КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри радіотехніки та радіоелектронних систем
від _____ 2026 року, протокол № ___.
В.о. завідувача кафедри кандидат фіз.-мат. наук,
доцент _____ Ігор БЕХ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
на тему:
«РОБОТИЗОВАНА ТРАНСПОРТНА ПЛАТФОРМА З АВТОНОМНОЮ НАВІГАЦІЄЮ»

Виконав:
студент 4-го курсу
денної форми здобуття освіти
спеціальності 172 Телекомунікації та радіотехніка
ОП Інформаційна безпека телекомунікаційних систем і мереж
Сюпюр Данило Сергійович _____

Науковий керівник:
кандидат фіз..-мат. наук, асистент
Богданов Роман Вікторович _____

Рецензент:
кандидат фіз.-мат. наук, доцент
кафедри комп'ютерної інженерії
Загороднюк Сергій Петрович _____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань

Студент _____Данило СЮПЮР

Київ – 2026

РЕФЕРАТ

Кваліфікаційна робота: 60 с., 13 табл., 17 рис., 26 джерел.

Ключові слова: МОБІЛЬНА РОБОТИЗОВАНА ПЛАТФОРМА, АВТОНОМНА НАВІГАЦІЯ, ESP32, GOTO, RSSI, TOF-ДАТЧИК, IMU, ВЕБІНТЕРФЕЙС.

Об'єкт дослідження – процес локальної навігації мобільної роботизованої платформи в середовищі з нескладними перешкодами.

Мета роботи – розробити роботизовану транспортну платформу з дистанційним керуванням і базовими функціями автономної навігації, достатніми для руху до заданої координати, виявлення перешкод та побудови простої карти оточення.

У роботі проаналізовано сучасні мобільні роботизовані платформи, обґрунтовано вибір апаратної архітектури, виконано перехід від Arduino Uno до ESP32, реалізовано ручний, автоматичний та координатний режими керування, а також вебінтерфейс оператора для налаштування й моніторингу роботи системи.

Програмна частина забезпечує оброблення показів енкодерів, IMU та далекомірів, визначення положення робота на дискретній карті, позначення перешкод і формування маршруту в режимі GOTO. Додатково реалізовано режим PHONE, у якому рівень бездротового сигналу RSSI використовується як допоміжна ознака для пошуку цільового пристрою.

Експериментальна перевірка показала, що розроблена система придатна для базового автономного керування в структурованому середовищі з нескладними перешкодами. Водночас встановлено, що запропонований підхід не є повноцінною навігаційною системою рівня SLAM і має природні обмеження щодо точності позиціювання, бокового контролю та роботи у складному середовищі.

ЗМІСТ

ВСТУП 5

1. АНАЛІЗ МОБІЛЬНИХ РОБОТИЗОВАНИХ ПЛАТФОРМ ТА МЕТОДІВ ЇХ НАВІГАЦІЇ 7

1.1 Історичний розвиток мобільних роботизованих систем 7

1.2 Актуальність наземних роботизованих комплексів для України .. 9

1.3 Класифікація мобільних роботизованих платформ 11

1.4 Сенсорні системи мобільних роботів 13

1.5 Методи навігації та уникнення перешкод 15

1.6 Використання рівня бездротового сигналу для пошуку цільового пристрою 17

1.7 Вимоги до розроблюваної мобільної платформи 18

2. ПОЧАТКОВА МОДЕЛЬ МОБІЛЬНОГО РОБОТА ТА АПАРАТНА МОДЕРНІЗАЦІЯ ПЛАТФОРМИ 19

2.1 Початкова конструкція мобільного робота 19

2.2 Режими роботи початкової моделі 20

2.3 Виявлені проблеми початкової конструкції 21

2.4 Проміжні рішення для покращення виявлення перешкод 22

2.5 Перехід до лазерних датчиків відстані 23

2.6 Модернізація мікроконтролерної частини 24

2.7 Додавання апаратних засобів для навігації 26

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ ТА НАВІГАЦІЇ МОБІЛЬНОГО РОБОТА 30

3.1 Загальна структура програмного забезпечення 30

3.2 Ініціалізація та обробка апаратних вузлів 31

3.3 Вебінтерфейс і обмін даними з оператором 32

3.4 Ручний та автономний режими 35

3.5 Координатний режим GOTO, карта і побудова перешкод 36

3.6 Режим PHONE: пошук пристрою за RSSI 38

3.7 Засоби безпеки та діагностики 39

3.8 Обмеження реалізованої програмної системи 40
4. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОБОТИ МОБІЛЬНОЇ РОБОТИЗОВАНОЇ ПЛАТФОРМИ 42
4.1 Методика проведення експериментальних досліджень 42
4.2 Дослідження точності прибуття в режимі GOTO 43
4.3 Дослідження роботи системи виявлення перешкод 44
4.4 Дослідження об'їзду перешкод у режимі GOTO 45
4.5 Дослідження об'їзду декількох перешкод 45
4.6 Дослідження автоматичного режиму AUTO 46
4.7 Дослідження впливу відстані та перешкод на рівень RSSI 47
4.8 Дослідження впливу орієнтації робота на RSSI 47
ВИСНОВКИ 49
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 51
ДОДАТОК А 53
ДОДАТОК Б 55

ВСТУП

Актуальність теми зумовлена зростанням ролі мобільних роботизованих платформ у цивільних, промислових і військових застосуваннях. Сучасні наземні роботи використовуються для транспортування вантажів, дистанційного обстеження небезпечних зон, інженерної розвідки, гуманітарного розмінування та виконання допоміжних логістичних операцій. Для таких систем важливими є не лише механічна надійність, а й доступні засоби локальної навігації, що дають змогу поєднати ручне керування з елементами автономної поведінки.

Для навчально-дослідних платформ особливе значення мають простота реалізації, відтворюваність апаратних рішень і зрозумілі алгоритми керування. У таких умовах доцільно будувати систему не як складний високорівневий навігаційний комплекс, а як практичну платформу з базовою автономністю, у якій оператор може задавати напрям руху, цільову координату або режим роботи, а робот самостійно виконує локальні дії: вимірює відстані, контролює положення, фіксує перешкоди та коригує рух.

Метою роботи є розробка роботизованої транспортної платформи з автономною навігацією, яка поєднує дистанційне керування, рух до заданої координати, виявлення перешкод, побудову простої карти середовища та засоби моніторингу через вебінтерфейс.

Для досягнення поставленої мети у роботі вирішено такі завдання: виконано аналіз мобільних роботизованих платформ і методів їх навігації; розглянуто початкову конструкцію робота та виявлено її обмеження; обґрунтовано модернізацію апаратної частини із переходом на ESP32; реалізовано програмну систему ручного, автоматичного та координатного керування; розроблено механізм побудови карти та відображення перешкод; проведено експериментальну перевірку точності руху й працездатності основних режимів.

Об'єктом дослідження є процес переміщення мобільної роботизованої платформи у середовищі з нескладними перешкодами. Предметом

дослідження є апаратні та програмні засоби реалізації базової автономної навігації, локального позиціювання, виявлення перешкод і дистанційної взаємодії з оператором.

Практичне значення одержаних результатів полягає у створенні працездатної навчально-дослідної платформи, на якій відпрацьовано типові задачі мобільної робототехніки: керування приводами, роботу сенсорної системи, оцінювання положення за енкодерами й IMU, формування локальної карти та організацію операторського вебінтерфейсу. Розроблена система не претендує на точність і універсальність складних навігаційних комплексів на кшталт SLAM, однак є достатньою для базового автономного керування в добре контрольованому середовищі.

Структурно робота складається з чотирьох розділів. У першому розділі розглянуто сучасні мобільні роботизовані платформи, сенсорні системи та методи навігації. Другий розділ присвячено аналізу початкової моделі робота та модернізації апаратної частини. У третьому розділі наведено програмну реалізацію системи керування, побудови карти та режимів навігації. У четвертому розділі подано результати експериментальної перевірки роботи розробленої платформи.

1. АНАЛІЗ МОБІЛЬНИХ РОБОТИЗОВАНИХ ПЛАТФОРМ ТА МЕТОДІВ ЇХ НАВІГАЦІЇ

1.1 Історичний розвиток мобільних роботизованих систем

Мобільна роботизована платформа є технічною системою, що здатна переміщуватися в середовищі, сприймати його за допомогою датчиків і формувати керувальні дії для виконавчих механізмів. На відміну від стаціонарних роботів, така платформа повинна одночасно розв'язувати задачі руху, оцінювання власного положення, контролю напряму та реакції на перешкоди. Саме тому в мобільній робототехніці поєднуються механічна конструкція, сенсорика, мікроконтролерне керування та алгоритм навігації [1].

Одними з перших відомих мобільних роботів, що демонстрували автономну поведінку, стали електронні «черепахи» Вільяма Ґрея Волтера Elmer і Elsie, створені наприкінці 1940-х років. Вони мали прості світлочутливі сенсори й могли рухатися до джерела світла, змінювати траєкторію та уникати зіткнень [2]. Зовнішній вигляд одного з таких роботів наведено на рисунку 1.1.

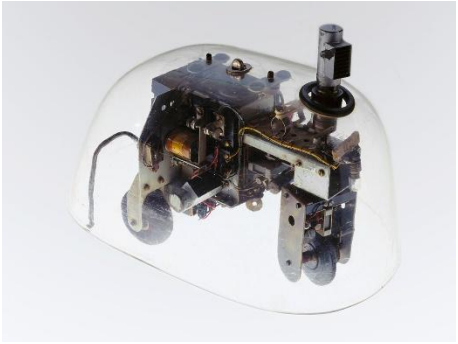

Рисунок 1.1 – Кібернетична «черепаха» В. Грея Волтера [2]

Ці роботи важливі не лише як історичний експеримент, а й як ранній приклад принципу, який залишається актуальним для мобільних платформ: поведінка машини виникає внаслідок взаємодії сенсорів, електронної схеми та двигунів із середовищем.

Наступним важливим етапом у розвитку мобільної робототехніки стали експерименти з платформою Stanford Cart (рис. 1.2), які проводилися в

Лабораторії штучного інтелекту Стенфордського університету. Спочатку її розглядали як дослідний візок для майбутнього місяцехода, а згодом використали для перевірки алгоритмів комп'ютерного бачення та планування руху. Одна з пізніших версій могла перетинати кімнату з перешкодами: робот проїжджав коротку ділянку, зупинявся, аналізував зображення з телевізійної камери і лише потім обчислював наступний фрагмент маршруту [3].

Рисунок 1.2 – Мобільна платформа Stanford Cart [3]

Ще одним важливим етапом у розвитку мобільної робототехніки став робот Shakey, розроблений у Стенфордському дослідницькому інституті (рис.1.3). Це був один із перших мобільних роботів, який не лише рухався, а й сприймав середовище, формував послідовність дій і виконував завдання, пов'язані з плануванням маршруту та взаємодією з об'єктами [4].

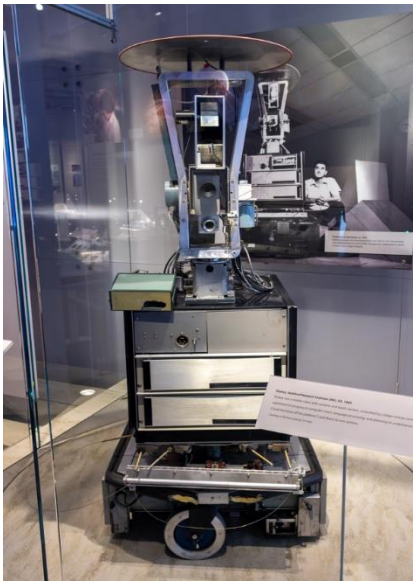

Рисунок 1.3 –Мобільний робот Shakey[5]

Розвиток мобільних роботів був пов’язаний не лише з науковими лабораторіями, а й із необхідністю виконання робіт у небезпечних для людини умовах. Прикладом є використання дистанційно керованих платформ у зонах техногенних аварій, радіаційного забруднення, завалів та інших середовищах, де безпосередня присутність оператора створює високий ризик для життя. Саме такі задачі продемонстрували практичну цінність мобільних роботів: вони можуть бути відносно простими за конструкцією, але водночас виконувати критично важливі функції у небезпечному середовищі.

У 1980–1990-х роках розвиток мікропроцесорної техніки та доступних мікроконтролерів зробив можливим створення недорогих навчальних і дослідницьких платформ. Такі роботи могли слідувати за лінією, виявляти перешкоди, рухатися за командами користувача або працювати в простому автономному режимі. Подібний підхід залишається актуальним і сьогодні, оскільки навчальна мобільна платформа дозволяє практично відпрацювати принципи сенсорного вимірювання, керування двигунами, побудови алгоритмів руху та аналізу похибок.

1.2 Актуальність наземних роботизованих комплексів для України

В умовах сучасної України наземні роботизовані комплекси мають не лише навчальне або промислове значення, а й безпосередню практичну актуальність. Повномасштабна війна, мінна небезпека, руйнування інфраструктури та необхідність виконання робіт у небезпечних зонах створюють потребу у системах, які можуть частково замінити людину під час виконання ризикованих завдань. До таких завдань належать розвідка місцевості, доставка вантажів, евакуація поранених, інженерне обстеження, гуманітарне розмінування та моніторинг зруйнованих об’єктів.

У Збройних силах України наземні роботизовані комплекси застосовуються для логістики, евакуації, розвідки та інженерних задач. За повідомленням Міністерства оборони України, у березні 2026 року з

використанням безпілотних наземних платформ було виконано понад 9000 завдань, а з початку року - майже 24500 місій [6]. Ці дані показують, що наземні роботи поступово переходять від окремих експериментальних рішень до практичного інструмента зменшення ризику для особового складу.

Наведені дані доповнюються ширшою функціональною структурою військового застосування наземних роботизований комплексів (НРК). У бойових умовах такі платформи використовуються як засоби логістики, евакуації поранених, інженерного забезпечення, розвідки, спостереження та, у спеціалізованих варіантах, вогневої підтримки. Найбільш практичним напрямом є доставка боєприпасів, води, спорядження й інших вантажів на передові позиції, оскільки це зменшує необхідність виходу особового складу в зони прямого ураження [7], [8].

Окрему роль відіграють логістично-евакуаційні комплекси, здатні рухатися під керуванням оператора або частково автономно, передавати відеоінформацію та виконувати завдання в умовах обмеженої видимості, складного рельєфу й впливу засобів радіоелектронної боротьби. Для таких систем важливими є стійкий канал зв’язку, можливість дистанційного контролю стану, резервні режими зупинки та зрозуміла взаємодія з оператором. Саме тому навіть навчальна мобільна платформа, що реалізує дистанційне керування, автономну зупинку перед перешкодою та передавання даних оператору, відтворює базові принципи побудови сучасних наземних роботизованих комплексів військового призначення [8].

Окрім оборонної сфери, значну роль такі системи відіграють у роботі рятувальних служб і підрозділів гуманітарного розмінування. Програма розвитку Організації Об’єднаних Націй у 2025 році передала Державній службі України з надзвичайних ситуацій шість дистанційно керованих машин для розмінування, що підкреслює важливість роботизованих платформ для пришвидшення очищення територій та зменшення небезпеки для саперів [9]. Приклад такого комплексу наведено на рисунку 1.4.

Рисунок 1.4 – Роботизований комплекс DIGITAL VANGUARD-S для розмінування, переданий підрозділам ДСНС України [10]

Розроблена в межах цієї бакалаврської роботи платформа розглядається як навчально-дослідний зразок, призначений для перевірки базових принципів побудови мобільних роботизованих систем. У ній реалізовано дистанційне керування, елементи автономного руху, виявлення перешкод, передавання даних оператору та прийняття простих локальних рішень під час переміщення. Тому практична цінність розробки полягає не у виконанні спеціалізованих бойових чи рятувальних задач, а у відпрацюванні підходів, які можуть використовуватися як основа для подальшого розвитку подібних роботизованих платформ.

1.3 Класифікація мобільних роботизованих платформ

Мобільні роботизовані платформи можна класифікувати за типом ходової частини, способом керування, рівнем автономності та призначенням. За типом ходової частини поширеними є колісні, гусеничні, крокуючі та комбіновані системи. Колісні роботи є конструктивно простими, енергоефективними та зручними для навчальних і лабораторних проєктів. Гусеничні платформи краще долають нерівності, сходинки, уламки та м’який

ґрунт, тому часто застосовуються у військових, рятувальних та інженерних задачах. Приклад гусеничного наземного робота PackBot наведено на рисунку 1.5.

Рисунок 1.5 – Гусеничний наземний робот PackBot, який використовувався для пошуково-рятувальних задач [11]

За способом керування мобільні роботи можна поділити на дистанційно керовані, автономні та комбіновані. Дистанційне керування передбачає, що оператор безпосередньо задає команди руху. Автономне керування означає, що робот самостійно приймає рішення на основі сенсорних даних і заданої мети. Комбінований режим поєднує обидва підходи: оператор може втручатися в роботу системи, але частина дій, наприклад об’їзд перешкод або зупинка перед небезпечним об’єктом, виконується автоматично. Порівняння наведено в таблиці 1.1

Таблиця 1.1 – Порівняння основних способів керування мобільними роботами

Спосіб керування	**Переваги**	**Недоліки**	**Приклад застосування**
Дистанційне	Простота реалізації, повний контроль оператора	Залежність від оператора та каналу зв’язку	ІЧ-пульт, веб-керування, телеметрія
Автономне	Можливість роботи без постійних команд	Вища складність алгоритмів і вимоги до датчиків	Рух до точки, об’їзд перешкод

Спосіб керування	Переваги	Недоліки	Приклад застосування
Комбіноване	Поєднання гнучкості оператора та автоматичних захистів	Складніша програмна логіка та більше станів системи	Ручне керування з автоматичною зупинкою перед небезпекою

Для розроблюваної навчально-дослідної платформи найбільш доцільним є комбінований підхід. Він дозволяє поєднати ручне керування для тестування, калібрування та аварійного втручання з автономними функціями, пов’язаними з рухом до цілі, виявленням перешкод і контролем безпечності маршруту. Така структура наближена до практичних роботизованих систем, де оператор задає загальну мету, а локальні рішення щодо безпечного руху робот виконує самостійно.

1.4 Сенсорні системи мобільних роботів

Сенсорна система є однією з базових складових будь-якого мобільного робота, оскільки саме вона забезпечує отримання інформації про зовнішнє середовище та власний стан платформи. Без сенсорного контуру робот не може надійно оцінювати відстані до об'єктів, контролювати напрямок руху, фіксувати зміну орієнтації або коректно реагувати на небезпечні ситуації. Тому якість роботи мобільної системи значною мірою визначається не лише механічною конструкцією, а й складом датчиків, способом їх розміщення та подальшою обробкою вимірювань.

У мобільній робототехніці сенсори зазвичай поділяють на зовнішні та внутрішні. Зовнішні датчики призначені для оцінювання навколишнього середовища: до них належать ультразвукові та інфрачервоні датчики, ToF-сенсори, камери, лідари, контактні вимикачі та інші засоби виявлення перешкод. Внутрішні датчики використовуються для оцінювання власного руху робота і містять енкодери, гіроскопи, акселерометри, магнітометри та сенсори контролю положення виконавчих механізмів.

Для задач навігації в приміщенні особливе значення мають датчики відстані, інерціальні модулі та енкодери коліс. Ультразвукові модулі привабливі простотою та низькою вартістю, однак їхня робота суттєво залежить від форми перешкоди, кута падіння хвилі та властивостей поверхні. ToF-датчики забезпечують швидкі вимірювання на близьких дистанціях і краще підходять для локального контролю простору навколо невеликої платформи. Інерціальні модулі дозволяють оцінювати зміну кута орієнтації, а енкодери - пройдений шлях. Оскільки жоден із цих каналів окремо не є ідеальним, на практиці їх доцільно поєднувати [13]. Узагальнену структуру мобільної роботизованої платформи наведено на рисунку 1.6.

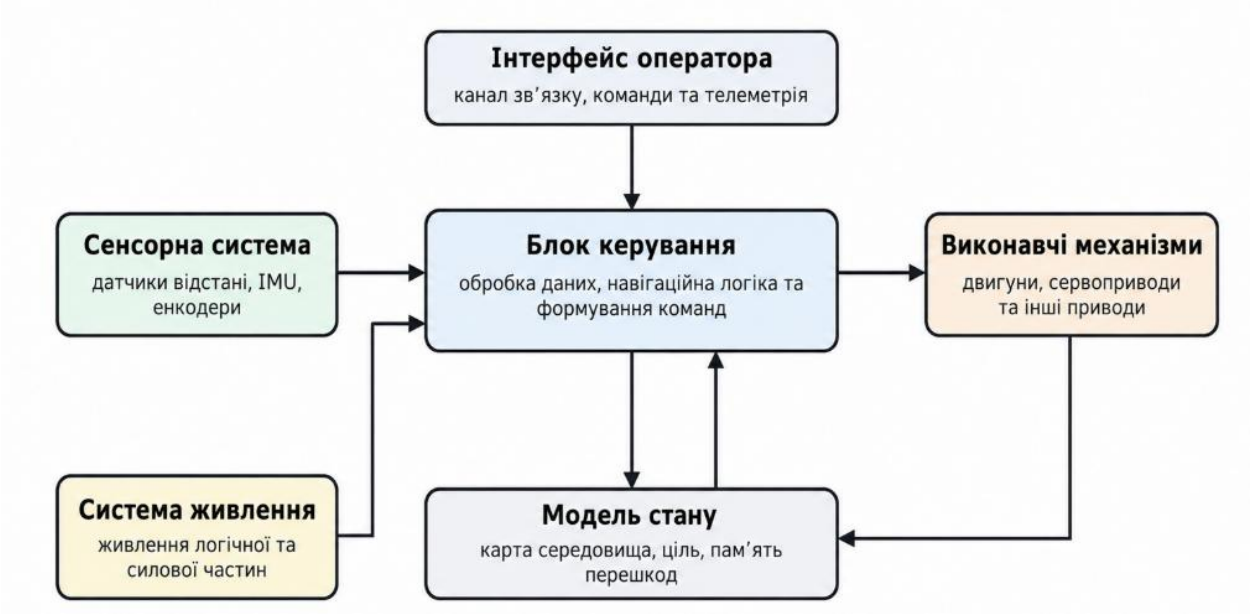

Рисунок 1.6 – Узагальнена структура мобільної роботизованої платформи

Як видно з рисунка 1.6, мобільна платформа поєднує сенсорну підсистему, блок керування, приводи, інтерфейс оператора, модель або карту середовища та систему живлення. У теоретичному сенсі саме взаємодія цих складових формує навігаційні можливості робота: одні вузли сприймають інформацію, інші її обробляють, а виконавча частина реалізує рішення у вигляді руху. Така логіка є спільною для більшості мобільних роботизованих систем, незалежно від їхнього масштабу та призначення.

1.5 Методи навігації та уникнення перешкод

Навігація мобільного робота охоплює визначення поточного стану платформи, вибір напряму руху та виконання траєкторії з урахуванням обмежень середовища. У найпростішому випадку робот діє реактивно: рухається вперед, а після виявлення перешкоди зупиняється, повертає і обирає новий напрямок. Такий підхід є конструктивно простим, однак має очевидні недоліки: він не враховує глобальну мету, може призводити до повторення однакових маневрів і не забезпечує стійкої поведінки в складнішому оточенні.

Складніші системи поєднують локалізацію та планування руху. Локалізація дає відповідь на питання, де приблизно перебуває робот, а планування визначає, яким чином він має рухатися до цілі. У повноцінних навігаційних комплексах для цього можуть застосовуватися карти середовища, алгоритми пошуку маршруту, оцінка ризику зіткнення та повторне планування під час зміни обстановки [14]. Для навчальних і малогабаритних платформ така архітектура часто спрощується, однак базова логіка залишається тією самою: робот повинен враховувати ціль, власне положення, орієнтацію та обмеження, накладені перешкодами.

Для колісного робота з диференціальним приводом одним із найпоширеніших способів оцінювання руху є одометрія. У загальному випадку, якщо ліве і праве колеса пройшли відстані sL та sR, а відстань між колесами становить b, то поступальне переміщення та зміну кута орієнтації можна наближено оцінити за виразом

$$s = (sR + sL)/2\,, \Delta\theta = (sR - sL)/b \qquad (1.5)$$

де s – оцінка поступального переміщення робота, sL та sR – переміщення лівого і правого коліс, b – відстань між колесами, $\Delta\theta$ – зміна кута орієнтації.

Наведена модель є базовою теоретичною основою для опису руху диференціальної платформи. Водночас реальна одометрія завжди містить похибку через проковзування коліс, люфти, нерівності поверхні та відмінності

між приводами. Саме тому на практиці одометричні дані часто комбінують з інерціальними вимірюваннями або зовнішніми сенсорами, щоб зменшити накопичення помилки [13].

З точки зору організації руху для малих мобільних роботів особливо доцільними є покрокові або посегментні схеми навігації. У такому разі робот не намагається пройти довгу траєкторію одним рухом, а періодично уточнює напрямок, перевіряє показання датчиків і за потреби коригує подальші дії. Подібний підхід є компромісом між простими реактивними алгоритмами та складними системами глобального планування.

Уникнення перешкод також може будуватися по-різному: від простих реактивних правил до локального планування руху. Одним із відомих підходів до локального планування руху є метод динамічного вікна (Dynamic Window Approach), запропонований Фоксом, Бургардом і Труном. У цьому методі команда руху обирається з урахуванням кінематики робота, доступного діапазону швидкостей, відстані до перешкод і напряму до цілі. [15]. Для невеликих навчальних платформ повна реалізація подібних методів часто є надмірною, однак їхня загальна ідея принципова: безпечний рух повинен враховувати не лише факт наявності перешкоди, а й можливість зупинки, обходу та продовження маршруту.

Особливо складними для невеликих мобільних роботів є вузькі та нерегулярні перешкоди: ніжки стільців, дроти, краї коробок, елементи меблів і предмети складної форми. Саме в таких випадках проявляються обмеження конкретної сенсорної конфігурації, кута огляду датчиків та похибок локалізації. Тому для практичної системи навігації важливо не лише вимірювати відстань у поточний момент, а й підтримувати хоча б спрощене уявлення про вже виявлені небезпечні зони. Це створює теоретичне підґрунтя для використання локальної карти або пам'яті перешкод у наступних розділах.

1.6 Використання рівня бездротового сигналу для пошуку цільового пристрою

Окремою задачею мобільної навігації може бути пошук цільового пристрою за рівнем бездротового сигналу. У такому випадку робот не має точних координат цілі, але може оцінювати зміну рівня сигналу під час переміщення. Для цього використовується показник RSSI (Received Signal Strength Indicator), який характеризує рівень прийнятого бездротового сигналу. Зазвичай RSSI вимірюється в дБм і має від'ємні значення: чим ближче значення до нуля, тим сильнішим є прийнятий сигнал.

Одним із відомих прикладів використання радіосигналу для локалізації в приміщенні є система RADAR, запропонована П. Бахлом і В. Н. Падманабханом [16]. У складніших системах RSSI може застосовуватися для приблизного визначення положення об'єкта або зони його розташування. Для навчально-дослідної мобільної платформи такий підхід доцільно спрощувати: не визначати точні координати пристрою, а використовувати зміну RSSI як допоміжну ознаку наближення до джерела сигналу.

Перевагою такого підходу є можливість використання вже наявних бездротових засобів мікроконтролера та телефона без додавання окремих модулів локалізації. Водночас RSSI не є стабільним показником відстані. У приміщенні рівень сигналу залежить не лише від віддалення між пристроями, а й від відбиттів від стін і меблів, екранування корпусом робота або людиною, орієнтації антени та наявності перешкод між пристроями.

Тому пошук за RSSI не слід розглядати як високоточний метод локалізації. У межах цієї роботи він використовується як додаткова функція платформи, що демонструє можливість застосування рівня бездротового сигналу для наближеного пошуку цільового пристрою. Для безпечної роботи такий режим має поєднуватися з виявленням перешкод і загальною логікою зупинки робота.

1.7 Вимоги до розроблюваної мобільної платформи

На основі проведеного аналізу можна сформулювати узагальнені вимоги до навчально-дослідної мобільної платформи. Вона повинна мати достатню рухомість, забезпечувати сприйняття середовища за допомогою комбінації сенсорів, підтримувати як ручне, так і автономне керування, а також містити засоби базової безпеки та можливість подальшого розширення функціональності. Інакше кажучи, така платформа має бути не максимально складною, а достатньо репрезентативною для перевірки типових задач мобільної робототехніки.

Таблиця 1.7 – Вимоги до розроблюваної мобільної платформи

Група вимог	Зміст вимоги	Обґрунтування
Керування	Ручний, автоматичний, координатний GOTO та режим пошуку цільового пристрою	Можливість перевірити різні рівні автономності на одній платформі
Сенсорика	Датчики відстані спереду, з боків і позаду, а також контроль небезпечного краю поверхні	Безпечний рух у приміщенні та захист від зіткнень або падіння
Орієнтація і шлях	Оцінювання переміщення за енкодерами та контроль кута повороту інерціальним датчиком	Зменшення похибки під час руху до заданої координати
Навігація	Покроковий рух, повторна перевірка перешкод і збереження заблокованих клітин карти	Запобігання повторному руху в уже виявлену перешкоду
Інтерфейс	Веб-інтерфейс із картою, телеметрією, командами руху, GOTO, HOME та очищенням карти	Оператор повинен бачити стан робота та швидко втручатися в роботу системи
Безпека	Аварійна зупинка, блокування руху при близькій перешкоді та контроль заднього ходу	Захист користувача, робота й навколишніх об’єктів під час експериментів

Сформульовані вимоги створюють логічний перехід від загального теоретичного огляду до практичної частини дипломної роботи. У наступному розділі розглядається початкова конструкція мобільного робота, її виявлені обмеження та ті апаратні зміни, які були потрібні для наближення платформи до наведених вище узагальнених вимог.

2. ПОЧАТКОВА МОДЕЛЬ МОБІЛЬНОГО РОБОТА ТА АПАРАТНА МОДЕРНІЗАЦІЯ ПЛАТФОРМИ

2.1 Початкова конструкція мобільного робота

Початкова версія мобільного робота була розроблена під час роботи над курсовим проєктом. Вона являла собою колісну мобільну систему на базі Arduino Uno, призначену для ручного керування та простого автономного руху з уникненням перешкод. У конструкції використовувалися готове шасі з колекторними двигунами постійного струму з редукторами, драйвер двигунів L298N, ультразвуковий датчик HC-SR04 на сервоприводі, інфрачервоний приймач для керування з пульта та сенсор контролю краю поверхні.

Загальний вигляд початкової конструкції наведено на рисунку 2.1. На цьому етапі основною задачею було забезпечити працездатність базової мобільної платформи: рух вперед і назад, повороти, зупинку, реакцію на перешкоди та захист від падіння з краю поверхні.

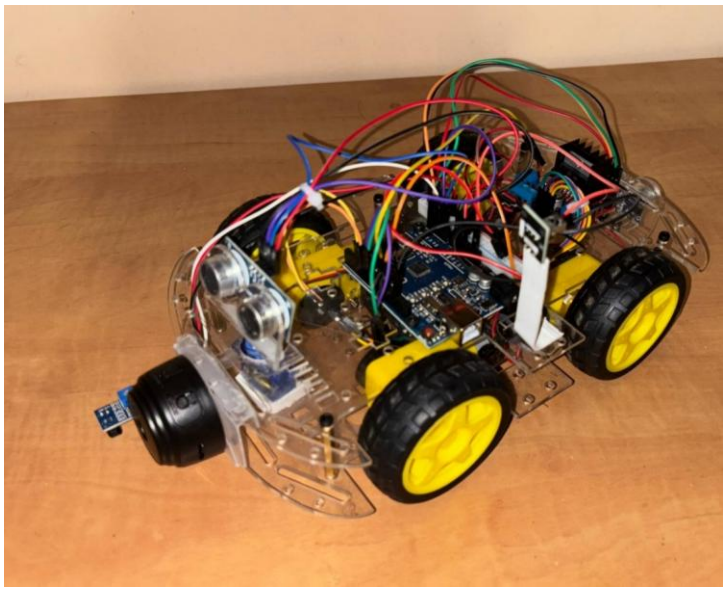

Рисунок 2.1 – Початкова конструкція мобільної роботизованої платформи

Вибір Arduino Uno був логічним для першого етапу, оскільки ця плата має достатній набір цифрових та аналогових входів-виходів для базової навчальної платформи [18]. Керування двигунами виконувалося через модуль L298N, побудований на двоканальному мостовому драйвері, який

призначений для керування двигунами постійного струму та кроковими двигунами [19]. Ультразвуковий модуль HC-SR04 застосовувався для первинного виявлення перешкод перед роботом [20].

У початковій моделі ультразвуковий датчик виконував роль основного засобу виявлення перешкод перед роботом. Сервопривід змінював напрямок його огляду, що давало змогу частково оцінювати простір спереду та з боків. Дистанційне керування виконувалося за допомогою інфрачервоного пульта, а перемикання режимів і команди руху оброблялися мікроконтролером Arduino Uno.

2.2 Режими роботи початкової моделі

Програмна частина початкової моделі передбачала два основні режими роботи: ручний та автономний. У ручному режимі оператор задавав команди руху за допомогою інфрачервоного пульта. Такий підхід був простим у реалізації та достатнім для перевірки працездатності двигунів, драйвера, живлення і базової логіки керування.

В автономному режимі робот рухався вперед до моменту виявлення перешкоди. Якщо ультразвуковий датчик фіксував об'єкт на небезпечній відстані, платформа зупинялася, від'їжджала назад і змінювала напрямок руху. Такий алгоритм належить до реактивних методів уникнення перешкод: рішення приймається за поточними показаннями сенсорів, без побудови карти та без запам'ятовування попередніх зіткнень.

Окремо в початковій конструкції використовувався датчик краю поверхні. Його призначенням було запобігання падінню робота зі столу або іншої підвищеної поверхні. При спрацюванні такого датчика робот повинен був зупинитися або змінити напрямок руху. Це підвищувало безпечність тестування, однак не вирішувало задачі навігації в приміщенні.

2.3 Виявлені проблеми початкової конструкції

Під час практичного використання початкової моделі було виявлено кілька обмежень, які заважали подальшому розвитку платформи. Одним із головних виявилася нестабільність живлення. Під час зіткнення робота з перешкодою колеса частково блокувалися, а двигуни продовжували намагатися обертатися. У цей момент збільшувалося навантаження на силову частину, що могло спричиняти короткочасне просідання напруги живлення. У результаті Arduino Uno працювала нестабільно, а в окремих випадках могла перезавантажуватися або припиняти коректне виконання програми.

Найімовірнішою причиною такої поведінки були короткочасні просідання напруги та електромагнітні завади, що виникали під час роботи двигунів. Якщо логічна частина і приводи живляться від спільного джерела, пусковий струм силової частини безпосередньо впливає на стабільність мікроконтролера.

Для зменшення впливу цієї проблеми до силової частини було додано електролітичний конденсатор ємністю 2200 мкФ. Це частково покращило стабільність роботи, однак повністю проблему не усунуло. Тому надалі було застосовано розділення живлення логічної та силової частин: мікроконтролер і сенсорні модулі живилися окремо від двигунів. Таке рішення зменшило взаємний вплив силових навантажень і логічної частини та підвищило надійність роботи платформи.

Окрема проблема стосувалася виявлення перешкод. HC-SR04 добре реагував на великі об'єкти, однак нестабільно виявляв вузькі перешкоди, наприклад ніжки стільців, дроти або предмети складної форми. Саме в таких випадках робот іноді стикався з об'єктом, тому що ультразвуковий датчик не встигав або не міг надійно зафіксувати перешкоду перед зупинкою.

Третьою проблемою була відсутність засобів для визначення власного положення і кута орієнтації. Початкова модель могла реагувати на поточні показання датчиків, але не могла точно визначати пройдений шлях, кут

повороту або координати. Тому реалізація координатної навігації та руху до заданої точки потребувала додаткової апаратної модернізації.

2.4 Проміжні рішення для покращення виявлення перешкод

Після виявлення обмежень ультразвукового датчика було проведено кілька проміжних спроб покращити виявлення перешкод. Одним із таких варіантів стало використання контактних датчиків удару в передній частині робота.

У передній частині було використано три контактні датчики удару. Залежно від того, який із них спрацьовував, робот міг приблизно визначити бік контакту та обрати напрямок від'їзду. Експериментальний варіант такої конструкції наведено на рисунку 2.2.

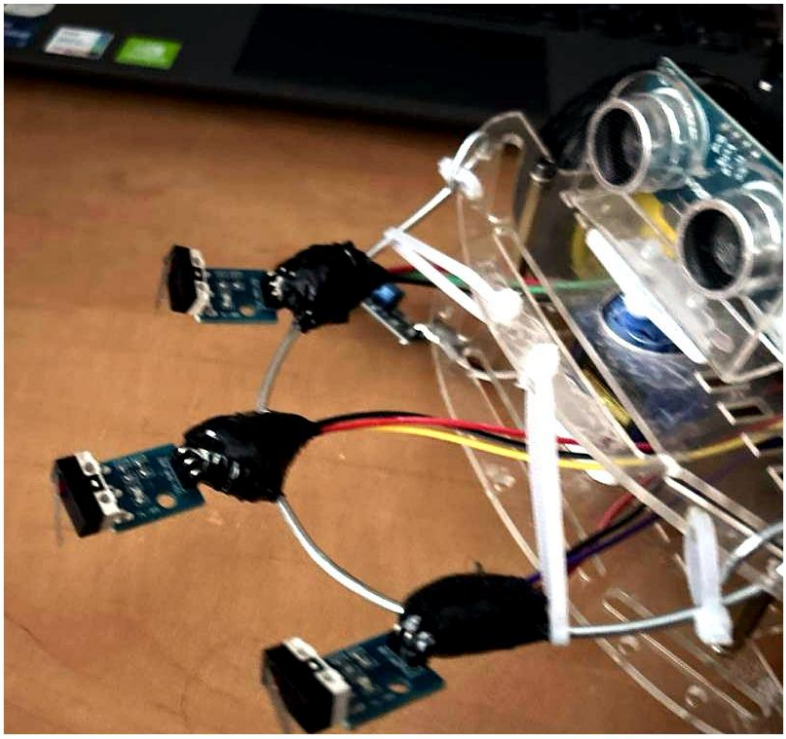

Рисунок 2.2 – Експериментальний варіант передньої частини робота з контактними датчиками удару

Попри простоту реалізації, контактний метод мав суттєві недоліки. Тонкий об'єкт міг потрапити між контактними елементами і не викликати спрацювання, а самі вузли залишалися механічно вразливими до ударів і

зачеплень. Тому контактні датчики були розглянуті лише як проміжне рішення.

Також було виконано спробу використання ІЧ-модуля виявлення перешкод FC-51. Типові модулі такого класу мають регульовану дальність спрацювання і застосовуються для виявлення близьких перешкод [21]. У практичних випробуваннях цей модуль іноді реагував на вузькі об'єкти швидше, ніж HC-SR04, однак результат залежав від кольору поверхні, її відбивної здатності та умов освітлення. Зокрема, темні й чорні об'єкти виявлялися нестабільно. Зовнішній вигляд модуля FC-51 наведено на рисунку 2.3.

Рисунок 2.3 – Інфрачервоний модуль виявлення перешкод FC-51

2.5 Перехід до лазерних датчиків відстані

У результаті порівняння різних способів виявлення перешкод було прийнято рішення перейти до використання лазерних датчиків відстані ToF. Такі модулі вимірюють відстань за часом проходження світлового імпульсу до об'єкта і назад. На відміну від простого ІЧ-модуля перешкод, ToF-сенсор не лише фіксує факт наявності об'єкта, а й дає числову оцінку відстані.

У вдосконаленій платформі для переднього, лівого та правого напрямків використано ToF-модулі VL53L1X. Для контролю простору позаду робота встановлено окремий модуль VL53L0X-V2 [22], [23]. Така схема дала змогу перейти від одного датчика з поворотом до кількох незалежних напрямків вимірювання.

Лазерні датчики виявилися практичнішими за ультразвуковий модуль для контролю близьких і вузьких перешкод. Вони швидше реагують у ближній зоні, краще підходять для розміщення кількох сенсорів на корпусі робота і дають змогу окремо контролювати передній, лівий, правий та задній напрямки. Водночас такі датчики не усувають проблему повністю: кожен модуль має обмежений кут огляду, тому між напрямками вимірювання залишаються сліпі зони. Через це ToF-сенсори потрібно розміщувати так, щоб їхні зони контролю взаємно доповнювали одна одну.

Водночас така сенсорна система не є повноцінною системою просторового сканування. Вона не формує детальну карту приміщення, а лише контролює кілька напрямків навколо робота. Для навчальної платформи і базового руху серед нескладних перешкод цього достатньо, однак така схема не замінює лідар, камеру глибини або повноцінну SLAM-систему.

Ультразвуковий датчик після модернізації не був повністю виключений із системи, але його роль змінилася. Він залишився допоміжним засобом для оцінювання простору під час повороту сервоприводу. Основне рішення про наявність близької перешкоди почали формувати лазерні датчики, а ультразвуковий модуль використовувався як додаткове джерело інформації.

2.6 Модернізація мікроконтролерної частини

Подальше розширення функціональності робота вимагало переходу з Arduino Uno на потужнішу мікроконтролерну платформу. Спочатку бездротове керування планувалося реалізувати на базі Arduino Uno із зовнішнім Bluetooth-модулем, зокрема модулем типу HC-06, і простим мобільним застосунком, створеним у MIT App Inventor. Передбачалося, що телефон під'єднуватиметься до робота через Bluetooth і надсилатиме команди руху.

Під час тестування такого підходу було виявлено, що зовнішній Bluetooth-модуль у поєднанні з Arduino Uno та мобільними пристроями працює недостатньо стабільно, ускладнює підключення, займає додаткові

виводи мікроконтролера й потребує окремого налаштування. Це стало однією з причин відмови від варіанта з окремим Bluetooth-модулем і переходу до платформи ESP32.

Для спрощення конструкції, зменшення кількості зовнішніх модулів та підвищення надійності системи було прийнято рішення перейти на платформу ESP32. У роботі застосовано плату розробника LuaNode 32 на базі модуля ESP32-WROOM-32, який має вбудовані засоби Wi-Fi та Bluetooth/BLE [24]. Завдяки цьому відпала потреба у використанні окремого Bluetooth-модуля, а керування мобільним роботом було реалізовано через Wi-Fi за допомогою веб-інтерфейсу.

Практичним результатом переходу стало створення вебінтерфейсу керування: ESP32 формує бездротове підключення, а оператор відкриває сторінку керування у браузері смартфона або ноутбука. Такий варіант виявився зручнішим за ІЧ-керування, оскільки не потребує прямої видимості, менш залежить від відбивань сигналу в приміщенні й дає змогу працювати на більшій відстані, ніж це практично можливо для ІЧ-пульта. Крім того, браузерний інтерфейс не прив'язує користувача до конкретної операційної системи телефона.

Рисунок 2.4 – Плата розробника LuaNode 32 на базі модуля ESP-WROOM-32

2.7 Додавання апаратних засобів для навігації

Після переходу на ESP32 і модернізації сенсорної системи виникла необхідність додати апаратні засоби, які дозволяють оцінювати не лише наявність перешкод, а й рух самої платформи. Для цього були використані оптичні енкодери коліс на основі компаратора LM393, інерціальний модуль LSM6DS3 та мультиплексор I2C PCA9548A.

Оптичні енкодери коліс на основі компаратора LM393 застосовуються для підрахунку імпульсів під час обертання коліс [17]. За кількістю імпульсів можна оцінити пройдений шлях і контролювати довжину окремих переміщень. У програмній конфігурації робота імпульси правого і лівого енкодерів зчитуються з окремих входів ESP32, що дозволяє використовувати їх для контролю переміщення та подальшої побудови координатної логіки руху.

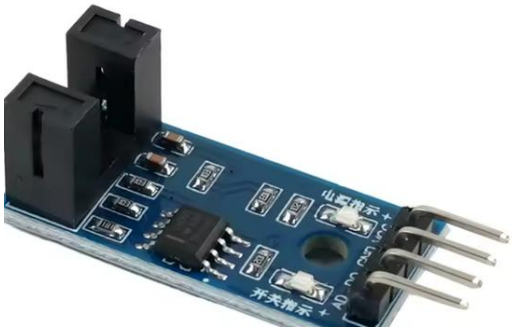

Рисунок 2.5 – Оптичний енкодер колеса на основі компаратора LM393

Інерціальний вимірювальний модуль IMU (Inertial Measurement Unit) LSM6DS3 було додано для контролю кута повороту. Цей модуль поєднує тривісний акселерометр і тривісний гіроскоп [25]. Початкова логіка руху виконувала повороти переважно за часом роботи двигунів, що є неточним способом: фактичний кут залежить від поверхні, заряду акумулятора, ковзання коліс і різниці швидкостей двигунів. Використання IMU дозволяє отримувати дані гіроскопа та оцінювати зміну кута орієнтації робота.

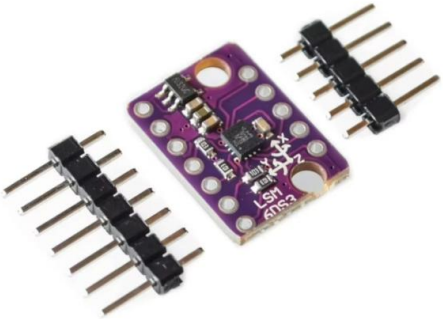

Рисунок 2.6 – Інерціальний модуль LSM6DS3

Для підключення кількох I2C-пристроїв використано мультиплексор PCA9548A [26]. Він дозволяє розвести одну I2C-шину на кілька незалежних каналів і програмно обирати, з яким модулем ESP32 обмінюється даними. Це було важливо для кількох ToF-датчиків з однаковими адресами, а також для підключення IMU через ту саму шину.

Зовнішній вигляд основних модулів, доданих для навігації, наведено на рисунках 2.5–2.7.

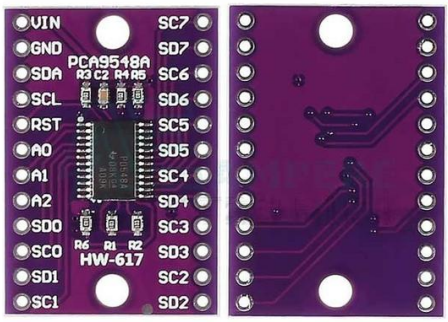

Рисунок 2.7 – I2C-мультиплексор PCA9548A

Після вибору модулів було сформовано загальну схему їх підключення до ESP32 (рис. 2.8). Двигуни підключено через драйвер L298N, лазерні датчики та IMU об'єднано через I2C-мультиплексор, а приймач ІЧ-пульта, енкодери та сервопривід використовують окремі сигнальні лінії. Такий

розподіл спрощує налагодження, оскільки кожен функціональний вузол можна перевіряти окремо перед запуском повної навігаційної логіки.

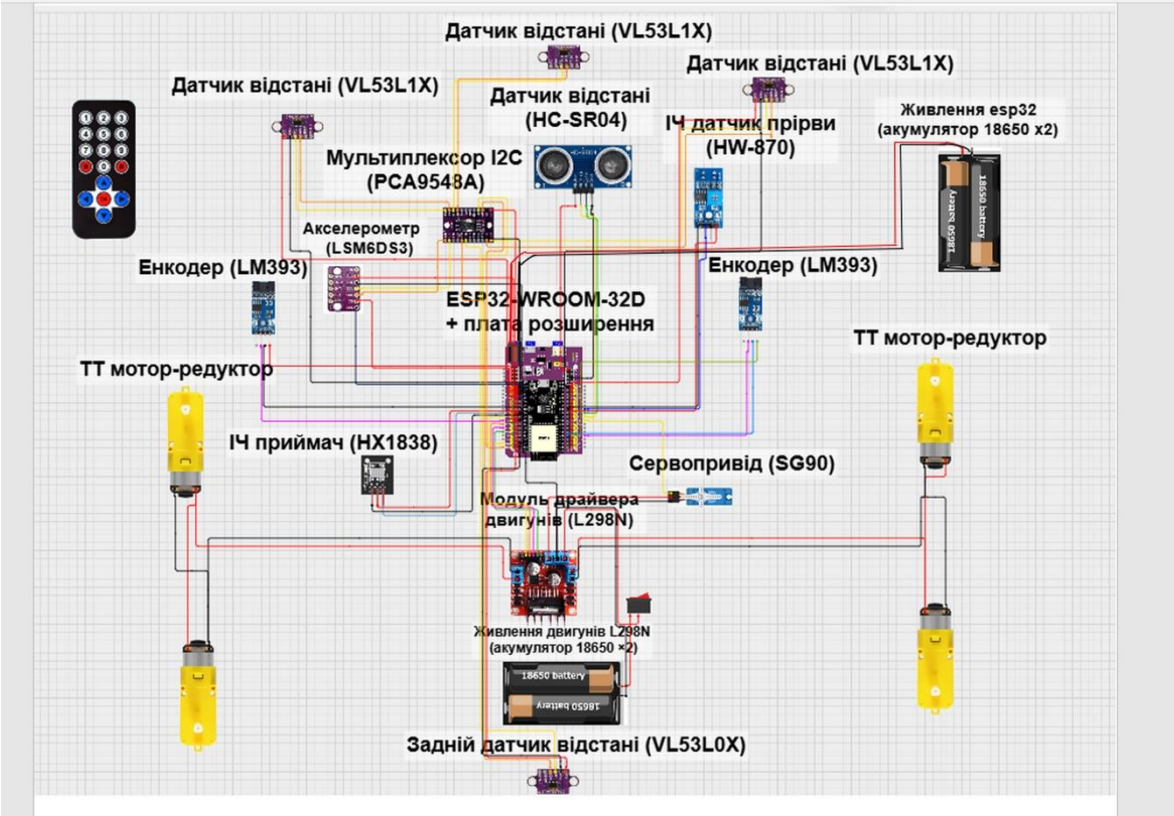

Рисунок 2.8 – Загальна схема підключення електронних вузлів модернізованої мобільної платформи

Схема відображає розподіл основних груп: ESP32 як центральний контролер, драйвер двигунів L298N, ToF-датчики через I2C-мультиплексор, інерціальний модуль, енкодери коліс, інфрачервоні елементи, сервопривід, живлення логічної частини та окреме живлення силової частини.

Узагальнено апаратна модернізація охопила три ключові напрями: підвищення стабільності живлення, заміну недостатньо ефективних сенсорів перешкод на точніші лазерні датчики та додавання елементів, необхідних для координатної навігації. Зовнішній вигляд модернізованої платформи наведено на рисунку 2.9.

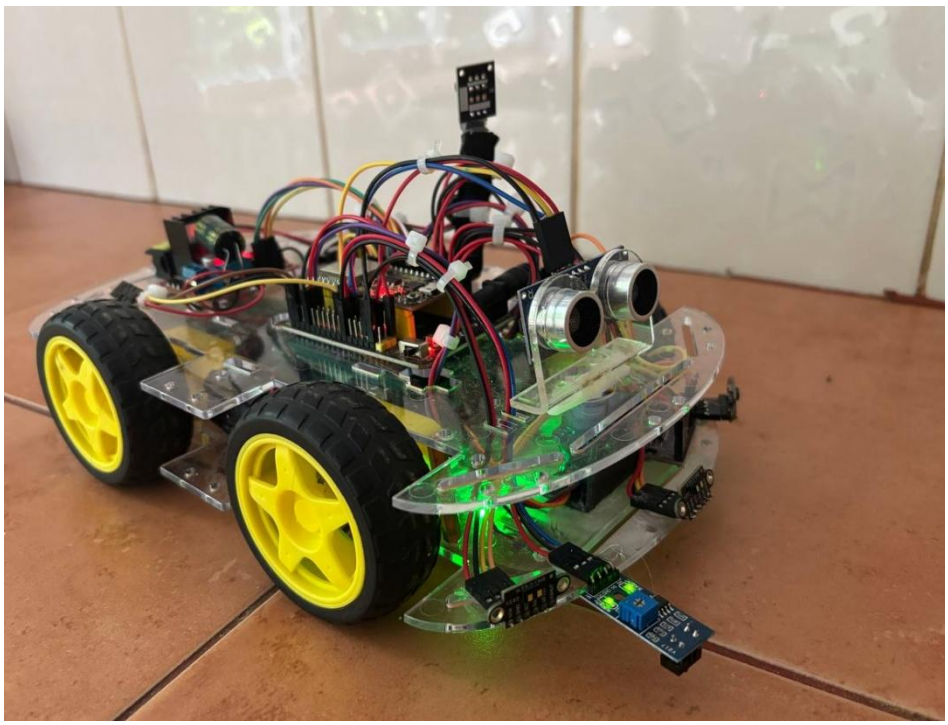

Рисунок 2.9 – Загальний вигляд модернізованої мобільної платформи

Саме ці зміни створили основу для переходу від простого реактивного робота до платформи з координатним рухом, картою та програмною логікою уникнення перешкод.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ ТА НАВІГАЦІЇ МОБІЛЬНОГО РОБОТА

3.1 Загальна структура програмного забезпечення

Програмна частина модернізованої платформи реалізована на базі ESP32 і виконує роль центрального рівня керування. Вона приймає команди оператора, зчитує датчики, формує сигнали для драйвера двигунів L298N, веде дискретні координати робота та передає стан системи у вебінтерфейс. На відміну від початкової версії, де основною була реакція на команду або просту перешкоду, у фінальній програмі поєднано ручне керування, автономний рух, координатний режим GOTO, пам'ять перешкод і експериментальний пошук телефона за рівнем Wi-Fi RSSI.

Код побудовано як набір взаємопов'язаних підсистем. Навігаційна частина зосереджена в просторі імен GotoNav: там зберігаються координати, стан GOTO, параметри руху, пам'ять перешкод, обробка IMU, енкодерів і ToF-датчиків. Окремо реалізовано обробники вебсерверу, ручний режим, автономний режим, режим пошуку телефона, ІЧ-керування та загальні захисні функції. Такий поділ важливий практично: низькорівневе керування двигунами не змішується безпосередньо з логікою маршруту, а налагодження можна виконувати по окремих блоках.

У програмі використано чотири режими роботи. Значення змінної mode визначає, який алгоритм виконується в головному циклі: 0 - MANUAL, 1 - AUTO, 2 - GOTO, 3 - PHONE. Перемикання режимів виконується через вебінтерфейс, HTTP-запити або ІЧ-пульт. При переході між режимами програма зупиняє двигуни та скидає тимчасові стани, щоб не залишалась стара команда руху.

Таблиця 3.1 – Основні режими роботи програми

Режим	**Призначення**	**Суть роботи**
MANUAL	Ручне керування оператором	Оператор утримує кнопки руху. Робот одразу реагує на команди, але блокує рух вперед або назад, якщо відповідний датчик показує небезпечну перешкоду.
AUTO	Автономний реактивний рух	Робот рухається короткими ділянками, контролює передній, бокові та задній напрямки, обирає вільніший бік і додає підтверджені перешкоди на карту.
GOTO	Рух до заданої координати	Задана оператором координата перетворюється на послідовність дискретних переміщень. Перед кожним сегментом робот вирівнює кут, перевіряє шлях і за потреби перебудовує маршрут.
PHONE	Пошук телефона за RSSI	Робот виконує короткі пробні переміщення, порівнює рівень Wi-Fi сигналу телефона та змінює напрямок за принципом 'тепліше - холодніше'.

Головний цикл loop() працює як диспетчер. Спочатку обслуговуються вебсервер, ІЧ-приймач, IMU, енкодери та датчики відстані. Після цього виконується лише та гілка, яка відповідає поточному режиму. Це не дозволяє, наприклад, одночасно виконувати GOTO і ручний рух, а також спрощує аналіз помилок під час тестування.

3.2 Ініціалізація та обробка апаратних вузлів

Під час запуску програма послідовно готує основні апаратні вузли: вмикає діагностичний порт, завантажує збережені параметри з пам'яті Preferences, налаштовує керування двигунами, ІЧ-приймач, сервопривід, бузер, сенсор краю поверхні, енкодери, I2C-шину, ToF-датчики та інерціальний модуль. Після цього створюється Wi-Fi точка доступу і запускається вебсервер. Така послідовність потрібна, щоб до переходу в режими руху всі критичні датчики вже були готові до роботи.

Лазерні датчики відстані підключені через мультиплексор PCA9548A: центральний, лівий і правий VL53L1X, а також задній VL53L0X-V2. Після вибору каналу I2C програма перевіряє наявність датчика, запускає безперервне вимірювання та зберігає останні стабільні значення у кеші. Для зменшення хибних спрацювань застосовується фільтрація: різкі далекі

стрибки не приймаються одразу, а близькі значення приймаються швидше, оскільки вони важливі для безпеки.

Для кожного ToF-напрямку зберігається не лише останнє значення відстані, а й службовий стан датчика: чи готовий модуль, чи валідне показання, скільки разів поспіль підтверджено блокування або очищення напрямку. Тому висновок про перешкоду не робиться за одним випадковим вимірюванням.

Енкодери підключені до окремих входів ESP32 і працюють через переривання. Кожен імпульс збільшує лічильник відповідного колеса. У програмі використано окремі коефіцієнти перерахунку імпульсів у сантиметри для лівого та правого коліс, оскільки реальні двигуни, колеса й поверхня не є ідеально однаковими. Для карти одна умовна клітинка відповідає 16 см, а каліброване значення TICKS_PER_CELL становить 26 імпульсів.

Інерціальний модуль LSM6DS3 використовується насамперед як гіроскоп для контролю кута повороту. У коді відстежується yaw - тобто кут повороту робота навколо вертикальної осі. Калібрування зміщення гіроскопа виконується лише тоді, коли платформа стоїть нерухомо. Під час руху активне калібрування не проводиться, щоб реальне обертання корпуса не було помилково прийняте за нульове зміщення.

3.3 Вебінтерфейс і обмін даними з оператором

Після запуску ESP32 створює Wi-Fi точку доступу і піднімає HTTP-сервер на порту 80. Оператор відкриває сторінку керування у браузері за адресою 192.168.4.1. Це рішення було обране замість окремого мобільного застосунку та замість ІЧ-керування, оскільки браузерний інтерфейс не потребує прямої видимості, не залежить так сильно від відбивань сигналу в приміщенні, дозволяє працювати на більшій відстані й не вимагає встановлення спеціальної програми на телефон.

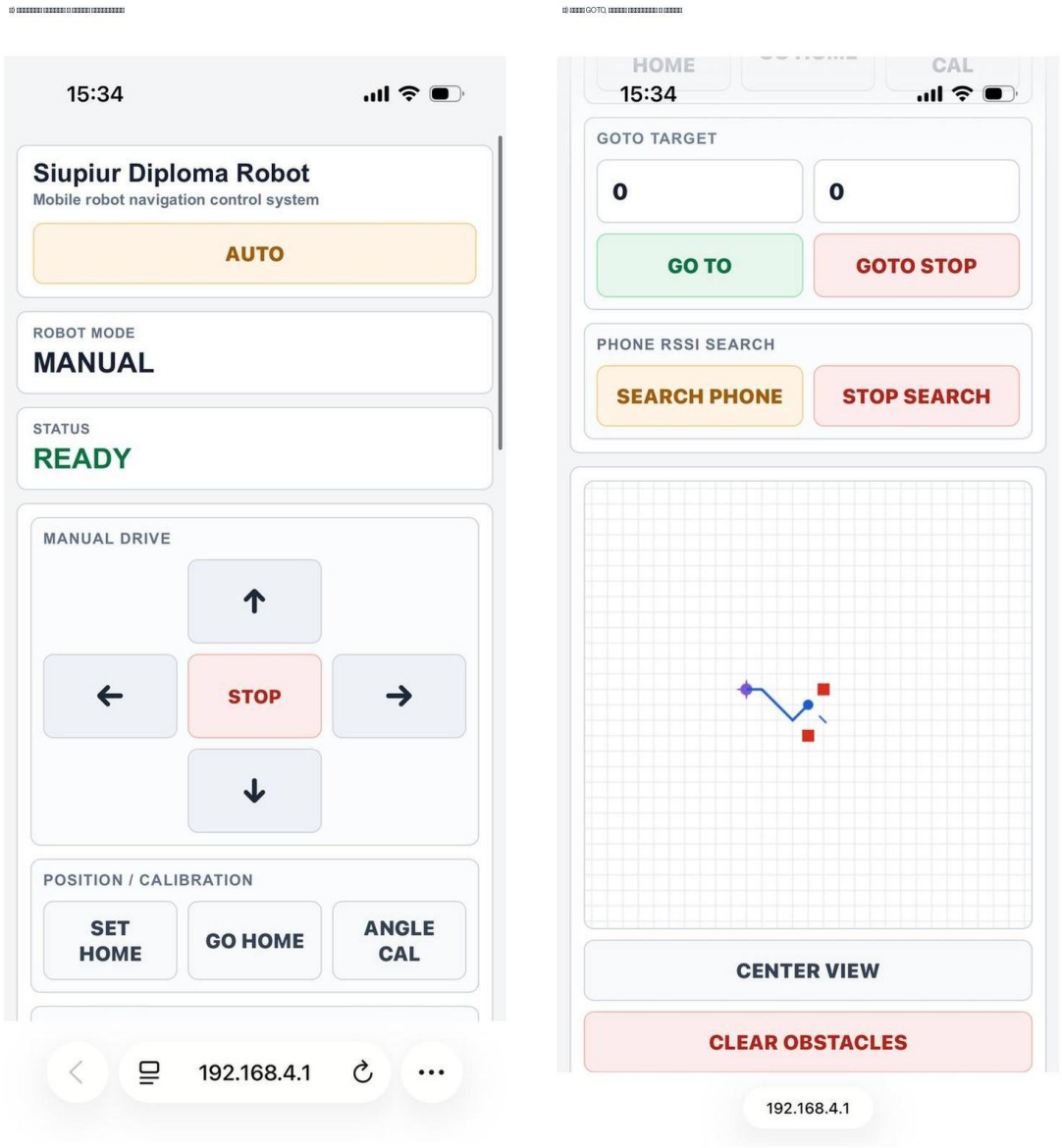

Рисунок 3.1 – Основні елементи вебінтерфейсу керування роботом

У вебінтерфейсі є блок ручного керування, кнопка переходу в AUTO, кнопки SET HOME, GO HOME та ANGLE CAL, поля для введення координати GOTO, керування пошуком телефона, а також елементи роботи з картою. Ручні кнопки працюють за принципом утримання: доки оператор тримає кнопку, браузер періодично надсилає команду руху; після відпускання надсилається STOP. Такий підхід зменшує ризик, що робот продовжить рух після втрати зв'язку або випадкового закриття сторінки.

Кнопка ANGLE CAL використовується для встановлення поточного кута орієнтації робота в нульове значення без зміни його координатного

положення на карті. Тобто після виконання цієї команди робот залишається в тій самій точці простору, але поточний напрямок приймається як новий початковий напрямок відліку кута.

Команда SET HOME виконує повне перевстановлення початкового стану: поточне положення робота приймається за координати (0, 0), а поточний кут орієнтації також встановлюється як нульовий. Таким чином, SET HOME одночасно обнуляє і координати, і кут, тоді як ANGLE CAL змінює лише кутовий відлік без зміни положення робота на карті.

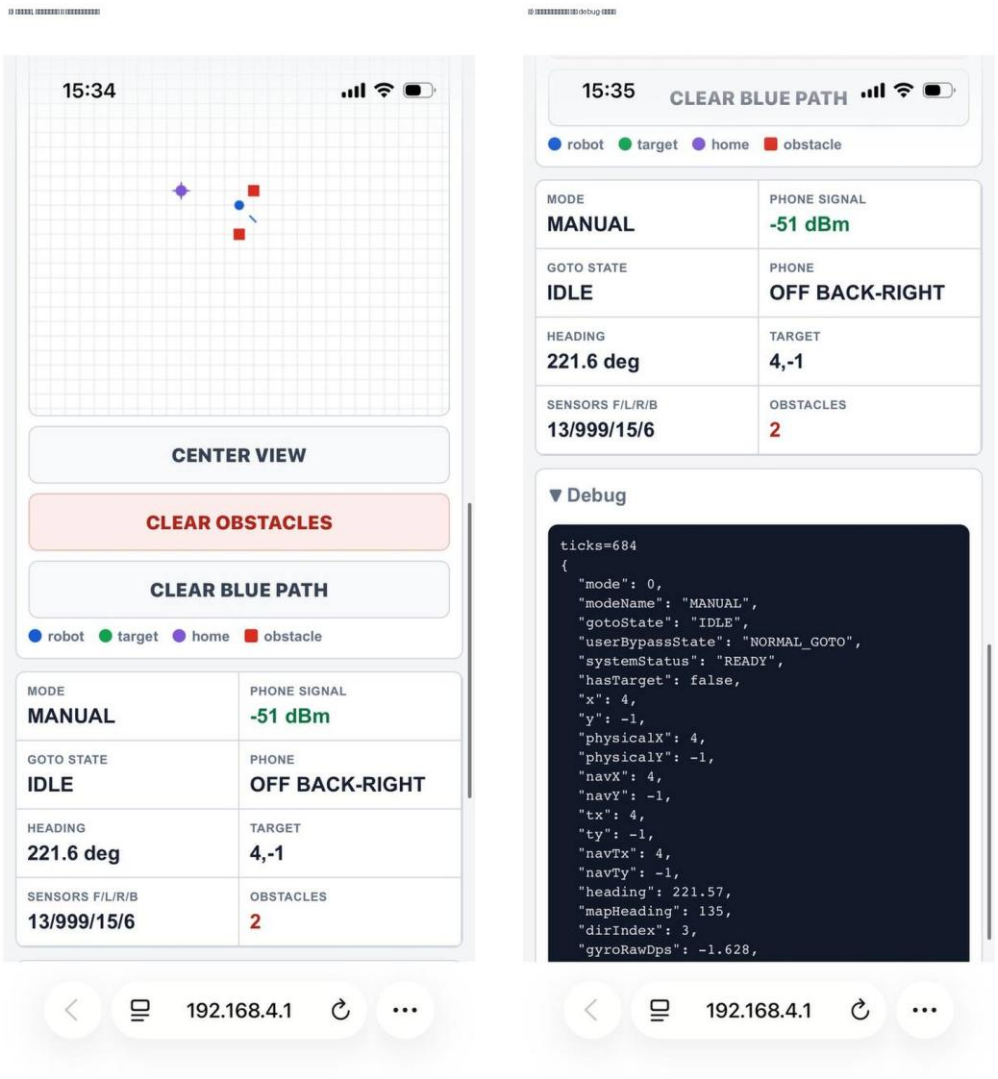

Рисунок 3.2 – Відображення карти, телеметрії та debug-стану у вебінтерфейсі

Сторінка періодично виконує запит /pose приблизно кожні 350 мс. У відповідь мікроконтролер передає JSON - тобто структурований текстовий

пакет даних, у якому містяться режим роботи, координати, кут, ціль, показання датчиків, рівень RSSI та список перешкод. Саме ці дані використовуються для оновлення телеметрії, стану GOTO і debug-блока.

Карта у вебінтерфейсі створюється безпосередньо на елементі HTML Canvas. Скрипт спочатку малює сітку простору, далі позначає HOME фіолетовою міткою, ціль - зеленою, запам'ятовані перешкоди - червоними квадратами, а поточне положення робота синьою точкою з коротким вектором напряму. Синя траєкторія накопичується з послідовних значень x та y, які надходять із /pose. Для налагодження одночасно передаються дві групи координат: physicalX/physicalY відображають безперервну оцінку руху за енкодерами та IMU, а navX/navY - дискретні координати навігаційної сітки, з якими працює режим GOTO.

3.4 Ручний та автономний режими

Режим MANUAL призначений для прямого керування роботом під час тестування, калібрування та аварійного втручання. За базовою логікою руху цей режим не відрізняється принципово від початкової моделі: команди вперед, назад, ліворуч і праворуч одразу перетворюються на PWM-сигнали двигунів. Водночас у фінальній версії ручний режим уже не є повністю «сліпим»: рух вперед блокується центральним переднім ToF-датчиком, а рух назад - заднім VL53L0X-V2.

Якщо під час ручного руху датчик підтверджує близьку перешкоду, програма зупиняє двигуни й може додати відповідну клітинку на карту. Передня перешкода в ручному режимі створюється переважно за центральним ToF-датчиком, а не за нестабільними боковими значеннями. Це зроблено для того, щоб карта не заповнювалася хибними червоними маркерами від бокових відблисків або сліпих зон датчиків.

Режим AUTO є реактивним автономним режимом. Робот рухається короткими ділянками, оцінює передню, бокові та задню зони, зупиняється перед підтвердженою перешкодою, за потреби від'їжджає назад і повертає у

вільніший бік. Порівняно з ранньою версією було зменшено ризик зациклення між кількома перешкодами: якщо алгоритм кілька разів поспіль намагається повертати в один і той самий бік, після третьої однакової спроби він примусово перевіряє протилежний напрямок. Це не робить AUTO складною системою планування, але помітно підвищує стійкість у простих кімнатних сценаріях.

3.5 Координатний режим GOTO, карта і побудова перешкод

Координатний режим GOTO є центральним засобом автономного переміщення в розробленій системі. Оператор задає координати X та Y у клітинках карти, після чого програма переводить ціль у навігаційну сітку. В одній клітинці прийнято крок 16 см, тому команди Y=1, Y=4 або X=6 відповідають переміщенню на 16, 64 та 96 см. Початок системи координат задається командою SET HOME, а всі подальші точки, траєкторія й перешкоди зберігаються відносно цього нуля.

Важливо підкреслити, що ця карта не є SLAM-картою. Робот не будує повну метричну модель кімнати, не виконує одночасну локалізацію і картографування за хмарами точок та не має огляду на 360 градусів. Карта є спрощеною дискретною моделлю: робот запам'ятовує власні кроки та ті клітинки, у яких датчики підтвердили перешкоду. Для складного невідомого середовища цього недостатньо, але для базового керування в приміщенні з нескладними перешкодами такий підхід є виправданим.

Побудова карти базується на поєднанні двох рівнів представлення. Нижній рівень зберігає фізичну оцінку положення шасі за енкодерами й IMU, тоді як верхній рівень - навігаційна сітка navGrid, яка використовує дискретні клітинки для планування маршруту. Саме navGrid визначає видиме положення робота в режимі GOTO на карті вебінтерфейсу та слугує основою для прокладання синього шляху.

Карта формується подієво. Коли робот успішно завершує черговий сегмент руху, його координата у navGrid оновлюється, і ця точка додається до синьої траєкторії. Коли ж датчики підтверджують блокування наступного

кроку, у пам'ять заноситься не абстрактна відстань у сантиметрах, а конкретна клітинка, яка реально заважає продовженню руху. Завдяки цьому карта показує не лише шлях, а й накопичений досвід руху робота в межах поточної сесії.

План GOTO будується як послідовність коротких сегментів. Перед початком кожного сегмента робот визначає потрібний напрямок, повертає корпус до відповідного кута, перевіряє стабільність повороту за IMU і лише після цього рухається вперед за енкодерами. Такий покроковий режим зменшує накопичення похибки й дає змогу частіше перевіряти наявність перешкод.

Таблиця 3.3 – Основні стани координатного режиму GOTO

Стан	**Призначення**	**Що виконується**
IDLE	Очікування	Ціль не активна, двигуни вимкнені.
WAIT_IMU	Пауза перед рухом	Робот стоїть, завершується стабілізація або калібрування гіроскопа.
PLAN	Планування	Будується наступний сегмент маршруту з урахуванням цілі та пам'яті перешкод.
TURN	Поворот	Платформа повертається до потрібного напрямку, IMU контролює heading.
VERIFY_TURN	Перевірка кута	Після зупинки програма чекає стабілізації та за потреби виконує малу докрутку.
MOVE	Рух сегментом	Робот їде вперед на задану кількість клітинок, контролюючи енкодери, heading і датчики.
SETTLE	Пауза після руху	Двигуни зупинені, робот чекає завершення інерційного руху.
ARRIVED	Досягнення цілі	Маршрут завершено, двигуни вимкнені, може подаватися звуковий сигнал.
BLOCKED/STUCK	Проблемна ситуація	Рух заблоковано або не вдалося знайти безпечне продовження маршруту.

Побудова перешкод у GOTO виконується обережно. Якщо під час руху вперед центральний ToF-датчик показує небезпечне наближення, робот спершу зупиняється. Після цього програма повторно зчитує датчики кілька разів у нерухомому стані. Лише підтверджена перешкода додається до пам'яті карти, щоб один випадковий відблиск або затримане вимірювання не ламали маршрут.

Координата перешкоди визначається не довільно, а відносно поточної клітинки робота і напряму запланованого руху. Перший маркер перешкоди ставиться саме в ту клітинку, яка реально блокує наступний крок. Бокові датчики в GOTO переважно допомагають вибрати вільніший бік для обходу, але самі по собі не передають сигнал про передню перешкоду без додаткового підтвердження.

Після появи нової клітинки перешкоди маршрут перебудовується вже не «поверх» карти, а з урахуванням пам'яті про заблоковані позиції. Якщо перешкод ще немає, програма може сформувати простий рух за осями X та Y. Якщо ж червоні клітинки вже існують, планувальник намагається прокласти шлях в обхід цих зон. Отже, карта в цій роботі є не лише елементом інтерфейсу, а й робочою частиною алгоритму.

Якщо запитана користувачем ціль уже зайнята підтвердженою перешкодою, програма не намагається в'їхати в неї буквально, а шукає найближчу вільну клітинку підходу. Завдяки цьому команда оператора не втрачає змісту, а робот не витрачає час на безглузду спробу досягти фізично заблокованої точки.

У внутрішньому стані системи зберігається як запитана оператором ціль, так і навігаційна ціль, до якої фактично прямує робот. Це дозволяє у вебінтерфейсі показувати початковий задум команди, а в алгоритмі руху - оперувати реально досяжною точкою підходу.

Окремо реалізовано захист від «тіньових» перешкод. Якщо на одному напрямку вже є відома червона клітинка, новий маркер далі за нею має бути відхилений, оскільки датчик не може достовірно бачити об'єкт за найближчою перешкодою. Це не робить карту ідеальною, але робить її стабільнішою для базових кімнатних сценаріїв.

3.6 Режим PHONE: пошук пристрою за RSSI

Режим PHONE є експериментальним доповненням до основних режимів руху. Телефон підключається до Wi-Fi точки доступу ESP32, після чого

мікроконтролер отримує рівень сигналу RSSI підключеної станції. Мета режиму не полягає в точному визначенні координат телефона. Робот працює за простішою логікою: виконує коротке пробне переміщення, зупиняється, чекає оновлення RSSI і порівнює, чи став сигнал сильнішим.

Якщо після пробного руху RSSI покращився, напрямок вважається перспективним і робот може продовжити рух. Якщо сигнал погіршився або не змінився, поточний напрямок тимчасово запам'ятовується як небажаний, а робот обирає інший напрямок. Значення RSSI оновлюється повільніше, ніж дані датчиків відстані, тому після переміщення передбачена пауза для стабілізації. Для зменшення впливу випадкових стрибків використовується фільтроване значення сигналу.

PHONE використовує ті самі засоби безпеки, що й інші режими: перевірку переднього і заднього напрямків, пам'ять перешкод, контроль небезпечного краю поверхні та зупинку при блокуванні. Якщо шлях до пробного напрямку перекритий, відповідний напрямок тимчасово позначається як no-go, а підтверджена перешкода додається на карту. Отже, режим RSSI не замінює GOTO, а демонструє інший тип пошукової поведінки на тій самій апаратній і програмній базі.

3.7 Засоби безпеки та діагностики

У програмі передбачено кілька рівнів безпеки. Перший рівень - це команда STOP, доступна з вебінтерфейсу, ІЧ-пульта і внутрішніх обробників режимів. Другий рівень - блокування небезпечного руху за датчиками: передній ToF не дозволяє їхати вперед у близьку перешкоду, задній ToF не дозволяє здавати назад у перешкоду, а бокові датчики враховуються під час автоматичного руху й обходу. Третій рівень - сенсор контролю краю поверхні на GPIO14, який використовується в AUTO, GOTO та PHONE.

Додатково реалізовано програмні watchdog-механізми. Якщо під час GOTO двигуни отримують команду, але енкодери або IMU не показують очікуваного прогресу, програма не повинна нескінченно продовжувати той

самий стан. Вона може повторно проаналізувати перешкоду, перебудувати план, виконати обмежену спробу відновлення або перейти у BLOCKED/STUCK. Це важливо для реальної платформи, де колеса можуть прослизати, батарея може просідати, а датчики іноді дають нестабільні значення.

Debug-блок у вебінтерфейсі показує внутрішній стан системи в розгорнутому вигляді. Для оператора це не основний елемент керування, однак саме тут видно, у якому режимі працює робот, який стан має машина GOTO, які координати та ціль зараз активні, що показують датчики, чи триває калібрування IMU і скільки перешкод уже внесено до пам'яті. Фактично цей блок пояснює, чому робот поїхав, зупинився або перейшов у BLOCKED/STUCK.

3.8 Обмеження реалізованої програмної системи

Розроблена система є навчально-дослідною і свідомо спрощеною. Вона не має складної SLAM-архітектури, не використовує камеру, лідар або глобальну локалізацію, а тому не може гарантувати точну карту приміщення в довготривалому русі. Положення оцінюється за енкодерами та IMU, тому похибка може накопичуватися через ковзання коліс, нерівну поверхню, люфт механіки або дрейф гіроскопа.

Сенсорна система також має обмеження. ToF-датчики добре працюють у ближній зоні, але бачать лише окремі напрямки. Між напрямками вимірювання залишаються сліпі зони; тонкі, темні, прозорі або розташовані під невдалим кутом об'єкти можуть виявлятися нестабільно. Саме тому програма використовує підтвердження перешкод, обережні короткі кроки та можливість ручного втручання.

Попри ці обмеження, для поставленої задачі така реалізація є достатньою. Вона дозволяє продемонструвати повний цикл роботи мобільного робота: отримання команди, оцінювання датчиків, рух за енкодерами, контроль кута за IMU, побудову простої карти, запам'ятовування перешкод,

рух до координати та відображення стану в браузері. Для базового управління в приміщенні з нескладними перешкодами цього рівня функціональності достатньо.

4. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОБОТИ МОБІЛЬНОЇ РОБОТИЗОВАНОЇ ПЛАТФОРМИ

4.1 Методика проведення експериментальних досліджень

Після завершення розроблення апаратної та програмної частин мобільної роботизованої платформи було проведено експериментальну перевірку її роботи. Метою досліджень було підтвердити працездатність основних режимів системи, оцінити точність переміщення в режимі GOTO, перевірити виявлення та об'їзд перешкод, а також оцінити роботу режимів AUTO і PHONE.

Експерименти виконувалися в кімнатних умовах на рівній твердій поверхні. Перед кожною спробою робот встановлювався у початкове положення, після чого через вебінтерфейс задавалася точка HOME. Це забезпечувало однакові стартові умови для повторних серій вимірювань.

Для кількісної перевірки режиму GOTO досліджувалися переміщення, що відповідали командам Y=1, Y=4, Y=6 та X=1, X=4, X=6. Для цих серій контрольними відстанями приймалися 16, 64 і 96 см. Для кожної координати виконувалося по 10 повторних спроб. Після завершення руху вимірювалися фактична пройдена відстань уздовж основної осі переміщення та бокове відхилення від заданої траєкторії.

Під час оброблення результатів визначалися середня фактична відстань, середня абсолютна похибка, максимальна абсолютна похибка та середнє бокове відхилення. Для оцінювання похибки використано співвідношення:

$$\Delta S = S\text{факт} - S\text{зад},$$

де Sфакт - фактично виміряна відстань після зупинки робота, см;Sзад - контрольне значення для відповідної серії дослідів, см.

Окремо були виконані дослідження роботи системи виявлення перешкод, алгоритму об'їзду, поведінки робота в автоматичному режимі AUTO та зміни рівня RSSI залежно від відстані, перешкод і орієнтації робота відносно телефона.

4.2 Дослідження точності прибуття в режимі GOTO

Першим етапом експериментальної перевірки було дослідження точності прибуття робота у задану координату в режимі GOTO. Рух уздовж осі Y відповідав прямолінійному переміщенню вперед, тоді як рух уздовж осі X додатково включав поворот платформи та подальший рух у новому напрямку. Саме тому для осі X очікувалося більше накопичення похибки. Повні таблиці експериментальних вимірювань для кожної координати наведено в додатку А.

Таблиця 4.1 – Узагальнені результати перевірки точності переміщення в режимі GOTO

Координата	Контрольна відстань, см	Середня фактична відстань, см	Середня абсолютна похибка, см	Максимальна абсолютна похибка, см	Середнє бокове відхилення, см
Y=1	16	18,55	2,55	5,50	0
Y=4	64	62,50	1,70	4,00	0,90
Y=6	96	97,25	1,45	4,00	0,40
X=1	16	16,88	1,72	3,00	0
X=4	64	60,47	3,93	7,00	0,60
X=6	96	91,02	4,98	9,00	1,40

Отримані результати показують, що під час руху вздовж осі Y робот у цілому стабільно відпрацьовує задану координату. Для найкоротшого переміщення Y=1 середня фактична відстань становила 18,55 см при контрольному значенні 16 см. На малій дистанції відносний вплив інерції після зупинки та локального проковзування коліс є відчутнішим, тому похибка виглядає більш помітною.

Для довших переміщень вздовж осі Y результати виявилися рівномірнішими. Для Y=4 середня абсолютна похибка становила 1,70 см, а для Y=6 - 1,45 см. Це свідчить про достатню точність прямолінійного руху за відсутності додаткового повороту.

Під час руху по осі X середня абсолютна похибка виявилася більшою. Для X=4 вона становила 3,93 см, а для X=6 - 4,98 см. Такий результат є очікуваним, оскільки перед початком руху по осі X робот має виконати поворот, а отже на кінцевий результат впливають не лише енкодери й проковзування коліс, а й похибка визначення кута за IMU.

Таким чином, режим GOTO забезпечує працездатне переміщення до заданої координати, однак найкраща точність досягається під час прямолінійного руху. Для маневрів, що включають зміну орієнтації, похибка помітно зростає, але залишається прийнятною для навчально-дослідної мобільної платформи.

Деталізація вимірювань для руху по осі Y наведена в додатку А.
Деталізація вимірювань для руху по осі X наведена в додатку А.

4.3 Дослідження роботи системи виявлення перешкод

Наступним етапом було дослідження роботи системи виявлення перешкод. Перевірка виконувалася для широких об'єктів, зокрема коробок, а також для тонких вертикальних перешкод типу ніжки стільця.

Експерименти показали, що широкі перешкоди, розташовані перед роботом, виявляються стабільно. Усі перевірені коробки на траєкторії руху фіксувалися надійно, а під час роботи GOTO робот зупинявся перед перешкодою та переходив до логіки об'їзду.

Для тонких об'єктів результат залежав від їхнього положення відносно центральної зони огляду. Якщо вузька перешкода потрапляла по центру траєкторії, виявлення було стабільним. Якщо ж вона зміщувалася вбік, спостерігалися пропуски, що пояснюється малою площею відбиття сигналу та обмеженою зоною огляду датчиків.

Таблиця 4.4 – Результати дослідження виявлення перешкод

Тип перешкоди	Положення перешкоди	Результат виявлення	Коментар
Коробка	Попереду робота	Стабільно	Перешкода виявлялася в усіх перевірених випадках.
Коробка	На маршруті GOTO	Стабільно	Робот зупинявся перед перешкодою та переходив до об'їзду.
Тонка перешкода	По центру траєкторії	Стабільно	Перешкода потрапляла в центральну зону виявлення.
Тонка перешкода	Зміщена вбік	Нестабільно	Можливі пропуски через малу площу об'єкта та геометрію відбиття.

Отже, система виявлення перешкод є працездатною для широких об'єктів і перешкод, розташованих у центральній зоні перед роботом.

Водночас тонкі об'єкти, зміщені вбік від траєкторії, виявляються менш надійно, що потрібно враховувати під час експлуатації платформи в реальному приміщенні.

4.4 Дослідження об'їзду перешкод у режимі GOTO

Для перевірки алгоритму об'їзду перешкод було проведено серію дослідів у режимі GOTO. Роботу задавалася цільова координата, а на шляху руху встановлювалася коробка. Під час кожної спроби оцінювалося, чи виявляє робот перешкоду, який напрямок обирає для об'їзду та чи досягає продовження руху без критичного збою.

Таблиця 4.5 – Результати об'їзду однієї перешкоди в режимі GOTO

№ спроби	Чи виявлено перешкоду	Напрямок об'їзду	Чи було зіткнення	Результат
1	Так	Праворуч	Ні	Об'їзд виконано
2	Так	Праворуч	Ні	Об'їзд виконано
3	Так	Праворуч	Ні	Об'їзд виконано
4	Так	Ліворуч	Ні	Об'їзд виконано
5	Так	Праворуч	Так	Після короткочасного контакту робот продовжив рух

У всіх п'яти спробах робот виявив перешкоду та виконав спробу об'їзду. У чотирьох випадках маневр було завершено без контакту з об'єктом, а в одній спробі зафіксовано короткочасне зіткнення. Цей випадок пов'язаний не з повною відмовою алгоритму, а з обмеженням сенсорної системи: під час складного розташування перешкод вузький об'єкт не був достатньо рано й надійно підтверджений ультразвуковим/боковим контролем. Водночас навіть після контакту програма не перейшла у критичний збій і завершила маневр.

4.5 Дослідження об'їзду декількох перешкод

Окремо було перевірено роботу алгоритму в складнішому сценарії, коли одна перешкода розташовувалася попереду робота, а інша - збоку. Такий дослід дає змогу оцінити, чи здатна система не лише зупинитися перед

перешкодою, а й вибрати вільний напрямок з урахуванням обмеженого простору.

Таблиця 4.6 – Результати об'їзду декількох перешкод

Умова експерименту	Результат	Коментар
Перешкода попереду та збоку	Об'їзд виконано	Робот обирав вільний напрямок для маневру.
Заблокований один із бокових напрямків	Рух у заблоковану сторону не виконувався	Алгоритм враховував поточні показники датчиків.
Декілька перешкод поруч	Зациклення не зафіксовано	Після маневру робот продовжував рух у допустимому напрямку.

Результати досліду показали, що алгоритм об'їзду працює не лише для одиничної перешкоди, а й у випадках, коли частина простору поруч із роботом уже заблокована. Це важливо для кімнатних сценаріїв, у яких об'єкти часто розміщені не ізольовано, а групами.

4.6 Дослідження автоматичного режиму AUTO

Для перевірки автономного режиму руху було проведено експеримент із кількома перешкодами в робочій зоні. У режимі AUTO робот рухався без заданої кінцевої координати, самостійно аналізував передній, бокові та задній напрямки й змінював траєкторію залежно від локальної ситуації.

Таблиця 4.7 – Результати перевірки режиму AUTO

Показник	Результат
Виявлення перешкод	Виконувалося в більшості випадків
Уникнення перешкод	Переважно успішне
Випадки зіткнення	1 випадок
Загальна працездатність режиму	Підтверджена

Під час перевірки режим AUTO у більшості випадків коректно реагував на перешкоди та змінював напрямок руху. Це підтверджує, що навіть без складного планування режим придатний для базового автономного переміщення в приміщенні. Водночас окремий випадок зіткнення показує, що при складному розташуванні вузьких об'єктів або недостатньому запасі часу на гальмування алгоритм може помилятися.

4.7 Дослідження впливу відстані та перешкод на рівень RSSI

Окремим етапом було досліджено роботу режиму пошуку телефона за рівнем RSSI. Під RSSI розуміється рівень прийнятого Wi-Fi сигналу: чим ближче значення до нуля, тим сильніший сигнал. Метою досліду було перевірити, як змінюється RSSI залежно від відстані між телефоном і роботом та наявності перешкод між ними.

Таблиця 4.8 – Вплив відстані та перешкод на рівень RSSI

Відстань	Без перешкоди, dBm	Мала перешкода, dBm	Велика перешкода, dBm
20 см	-49	-53	-52
50 см	-56	-60	-60...-62
80 см	-57	-58	-64

Результати показали, що у відкритішому просторі сигнал загалом слабшав зі збільшенням відстані, однак ця зміна не була лінійною. Наприклад, різниця між 50 см і 80 см без перешкоди виявилася порівняно невеликою, хоча фізична відстань зросла. Це пояснюється впливом відбиттів від стін, підлоги, меблів та інших об'єктів у кімнаті.

За наявності перешкод нестабільність RSSI ставала ще помітнішою. У ряді випадків сигнал послаблювався очікувано, однак в окремих серіях різниця між малою та великою перешкодою була не настільки вираженою, як можна було б припустити. Отже, RSSI не може розглядатися як точний параметр відстані; це лише орієнтовний індикатор наближення або віддалення від джерела сигналу.

4.8 Дослідження впливу орієнтації робота на RSSI

Додатково було перевірено вплив орієнтації робота відносно телефона на рівень RSSI. Для цього телефон розміщувався на відстані 50 см, після чого робот по черзі орієнтувався передом, задом, правим та лівим боком до телефона.

Таблиця 4.9 – Вплив орієнтації робота на рівень RSSI при відстані 50 см

Орієнтація робота відносно телефона	**RSSI, dBm**
Передня частина до телефона	-58
Задня частина до телефона	-52
Правим боком до телефона	-57
Лівим боком до телефона	-50

Найсильніший сигнал у цьому досліді було зафіксовано при орієнтації робота лівим боком до телефона, тоді як положення передом до телефона не дало найкращого результату. Це підтверджує, що RSSI залежить не лише від геометричної відстані, а й від положення антени ESP32, конструкції корпусу та перевідбиття сигналу в приміщенні.

Отже, орієнтація робота суттєво впливає на рівень RSSI. Саме тому цей параметр не може використовуватися як надійний показник точного напряму на телефон і в роботі розглядається лише як допоміжний критерій пошуку.

ВИСНОВКИ

Шляхом розроблення та експериментальної перевірки роботизованої транспортної платформи встановлено, що поєднання мікроконтролера ESP32, енкодерів, інерціального модуля, датчиків відстані та вебінтерфейсу оператора дозволяє реалізувати базову автономну навігацію в умовах структурованого приміщення. Розроблена система забезпечує дистанційне керування, рух до заданої координати в режимі GOTO, виявлення перешкод, побудову спрощеної карти середовища, автоматичний режим руху та допоміжний режим пошуку цільового пристрою за рівнем RSSI.

У результаті модернізації початкової конструкції було підвищено функціональні можливості платформи порівняно з початковою моделлю на базі Arduino Uno. Перехід до ESP32 дав змогу реалізувати вебінтерфейс, оброблення більшої кількості сенсорних даних, координатний режим руху, збереження інформації про перешкоди та передавання оператору поточного стану робота. Це підтвердило доцільність використання ESP32 як центрального керуючого модуля для навчально-дослідної мобільної платформи.

Експериментальна перевірка режиму GOTO показала, що робот здатний виконувати переміщення до заданих координат із прийнятною для навчально-дослідної платформи точністю. Середня абсолютна похибка переміщення в проведених дослідах знаходилася в межах від 1,45 см до 4,98 см, а максимальна абсолютна похибка не перевищувала 9 см. Встановлено, що під час руху вздовж осі Y похибка є меншою, оскільки такий рух переважно прямолінійний. Під час руху вздовж осі X похибка зростає через необхідність повороту, накопичення похибок орієнтації, нерівномірність роботи приводів і можливе проковзування коліс.

Дослідження системи виявлення перешкод показало, що платформа надійно реагує на достатньо великі об’єкти перед роботом, зупиняється перед ними, позначає перешкоди на карті та може виконувати їх об’їзд у режимі GOTO. Водночас встановлено, що вузькі, бокові або складні за формою

перешкоди можуть виявлятися менш стабільно. Це пов’язано з обмеженою зоною огляду датчиків, дискретним характером карти та геометричними розмірами самої платформи.

Автоматичний режим AUTO підтвердив працездатність базової реактивної поведінки робота, за якої платформа самостійно рухається, контролює простір перед собою та змінює напрямок у разі виявлення перешкод. Однак цей режим не забезпечує руху до конкретної цілі й має допоміжний характер, оскільки рішення приймаються переважно на основі поточних показів сенсорів без повноцінного планування маршруту.

Шляхом дослідження рівня бездротового сигналу RSSI встановлено, що цей параметр може використовуватися лише як допоміжна ознака наближення до цільового пристрою. Результати показали, що RSSI залежить не тільки від відстані до телефона, а й від орієнтації робота, положення антени ESP32, наявності перешкод і перевідбиття сигналу в приміщенні. Тому режим PHONE не може розглядатися як точний метод локалізації, але демонструє можливість використання бездротового сигналу як додаткового інформаційного каналу для мобільної роботизованої платформи.

Отримані результати підтверджують, що розроблена платформа є працездатним навчально-дослідним зразком для перевірки базових принципів мобільної робототехніки: керування приводами, оброблення сенсорних даних, оцінювання положення, контролю орієнтації, виявлення перешкод, побудови локальної карти та взаємодії з оператором через вебінтерфейс. Поставлену мету роботи досягнуто.

Разом з тим розроблена система не є повноцінною навігаційною системою рівня SLAM і має природні обмеження щодо точності позиціювання, стабільності бокового контролю та роботи у складному або динамічному середовищі. Подальше вдосконалення платформи доцільно спрямувати на підвищення точності одометрії, покращення фільтрації даних IMU, розширення зони огляду сенсорів, удосконалення алгоритмів планування маршруту та використання більш надійних методів локалізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Correll N., Hayes B., Heckman C., Roncone A. Introduction to Autonomous Robots. LibreTexts, 2025. С. 1-6.
2. IEEE Spectrum. Meet Roomba's Ancestor: the Cybernetic Tortoise [Electronic resource].
3. Computer History Museum. Robots Are a Few of My Favorite Things [Electronic resource].
4. SRI International. Shakey the Robot [Electronic resource].
5. Wikimedia Commons. SRI Shakey robot, 1969, Computer History Museum [Image].
6. Ministry of Defence of Ukraine. Over 9,000 frontline missions in March: the Defence Forces continue to expand the use of unmanned ground vehicles [Electronic resource].
7. Міністерство оборони України. Міноборони прискорює постачання наземних роботизованих комплексів для фронту [Електронний ресурс].
8. DOU. Наземні роботизовані комплекси: досвід застосування [Електронний ресурс].
9. United Nations Development Programme in Ukraine. Six new demining machines to boost Ukraine's mine clearance efforts [Electronic resource].
10. Wikimedia Commons. Demining robot DIGITAL VANGUARD-S (2023-11-03) 01 [Image].
11. Wikimedia Commons. Packbot [Image].
12. Andreasson H., Grisetti G., Stoyanov T., Pretto A. Sensors for Mobile Robots. arXiv, 2022. С. 2-6.
13. Borenstein J., Everett H. R., Feng L. Where Am I? Sensors and Methods for Mobile Robot Positioning. University of Michigan, 1996. С. 1-9.
14. LaValle S. M. Planning Algorithms. Cambridge University Press, 2006. С. 1-8, 185-192.
15. Fox D., Burgard W., Thrun S. The Dynamic Window Approach to Collision Avoidance. IEEE Robotics & Automation Magazine, 1997. С. 23-33.
16. Bahl P., Padmanabhan V. N. RADAR: An In-Building RF-Based User Location and Tracking System. IEEE INFOCOM, 2000. С. 775-784.
17. Texas Instruments. LM393 Dual Differential Comparator Datasheet. С. 1-5.
18. Arduino. Arduino Uno Rev3. С. 1-3.
19. STMicroelectronics. L298 Dual Full-Bridge Driver Datasheet. С. 1-4.
20. ElecFreaks. Ultrasonic Ranging Module HC-SR04 Datasheet. С. 1-2.
21. Components101. IR Sensor Module Pinout, Features & Datasheet [Electronic resource].
22. STMicroelectronics. VL53L1X Time-of-Flight Ranging Sensor Datasheet. С. 1-3.
23. STMicroelectronics. VL53L0X Time-of-Flight Ranging Sensor Datasheet. С. 1-3.
24. Espressif Systems. ESP32-WROOM-32 Datasheet. С. 1-5.
25. STMicroelectronics. LSM6DS3 iNEMO Inertial Module Datasheet. С. 1-5.

26. NXP Semiconductors. PCA9548A 8-channel I2C-bus switch with reset Datasheet. C. 1-4.

ДОДАТОК А

Деталізовані результати експериментальної перевірки режиму GOTO

У додатку наведено дві великі таблиці з повними серіями вимірювань, які перенесено з четвертого розділу, щоб розвантажити основний текст роботи та зберегти зручність його читання.

Таблиця А.1 – Деталізація вимірювань для руху по осі Y

Координата	№ спроби	Контрольне значення, см	Фактично, см	ΔS, см	|ΔS|, см	Бокове відхилення, см
Y=1	1	16	19	3	3	0
Y=1	2	16	16,50	0,50	0,50	0
Y=1	3	16	17,50	1,50	1,50	0
Y=1	4	16	21	5	5	0
Y=1	5	16	16,50	0,50	0,50	0
Y=1	6	16	21,50	5,50	5,50	0
Y=1	7	16	20	4	4	0
Y=1	8	16	17,50	1,50	1,50	0
Y=1	9	16	17,50	1,50	1,50	0
Y=1	10	16	18,50	2,50	2,50	0
Y=4	1	64	60	-4	4	3
Y=4	2	64	61	-3	3	0
Y=4	3	64	63	-1	1	1
Y=4	4	64	64	0	0	2
Y=4	5	64	64	0	0	2
Y=4	6	64	64	0	0	0
Y=4	7	64	64	0	0	0
Y=4	8	64	65	1	1	0
Y=4	9	64	60	-4	4	0
Y=4	10	64	60	-4	4	1
Y=6	1	96	96	0	0	1
Y=6	2	96	98	2	2	0
Y=6	3	96	98,50	2,50	2,50	1,50
Y=6	4	96	95	-1	1	0
Y=6	5	96	100	4	4	0
Y=6	6	96	96,50	0,50	0,50	0
Y=6	7	96	98,50	2,50	2,50	0
Y=6	8	96	96	0	0	0
Y=6	9	96	98	2	2	0,50
Y=6	10	96	96	0	0	1

Таблиця А.2 – Деталізація вимірювань для руху по осі X

Координата	№ спроби	Контрольне значення, см	Фактично, см	ΔS, см	|ΔS|, см	Бокове відхилення, см
X=1	1	16	15,50	-0,50	0,50	0
X=1	2	16	13,30	-2,70	2,70	0
X=1	3	16	17	1	1	0
X=1	4	16	19	3	3	0
X=1	5	16	16,50	0,50	0,50	0
X=1	6	16	19	3	3	0
X=1	7	16	19	3	3	0

X=1	8	16	17,50	1,50	1,50	0
X=1	9	16	15	-1	1	0
X=1	10	16	17	1	1	0
X=4	1	64	61,50	-2,50	2,50	0
X=4	2	64	66	2	2	0
X=4	3	64	60,50	-3,50	3,50	0
X=4	4	64	57	-7	7	0
X=4	5	64	58,50	-5,50	5,50	0,50
X=4	6	64	59,50	-4,50	4,50	2
X=4	7	64	58,50	-5,50	5,50	0,50
X=4	8	64	61,50	-2,50	2,50	3
X=4	9	64	61,50	-2,50	2,50	0
X=4	10	64	60,20	-3,80	3,80	0
X=6	1	96	88,20	-7,80	7,80	2
X=6	2	96	91	-5	5	1
X=6	3	96	92	-4	4	4
X=6	4	96	90	-6	6	1
X=6	5	96	96	0	0	0
X=6	6	96	96	0	0	0
X=6	7	96	88	-8	8	3
X=6	8	96	92	-4	4	3
X=6	9	96	87	-9	9	0
X=6	10	96	90	-6	6	0

ДОДАТОК Б

Основні фрагменти програмного коду системи керування

У додатку наведено лише ті фрагменти програми, які безпосередньо відображають логіку роботи веб-інтерфейсу, оновлення карти, обробки команд оператора, побудови маршруту GOTO та фіксації перешкод.

Фрагмент Б.1 – Налаштування HTTP-маршрутів веб-інтерфейсу

```
      server.on("/", handleRoot);
  server.on("/pose", handlePose);
  server.on("/cmd", handleCmd);
  server.on("/goto", handleGoto);
  server.on("/beep/test", handleBuzzerTest);
  server.on("/phone/search", handlePhoneSearch);
  server.on("/home/set", handleHomeSet);
  server.on("/home/go", handleHomeGo);
  server.on("/angle/cal", handleAngleCal);
  server.on("/map/clear", handleMapClear);
```

Фрагмент Б.2 – Формування JSON-стану для карти і телеметрії

```
      void handlePose() {
  server.sendHeader("Cache-Control", "no-store");
  syncLaserCacheFromGoto();
  updatePhoneRssiIfNeeded();
  float mapX = (float)viewX();
  float mapY = (float)viewY();
  float mapHeading = dirIndexToMapHeadingDeg(dirIndex);
  if (::mode == 2) {
    mapX = GotoNav::navGridXCoord();
    mapY = GotoNav::navGridYCoord();
    mapHeading = GotoNav::headingDeg;
  }
  String json;
  json.reserve(1900);
  json += "{";
  json += "\"mode\":"; json += String(mode); json += ",";
  json += "\"modeName\":\""; json += GotoNav::modeStr(); json += "\",";
  json += "\"gotoState\":\""; json += GotoNav::gotoStateStr(); json += "\",";
  json += "\"userBypassState\":\""; json += GotoNav::userBypassStateStr(); json +=
"\",";
  json += "\"systemStatus\":\""; json += GotoNav::systemStatusStr(); json += "\",";
  json += "\"hasTarget\":"; json += GotoNav::hasTarget() ? "true" : "false"; json +=
",";
  json += "\"x\":"; json += String(mapX, 2); json += ",";
  json += "\"y\":"; json += String(mapY, 2); json += ",";
  json += "\"physicalX\":"; json += String(GotoNav::viewXCoord(), 2); json += ",";
  json += "\"physicalY\":"; json += String(GotoNav::viewYCoord(), 2); json += ",";
  json += "\"navX\":"; json += String(GotoNav::navGridXCoord(), 2); json += ",";
  json += "\"navY\":"; json += String(GotoNav::navGridYCoord(), 2); json += ",";
  json += "\"tx\":"; json += String(GotoNav::targetXCoord(), 2); json += ",";
  json += "\"ty\":"; json += String(GotoNav::targetYCoord(), 2); json += ",";
  json += "\"navTx\":"; json += String(GotoNav::navTargetXCoord(), 2); json += ",";
  json += "\"navTy\":"; json += String(GotoNav::navTargetYCoord(), 2); json += ",";
  json += "\"heading\":"; json += String(GotoNav::headingDeg, 2); json += ",";
  json += "\"mapHeading\":"; json += String(mapHeading, 2); json += ",";
  json += "\"dirIndex\":"; json += String(dirIndex); json += ",";
  json += "\"gyroRawDps\":"; json += String(GotoNav::imuGyroZRawDps, 3); json += ",";
  json += "\"gyroCorrectedDps\":"; json += String(GotoNav::imuGyroZCorrectedDps, 3);
json += ",";
  json += "\"yawLocked\":"; json += GotoNav::imuYawLockedByStillness ? "true" :
"false"; json += ",";
  json += "\"imuCalibrating\":"; json += GotoNav::imuCalibrating ? "true" : "false";
json += ",";
  json += "\"targetHeading\":"; json += String(GotoNav::gotoCommandHeadingDeg, 2);
```

```
json += ",";
  json += "\"remainingCells\":"; json += String(GotoNav::remainingCoord(), 2); json +=
",";
  json += "\"remainingCm\":"; json += String(GotoNav::remainingCm(), 1); json += ",";
  json += "\"physicalRemainingCm\":"; json += String(GotoNav::physicalRemainingCm(),
1); json += ",";
  json += "\"frontCm\":"; json += String(cachedLaser); json += ",";
  json += "\"leftCm\":"; json += String(cachedLaserLeft); json += ",";
  json += "\"rightCm\":"; json += String(cachedLaserRight); json += ",";
  json += "\"backCm\":"; json += String(cachedLaserBack); json += ",";
  json += "\"cliff\":"; json += cliff ? "true" : "false"; json += ",";
  json += "\"cliffRaw\":"; json += cliffRawNow ? "true" : "false"; json += ",";
  json += "\"obstacleNow\":"; json += GotoNav::obstacleNow() ? "true" : "false"; json
+= ",";
  json += "\"phoneSearch\":"; json += phoneSearchActive ? "true" : "false"; json +=
",";
  json += "\"phoneState\":\""; json += phoneSearchStateStr(); json += "\",";
  json += "\"phoneRssiDbm\":"; json += String(phoneRssiDbm); json += ",";
  json += "\"phoneRssiFiltered\":"; json += String(phoneRssiFiltered, 1); json += ",";
  json += "\"phoneBestRssiDbm\":"; json += String(phoneBestRssiDbm); json += ",";
  json += "\"phoneStationCount\":"; json += String(phoneStationCount); json += ",";
  json += "\"phoneRssiValid\":"; json += phoneRssiValid ? "true" : "false"; json +=
",";
  json += "\"phoneDir\":\""; json += phoneDirStr(); json += "\",";
  json += "\"phoneStep\":"; json += String(phoneStepCounter); json += ",";
  json += "\"phoneVisitedCount\":"; json += String(phoneVisitedCount); json += ",";
  json += "\"phoneBlockedStreak\":"; json += String(phoneBlockedStreak); json += ",";
  json += "\"phoneTrailEpoch\":"; json += String(phoneTrailEpoch); json += ",";
  json += "\"phoneLiveMapActive\":"; json += phoneLiveMapActive ? "true" : "false";
json += ",";
  json += "\"homeX\":0,";
  json += "\"homeY\":0,";
  json += "\"obstacleCount\":"; json += String(GotoNav::obstacleMemoryCount); json +=
",";
  json += "\"targetBlocked\":"; json += GotoNav::navTargetBlocked ? "true" : "false";
json += ",";
  json += "\"robotWidthCm\":"; json += String(GotoNav::ROBOT_BODY_DISPLAY_WIDTH_CM,
1); json += ",";
  json += "\"robotLengthCm\":"; json += String(GotoNav::ROBOT_BODY_DISPLAY_LENGTH_CM,
1); json += ",";
  json += "\"obstacles\":[";
  for (uint8_t i = 0; i < GotoNav::obstacleMemoryCount; ++i) {
    if (i) json += ",";
    json += "{\"x\":"; json += String(GotoNav::obstacleMemory[i].x);
    json += ",\"y\":"; json += String(GotoNav::obstacleMemory[i].y);
    json += ",\"hits\":"; json += String(GotoNav::obstacleMemory[i].hits);
    json += "}";
  }
  json += "]";
  json += "}";
  server.send(200, "application/json; charset=utf-8", json);
}
```

Фрагмент Б.3 – Обробка команд ANGLE CAL, GOTO, SET HOME та GO HOME

```
      void handleAngleCal() {
  server.sendHeader("Cache-Control", "no-store");
  bool ok = GotoNav::requestIdleGyroBiasCalibration("web-button");
  server.send(ok ? 200 : 409, "text/plain; charset=utf-8", ok ? "ANGLE CAL STARTED" :
"ANGLE CAL REJECTED: stop robot and cancel GOTO");
}
void handleGoto() {
  server.sendHeader("Cache-Control", "no-store");
  if (server.hasArg("x") && server.hasArg("y")) {
    String sx = server.arg("x");
    String sy = server.arg("y");
    sx.trim();
    sy.trim();
```

```
    if (sx.length() == 0 || sy.length() == 0) {
      server.send(400, "text/plain; charset=utf-8", "Bad GOTO: empty x/y");
      return;
    }
    float vx = sx.toFloat();
    float vy = sy.toFloat();
    phoneSearchStopOnly();
    int previousMode = mode;
    mode = 2;
    manualDrive = M_STOP;
    stopMotors();
    servoHome();
    GotoNav::startGotoCoord(vx, vy);
    server.send(200, "text/plain; charset=utf-8", "GOTO SET");
    return;
  }
  server.send(400, "text/plain; charset=utf-8", "Need x and y");
}
void handleHomeSet() {
  phoneSearchStopOnly();
  stopMotors();
  manualDrive = M_STOP;
  if (mode == 2 || GotoNav::hasTarget()) {
    GotoNav::syncLegacyGridFromNavPose();
  }
  homeRawX = rawX;
  homeRawY = rawY;
  prefs.putLong("homeRawX", homeRawX);
  prefs.putLong("homeRawY", homeRawY);
  dirIndex = 0;
  resetManualPoseTimer();
  GotoNav::setHomeHere();
  phoneLiveMapActive = false;
  phoneLiveMapX = 0.0f;
  phoneLiveMapY = 0.0f;
  GotoNav::clearObstacleMemory();
  phoneClearVisitedMemory();
  mapResetEpoch++;
  phoneTrailEpoch++;
  server.sendHeader("Cache-Control", "no-store");
  server.send(200, "text/plain; charset=utf-8", "HOME SET AT CURRENT ROBOT POSITION");
}
void handleHomeGo() {
  phoneSearchStopOnly();
  int previousMode = mode;
  mode = 2;
  manualDrive = M_STOP;
  stopMotors();
  servoHome();
  GotoNav::goHome();
  server.sendHeader("Cache-Control", "no-store");
  server.send(200, "text/plain; charset=utf-8", "GO HOME");
}
```

Фрагмент Б.4 – Скидання домашньої позиції та локальної системи координат

```
      void setHomeHere() {
  navGridX = 0.0f;
  navGridY = 0.0f;
  gotoTargetGridX = 0.0f;
  gotoTargetGridY = 0.0f;
  requestedTargetGridX = 0.0f;
  requestedTargetGridY = 0.0f;
  navApproachTargetActive = false;
  homePosX = 0.0f;
  homePosY = 0.0f;
  posX = 0.0f;
  posY = 0.0f;
```

```
  syncLegacyGridFromNavPose();
  idleHeadingZeroMode = true;
  navHaveLastSegmentHeading = false;
  navLastSegmentHeadingDeg = 0.0f;
  gotoPlanCount = 0;
  gotoPlanIndex = 0;
  resetHeadingReference();
  if (!haveTarget && imuReady && !imuCalibrating) {
    startIMUCalibration(true);
  }
}
```

Фрагмент Б.5 – Побудова дискретного маршруту в режимі GOTO

```
      bool buildDiscreteGotoPlan() {
  gridBugPlanActive = false;
  resetGridBugActiveObstacleLock();
  gotoPlanCount = 0;
  gotoPlanIndex = 0;
  gotoPlanStartGridX = navGridX;
  gotoPlanStartGridY = navGridY;
  long startX = lroundf(navGridX);
  long startY = lroundf(navGridY);
  if (NAV_MEMORY_ROUTE_ENABLED && obstacleMemoryCount > 0) {
    if (buildMemoryAwareRoutePlan("v151-build-discrete-memory-route")) {
      return true;
    }
    gotoPlanCount = 0;
    gotoPlanIndex = 0;
    gotoPlanStartGridX = navGridX;
    gotoPlanStartGridY = navGridY;
  }
  if (obstacleMemoryBlocked(lroundf(requestedTargetGridX),
lroundf(requestedTargetGridY))) {
    if (!applyOccupiedTargetApproachIfNeeded("build-discrete-plan")) {
      gotoPlanCount = 0;
      gotoPlanIndex = 0;
      gotoState = GOTO_BLOCKED;
      return false;
    }
  } else {
    navApproachTargetActive = false;
    gotoTargetGridX = requestedTargetGridX;
    gotoTargetGridY = requestedTargetGridY;
    gotoFinalTargetX = homePosX + cellsToCm(gotoTargetGridX);
    gotoFinalTargetY = homePosY + cellsToCm(gotoTargetGridY);
  }
  long targetX = lroundf(gotoTargetGridX);
  long targetY = lroundf(gotoTargetGridY);
  long dy = targetY - startY;
  long dx = targetX - startX;
  if (obstacleMemoryCount > 0) {
  }
  bool yFirst = true;
  if (NAV_NO_IMMEDIATE_OPPOSITE_MOVES) {
    bool yOpposite = (dy != 0 && navLastCompletedMoveDy != 0 && ((dy > 0) !=
(navLastCompletedMoveDy > 0)));
    bool xAvailable = (dx != 0);
    if (yOpposite && xAvailable) {
      yFirst = false;
    }
  }
  if (yFirst) {
    if (!addAxisPlan(AXIS_Y, dy)) return false;
    if (!addAxisPlan(AXIS_X, dx)) return false;
  } else {
    if (!addAxisPlan(AXIS_X, dx)) return false;
    if (!addAxisPlan(AXIS_Y, dy)) return false;
  }
  for (uint8_t i = 0; i < gotoPlanCount; ++i) {
```

```
  }
  return gotoPlanCount > 0;
}
struct GridVector {
  int dx;
  int dy;
};
```

Фрагмент Б.6 – Запам'ятовування перешкод на карті

```
       void rememberObstacleCell(long x, long y, const char *reason) {
  if (!NAV_OBSTACLE_MEMORY_ENABLED) return;
  if (NAV_CONSERVATIVE_OBSTACLE_MAPPING && reason) {
    bool isPhysicalProjection = strstr(reason, "physical") != nullptr;
    bool isSideProjection = strstr(reason, "left-side") != nullptr || strstr(reason,
"right-side") != nullptr ||
                            strstr(reason, "left-trigger") != nullptr ||
strstr(reason, "right-trigger") != nullptr;
    if ((isPhysicalProjection && !NAV_MAP_PHYSICAL_SENSOR_MARKERS) ||
        (isSideProjection && !NAV_MAP_SIDE_SENSOR_MARKERS)) {
      return;
    }
  }
  if (reason && (strstr(reason, "front") != nullptr || strstr(reason, "map-") !=
nullptr || strstr(reason, "tof") != nullptr || strstr(reason, "center") != nullptr ||
strstr(reason, "ultra") != nullptr)) {
    bool forceLocalMarker = (strstr(reason, "force-local") != nullptr);
    int nearIdx = -1;
    if (forceLocalMarker) {
      nearIdx = findObstacleMemoryCell(x, y);
    } else {
      for (uint8_t i = 0; i < obstacleMemoryCount; ++i) {
        if (labs(x - obstacleMemory[i].x) <= NAV_OBSTACLE_MEMORY_MERGE_CELLS &&
            labs(y - obstacleMemory[i].y) <= NAV_OBSTACLE_MEMORY_MERGE_CELLS) {
          nearIdx = i;
          break;
        }
      }
      if (nearIdx < 0) {
        long rx = lroundf(viewXCoord());
        long ry = lroundf(viewYCoord());
        nearIdx = findExistingObstacleOnSameViewRay(x, y, rx, ry);
      }
    }
    int idx = nearIdx;
    if (idx < 0) {
      if (obstacleMemoryCount >= NAV_OBSTACLE_MEMORY_MAX) {
        return;
      }
      idx = obstacleMemoryCount++;
      obstacleMemory[idx].x = x;
      obstacleMemory[idx].y = y;
      obstacleMemory[idx].hits = 0;
    }
    if (obstacleMemory[idx].hits < NAV_OBSTACLE_MEMORY_HIT_LIMIT)
obstacleMemory[idx].hits++;
    obstacleMemory[idx].lastSeenMs = millis();
    obstacleMemoryLastUpdateMs = millis();
    return;
  }
  bool forceLocalMarker = (reason && strstr(reason, "force-local") != nullptr);
  int idx = findObstacleMemoryCell(x, y);
  if (idx < 0 && !forceLocalMarker) {
    for (uint8_t i = 0; i < obstacleMemoryCount; ++i) {
      if (labs(x - obstacleMemory[i].x) <= NAV_OBSTACLE_MEMORY_MERGE_CELLS &&
          labs(y - obstacleMemory[i].y) <= NAV_OBSTACLE_MEMORY_MERGE_CELLS) {
        idx = i;
        break;
      }
    }
```

```
  }
  if (idx >= 0) {
    if (obstacleMemory[idx].hits < NAV_OBSTACLE_MEMORY_HIT_LIMIT)
obstacleMemory[idx].hits++;
    obstacleMemory[idx].lastSeenMs = millis();
  } else {
    if (obstacleMemoryCount < NAV_OBSTACLE_MEMORY_MAX) {
      idx = obstacleMemoryCount++;
    } else {
      return;
    }
    obstacleMemory[idx].x = x;
    obstacleMemory[idx].y = y;
    obstacleMemory[idx].hits = 1;
    obstacleMemory[idx].lastSeenMs = millis();
  }
  obstacleMemoryLastUpdateMs = millis();
}
```