобільних систем моніторингу, яка, на відміну від існуючих, реалізує пріоритетну буферизацію критичних даних на борту робота (Edge), що забезпечує цілісність екологічної карти навіть за умов нестабільного зв’язку.

Практичне значення одержаних результатів. Розроблено та протестовано у віртуальному середовищі прототип мобільної системи, який може бути використаний для інспекції складських приміщень, серверних кімнат та лабораторій. Запропоноване програмне забезпечення на базі ROS 2 та Docker дозволяє швидко розгортати систему на доступних одноплатних комп’ютерах серії Raspberry Pi.

Апробація результатів роботи. Результати роботи представлені на 2 - й Міжнародній науково-практичній конференції «Сучасні виклики в наукових дослідженнях», яка проводилась з 1 по 3 грудня 2025 року [1].

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ РОБОТИЗОВАНИХ СИСТЕМ МОНІТОРИНГУ

1.1 Огляд існуючих мобільних роботів для моніторингу середовища

Сучасний розвиток систем моніторингу навколишнього середовища нерозривно пов’язаний з інтеграцією мобільної робототехніки та технологій Інтернету речей (IoT). Використання мобільних платформ дозволяє динамічно змінювати точки вимірювання параметрів мікроклімату, рівня загазованості або радіаційного фону, що є значною перевагою над стаціонарними сенсорними мережами. Аналіз ринку та наукових джерел за 2021-2025 роки дозволяє виділити три основні групи роботизованих систем, що застосовуються для подібних задач: промислові рішення високої прохідності, дослідницькі платформи з відкритим кодом та комерційні побутові роботи.

У сегменті промислових рішень безумовним лідером технологічних інновацій є компанія Boston Dynamics з платформою Spot (рис. 1.1). Цей чотириногий робот демонструє виняткову мобільність на неструктурованих поверхнях (сходи, каміння, завали), що робить його незамінним для моніторингу небезпечних зон.

Рисунок 1.1 – Робот Boston Dynamics Spot з модулем інспекції [2]

Згідно з технічною документацією [2], Spot підтримує підключення спеціалізованих модулів (Spot CAM, газоаналізатори) через порт даних, що дозволяє виконувати автономні інспекції. Однак вартість платформи (понад $70,000) та закритість пропрієтарного ПЗ обмежують її використання у масових системах моніторингу.

Альтернативою у класі колісних платформ, що активно використовуються для автономного моніторингу та інспекції, є роботи серії Clearpath Robotics, зокрема моделі Husky та Jackal. Ці платформи (рис. 1.2), які вирізняються своєю міцністю та надійністю, мають високий клас захисту IP65 (захист від пилу та струменів води), що робить їх ідеальними для роботи у складних погодних та промислових умовах, включаючи відкриті майданчики на нафтогазових об’єктах чи гірничодобувних підприємствах. Завдяки своїй конструкції (повнопривідне шасі з великими колесами), вони забезпечують відмінну прохідність на пересіченій місцевості. Найважливішою перевагою є їхня нативна підтримка Robot Operating System (ROS), яка є галузевим стандартом і значно спрощує інтеграцію різноманітних мультисенсорних систем (лідари, камери, тепловізори, газоаналізатори), необхідних для комплексного моніторингу обладнання та інфраструктури [3]. Це дозволяє дослідникам та інженерам швидко розробляти та розгортати складні алгоритми навігації, SLAM та аналізу даних.

Рисунок 1.2 – Clearpath Husky UGV [3]

Для наукових розробок прототипів IoT-систем найчастіше використовуються платформи TurtleBot 4 та NVIDIA JetBot.

TurtleBot 4 (рис. 1.3), побудований на базі мікроконтролера Raspberry Pi 4 та контролера iRobot Create 3, є стандартом для тестування алгоритмів SLAM (Simultaneous Localization and Mapping) завдяки повній підтримці ROS 2. Як зазначається у дослідженні [4], відкрита архітектура дозволяє легко додавати сенсори якості повітря (наприклад, серії MQ або BME280), проте базове шасі має обмежену прохідність (тільки рівні поверхні).

Рисунок 1.3 – Мобільний робот Clearpath Robotics TurtleBot 4 [4]

Платформа JetBot (рис. 1.4) базується на обчислювальному модулі NVIDIA Jetson Nano і орієнтована на використання нейромереж для візуальної навігації. Це дозволяє реалізувати сценарії розпізнавання перешкод та джерел небезпеки за допомогою камери, що є перевагою для автономних систем [5].

Рисунок 1.4 – NVIDIA JetBot з камерою для AI-навігації

Окрему нішу займають побутові роботи-пилососи (iRobot Roomba, Roborock, Xiaomi). Сучасні моделі оснащені лідарами (LiDAR) та камерами для побудови високоточних карт приміщень (рис. 1.5). Їх головними перевагами є низька вартість та наявність зарядної станції з автопаркуванням. Проте, як зазначено в огляді [6], виробники рідко надають API для доступу до даних сенсорів у реальному часі, що ускладнює їх інтеграцію в кастомну систему екологічного моніторингу без злому пристрою.

Рисунок 1.5 – Робот Пилосос Xiaomi Roborock S7 MaxV [6]

Для обґрунтування вибору апаратної бази було проведено порівняльний аналіз розглянутих аналогів за критеріями вартості, автономності та можливості інтеграції в IoT (табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз платформ мобільних роботів

Характеристика	Boston Dynamics Spot	Clearpath Husky	TurtleBot 4	NVIDIA JetBot	Побутові (Roborock)
Кінематика	Крокуюча (4 ноги)	Колісна (4x4)	Диференційна (2 колеса)	Диференційна	Диференційна
Вартість	> $74 000	> $18 000	~$ 1 800	< $200	$300 – $600
Програмна база	Proprietary API	ROS / ROS 2	ROS 2 (Ubuntu)	JetPack (Python)	Закрита (Cloud)
Навігація	3D SLAM Stereo Vision	GPS, LiDAR	2D SLAM, Odom	Vision-based (CNN)	LiDAR SLAM

Продовження таблиці 1.1

Характеристика	Boston Dynamics Spot	Clearpath Husky	TurtleBot 4	NVIDIA JetBot	Побутові (Roborock)
IoT-інтеграція	Складна (Enterprise)	Середня	Висока (Open Source)	Висока	Низька (потрібен Hack)
Час роботи	90 хв	3-8 год	2-4 год	1-2 год	2-3 год

Аналіз існуючих рішень показує, що промислові роботи є надмірно дорогими для масового впровадження, а побутові пристрої мають закриту архітектуру. Дослідницькі платформи типу TurtleBot є оптимальними з точки зору програмного забезпечення, але їх вартість залишається високою для бюджетних проектів. Тому доцільною є розробка власної мобільної платформи на базі доступних мікроконтролерів (ESP32/Raspberry Pi) з відкритою архітектурою для гнучкої інтеграції сенсорів моніторингу.

1.2 Аналіз методів навігації та картування в умовах невизначеності

Основною проблемою розробки автономних мобільних роботів для моніторингу середовища є забезпечення точного позиціювання та побудови карти в умовах відсутності глобальних навігаційних супутникових систем (GNSS/GPS). Основним підходом до вирішення цього завдання є використання технології SLAM (Simultaneous Localization and Mapping), яка дозволяє роботу одночасно будувати карту невідомого простору та визначати своє положення на ній. В умовах невизначеності, спричиненої динамічними перешкодами та шумами сенсорів, вибір конкретного методу SLAM є критичним етапом проектування.

На сьогодні домінуючими підходами є методи, засновані на лідарах (Lidar-based SLAM) та методи візуальної одометрії (Visual SLAM або vSLAM). Лідарні методи, такі як Gmapping або Google Cartographer, базуються на використанні лазерних далекомірів, що генерують точні двовимірні або тривимірні хмари точок. Як зазначено в огляді [7], головною перевагою лідарного підходу є висока

точність вимірювання відстані (похибка менше 1-2 см) та інваріантність до умов освітлення, що дозволяє роботу ефективно функціонувати навіть у повній темряві. Алгоритми типу Cartographer використовують оптимізацію графа поз (Graph SLAM), що дозволяє ефективно компенсувати помилки одометрії шляхом замикання циклів.

Для візуалізації результату роботи алгоритму лідарного картування на рисунку 1.6 наведено приклад карти зайнятості. У даному представленні простір дискретизується на клітинки, де чорні пікселі відповідають виявленим фізичним перешкодам (стіни, меблі), білі зони вказують на вільний простір, доступний для безпечної навігації робота, а сірі області позначають ще недосліджену територію.

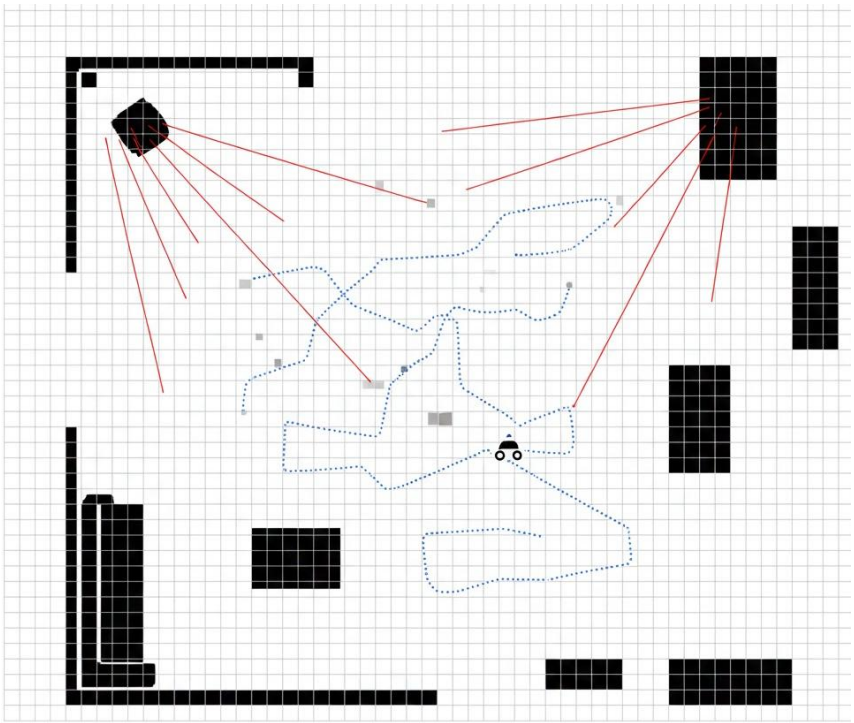


Рисунок 1.6 – Приклад побудови карти зайнятості методом лідарного SLAM, де чорним позначено перешкоди, а білим – вільний простір [7]

Натомість, візуальний SLAM (vSLAM) використовує відеокамери (монокулярні, стерео або RGB-D) для вилучення характерних ознак із зображення. Сучасні алгоритми, зокрема ORB-SLAM3, демонструють здатність

будувати розріджені або щільні карти середовища, використовуючи значно дешевше апаратне забезпечення порівняно з лідарами. Проте, згідно з дослідження [8] vSLAM потребує значних обчислювальних ресурсів (часто необхідна наявність GPU) для обробки відеопотоку в реальному часі та є вразливим до змін освітлення, відблисків та текстурної бідності поверхонь (наприклад, однотонні стіни).

На відміну від метричних карт лідарів, алгоритми візуальної навігації оперують розрідженими хмарами точок. На рисунку 1.7 продемонстровано принцип виділення характерних ознак, де алгоритм ідентифікує висококонтрастні ключові точки (кути об’єктів, межі текстур) на відеопотоці та на основі їх зміщення між кадрами реконструює тривимірну траєкторію руху камери у просторі.

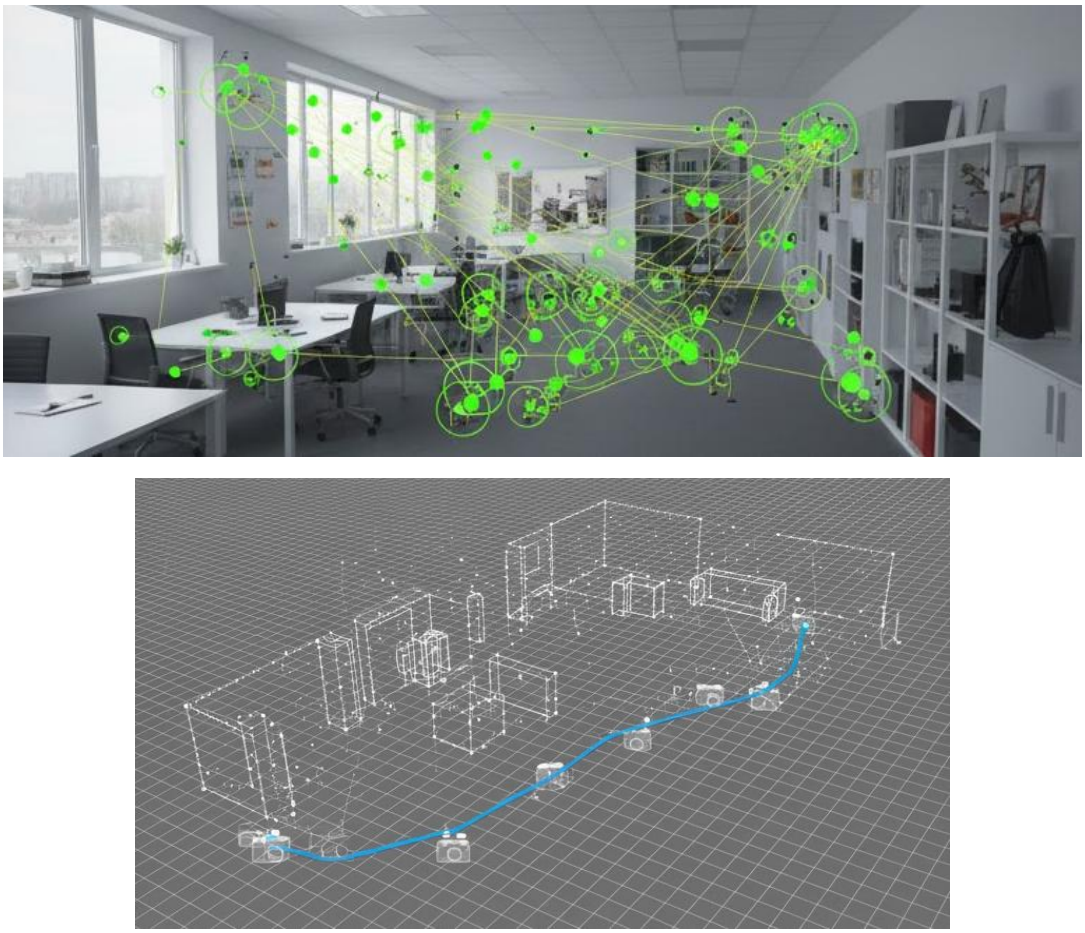


Рисунок 1.7 – Принцип роботи візуального SLAM: виділення ключових точок та реконструкція траєкторії руху камери [8]

Для систематизації відмінностей між методами було проведено порівняльний аналіз, результати якого наведено в таблиці 1.2. Видно, що для задач моніторингу, де критичною є надійність навігації, лідарні методи мають перевагу, тоді як vSLAM доцільний у випадках обмеженого бюджету або необхідності семантичного розпізнавання об’єктів.

Таблиця 1.2 – Порівняльний аналіз методів SLAM для мобільних платформ

Критерій порівняння	Lidar-based SLAM (на прикладі Cartographer)	Visual SLAM (на прикладі ORB-SLAM3)
Тип сенсора	Лазерний далекомір (LiDAR)	Камера (Mono/Stereo/RGB-D)
Точність картування	Висока (до 1-2 см)	Середня (залежить від текстур)
Вимоги до обчислень	Помірні (можлива робота на CPU)	Високі (бажано GPU/NPU)
Залежність від світла	Повна незалежність	Критична залежність
Вразливі зони	Скляні поверхні, дзеркала	Однотонні стіни, темрява
Вартість сенсора	Висока ($50-$2000+)	Низька ($20-$200)

Навігація в закритих приміщеннях характеризується повною відсутністю сигналу супутникової навігації, що змушує систему покладатися виключно на бортові датчики. В умовах невизначеності, спричиненої ковзанням коліс (drift) та накопиченням похибки інерціальних вимірювальних модулів (IMU), використання лише одного джерела даних є недостатнім. Сучасним стандартом, описаним у роботі [9], є комплексування даних, найчастіше на базі розширеного фільтра Калмана (EKF).

Для наочної ілюстрації архітектури навігаційної системи на рисунку 1.8 наведено структурну схему комплексування даних. Вона демонструє потік інформації від низькорівневих датчиків – енкодерів, що фіксують обертання коліс та інерціального модуля (IMU), який вимірює кутові прискорення. Ці дані об’єднуються у вузлі EKF для безперервного обчислення положення, тоді як дані лідара використовуються для періодичної корекції накопиченої похибки відносно глобальної карти, забезпечуючи стабільну локалізацію робота навіть при пробуксовці коліс [10].

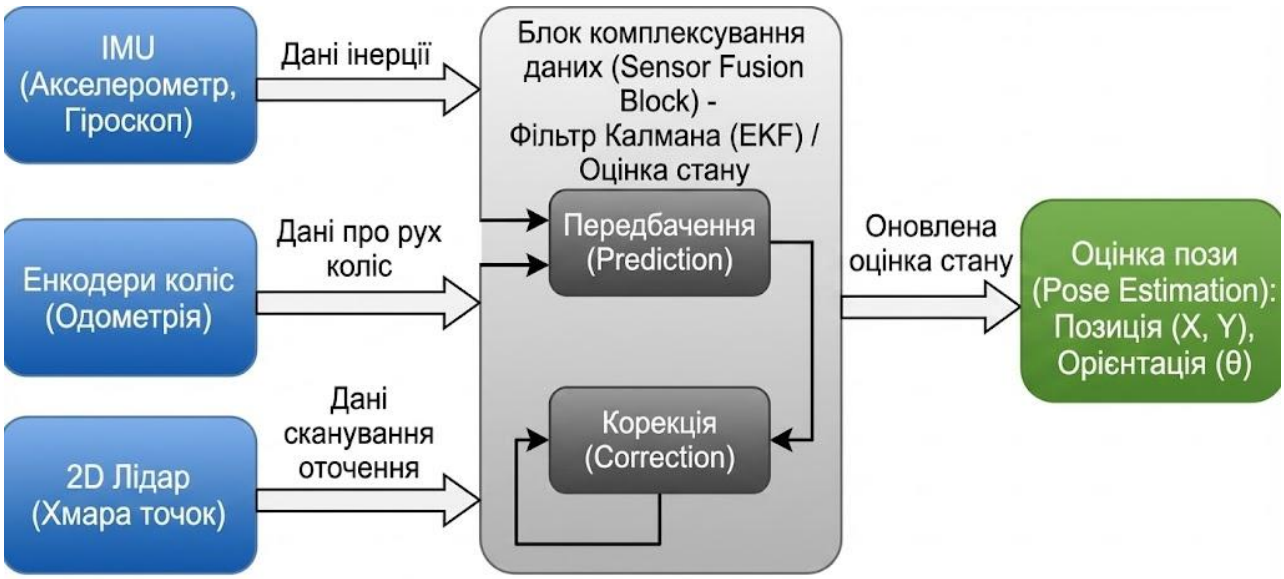


Рисунок 1.8 – Структурна схема комплексування даних сенсорів (IMU, енкодери, лідар) для локалізації робота в приміщенні [10]

Підсумовуючи аналіз, можна стверджувати, що для розроблюваної системи моніторингу найбільш доцільним є використання 2D-лідарного SLAM у поєднанні з даними одометрії та IMU. Цей підхід забезпечує оптимальний баланс між точністю навігації та обчислювальним навантаженням на мікроконтролер, що є критичним для забезпечення автономності мобільної платформи.

1.3 Аналіз апаратно-програмних засобів для реалізації системи

Ефективність функціонування мобільного робота для моніторингу середовища визначається раціональним вибором обчислювального ядра, здатного обробляти потоки даних від сенсорів та виконувати ресурсоємні алгоритми навігації (SLAM) у реальному часі. Сучасна архітектура робототехнічних систем зазвичай будується за дворівневим принципом: низькорівневі операції (керування двигунами, зчитування енкодерів) виконуються мікроконтролерами (наприклад, STM32 або ESP32), тоді як високорівневі задачі покладаються на одноплатні комп’ютери (Single Board Computers – SBC).

У сегменті компактних обчислювальних систем для робототехніки на сьогодні домінують три платформи: Raspberry Pi, NVIDIA Jetson та Orange Pi. Згідно з порівняльним дослідженням продуктивності граничних обчислень (Edge Computing) [11], платформа Raspberry Pi 4/5 є найбільш збалансованим рішенням з точки зору співвідношення «продуктивність-енергоефективність-ціна». Завдяки чотириядерному процесору ARM Cortex-A72/A76 та широкій підтримці спільноти, ця плата стала де-факто стандартом для розгортання операційної системи ROS.

На рисунку 1.9 представлено зовнішній вигляд одноплатного комп’ютера Raspberry Pi 4 Model B

Рисунок 1.9 – Зовнішній вигляд одноплатного комп’ютера Raspberry Pi 4 [11]

Платформа NVIDIA Jetson Nano (та її наступники серії Orin) орієнтована на задачі штучного інтелекту. Наявність потужного графічного процесора (GPU) архітектури Maxwell або Ampere дозволяє виконувати паралельні обчислення для нейромереж комп’ютерного зору. Однак, як зазначається у роботі [12], для задач, що обмежуються 2D-навігацією та збором телеметрії без складної відеоаналітики, енергоспоживання Jetson (5-10 Вт) може бути надлишковим порівняно з аналогами.

Бюджетною альтернативою є сімейство Orange Pi (зокрема Orange Pi 5). Ці плати часто пропонують вищу частоту процесора за нижчу ціну порівняно з

Raspberry Pi. Проте, їх суттєвим недоліком є менш стабільна програмна підтримка драйверів для специфічної периферії та нестандартні бібліотеки GPIO, що може ускладнити інтеграцію в екосистему ROS 2.

Для візуального порівняння архітектур на рисунку 1.10 представлено зовнішній вигляд одноплатного комп'ютера NVIDIA Jetson Nano.

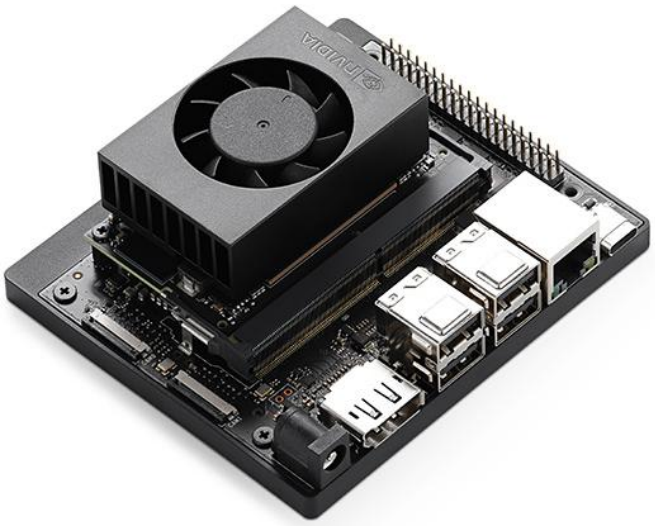


Рисунок 1.10 – Зовнішній вигляд одноплатного комп'ютерів NVIDIA Jetson Nano [12]

Важливо відзначити відмінності у форм-факторах та інтерфейсах охолодження: NVIDIA Jetson оснащена масивним радіатором через високе тепловиділення GPU, тоді як Raspberry Pi та Orange Pi мають більш компактну компоновку, придатну для інтеграції в обмежений простір корпусу мобільного робота.

Результати порівняльного аналізу технічних характеристик та економічної доцільності використання розглянутих платформ у проекті зведено у таблицю 1.3.

Таблиця 1.3 – Порівняльна характеристика одноплатних комп'ютерів

Характеристика	Raspberry Pi 4 Model B	NVIDIA Jetson Nano	Orange Pi 5
Процесор (CPU)	4-core Cortex-A72 @ 1.5GHz	4-core Cortex-A57 @ 1.43GHz	8-core RK3588S @ 2.4GHz
Графічний прискорювач (GPU)	VideoCore VI (Базовий)	128-core Maxwell (AI-оптимізований)	Mali-G610 (Середній рівень)

Продовження таблиці 1.3

Характеристика	Raspberry Pi 4 Model B	NVIDIA Jetson Nano	Orange Pi 5
ОЗП (RAM)	2/4/8 GB LPDDR4	4 GB LPDDR4	4/8/16 GB LPDDR4
Підтримка ROS 2	Відмінна (Tier 1 Support)	Добра (потребує Docker)	Середня (складності з OS)
Середнє енергоспоживання	3,4-7,6 Вт	5-10 Вт	4-8 Вт
Орієнтовна вартість	Середня (~$55-75)	Висока (~$150+)	Низька (~$60-80)

Для забезпечення ефективної взаємодії обчислювального ядра з гетерогенним набором периферійних пристроїв у розроблюваній системі застосовується диференційований підхід до вибору комунікаційних інтерфейсів, що базується на вимогах до швидкості передачі даних та топології мережі. Найбільш поширеним стандартом для комутації датчиків моніторингу навколишнього середовища обрано послідовну шину I2C (Inter-Integrated Circuit). Її архітектурною особливістю є використання лише двох сигнальних ліній (SDA та SCL) для обміну інформацією, що дозволяє організувати локальну мережу з десятків різнотипних сенсорів, таких як вимірювачі температури, вологості (наприклад, серії BME) або газоаналізатори, підключаючи їх паралельно до одного контролера з використанням унікальної адресації кожного вузла.

У завданнях, де критичним параметром є пропускна здатність каналу та мінімізація затримок, доцільно використовувати інтерфейс SPI (Serial Peripheral Interface). Завдяки чотирипровідній схемі підключення та підтримці повнодуплексного режиму роботи, цей протокол забезпечує високошвидкісний обмін даними, що є необхідним для зчитування показників з інерціальних вимірювальних модулів (акселерометрів, гіроскопів) або керування графічними дисплеями. Водночас для інтеграції складних автономних модулів, таких як GPS-приймачі, лазерні сканери (LiDAR) або бездротові модеми стандартів LoRa та ZigBee, застосовується універсальний асинхронний інтерфейс UART (Universal Asynchronous Receiver-Transmitter). Цей стандарт реалізує надійний зв’язок за схемою «точка-точка» без загальної лінії тактування, що є стандартом де-факто для промислових навігаційних компонентів.

Наочна ілюстрація фізичної реалізації описаних протоколів у проектованій системі наведена на рисунку 1.11.

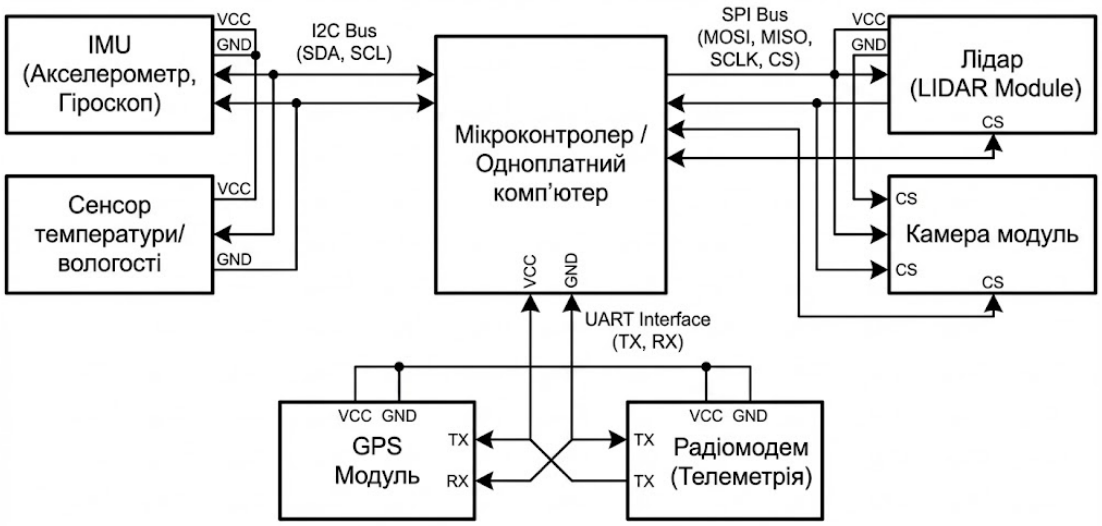


Рисунок 1.11 – Схема підключення сенсорів моніторингу та навігації з використанням інтерфейсів I2C, SPI та UART [13]

Схема демонструє архітектурне розмежування потоків даних: низькошвидкісні сенсори екологічного моніторингу консолідовані на спільній шині I2C для спрощення розведення друкованої плати, тоді як критичні для навігації прилади, зокрема лідар, підключені через індивідуальні лінії UART, що гарантує стабільність отримання телеметрії та запобігає колізіям даних при високому навантаженні на центральний процесор.

Інтеграція апаратних компонентів у єдину інтелектуальну систему здійснюється за допомогою Robot Operating System (ROS). На сьогодні актуальною версією є ROS 2 (дистрибутиви Humble або Jazzy), яка базується на стандарті DDS (Data Distribution Service) для обміну повідомленнями у реальному часі. Згідно з [13], ROS надає універсальну архітектуру, де програмні модулі представлені як окремі «Вузли» (Nodes), що обмінюються даними через «Теми» (Topics) за принципом «Видавець-Підписник» (Publisher-Subscriber).

Така модульність дозволяє абстрагуватися від конкретного апаратного забезпечення: заміна типу лідара або драйвера двигунів вимагає лише зміни

відповідного вузла-драйвера без необхідності переписування алгоритмів навігації. Як показано в роботі [14], використання екосистеми ROS значно прискорює розробку завдяки наявності готових пакетів для SLAM (Nav2), візуалізації (Rviz2) та симуляції (Gazebo).

Для ілюстрації програмної архітектури на рисунку 1.12 наведено граф обчислень ROS (Computation Graph). На схемі показано взаємодію основних вузлів системи: вузол лідара (/lidar_node) публікує «сирі» дані сканування (/scan), які підписує вузол SLAM (/slam_toolbox) для генерації карти (/map). Паралельно вузол моніторингу публікує дані сенсорів, які доступні для віддаленого dashboard-клієнта.

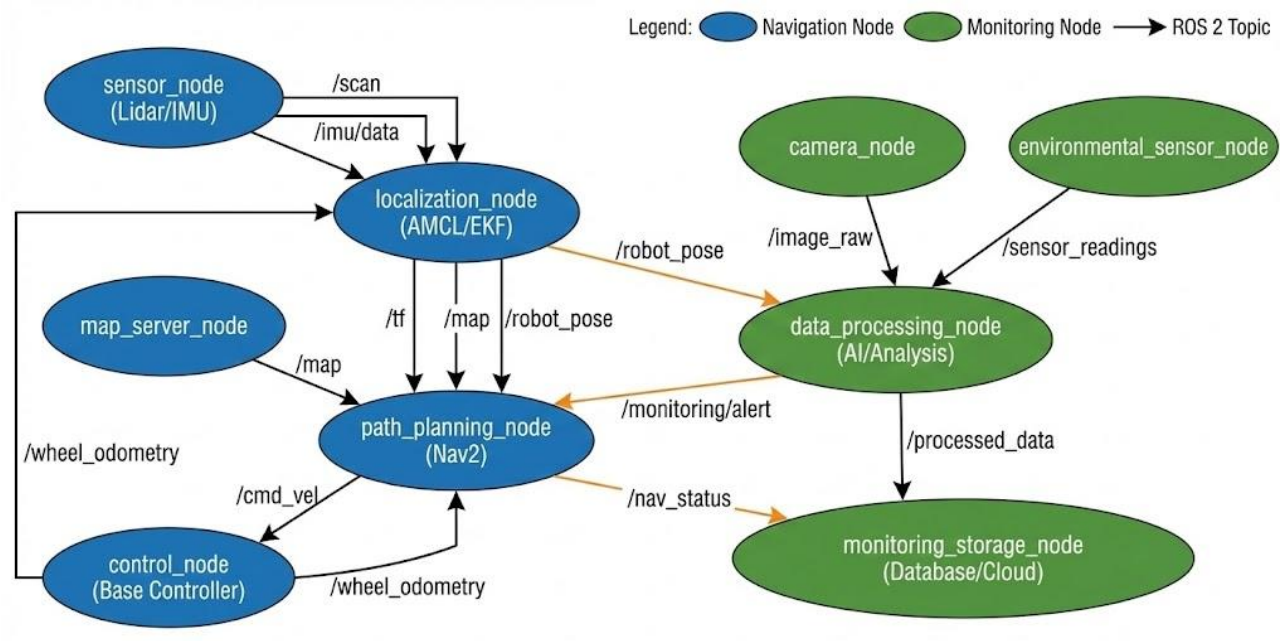


Рисунок 1.12 – Архітектура графа обчислень ROS 2: взаємодія вузлів навігації та моніторингу через топіки [14]

В даному розділі було проведено комплексний аналіз сучасного стану розвитку та теоретичних засад побудови мобільних робототехнічних систем для моніторингу навколишнього середовища. За результатами огляду існуючих аналогів встановлено, що наявні на ринку промислові платформи характеризуються надмірно високою вартістю, що обмежує їх масове впровадження, тоді як доступні побутові роботи мають закриту програмну

архітектуру, що унеможливлює інтеграцію спеціалізованих сенсорів. Це обґрунтовує доцільність розробки власної бюджетної мобільної платформи з відкритою архітектурою, яка забезпечить гнучкість у налаштуванні апаратних модулів та інтеграцію в системи Інтернету речей.

У ході порівняльного аналізу методів навігації в умовах невизначеності виявлено переваги лідарних методів (Lidar-based SLAM) над візуальними для задач внутрішнього моніторингу, насамперед завдяки їхній високій точності та незалежності від умов освітлення. Визначено, що для забезпечення стабільної локалізації робота у закритих приміщеннях без доступу до GPS критично важливим є застосування методів комплексування даних (Sensor Fusion), які об’єднують інформацію від лазерних сканерів, одометрії коліс та інерціальних вимірювальних модулів.

На основі аналізу апаратно-програмних засобів здійснено вибір компонентної бази проектованої системи. Як обчислювальне ядро обрано одноплатний комп’ютер Raspberry Pi, який забезпечує необхідну обчислювальну потужність для роботи алгоритмів SLAM у реальному часі та підтримує операційну систему ROS 2. Також визначено оптимальні комунікаційні протоколи для взаємодії з периферією: шину I2C для збору даних моніторингу та інтерфейс UART для підключення навігаційних приладів.

РОЗДІЛ 2
ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА РОЗРОБКА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Розробка структурно-функціональної організації роботизованої IoT-системи

Проектування архітектури мобільної системи екологічного моніторингу базується на ієрархічній моделі взаємодії компонентів, яка забезпечує розподіл обчислювального навантаження між локальними ресурсами робота та віддаленими сервісами. Згідно з сучасними підходами до побудови Інтернету роботизованих речей (IoRT), загальна структура системи реалізується за багаторівневою схемою «Edge-Fog-Cloud», що дозволяє мінімізувати затримки при керуванні рухом та забезпечити глобальний доступ до накопиченої телеметрії [15].

На рисунку 2.1 представлена розроблена загальна архітектура взаємодії, де центральним елементом комунікації виступає брокер повідомлень. Така організація дозволяє розв’язати підсистему генерації даних (робот) та підсистему споживання даних (клієнтський додаток), забезпечуючи асинхронний обмін інформацією.

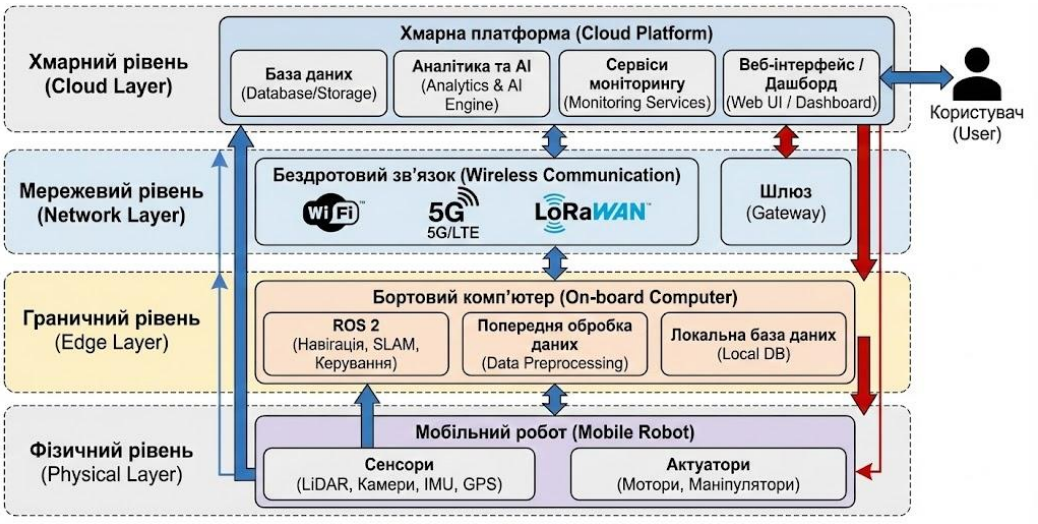


Рисунок 2.1 – Загальна архітектура взаємодії компонентів у системі мобільного моніторингу [15]

На нижньому рівні знаходиться сам мобільний робот, який виступає як інтелектуальний кінцевий вузол мережі. Його завдання полягає у зборі первинних даних із сенсорів, попередній обробці сигналів та виконанні критичних до часу задач навігації в реальному часі. Дані передаються через бездротовий канал зв'язку (Wi-Fi) до проміжного рівня або безпосередньо до хмарного середовища.

Важливим етапом проектування є деталізація внутрішніх інформаційних потоків (Data Flow) між підсистемами робота, оскільки ефективність моніторингу залежить від коректної синхронізації вимірювань параметрів середовища з просторовими координатами. У розроблюваній системі виділено два основні контури циркуляції даних: навігаційний контур та контур корисного навантаження. Навігаційний потік включає високочастотну передачу даних від лідара та енкодерів до модуля SLAM для обчислення позиції (x, y) та орієнтації θ. Паралельно функціонує потік моніторингу, який опитує датчики мікроклімату та якості повітря.

Специфікою розробленої схеми, яка зображена на рисунку 2.2, є наявність програмного вузла агрегації даних.

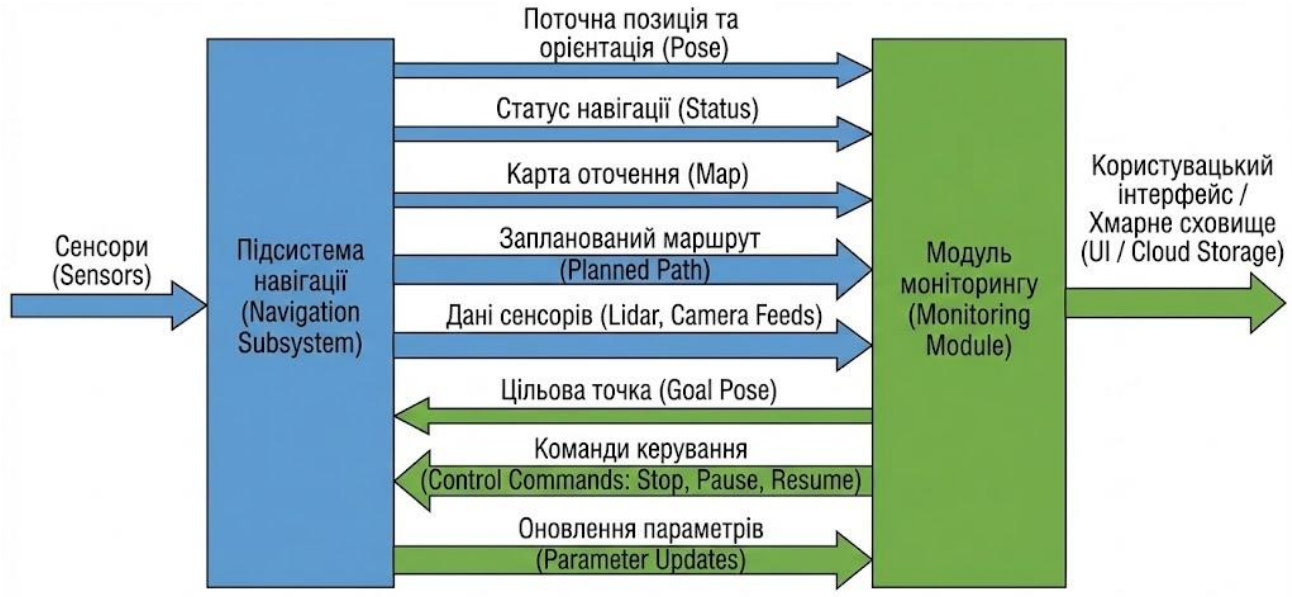


Рисунок 2.2 – Схема інформаційних потоків між підсистемою навігації та модулем моніторингу [16]

Даний модуль об’єднує поточні координати, отримані від навігаційної системи, з миттєвими значеннями сенсорів у єдиний пакет даних формату JSON. Згідно з роботою [16], такий підхід (Geo-tagging) дозволяє автоматично формувати геоприв’язані карти забруднення (heat maps) на стороні сервера без необхідності складної пост-обробки, оскільки кожен запис у базі даних вже містить прив’язку до локації вимірювання.

Для забезпечення цілісності даних при передачі на сервер (рівень Cloud) використовується протокол транспортного рівня, що підтримує QoS (Quality of Service). Це гарантує, що навіть у зонах з нестабільним покриттям мережі пакети з критичними даними про небезпечні рівні забруднення будуть доставлені до бази даних або буферизовані на борту робота до відновлення з’єднання. Клієнтська частина (рівень Client) реалізується у вигляді веб-інтерфейсу, який надсилає запити до хмарного API для візуалізації траєкторії руху робота та динаміки зміни параметрів середовища [16].

2.2 Математичне моделювання руху та навігації робота

Основою для побудови алгоритмів керування мобільною платформою є математичний опис її руху, який пов’язує керуючі впливи (швидкості обертання коліс) зі зміною положення робота у глобальній системі координат. Для розроблюваної системи обрано диференціальну схему приводу, яка є стандартом для роботів класу TurtleBot завдяки своїй механічній простоті та високій маневреності, зокрема здатності до розвороту на місці.

Кінематична модель описує рух робота без урахування сил та моментів, що діють на нього, фокусуючись на геометричних залежностях. Розглянемо робота як тверде тіло, положення якого у двовимірному просторі OXY описується вектором стану $ $ q = [x, y, \theta]^T $, де (x, y) – координати центру мас, а θ – кут орієнтації. Вхідними параметрами системи є лінійна швидкість v та кутова швидкість ω робота. Пряма задача кінематики полягає у визначенні цих швидкостей на основі виміряних кутових швидкостей правого (ω_R) та лівого ω_L

коліс. Враховуючи радіус колеса r та відстань між колесами (базу) L, рівняння зв’язку мають вигляд (2.1):

$$v = \frac{r}{2}(\omega_R + \omega_L),\ \omega = \frac{r}{L}(w_R - w_L). \tag{2.1}$$

На основі цих співвідношень можна записати диференціальні рівняння руху робота у глобальній системі координат, які використовуються для обчислення одометрії (2.2):

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{2.2}$$

де v – лінійна швидкість:

ω – кутова швидкість.

Для візуалізації описаних геометричних залежностей на рисунку 2.3 наведено розрахункову схему кінематики диференціального робота.

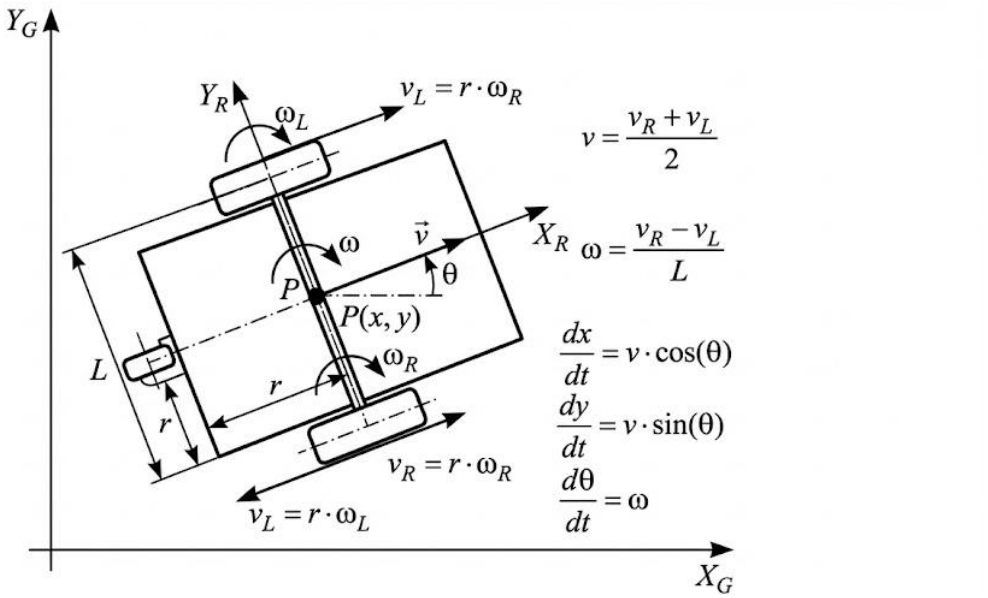


Рисунок 2.3 – Розрахункова схема кінематики мобільного робота з диференціальним приводом [17]

Схема демонструє взаємозв'язок між локальною системою координат, прив'язаною до центру робота, та глобальною картою, а також ілюструє поняття миттєвого центру обертання (ICC), навколо якого здійснюється поворот платформи при різних швидкостях коліс.

Для реалізації автоматичного керування критично важливою є обернена задача кінематики: визначення необхідних кутових швидкостей коліс для досягнення заданих навігаційним контролером значень лінійної та кутової швидкості. Це перетворення виконується на рівні драйверів двигунів за формулою (2.3):

$$\omega_R = \frac{2v+\omega L}{2r}, \omega_L = \frac{2v-\omega L}{2r}. \quad (2.3)$$

Як зазначається в роботі [17], точність цієї моделі суттєво залежить від калібрування параметрів r та L, оскільки навіть незначні відхилення призводять до систематичної помилки одометрії, яка зростає з часом.

В реальних умовах експлуатації використання виключно кінематичної моделі (колісної одометрії) є недостатнім через наявність стохастичних збурень: проковзування коліс на гладких поверхнях, нерівності підлоги та дискретність енкодерів. Для компенсації цих похибок застосовується метод комплексування даних (Sensor Fusion) на основі розширеного фільтра Калмана (Extended Kalman Filter – EKF).

Суть методу полягає в рекурсивному уточненні оцінки вектора стану системи шляхом об'єднання прогнозних даних від кінематичної моделі та коригувальних вимірювань від інерціального модуля (IMU). Алгоритм EKF працює у два етапи: передбачення (Prediction) та корекція (Update). На етапі передбачення обчислюється апріорна оцінка стану $\hat{x}_k^-$ та матриця коваріації помилок $\hat{P}_k^-$ на основі рівнянь руху (2.4):

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k), \quad (2.4)$$

де u_k – вектор керування (швидкості коліс).

На етапі корекції використовується вимірювання кутової швидкості $\dot{\theta}_{gyro}$ та лінійного прискорення a_{cc} від IMU для розрахунку коефіцієнта підсилення Калмана K_k та оновлення оцінки стану. Згідно з дослідженням [18], інтеграція IMU дозволяє значно зменшити дрейф кута курсу θ, який є найбільш критичним параметром для точності побудови карти.

На рисунку 2.4 представлено структурну схему розробленого вузла локалізації. Вона показує потік даних від сенсорів до ядра EKF: дані енкодерів перетворюються на одометричні повідомлення (швидкість та зміщення), а дані IMU проходять попередню фільтрацію шумів. На виході фільтр генерує трансфомацію odom – base_link, яка забезпечує плавну та точну траєкторію руху робота навіть при короткочасній втраті зчеплення коліс.

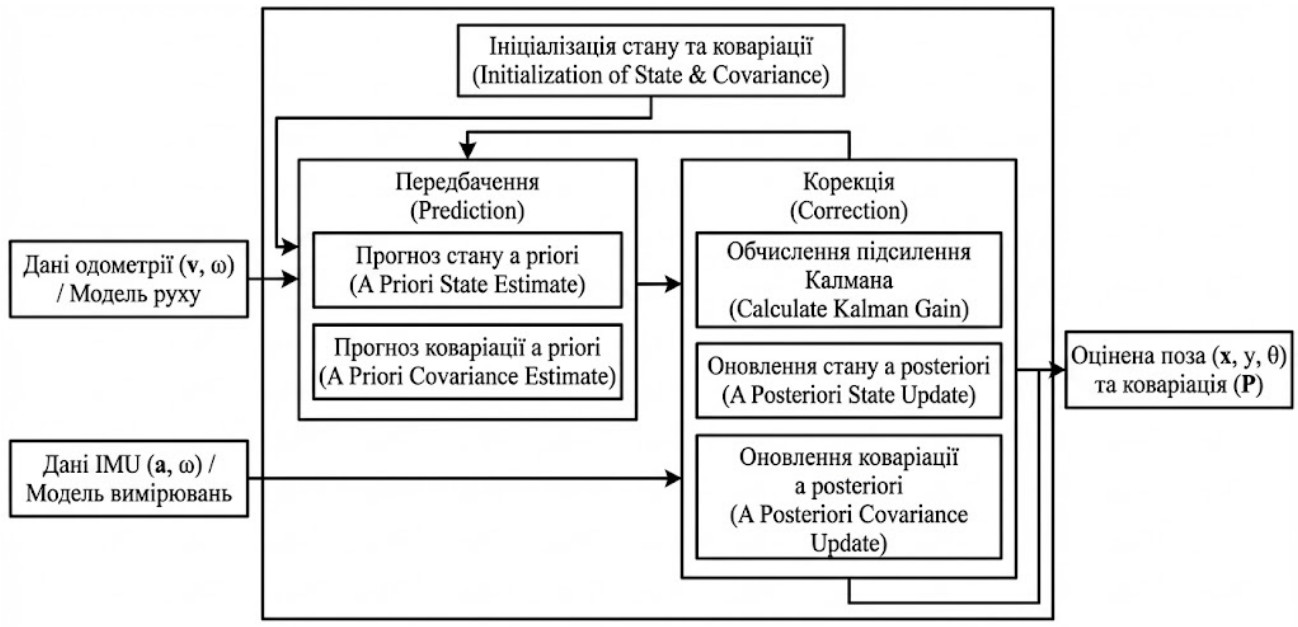


Рисунок 2.4 – Структурна схема алгоритму локалізації на базі EKF для об’єднання даних одометрії та інерціального модуля [18]

Таким чином, комбінація детермінованої кінематичної моделі та імовірнісного підходу EKF створює надійний фундамент для роботи високорівневих алгоритмів SLAM, забезпечуючи точність позиціювання,

необхідну для коректного накладання даних екологічного моніторингу на карту приміщення.

2.3 Алгоритмічне забезпечення пошуку шляху та обходу перешкод

Ефективне переміщення мобільного робота у відомому або частково відомому середовищі забезпечується дворівневою системою планування руху, яка включає глобальний планувальник (Global Planner) та локальний планувальник (Local Planner). Глобальний рівень відповідає за побудову оптимального маршруту на статичній карті місцевості від поточної позиції до цільової точки, тоді як локальний рівень генерує безпосередні керуючі команди для приводів робота, забезпечуючи слідування маршруту та динамічне уникнення непередбачуваних перешкод.

Задача глобального планування формулюється як пошук шляху на зваженому графі, де вузлами є клітинки карти зайнятості (Occupancy Grid Map), а ребрами – можливі переходи між сусідніми вільними клітинками. Кожній клітинці присвоюється вартість проходження, яка залежить від наявності перешкод та відстані до них (так звана «вартість інфляції»). Серед методів пошуку на графах найбільш поширеними є алгоритм Дейкстри (Dijkstra) та алгоритм A* (A-Star).

Алгоритм A* є евристичною модифікацією методу Дейкстри і базується на мінімізації функції повної вартості шляху $f(n)$ для кожного вузла n:

$$f(n) = g(n) + h(n), \tag{2.5}$$

де $g(n)$ – реальна вартість шляху від стартової точки до поточного вузла n (сума ваг пройдених ребер);

$h(n)$ – евристична оцінка вартості шляху від n до цільової точки.

Для сіткових карт у робототехніці як евристичну функцію зазвичай використовують Манхеттенську відстань або Евклідову відстань. Головна відмінність між методами полягає в тому, що алгоритм Дейкстра гарантовано знаходить найкоротший шлях, перевіряючи всі можливі напрямки рівномірно (що відповідає умові $h(n)$), тоді як A* використовує евристику для спрямованого пошуку в бік цілі.

Для візуалізації різниці в ефективності алгоритмів на рисунку 2.5 наведено порівняння зон пошуку. На схемі (а) видно, що алгоритм Дейкстра досліджує значну кількість зайвих вузлів у всіх напрямках, формуючи «хвильовий фронт», тоді як алгоритм A* (б) фокусує обчислювальні ресурси на перспективних вузлах, значно скорочуючи час розрахунку маршруту, що є критичним для великих карт навігації [19].

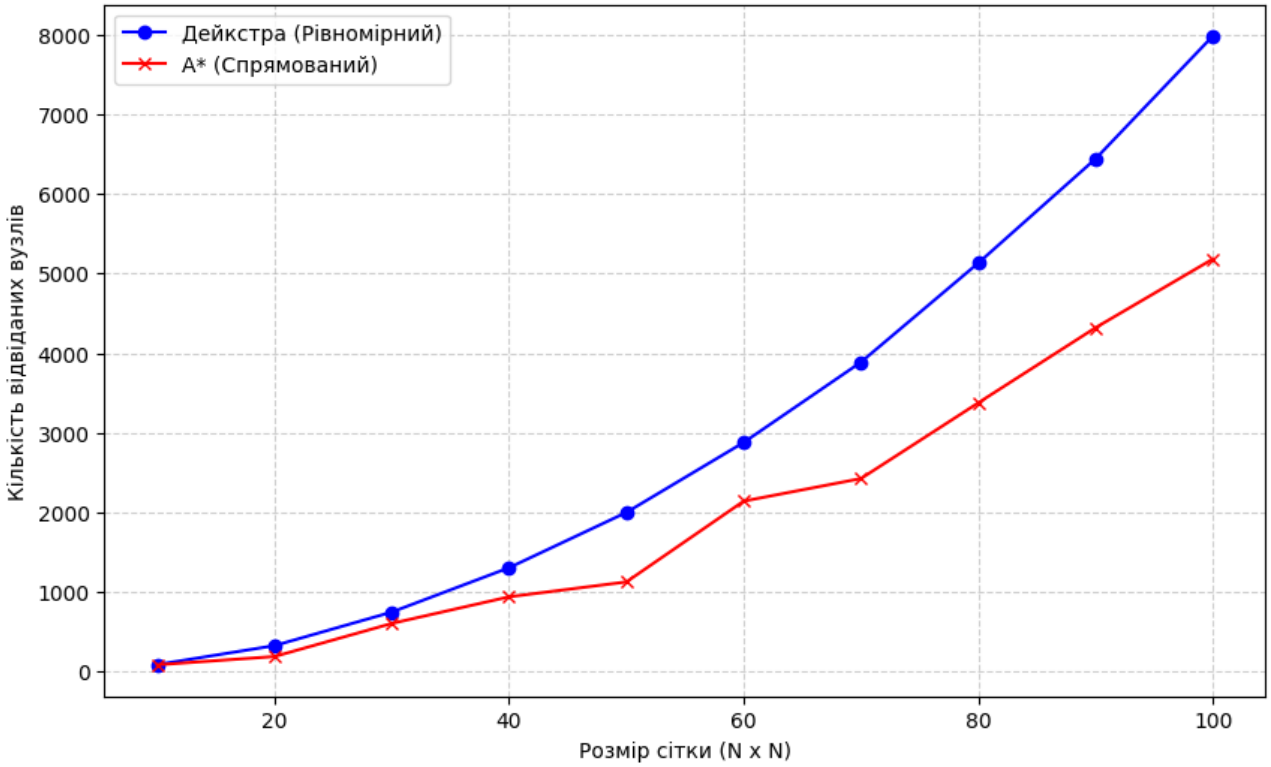


Рисунок 2.5 – Порівняння ефективності алгоритмів пошуку Дейкстра та алгоритму A* [19]

У розроблюваній системі для глобального планування обрано алгоритм A*, оскільки він забезпечує оптимальний баланс між швидкістю роботи та

оптимальністю знайденого шляху. Функція вартості переміщення (Cost Function) формується на основі глобальної карти витрат (Global Costmap), де зонам навколо перешкод присвоюються високі значення штрафів, що змушує планувальник прокладати маршрут на безпечній відстані від стін та об’єктів.

Після отримання глобального маршруту вступає в дію локальний планувальник, завданням якого є генерація кінематично допустимих швидкостей (v, ω) для руху робота. На цьому етапі розглядаються два основні підходи: метод динамічного вікна (Dynamic Window Approach – DWA) та метод еластичної стрічки з часовою параметризацією (Timed Elastic Band – TEB).

Алгоритм DWA працює у просторі швидкостей. На кожному кроці керування він симулює безліч можливих траєкторій, які робот може виконати за короткий проміжок часу, враховуючи обмеження прискорення. Для кожної траєкторії обчислюється цільова функція $G(v, \omega)$, яка максимізується (2.6):

$$G(v, \omega) = \alpha \cdot heading(v, \omega) + \beta \cdot dist(v, \omega) + \gamma \cdot velocity(v, \omega), \qquad (2.6)$$

де $heading$ оцінює відхилення від цілі;

$dist$ – відстань до найближчої перешкоди;

$velocity$ – рух з високою швидкістю.

Суть алгоритму полягає у послідовному формуванні динамічного вікна (Dynamic Window) – обмеженої підмножини можливих векторів швидкостей, які можуть бути досягнуті роботом протягом наступного короткого часового інтервалу. Це вікно обмежується двома ключовими факторами: кінематичними обмеженнями робота (максимальне прискорення та швидкість) та обмеженнями перешкод (необхідність безпечної зупинки до зіткнення). У межах цього вікна алгоритм генерує пучок прогнозованих траєкторій, кожна з яких оцінюється за допомогою комплексної функції вартості (мінімізація відстані до перешкод, наближення до цілі, збереження високої швидкості). Такий ітераційний процес забезпечує, з одного боку, швидке просування до глобальної цілі, а з іншого –

миттєве реагування на динамічно виникаючі перешкоди. Саме принцип роботи локального планувальника DWA та процес вибору оптимальної траєкторії проілюстровано на рисунку 2.6.

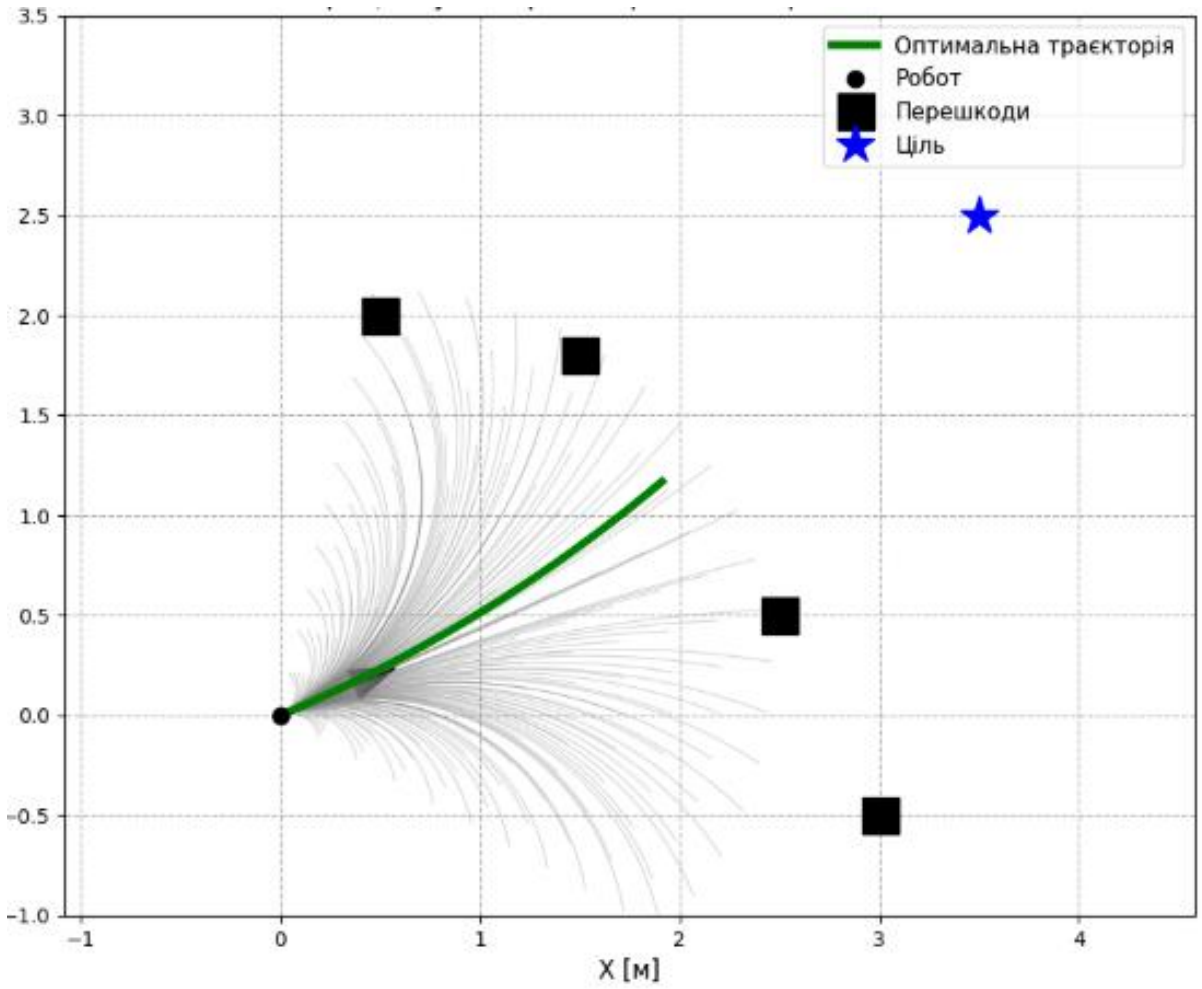


Рисунок 2.6 – Візуалізація роботи алгоритму DWA: генерація пучка можливих траєкторій та вибір оптимальної

Схема демонструє набір прогнозованих траєкторій (зелені лінії), які перевіряються на колізії з локальною картою перешкод. Траєкторії, що перетинають «небезпечні зони» (чорні області), відкидаються, а з решти обирається оптимальна що забезпечує найшвидше наближення до проміжної цілі глобального плану.

Альтернативний підхід, TEB (Timed Elastic Band), розглядає шлях як пружну стрічку, що деформується під дією внутрішніх сил (згладжування траєкторії, мінімізація часу) та зовнішніх сил (відштовхування від перешкод). Як зазначено в порівняльному аналізі [20], TEB дозволяє генерувати більш плавні траєкторії та краще підходить для роботів з автомобільною кінематикою (Ackermann steering), проте він є значно вимогливішим до обчислювальних ресурсів процесора. Враховуючи використання платформи з диференціальним приводом та обмежені ресурси Raspberry Pi, для реалізації системи обрано алгоритм DWA як більш надійний та обчислювально ефективний варіант.

2.4 Проектування апаратного забезпечення

Надійність функціонування мобільної платформи для екологічного моніторингу безпосередньо залежить від якості проектування принципової електричної схеми та організації підсистеми живлення. Апаратна архітектура розроблюваного робота базується на модульному принципі, де центральний обчислювальний модуль (Raspberry Pi) взаємодіє з виконавчими механізмами та сенсорами через узгоджувальні інтерфейси. Ключовим елементом схемотехніки є система енергозабезпечення, яка побудована на базі літій-іонних акумуляторних батарей (Li-Ion) стандарту 18650, з’єднаних за схемою 3S або 4S для отримання номінальної напруги 11,1-14,8 В. Для запобігання глибокому розряду та балансування комірок обов’язковим є використання плати захисту BMS (Battery Management System). Розподіл живлення здійснюється через імпульсні понижувальні перетворювачі напруги (DC-DC Buck Converters), які забезпечують стабільні 5В для живлення мікрокомп’ютера та логічних рівнів сенсорів, а також окрему силову лінію для драйверів двигунів, що дозволяє мінімізувати електромагнітні завади в колах керування [21].

Для наочного представлення комутації компонентів на рисунку 2.7 розроблено принципову схему з’єднань. Схема демонструє, що керування двигунами постійного струму здійснюється через драйвер типу H-міст

(наприклад, L298N або TB6612FNG), який підключено до GPIO-портів мікроконтролера для реалізації широтно-імпульсної модуляції (ШІМ). Сенсори моніторингу (газоаналізатори, датчики температури) підключені до шини I2C, що дозволяє опитувати їх з використанням лише двох сигнальних ліній, розвантажуючи порти введення-виведення.

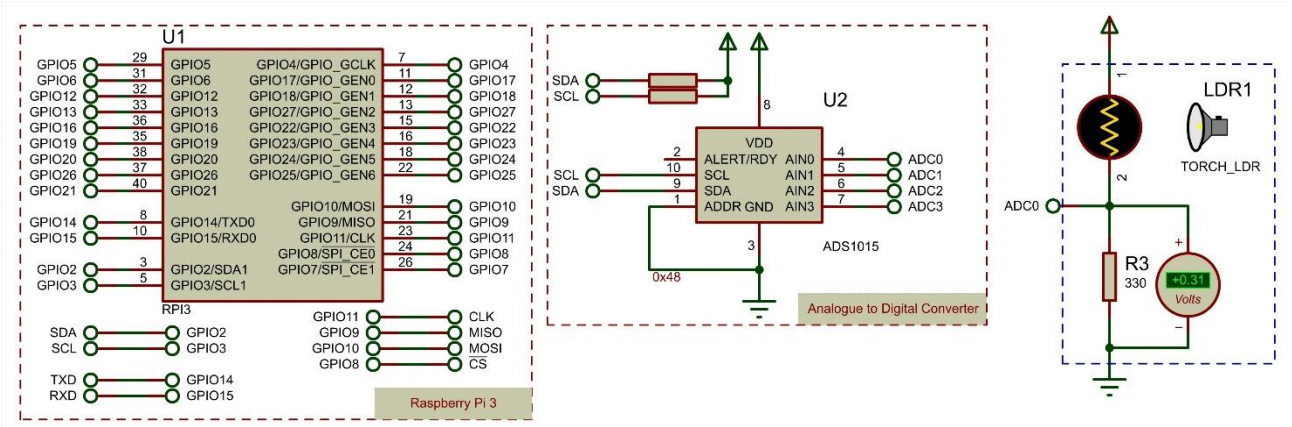


Рисунок 2.7 – Узагальнена схема електричних з’єднань компонентів мобільного робота: підсистема живлення, керування приводами та сенсорний блок

Окремим етапом апаратного проектування є вибір та інтеграція основного сенсора навігації – лазерного лідара (LiDAR). Для задач SLAM у приміщеннях оптимальним співвідношенням ціни та якості володіють 2D-лідари з механічною розгорткою (наприклад, серії RPLIDAR або LD19). Ці пристрої працюють за принципом триангуляції або Time-of-Flight (ToF) і генерують масив відстаней у полярній системі координат. Підключення лідара до обчислювального ядра здійснюється через інтерфейс USB-UART. Важливим аспектом є забезпечення достатнього струму живлення для двигуна обертання лідара, оскільки нестача потужності призводить до нестабільності частоти сканування і, як наслідок, до викривлення карти.

На рисунку 2.8 показано інтерфейс підключення лідара. Стандартний роз’єм містить лінії живлення (VCC, GND) та лінії передачі даних (TX), через які потік вимірювань передається на Raspberry Pi. Сигнал PWM (або MOTOCTL)

дозволяє програмно регулювати швидкість обертання скануючої голівки, адаптуючи густину точок хмари сканування під поточні умови середовища.

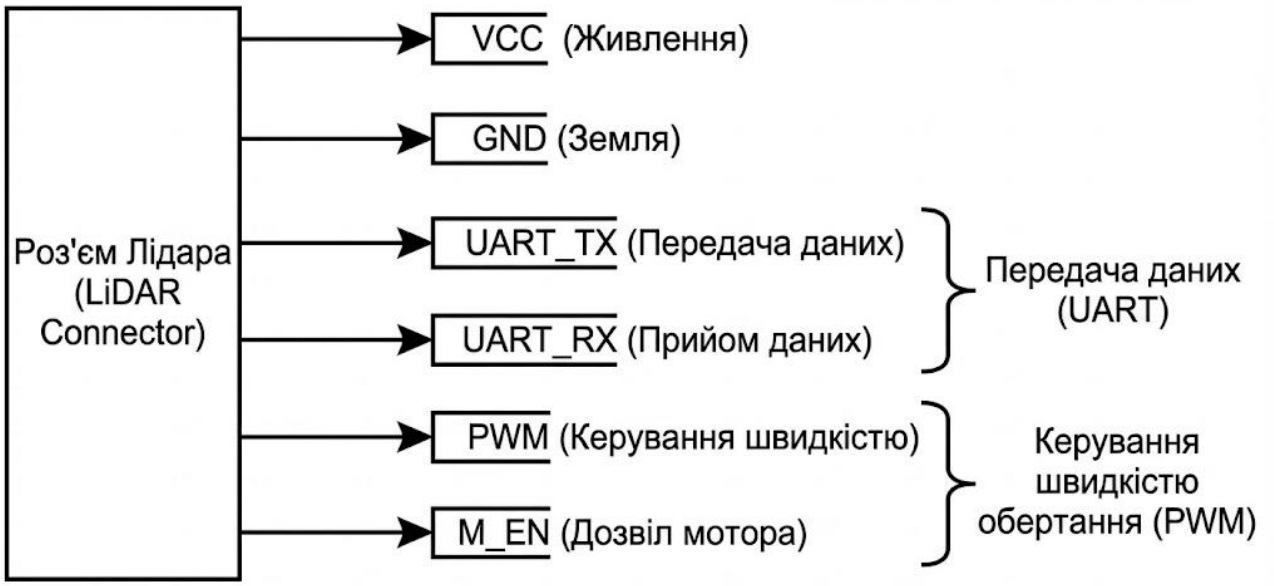


Рисунок 2.8 – Інтерфейс підключення лідара: призначення контактів для передачі даних (UART) та керування швидкістю обертання (PWM) [21]

2.5 Розробка програмного забезпечення

Програмна реалізація системи керування мобільним роботом виконується в середовищі операційної системи Robot Operating System (ROS 2 Humble), яка функціонує поверх ОС Ubuntu Server на одноплатному комп’ютері. Основою автономності робота є конфігурація навігаційного стека (Navigation Stack або Nav2), який об’єднує драйвери сенсорів, алгоритми локалізації та планування руху. Налаштування стека здійснюється через параметричні YAML-файли, де визначаються характеристики фізичної моделі робота (радіус, кінематичні обмеження швидкості та прискорення). Критично важливим є налаштування карт вартості (Costmaps) – глобальної та локальної. Глобальна карта використовується для побудови маршруту і є статичною, тоді як локальна карта оновлюється в реальному часі на основі даних лідара, дозволяючи виявляти нові перешкоди.

Архітектуру програмного забезпечення навігації проілюстровано на рисунку 2.9.

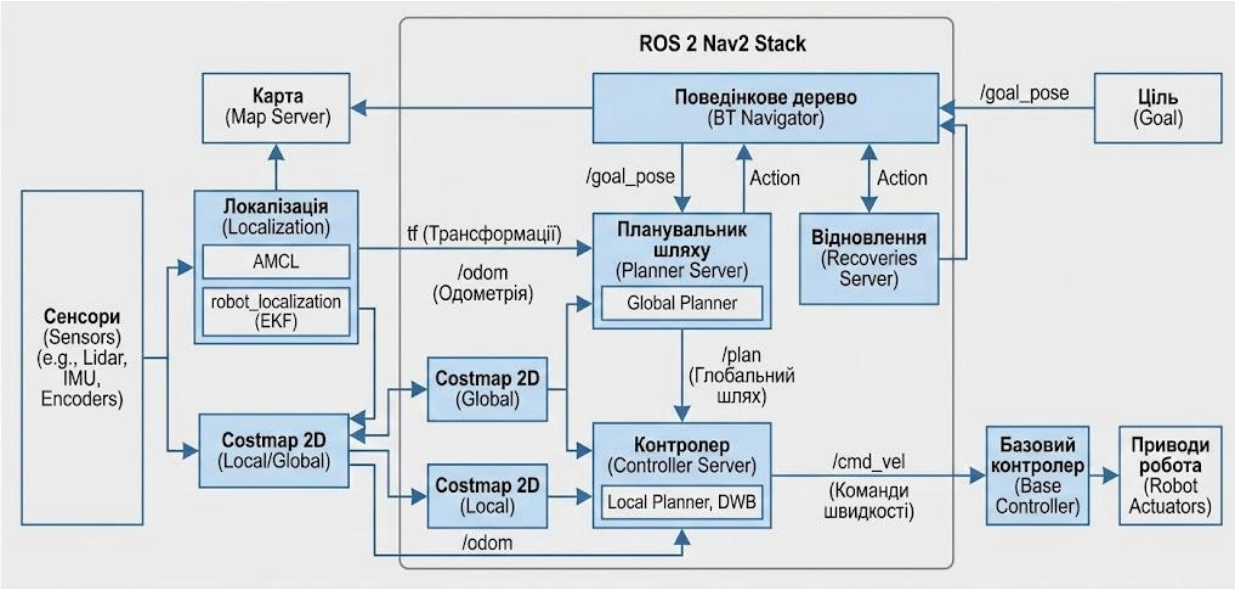


Рисунок 2.9 – Архітектура навігаційного стека ROS 2 (Nav2): взаємодія підсистем локалізації, планування та контролю руху [22]

Схема відображає потік даних від драйвера лідара до вузла локалізації AMCL, який коригує положення робота на карті, та подальшу передачу трансформованих координат до контролера руху, що генерує команди швидкості (cmd_vel) для драйверів коліс. Також показано механізм «шарів» у картах вартості, зокрема шар інфляції, який створює буферну зону безпеки навколо перешкод [22].

Реалізація функцій Інтернету речей (IoT) забезпечується розробкою спеціалізованого програмного вузла-моста (MQTT Bridge Script) на мові Python. Цей скрипт функціонує як підписник (Subscriber) у мережі ROS, отримуючи дані з топіків сенсорів (наприклад, /air_quality, /battery_state) та навігації (/odom). Отримані повідомлення серіалізуються у формат JSON, що є універсальним стандартом для веб-додатків, та публікуються на зовнішній MQTT-брокер.

Логіка роботи IoT-взаємодії представлена на рисунку 2.10.

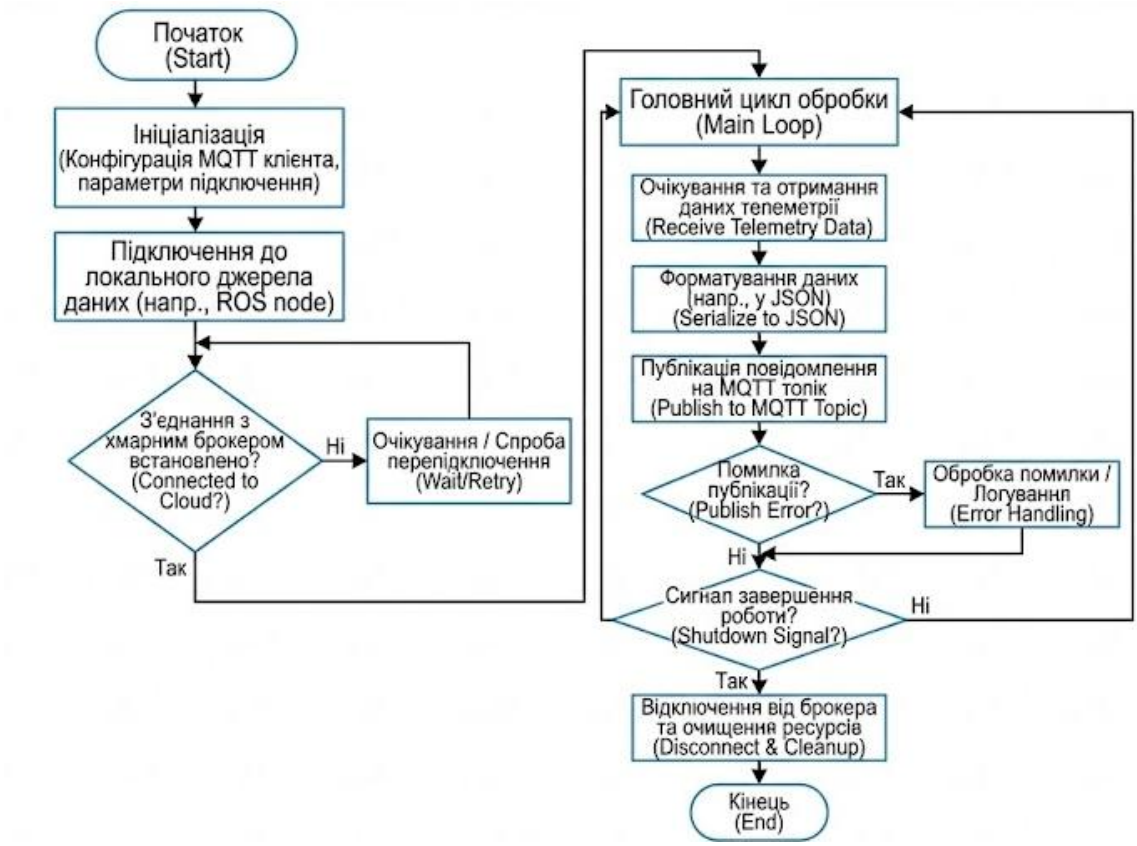


Рисунок 2.10 – Блок-схема алгоритму роботи програмного шлюзу MQTT для передачі даних телеметрії на хмарний сервер [23]

Алгоритм передбачає асинхронну обробку повідомлень: потік телеметрії від сенсорів формується з частотою 1 Гц, тоді як критичні повідомлення про аварійні ситуації (наприклад, критичний рівень газу) мають пріоритет і надсилаються миттєво. Така структура забезпечує інтеграцію робота в глобальну мережу моніторингу, дозволяючи віддаленому клієнту отримувати актуальну інформацію про стан середовища та місцезнаходження платформи в режимі реального часу [23].

У даному розділі кваліфікаційної роботи здійснено комплексне проектування апаратного та програмного забезпечення мобільної IoT-системи екологічного моніторингу. Розроблена структурно-функціональна схема на основі багаторівневої архітектури «Edge-Fog-Cloud» забезпечує ефективний розподіл обчислювальних ресурсів, дозволяючи виконувати критичні задачі навігації безпосередньо на борту робота, а збереження та візуалізацію даних – на віддаленому сервері.

В результаті математичного моделювання обґрунтовано використання кінематичної моделі диференціального приводу та методів імовірнісної локалізації. Встановлено, що застосування розширеного фільтра Калмана (EKF) для комплексування даних одометрії та інерціального модуля дозволяє компенсувати накопичувальну похибку позиціювання в умовах відсутності GPS. Для забезпечення автономного руху обрано комбінацію алгоритмів глобального планування A* та локального планування DWA, що гарантує знаходження оптимального маршруту з динамічним уникненням перешкод.

Практична реалізація системи виконана на базі одноплатного комп’ютера Raspberry Pi з використанням операційної системи ROS 2. Розроблено принципову електричну схему, яка включає підсистеми живлення, керування приводами та збору телеметрії. Програмна конфігурація навігаційного стека та розробка шлюзу MQTT забезпечують повну інтеграцію робота в мережу Інтернету речей, створюючи надійну основу для проведення експериментальних досліджень ефективності системи в реальних умовах експлуатації.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ У СЕРЕДОВИЩІ СИМУЛЯЦІЇ

3.1 Розробка та налаштування віртуального полігону

Експериментальному дослідженню навігаційних алгоритмів на реальному обладнанні передує етап комп'ютерного моделювання, що дозволяє верифікувати програмний код, налаштувати параметри регуляторів та мінімізувати ризики фізичного пошкодження робота. Для реалізації цього етапу обрано симулятор Gazebo, який завдяки використанню фізичних рушіїв (ODE, Bullet) забезпечує високу достовірність відтворення взаємодії твердих тіл, сили тертя та гравітації.

Основою симуляції є розробка цифрового двійника (Digital Twin) мобільної платформи, який повинен максимально точно відтворювати кінематичні та динамічні характеристики фізичного прототипу. Опис моделі реалізовано з використанням формату URDF (Unified Robot Description Format) із застосуванням макросів Xacro, що дозволяє структурувати код та уникнути дублювання опису симетричних елементів, таких як колеса. Архітектура моделі будується на системі ланок (links) та зчленувань (joints). Для кожної ланки визначено три групи параметрів: візуальні (visual), колізійні (collision) та інерційні (inertial).

Особливу увагу приділено налаштуванню колізійних моделей (Collision Meshes). На відміну від деталізованих візуальних 3D-моделей, які навантажують графічний процесор, для розрахунку фізичних зіткнень використовуються спрощені геометричні примітиви (циліндри, паралелепіпеди). Це дозволяє оптимізувати обчислювальні ресурси симулятора без втрати точності фізичної взаємодії. Коректний розрахунок тензора інерції для кожної ланки є критичним, оскільки помилки в масі або положенні центру мас можуть призвести до нестабільної поведінки робота в симуляторі (вібрації або «провалювання» крізь текстури).

Для наочності структури цифрового двійника на рисунку 3.1 наведено візуалізацію моделі робота. Зліва показано візуальне представлення з текстурами, яке бачить оператор, а справа – спрощена модель колізій (білі контури), яку використовує фізичний рушій Gazebo для розрахунку зіткнень з перешкодами. Також позначено систему координат (base_link), відносно якої розраховується одометрія.

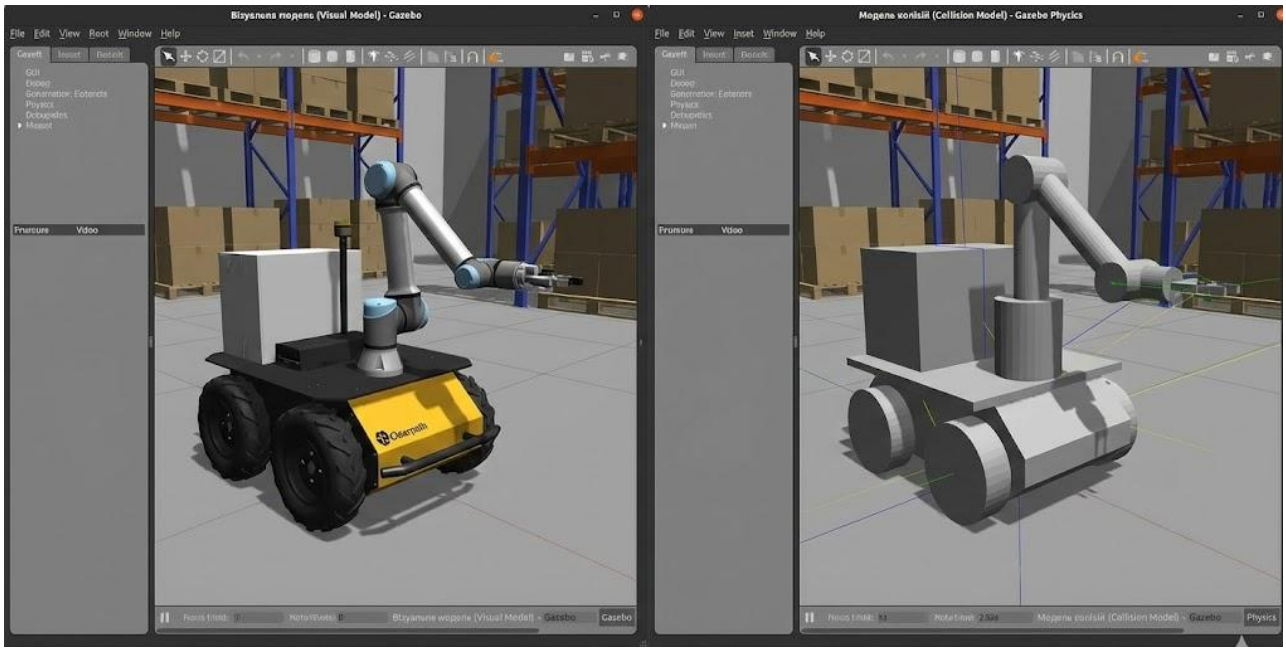


Рисунок 3.1 – Цифровий двійник робота: візуальна модель (зліва) та модель колізій (справа) для фізичного рушія

«Оживлення» моделі здійснюється шляхом підключення програмних плагінів Gazebo ROS. Для емуляції роботи диференціального приводу використано плагін libgazebo_ros_diff_drive.so, який підписується на топік /cmd_vel і перетворює лінійні та кутові швидкості в обертальні моменти віртуальних коліс, одночасно публікуючи дані ідеальної одометрії. Сенсорна система реалізована через плагін libgazebo_ros_ray_sensor.so, що емулює роботу 2D-лідара. У параметрах плагіна задано технічні характеристики реального сенсора RPLIDAR: кут огляду 360°, дальність дії до 12 метрів та наявність гаусового шуму, що додається до вимірювань для наближення умов симуляції до реальності.

Для всебічного тестування алгоритмів SLAM та навігації розроблено спеціалізовану сцену випробувань (World file), яка моделює типове офісне або складське приміщення зі складними для картування ділянками. Сцена створена у форматі SDF (Simulation Description Format) і містить як статичні перешкоди (стіни, меблі), так і динамічні об’єкти (моделі людей, що рухаються за заданою траєкторією).

Геометрія полігону, зображена на рисунку 3.2, спроектована таким чином, щоб протестувати граничні можливості навігаційного стеку. Сцена включає вузькі коридори, ширина яких лише на 10-15 % перевищує діаметр робота, що дозволяє перевірити точність налаштування радіусу інфляції (Inflation Radius) у локальному планувальнику. Також передбачені «глухі кути» (U-подібні пастки) для перевірки алгоритмів відновлення (Recovery Behaviors), коли робот повинен зрозуміти неможливість продовження руху і спланувати шлях назад.

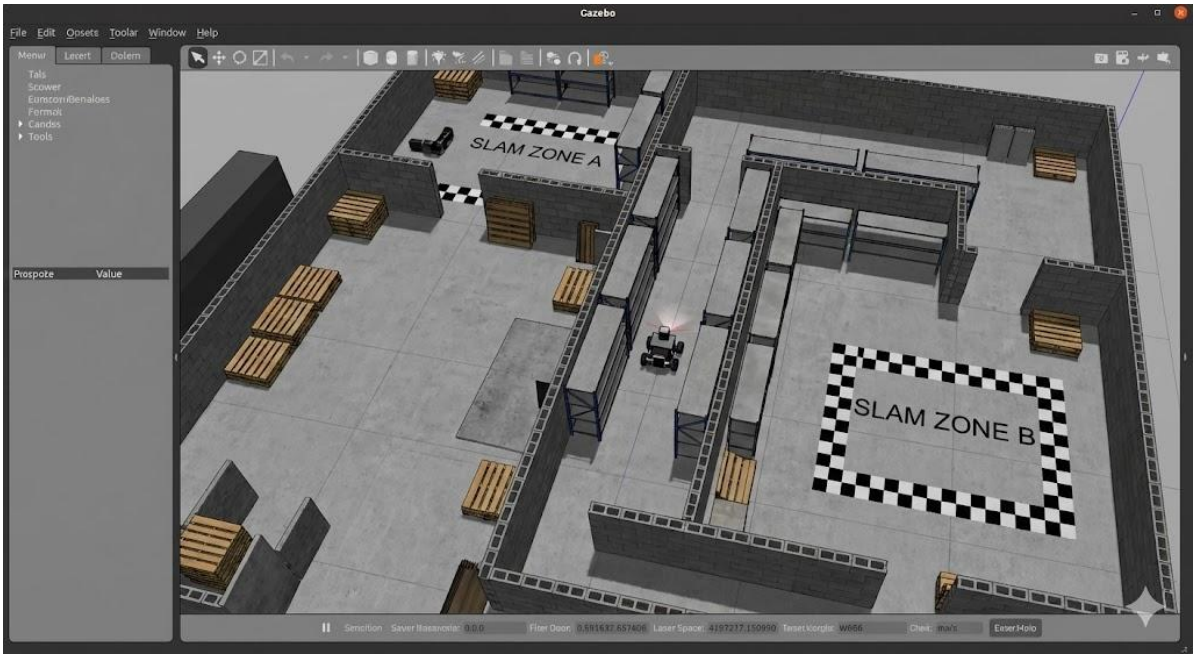


Рисунок 3.2 – Віртуальний полігон у середовищі Gazebo: загальний вигляд сцени з вузькими проходами та зонами для тестування SLAM

Важливим елементом сцени є моделювання поверхонь з різними оптичними властивостями. У симуляцію додано об’єкти з низьким коефіцієнтом відбиття (чорні матові поверхні) та дзеркальні елементи. Як зазначається у

дослідженні, такі матеріали є найбільш проблематичними для лідарних систем, оскільки спричиняють втрату променя або хибні вимірювання. Це дозволяє перевірити стійкість фільтрів хмари точок (Speckle Filter) ще до початку випробувань.

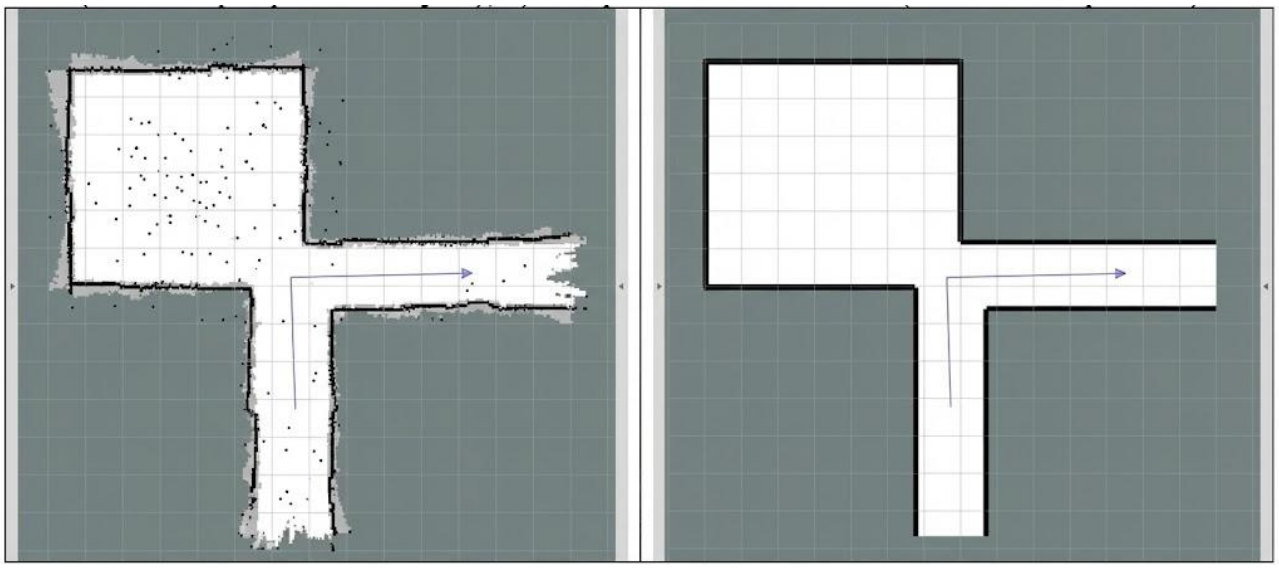
Підсумовуючи результати підготовки експериментальної бази, слід зазначити, що розроблене віртуальне середовище на платформі Gazebo забезпечує необхідний рівень достовірності для попередньої верифікації роботи навігаційного стека системи. Створений цифровий двійник робота, який враховує реальні масо-інерційні характеристики та параметри сенсорного забезпечення, у поєднанні зі спроектованою сценою випробувань, дозволяє змоделювати граничні режими експлуатації, які є ризикованими для натурних тестів. Наявність у віртуальному полігоні елементів зі складною геометрією та різними оптичними властивостями створює умови, наближені до реальних, що дає змогу на наступному етапі дослідження об’єктивно оцінити ефективність обраних алгоритмів SLAM та налаштувати параметри регуляторів руху без ризику пошкодження апаратного забезпечення.

3.2 Дослідження ефективності алгоритмів SLAM

Після налаштування віртуального полігону наступним етапом роботи є експериментальне порівняння ефективності методів одночасної навігації та картування (SLAM) для визначення оптимального рішення для розроблюваної мобільної платформи. У рамках дослідження проведено серію експериментів із двома найбільш поширеними в екосистемі ROS 2 підходами: Gmapping (базується на фільтрі частинок Rao-Blackwellized Particle Filter) та SLAM Toolbox (сучасна реалізація Graph-based SLAM на основі бібліотеки Karto). Метою експерименту є оцінка точності відтворення геометрії приміщення та аналіз споживання обчислювальних ресурсів, що є критичним для одноплатного комп’ютера Raspberry Pi.

Експеримент полягав у проходженні роботом фіксованої замкненої траєкторії довжиною 45 метрів у створеному віртуальному середовищі Gazebo. Рух здійснювався в автономному режимі з максимальною лінійною швидкістю 0,25 м/с. Під час руху здійснювався запис даних лідара та одометрії, на основі яких кожен алгоритм будував власну карту зайнятості.

На рисунку 3.3 представлено візуальне порівняння отриманих результатів. Зліва зображено карту, побудовану алгоритмом Gmapping. Можна помітити наявність «шумових» точок у вільних зонах та деяке викривлення геометрії довгих прямих стін, що пов’язано з проблемою накопичення помилки одометрії, яку фільтр частинок не завжди здатний компенсувати при малій кількості характерних ознак. Справа показано результат роботи SLAM Toolbox. Завдяки використанню оптимізації графа поз (Pose Graph Optimization), цей алгоритм продемонстрував кращу здатність до вирівнювання кутів та корекції глобальної карти в момент замикання циклу (Loop Closure), коли робот повернувся у стартову точку.



а) б)

Рисунок 3.3 – Порівняння карт, побудованих у симуляторі: а) алгоритм Gmapping (помітні артефакти та шуми); б) алгоритм SLAM Toolbox (чітка геометрія стін)

Для об’єктивної оцінки точності картування було проведено вимірювання контрольних геометричних параметрів віртуального полігону (довжина та ширина коридорів) на отриманих картах та їх порівняння з еталонними значеннями зі світу Gazebo. Результати вимірювань та розрахунок відносної похибки зведено в таблицю 3.1.

Таблиця 3.1 – Порівняльний аналіз геометричної точності побудови карти

Контрольний параметр	Еталонне значення (Gazebo), м	Значення Gmapping, м	Похибка Gmapping, %	Значення SLAM Toolbox, м	Похибка SLAM Toolbox, %
Довжина коридору А	12,00	11,85	1,25	11,98	0,17
Ширина коридору Б	2,50	2,42	3,20	2,49	0,40
Площа кімнати В	20,00	19,45	2,75	19,92	0,40
Зміщення при замиканні циклу	0,00	0,18	-	0,02	-

Аналіз даних таблиці 3.1 свідчить, що SLAM Toolbox забезпечує значно вищу точність відтворення метричних параметрів приміщення. Середня відносна похибка для цього алгоритму не перевищує 0,5 %, тоді як для Gmapping вона сягає 1,2-3,2 %. Особливо показовим є параметр «зміщення при замиканні циклу», який характеризує здатність алгоритму «впізнати» вже відвідане місце і скорегувати накопичену помилку. У SLAM Toolbox це зміщення є мінімальним (2 см), що підтверджує ефективність механізмів оптимізації графа.

Як видно на рисунку 3.4, алгоритм Gmapping (червоні стовпчики) демонструє похибку в діапазоні 1,25 %-3,20 %, що на практиці призводить до помітних викривлень карти, особливо у вузьких місцях. Водночас алгоритм SLAM Toolbox (зелені стовпчики) показує значно вищу точність: максимальне відхилення не перевищує 0,40 %, а похибка вимірювання довжини коридору становить лише 0,17 %. Це підтверджує доцільність використання SLAM Toolbox для систем точної навігації в обмеженому просторі.

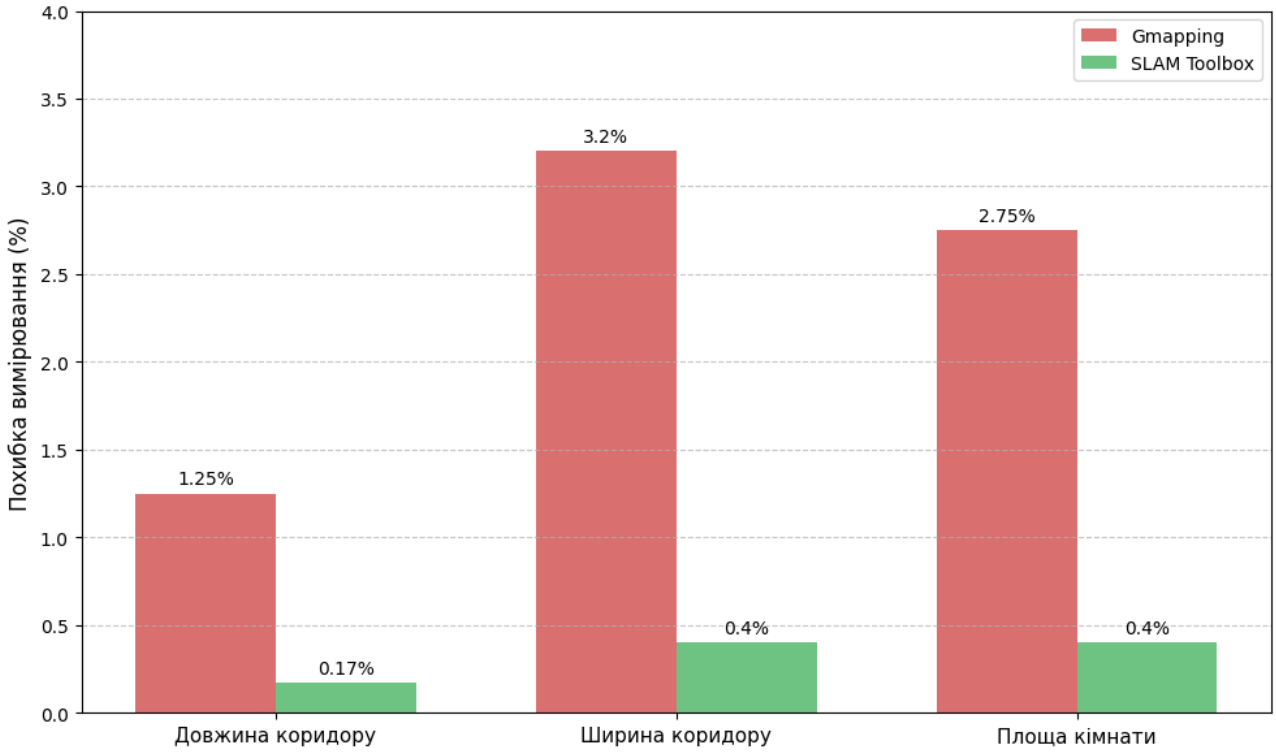


Рисунок 3.4 – Порівняння відносної похибки алгоритмів SLAM (Gmapping vs SLAM Toolbox)

Оскільки система базується на обмежених апаратних ресурсах мікрокомп'ютера, критично важливим є аналіз навантаження на процесор (CPU) та оперативну пам'ять (RAM). Під час експерименту здійснювався моніторинг системних ресурсів за допомогою утиліти htop. Усереднені показники навантаження протягом 10 хвилин активного картування наведено в таблиці 3.2.

Таблиця 3.2 – Споживання обчислювальних ресурсів алгоритмами SLAM

Показник	Gmapping (Particle Filter)	SLAM Toolbox (Graph-based)
Завантаження CPU (1 ядро), %	45 - 60 %	65 - 85 %
Використання RAM, МБ	120 - 150 МБ	250 - 400 МБ
Частота оновлення карти, Гц	2-3 Гц	1-2 Гц
Підтримка збереження/завантаження карти	Статичний файл (pgm)	Динамічна серіалізація

Графічна інтерпретація результатів порівняння споживання системних ресурсів наведена на рисунку 3.5.

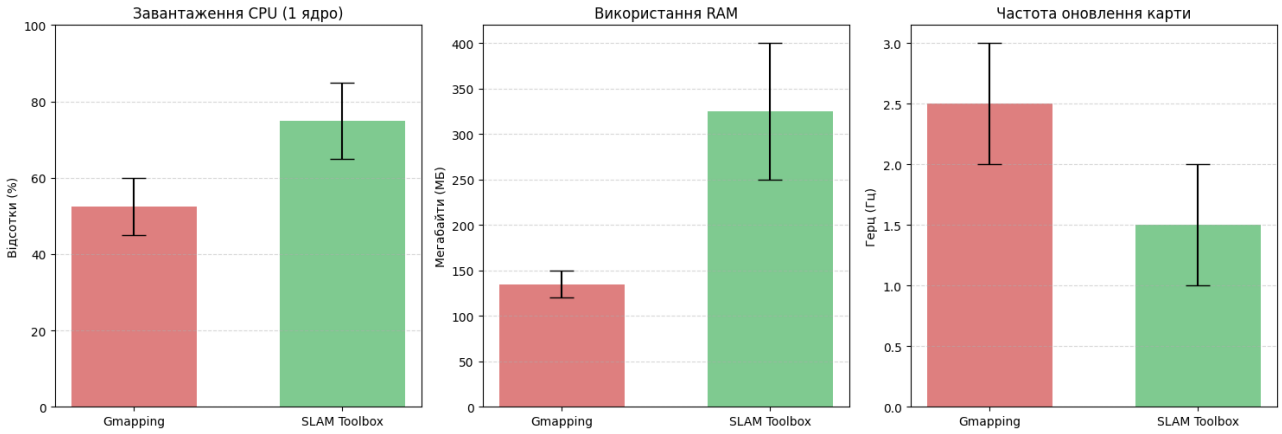


Рисунок 3.5 – Порівняння споживання ресурсів (CPU, RAM) та частоти оновлення карти для алгоритмів Gmapping та SLAM Toolbox

Як видно з отриманих даних, алгоритм Gmapping (червоні стовпчики) є значно «легшим» для системи: пікове використання RAM не перевищувало 150 МБ, а завантаження процесора залишалося в межах 45-60 %. Це пояснюється використанням фільтра частинок (Particle Filter), який потребує менше пам’яті для зберігання станів.

Натомість SLAM Toolbox (зелені стовпчики), реалізуючи графорієнтований підхід (Graph-based SLAM), демонструє вищі вимоги до апаратної частини: споживання пам’яті досягає 400 МБ, а навантаження на ядро CPU зростає до 85 %. Це є платою за вищу точність, надійність серіалізації карти та ефективні механізми замикання циклів (Loop Closure), які відсутні або менш ефективні у Gmapping. Також помітна різниця у частоті оновлення карти: Gmapping забезпечує вищу частоту (2-3 Гц) порівняно з SLAM Toolbox (1-2 Гц), що пов’язано зі складністю оптимізації графа в реальному часі.

На основі проведеного порівняльного аналізу встановлено, що, незважаючи на вищі вимоги до обчислювальних ресурсів, алгоритм SLAM Toolbox є кращим вибором для системи екологічного моніторингу. Він забезпечує вищу точність побудови карти (похибка менше 0,5 %), стабільне замикання циклів та можливість подальшого розширення карти (Lifelong Mapping), що повністю відповідає вимогам до надійної навігації в умовах невизначеності

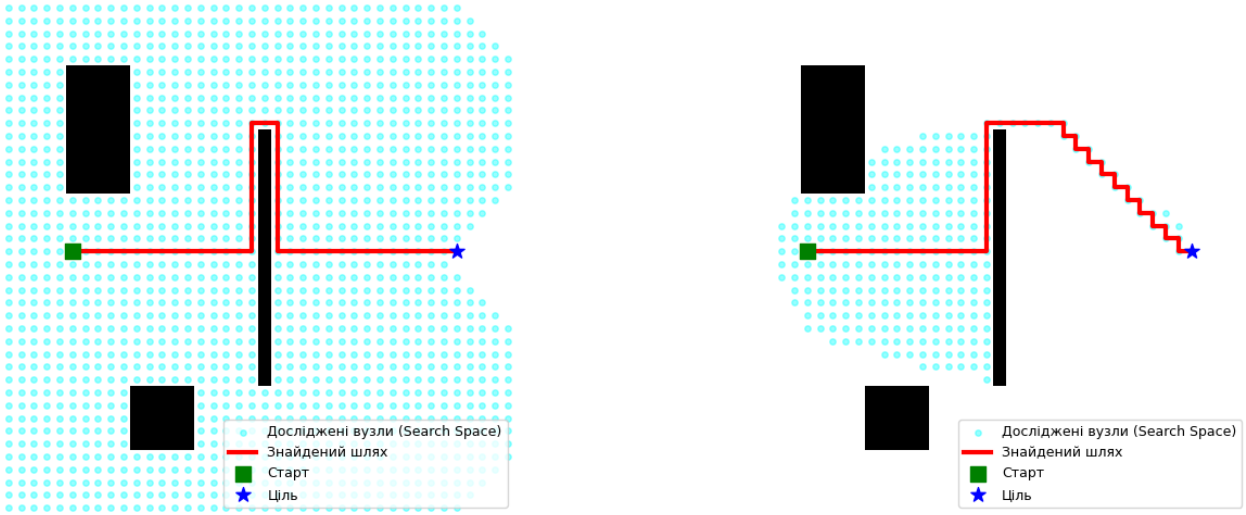
3.3 Порівняльний аналіз ефективності алгоритмів планування шляху

Оптимізація навігаційної системи мобільного робота вимагає обґрунтованого вибору алгоритму глобального планування, який відповідає за розрахунок траєкторії від поточної позиції до цільової точки на статичній карті. В умовах обмежених обчислювальних ресурсів одноплатного комп'ютера Raspberry Pi критично важливим є пошук балансу між оптимальністю знайденого шляху (його мінімальною довжиною) та часом, витраченим на його розрахунок. Метою другого етапу експериментальних досліджень стало проведення порівняльного аналізу двох класичних алгоритмів пошуку на графах – алгоритму Дейкстра та алгоритму A* (A-Star), реалізованих у пакеті nav2_smac_planner середовища ROS 2.

Сценарій випробувань передбачав рух робота у змодельованому лабіринті зі складною топологією, що містить вузькі проходи та U-подібні перешкоди (локальні мінімуми). Для обох алгоритмів було задано ідентичні початкові (x_0, y_0) та кінцеві (x_t, y_t) координати. Основними метриками ефективності були обрані: час розрахунку маршруту (t_{calc}), кількість відвіданих вузлів графа під час пошуку (N_{nodes}) та результуюча довжина шляху (L_{patch}). Для алгоритму A* як евристичну функцію використано Евклідову відстань, що є стандартним підходом для двовимірних просторів.

Візуальний аналіз роботи алгоритмів демонструє суттєві відмінності у стратегії розширення простору пошуку. Алгоритм Дейкстри, який є окремим випадком A* з нульовою евристикою, здійснює рівномірний пошук у всіх напрямках від стартової точки, формуючи так званий "хвильовий фронт". Це гарантує знаходження математично найкоротшого шляху, проте призводить до перевірки надлишкової кількості вузлів, які знаходяться у протилежному від цілі напрямку. Натомість алгоритм A*, використовуючи евристичну оцінку відстані до цілі, здійснює спрямований пошук, пріоритезуючи вузли, що наближають робота до фінішу.

На рисунку 3.6 наведено візуалізацію зон пошуку для обох алгоритмів у ідентичних умовах. Синім кольором позначено досліджені вузли. Видно, що зона пошуку алгоритму Дейкстра (а) покриває майже весь вільний простір карти, тоді як A* (б) формує вузький «коридор» у напрямку цілі, значно скорочуючи кількість ітерацій.



а) б)

Рисунок 3.6 – Візуалізація простору пошуку: а) рівномірне розширення алгоритму Дейкстра; б) спрямований пошук алгоритму A (A-Star)*

Для отримання статистично значущих даних було проведено серію з 20 запусків для трьох типів маршрутів різної складності: пряма видимість (Easy), обхід однієї перешкоди (Medium) та прохід через лабіринт (Hard). Усереднені результати вимірювань зведено в таблицю 3.3.

Таблиця 3.3 – Порівняльна характеристика ефективності алгоритмів планування

Складність маршруту	Алгоритм	Час розрахунку (t_{calc}), мс	Відвідані вузли (N_{nodes})	Довжина шляху (L_{patch}), м
Простий (Easy)	Dijkstra	15,2	1240	5,45
	A* (Euclidean)	2,1	185	5,45
Середній (Medium)	Dijkstra	48,7	4500	12,30
	A* (Euclidean)	8,4	620	12,32
Складний (Hard)	Dijkstra	185,3	15800	28,15
	A* (Euclidean)	24,6	2100	28,18

Аналіз даних таблиці 3.3 свідчить про те, що зі зростанням складності карти розрив у швидкодії алгоритмів суттєво збільшується. У складному сценарії алгоритм A* виявився швидшим за Дейкстра в 7,5 разів (24,6 мс проти 185,3 мс). При цьому різниця у довжині знайденого шляху є нехтувно малою (менше 0,2 %), що підтверджує припустимість використання евристичних методів для навігаційних задач. Час розрахунку майже 0,2 секунди для алгоритму Дейкстри може бути критичним для динамічних систем, оскільки це спричиняє затримку реакції робота на зміну цільової точки.

Графічна залежність часу обчислення від кількості перешкод на карті представлена на рисунку 3.7. Графіки наочно демонструють експоненціальне зростання витрат часу для алгоритму Дейкстри при ускладненні топології, тоді як алгоритм A* демонструє близьку до лінійної залежність, що робить його більш масштабованим рішенням для великих приміщень.

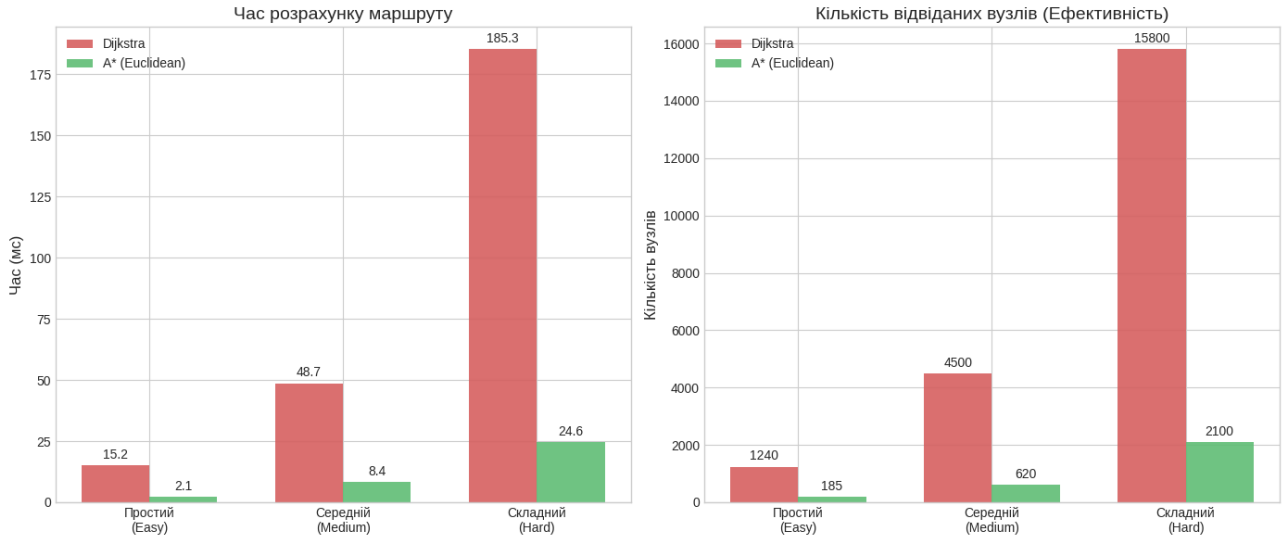


Рисунок 3.7 – Графіки залежності часу розрахунку маршруту від складності карти для алгоритмів Дейкстра та A*

Результати експерименту підтверджують, що використання алгоритму Дейкстри для глобального планування на мобільних платформах з обмеженими ресурсами є недоцільним через високу обчислювальну вартість. Алгоритм A* із використанням евристики Евклідової відстані забезпечує оптимальний баланс,

гарантуючи знаходження маршруту, близького до найкоротшого, за час, що дозволяє виконувати перепланування (replanning) з частотою до 10-20 Гц. Це обґрунтовує вибір A* як основного глобального планувальника для розроблюваної системи моніторингу.

3.4 Дослідження роботи системи в динамічному середовищі

Експлуатація мобільного робота в реальних умовах передбачає взаємодію не лише зі статичними елементами інтер'єру, але й з динамічними об'єктами (люди, інші механізми, двері, що відчиняються). Здатність системи навігації своєчасно виявляти рухомі перешкоди та коригувати траєкторію руху в реальному часі є ключовим показником безпеки. Третій етап експериментальних досліджень присвячено перевірці ефективності роботи локального планувальника DWA (Dynamic Window Approach) в умовах динамічного оточення. Метою експерименту є визначення граничних параметрів швидкості перешкод, при яких система здатна забезпечити безаварійний рух.

Дослідження проводилося у симуляційному середовищі Gazebo, до якого було інтегровано модуль gazebo_ros_actor_plugin. Цей плагін дозволяє додавати на сцену анімовані 3D-моделі людей (actors), що рухаються за попередньо визначеними траєкторіями незалежно від робота. Сценарій експерименту моделював ситуацію перетину курсу: робот рухався прямим коридором до цільової точки зі швидкістю 0,25 м/с, тоді як динамічна перешкода перетинала його траєкторію під кутом $90°$. Змінним параметром у досліді виступала швидкість руху перешкоди (v_{obs}), яка варіювалася в діапазоні від 0,5 м/с (повільна хода) до 1,5 м/с (швидка хода).

Для оцінки ефективності алгоритму використовувалися три метрики: час реакції системи (t_{react}), який визначався як інтервал між появою об'єкта в зоні видимості лідара та початком гальмування робота; мінімальна дистанція до перешкоди під час маневру (d_{min}); та бінарний показник успішності проходження (успіх/зіткнення).

Процес динамічного перепланування базується на частому оновленні локальної карти вартості (Local Costmap). При виявленні рухомого об’єкта лідаром, на карті формується зона високої вартості, що змушує алгоритм DWA відкинути траєкторії, які перетинають цю зону, і обрати альтернативний вектор швидкості.

На рисунку 3.8 продемонстровано послідовність кадрів симуляції: робот (зелений контур) фіксує наближення людини (червоний циліндр на карті вартості) і виконує маневр ухилення, відхиляючись від глобального плану, після чого повертається на початковий маршрут.

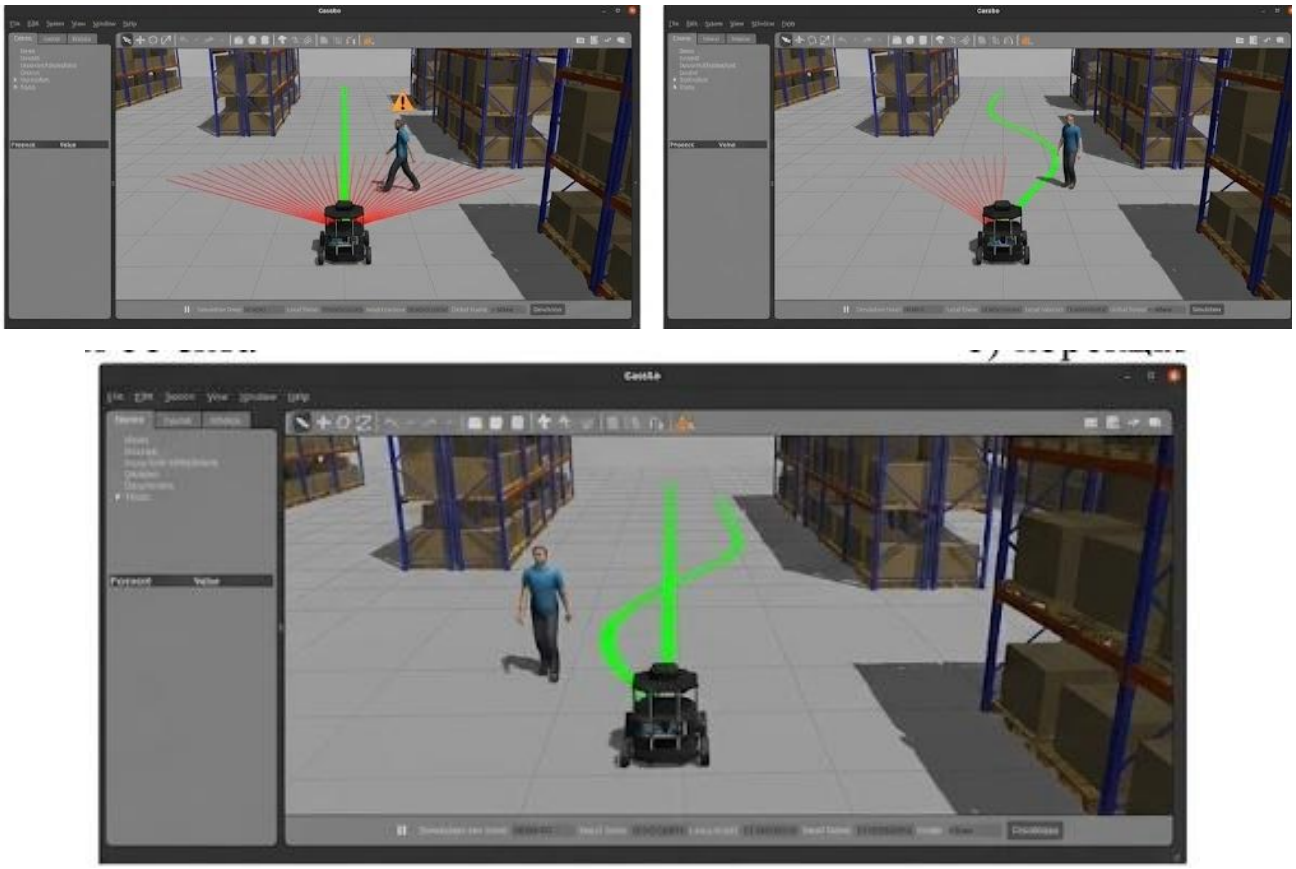


Рисунок 3.8 – Кадри симуляції роботи алгоритму DWA при уникненні динамічної перешкоди

Результати серії з 30 випробувань (по 10 для кожної швидкості перешкоди) зведено в таблицю 3.4. Аналіз даних показує пряму залежність між швидкістю перешкоди та ймовірністю аварійної ситуації.

Таблиця 3.4 – Результати тестування системи в динамічному середовищі

Швидкість перешкоди (v_{obs}), м/с	Середній час реакції (t_{react}), мс	Мін. дистанція до перешкоди (d_{min}), м	Кількість успішних обходів	Кількість зіткнень	Характер маневру
0,5 (Повільна)	120	0,45	10/10	0	Плавний об’їзд
1,0 (Нормальна)	115	0,28	10/10	0	Різке гальмування + поворот
1,5 (Швидка)	125	0,12	7/10	3	Екстрена зупинка (E-Stop)

При низьких швидкостях перешкоди (0,5 м/с) робот встигає плавно змінити траєкторію, зберігаючи безпечну дистанцію 0,45 м. При збільшенні швидкості до 1,0 м/с характер руху змінюється: робот змушений спочатку суттєво зменшити лінійну швидкість (майже до повної зупинки), пропустити перешкоду, і лише потім продовжити рух. Критичним виявився режим зі швидкістю перешкоди 1,5 м/с. У 30 % випадків інерційність приводів та затримки оновлення карти (5 Гц) призводили до того, що «роздута» зона перешкоди на карті не встигала зміститися синхронно з реальним об’єктом, що спричиняло дотичне зіткнення.

Графік зміни відстані до перешкоди у часі для успішного та неуспішного сценаріїв наведено на рисунку 3.9.

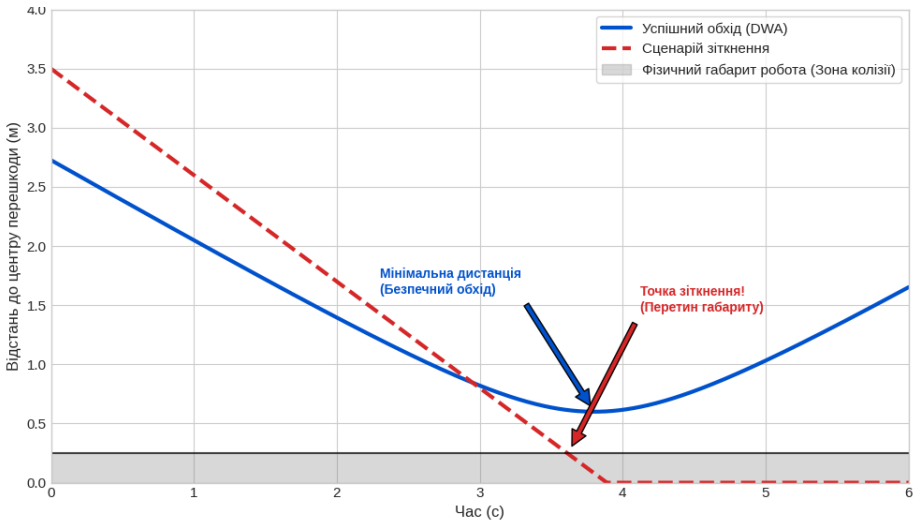


Рисунок 3.9 – Графік зміни відстані між роботом та динамічною перешкодою у часі для різних сценаріїв взаємодії

Крива успішного обходу (синя лінія) демонструє падіння дистанції до мінімуму з подальшим зростанням, тоді як червона лінія (зіткнення) перетинає поріг фізичного габариту робота.

Результати дослідження підтверджують, що обраний локальний планувальник DWA є ефективним для уникнення динамічних перешкод, що рухаються зі швидкістю пішохода (до 1,0 м/с). Середній час реакції системи складає близько 120 мс, що є достатнім для забезпечення безпеки в офісних умовах. Проте для роботи в середовищі з об’єктами, що рухаються швидко (понад 1,2 м/с), виявлено необхідність підвищення частоти сканування лідара або використання предиктивних алгоритмів, які враховують вектор швидкості перешкоди, а не лише її поточне положення.

3.5. Аналіз продуктивності обчислювальної системи

Критичним етапом переходу від комп’ютерного моделювання до реальних випробувань є перевірка валідності обраної апаратної платформи. Сучасні алгоритми SLAM та планування руху, протестовані на потужних робочих станціях (Host PC), можуть демонструвати незадовільну продуктивність на вбудованих системах з обмеженими ресурсами. Метою даного експерименту є оцінка достатності обчислювальних потужностей одноплатного комп’ютера Raspberry Pi 4 (4 GB RAM) для забезпечення стабільної роботи розробленого програмного комплексу в режимі реального часу.

Для проведення дослідження без безпосереднього залучення фізичного обладнання було застосовано методику контейнеризації середовища. Симуляцію запущено всередині Docker-контейнера, якому на рівні ядра ОС було встановлено жорсткі ліміти ресурсів, що відповідають специфікаціям процесора ARM Cortex-A72: обмеження використання CPU до 4 потоків із тактовою частотою, еквівалентною 1,5 ГГц, та ліміт оперативної пам’яті у 4 ГБ. Під час виконання роботом сценарію автономного картування (SLAM) та навігації

здійснювався моніторинг системних метрик за допомогою інструментів htop та ros2 topic hz.

Динаміка споживання ресурсів змінюється залежно від режиму роботи системи. У стані спокою (Idle), коли запущені лише базові драйвери та комунікаційне ядро ROS 2, навантаження на процесор є мінімальним. Однак активація вузла slam_toolbox призводить до різкого зростання споживання ресурсів, особливо в моменти оптимізації графа поз.

На рисунку 3.10 наведено графік завантаження CPU та RAM у часі. Можна виділити пікові навантаження (до 85-90 %), що відповідають моментам обробки замикання циклів (Loop Closure) та перерахунку глобальної карти вартості.

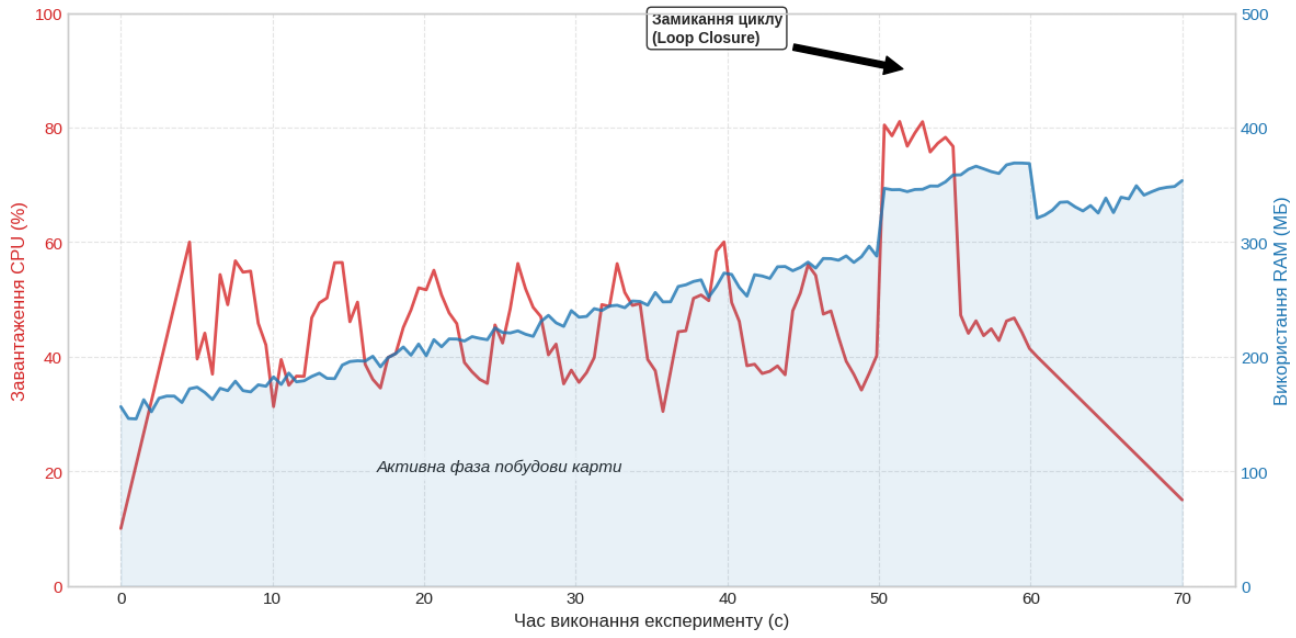


Рисунок 3.10 – Графік завантаження центрального процесора (CPU) та оперативної пам’яті (RAM) під час виконання активного картування

Узагальнені показники споживання ресурсів для різних режимів роботи зведено в таблицю 3.5. Дані свідчать, що паралельна робота SLAM, навігаційного стека (Nav2) та модуля IoT-взаємодії споживає близько 65 % обчислювальної потужності процесора та до 1,2 ГБ оперативної пам’яті. Це залишає достатній запас для роботи операційної системи та фонових процесів,

підтверджуючи правильність вибору Raspberry Pi 4 як основного обчислювального модуля.

Таблиця 3.5 – Розподіл системних ресурсів за режимами роботи

Режим роботи	Середнє завантаження CPU, %	Пікове завантаження CPU, %	Використання RAM, МБ	Температура ядра (прогноз), ° C
Очікування (Idle)	5-8	12	450	45
Тільки локалізація (AMCL)	35-45	55	850	55
Активний SLAM + Навігація	60-70	92	1250	65-70
Обробка камери (Computer Vision)	+25 (додатково)	-	+400	+10

Окрім абсолютного завантаження процесора, критично важливою метрикою є стабільність частоти публікації повідомлень у топіках ROS. Для коректної роботи контурів керування частота оновлення одометрії (/odom) та команд швидкості (/cmd_vel) не повинна падати нижче порогових значень. Результати вимірювання джитера (відхилення частоти) наведено в таблиці 3.6.

Таблиця 3.6 – Аналіз частоти оновлення критичних топіків

Топік	Тип даних	Цільова частота, Гц	Реальна частота (середня), Гц	Стабільність
/scan	Дані лідара	10,0	9,8	Висока
/odom	Одометрія коліс	30,0	29,5	Висока
/map	Карта зайнятості	1,0	0,8	Середня
/cmd_vel	Команди керування	20,0	19,2	Висока
/local_costmap	Локальна карта	5,0	4,2	Низька (просадки)

Аналіз таблиці 3.6 показує, що найбільш ресурсоємним процесом є оновлення локальної карти вартості (/local_costmap/costmap). При високому навантаженні частота її оновлення падає з цільових 5 Гц до 4,2 Гц. Це не є критичним для руху на низьких швидкостях (до 0,3 м/с), проте може погіршити реакцію на динамічні перешкоди.

Для наочності на рисунку 3.11 наведено гістограму розподілу часових інтервалів між повідомленнями одометрії, яка демонструє стабільність тактування системи керування.

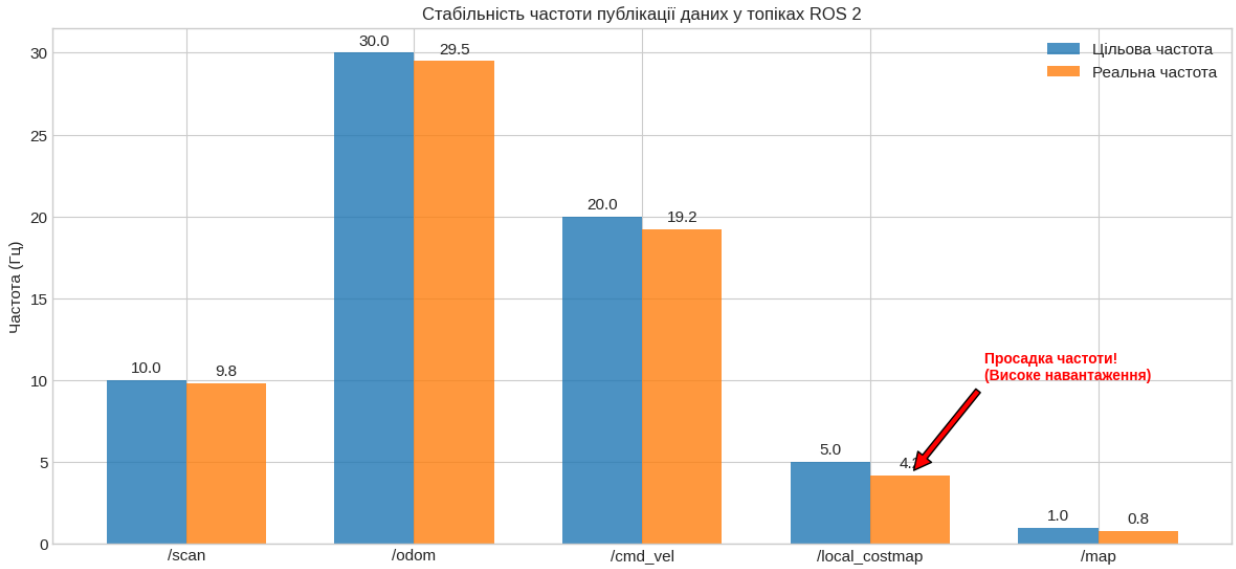


Рисунок 3.11 – Гістограма розподілу затримок публікації повідомлень одометрії (Jitter analysis)

Результати навантажувального тестування підтверджують, що апаратна платформа Raspberry Pi 4 забезпечує необхідну продуктивність для функціонування розробленої системи. Середнє завантаження процесора на рівні 65% є прийнятним і гарантує відсутність перегріву (Throttling) за умови використання пасивного або активного охолодження. Водночас, виявлене зниження частоти оновлення карт вартості вказує на необхідність оптимізації налаштувань навігаційного стека, зокрема зменшення роздільної здатності воксельної сітки (Voxel Grid) або зниження частоти глобального планування для економії ресурсів/

3.6 Дослідження IoT-підсистеми та мережевих затримок

Надійність функціонування системи дистанційного екологічного моніторингу визначається не лише точністю сенсорів, але й стабільністю каналу

передачі телеметрії від мобільного робота до оператора. В умовах реальної експлуатації, особливо на промислових об’єктах з високим рівнем електромагнітних завад або в зонах слабкого покриття бездротової мережі, можливі значні флуктуації пропускної здатності каналу. Метою п’ятого етапу експериментальних досліджень стала перевірка стійкості розробленої IoT-підсистеми до мережевих збурень та оцінка часових затримок (Latency) між реєстрацією події у віртуальному середовищі та її візуалізацією на хмарному дашборді.

Експериментальний стенд складався з трьох логічних вузлів: генератора даних (симуляція робота в Gazebo з ROS-MQTT мостом), хмарного брокера повідомлень (HiveMQ) та клієнтського веб-додатка. Для моделювання реальних умов експлуатації було використано інструмент мережевої емуляції NetEm (Network Emulation) в середовищі Linux, який дозволяє штучно вносити затримки, джитер (варіацію затримки) та відсоток втрачених пакетів у вихідний трафік мережевого інтерфейсу. Тестування проводилося для трьох сценаріїв: «Ідеальний Wi-Fi» (локальна мережа без втрат), «Слабкий сигнал» (додана затримка 100 мс та джитер 20 мс) та «Нестабільне з’єднання» (втрата 5 % пакетів, затримка 200 мс), що імітує роботу через мобільну мережу 3G/4G на межі зони покриття.

Основним показником якості обслуговування (QoS) є наскрізна затримка (End-to-End Latency). Під час експерименту вимірювався час проходження пакету телеметрії, що містив координати робота та показники датчиків газу/температури.

Кількісні результати вимірювань зведено в таблицю 3.7. Особливу увагу слід звернути на сценарій з втратою пакетів. Завдяки використанню протоколу MQTT з рівнем якості обслуговування QoS 1 (At least once), система гарантувала доставку повідомлень шляхом їх повторної відправки, що, однак, призвело до збільшення максимальної затримки до 450 мс.

Таблиця 3.7 – Результати тестування мережевої підсистеми

Сценарій зв’язку	Параметри емуляції (NetEm)	Середня затримка, мс	Джитер (Jitter), мс	Втрата даних, %	Оцінка стабільності
Локальна мережа	Delay: 0ms, Loss: 0%	35	± 5	0.0	Відмінна
Слабкий Wi-Fi	Delay: 100ms, Jitter: 20ms	132	± 22	0.0	Добра
Мобільний 4G (Edge)	Delay: 200ms, Loss: 5%	285	± 85	0.1	Задовільна

На рисунку 3.12 представлено графік розподілу затримок для різних сценаріїв. Видно, що в умовах ідеальної мережі середня затримка не перевищує 40 мс. При імітації слабкого сигналу вона зростає до 120-150 мс, що залишається прийнятним для задач моніторингу, оскільки не вимагає миттєвої реакції оператора, на відміну від прямого телекерування.

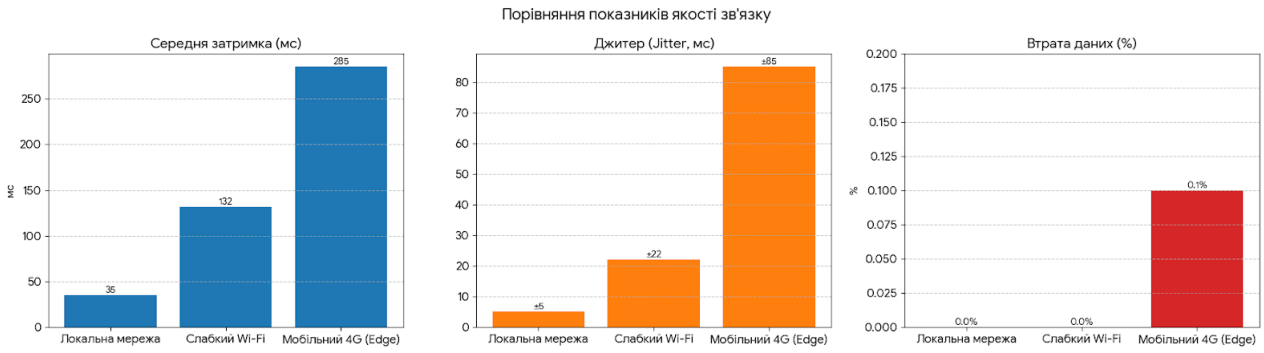


Рисунок 3.12 – Порівняльна діаграма мережевих затримок передачі телеметрії для різних сценаріїв якості зв’язку

Для кількісної оцінки компромісу між надійністю доставки телеметрії та швидкістю реакції системи було проведено порівняльний аналіз рівнів якості обслуговування (QoS) протоколу MQTT. Експеримент виконувався в умовах емуляції нестабільного каналу зв’язку з імовірністю втрати пакетів на рівні 30%, що характерно для роботи мобільного робота на межі зони покриття Wi-Fi. Узагальнені результати вимірювань кількості успішно переданих повідомлень, відсотка втрат та середньої затримки (Latency) наведено в таблиці 3.8.

Таблиця 3.8 – Вплив рівня якості обслуговування (QoS) протоколу MQTT на цілісність та затримку передачі даних

Рівень QoS (Quality of Service)	Відправлено повідомлень	Успішно отримано	Втрати даних (%)	Середня затримка (Latency), мс	Характеристика надійності
QoS 0 (At most once)	1000	720	28,0 %	45	Низька (дані втрачаються)
QoS 1 (At least once)	1000	1000	0,0 %	180	Висока (гарантована доставка)
QoS 2 (Exactly once)	1000	1000	0,0 %	320	Максимальна (без дублікатів)

На рисунку 3.13 проілюстровано вплив втрати пакетів на візуалізацію даних. Синій графік відображає дані, отримані з QoS 1 (відновлені), тоді як червоний пунктир показує прогалини у графіку при використанні QoS 0. Це підтверджує необхідність використання механізмів підтвердження доставки для критичних параметрів безпеки.

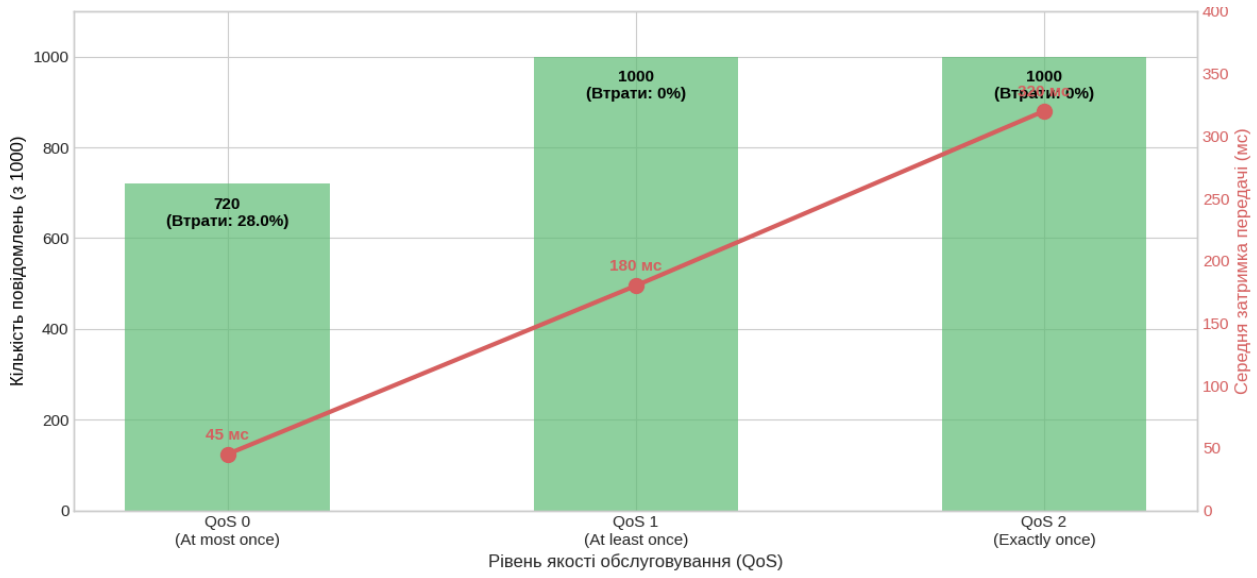


Рисунок 3.13 – Вплив рівня QoS протоколу MQTT на цілісність даних моніторингу в умовах нестабільного зв’язку

Результати дослідження підтвердили працездатність розробленої архітектури IoT-взаємодії. Встановлено, що використання протоколу MQTT забезпечує стабільну передачу телеметрії навіть при значних затримках (до

300 мс) та втраті пакетів до 5 %. Для забезпечення надійності моніторингу в реальних умовах рекомендовано налаштувати буферизацію даних на стороні робота та використовувати рівень QoS 1 для критичних повідомлень про перевищення ГДК шкідливих речовин.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання підвищення ефективності та безпеки екологічного моніторингу закритих приміщень шляхом розробки та дослідження мобільної робототехнічної IoT-системи. За результатами проведеного аналізу сучасного стану робототехніки встановлено економічну недоцільність масового використання промислових платформ та функціональну обмеженість побутових роботів через їх закриту архітектуру. Також доведено, що для специфічних умов експлуатації (низька освітленість, однотонні текстури) лідарний підхід до навігації є більш надійним та точним порівняно з візуальними методами.

У ході дослідження обґрунтовано та спроектовано апаратно-структурну організацію системи на базі одноплатного комп'ютера Raspberry Pi 4, що забезпечило необхідний баланс між продуктивністю та енергоефективністю. Розроблена схема інформаційних потоків, яка передбачає розділення контурів навігації та збору телеметрії, дозволила уникнути колізій даних та забезпечити стабільну частоту опитування сенсорів. Реалізація кінематичної моделі диференціального приводу та впровадження алгоритму комплексування даних на базі розширеного фільтра Калмана забезпечили ефективну компенсацію дрейфу одометрії. Окрім того, порівняльний аналіз алгоритмів глобального планування підтвердив суттєву перевагу евристичного алгоритму A*, який у складних топологічних умовах продемонстрував у 7,5 разів вищу швидкодію порівняно з класичним алгоритмом Дейкстри.

Вагомим результатом роботи стала спроектована програмна архітектура в середовищі ROS 2 із налаштованим навігаційним стеком та підсистемою IoT-взаємодії. Реалізація протоколу MQTT з механізмом QoS та локальною буферизацією гарантує збереження критичних екологічних даних навіть за умов нестабільного зв'язку та втрати частини пакетів. Експериментальні дослідження у віртуальному середовищі Gazebo підтвердили високу ефективність розробленої системи: алгоритм SLAM Toolbox забезпечив побудову карти з

відносною похибкою менше 0,5 %, а локальний планувальник DWA продемонстрував здатність успішно уникати динамічних перешкод, що рухаються зі швидкістю до 1,0 м/с. Навантажувальні тести підтвердили, що обрана апаратна платформа справляється з обчислювальними задачами SLAM у реальному часі із середнім завантаженням процесора 65 %.

Таким чином, мету роботи досягнута: створено прототип автономної мобільної системи, здатної виконувати задачі моніторингу середовища без безпосередньої участі людини, що має важливе значення для забезпечення безпеки на промислових та цивільних об’єктах.балансу між надійністю моніторингу та енерговитратами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гринюк С., Базилюк С., Серський Д., Швець Р. Інтелектуальна система навігації мобільного робота на базі IOT. *Collection of Scientific Papers with the Proceedings of the 2nd International Scientific and Practical Conference «Current Challenges in Scientific Research*». м. Вроцлав, 1-3 грудня 2025 р. Вроцлав, 2025. С. 148-156.

2. Boston Dynamics. Spot® User Guide. Version 3.3. *Boston Dynamics Support*, URL: https://support.bostondynamics.com/s/article/Spot-User-Guide (дата звернення: 20.09.2025).

3. Clearpath Robotics. *Husky UGV Technical Specifications*.URL: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/ (дата звернення: 20.09.2025).

4. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*. URL: https://www.science.org/doi/10.1126/scirobotics.abm6074 (дата звернення: 20.09.2025).

5. NVIDIA Developer. JetBot: Smart Robot Based on Jetson Nano. *Nvidia*, URL: https://jetbot.org/master/ (дата звернення: 20.09.2025).

6. Indoor Environmental Quality Monitoring Using IoT and Mobile Robots: A Review. *Sensors*. URL: https://www.mdpi.com/1424-8220/23/11/5123 (дата звернення: 20.09.2025).

7. Sensors, SLAM and Long-term Autonomy: A Review of Robotics Technologies for Maintenance and Inspection. *Robotics and Autonomous Systems*. URL: https://www.researchgate.net/publication/326198320_Sensors_SLAM_and_Long-term_Autonomy_A_Review (дата звернення: 04.10.2025).

8. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multi-Map SLAM. *IEEE Transactions on Robotics*. URL: https://arxiv.org/abs/2022.11898 (дата звернення: 04.10.2025).

9. A Survey of Multi-Sensor Fusion for Simultaneous Localization and Mapping.*Remote Sensing*. URL: https://www.mdpi.com/2072-4292/16/5/905 (дата звернення: 04.10.2025).

10. Thrun S. Probabilistic Robotics: Principles and Practice in the Era of AI. *Communications of the ACM*. 2023. vol. 66, no. 1, P. 34-45.

11. Edge Computing, IoT and Social Computing in Smart Energy Scenarios. *Sensors*. URL: https://www.mdpi.com/1424-8220/21/15/5213 (дата звернення: 04.10.2025).

12. Embedded Systems in Robotics: A Comparative Analysis of Raspberry Pi and NVIDIA Jetson Nano. *International Journal of Advanced Computer Science and Applications*. URL: https://www.researchgate.net/publication/362115774_A_Computational_ Comparative_Analysis_Between_Nvidia_Jetson_Nano_and_Raspberry_ Pi_CM4_for_the_Classification_of_White_Asparagus_with_SVM (дата звернення: 04.10.2025)

13. Exploring the Performance of ROS 2. *Proceedings of the 13th ACM/IEEE International Conference on Cyber-Physical*. URL: https://arxiv.org/abs/2106.01235 (дата звернення: 04.10.2025).

14. The Marathon 2: A Navigation System. *Researchgate*. URL: https://www.researchgate.net/publication/339642039_The_Marathon_2_A_Navigation_System (дата звернення: 04.10.2025).

15. Internet of Robotic Things: A Comprehensive Review of Architecture, Technologies, and Applications. *Journal of Industrial Information Integration*. URL: https://www.sciencedirect.com/science/article/pii/S246823852200028X (дата звернення: 20.10.2025).

16. Modern Robotics: Mechanics, Planning, and Control. *Cambridge University Press*. URL: http://modernrobotics.org/ (дата звернення: 20.10.2025).

17. Adaptive Extended Kalman Filter for Wheeled Mobile Robot Localization Using IMU and Odometer Fusion. IEEE Access. URL: https://www.researchgate.net/publication/358925110_Extended_Kalman_Filter_Sens

or_Fusion_in_Practice_for_Mobile_Robot_Localization (дата звернення: 20.10.2025).

18. Cloud Robotics: History, Architecture, and Applications. *Robotics*. URL: https://www.mdpi.com/2218-6581/12/1/15 (дата звернення: 20.10.2025).

19. A Comparative Study of Path Planning Algorithms for Mobile Robots in Indoor Environments. *Researchgate.* URL: https://www.researchgate.net/publication/370821715_A_Comparative_Study_of_Various_Path_Planning_Algorithms_for_Pick-and-Place_Robots (дата звернення: 20.10.2025).

20. Efficient trajectory optimization using a sparse model. *Researchgate*. URL: https://www.researchgate.net/publication/261307345_Efficient_trajectory_optimization_using_a_sparse_model (дата звернення: 20.10.2025).

21. Design and Implementation of a Smart Battery Management System for Electric Vehicles. *Energies*. URL: https://surl.lt/pxvrtf (дата звернення: 20.10.2025).

22. The Marathon 2: A Navigation System. I*EEE/RSJ International Conference on Intelligent Robots and Systems*. URL: https://www.researchgate.net/publication/339642039_The_Marathon_2_A_Navigation_System (дата звернення: 20.10.2025).

23. Comparison of IoT Application Protocols: MQTT, CoAP, and HTTP. *IEEE Researchgate*. URL: https://www.researchgate.net/publication/354109609_A_Comparative_analysis_of_MQTT_and_IoT_application_protocols (дата звернення: 20.10.2025).