Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризированих технологій
(повна назва)
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації
(повна назва)
та робототехніки

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)
(рівень вищої освіти)

Розробка гусеничної роботизованої платформи на базі Raspberry PI
(тема)

Виконав:
Студент 3 (прискореного) курсу,
групи АКТСІу-21-1
Миронов Юрій Дмитрович
(прізвище, ім'я, по батькові)
Спеціальності 151 Автоматизація та
комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми Освітньо-професійна
Освітня програма Системна інженерія
(назва)
Керівник доц. Янушкевич Д.А.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. Кафедри КІТАР _____
(підпис)
«____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Миронову Юрію дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка гусеничної роботизованої платформи на базі Raspberry PI

Затверджена наказом університету від 20.05.2024 р. № 478 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 2024 р
3. Вихідні дані до роботи:
3.1 Одноплатні комп'ютери: Raspberry Pi 3 B+ ;
3.2 Середовище розробки : Raspbian OS,
3.3 Мова програмування Python;
3.4 Можливість апаратної реалізації

4. Перелік питань, що потрібно опрацювати в роботі:
4.1 Вступ;
4.2 Визначення мети , об'єкту та предмету розробки;
4.3 Аналіз предметної області;
4.4 Розробка схеми та вибір обладнання для наземної роботизованої платформи;
4.5 Розробка програмного забезпечення для наземної роботизованої платформи;
4.6 Висновки.

5. Графічний демонстраційний матеріал в форматі powerpoint(*.ppt) формату А4 –15 сторінок.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		Підпис	Дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Актуальність роботи, постановка задачі	22.04.24 – 30.04.24	виконано
2	Визначення мети об'єкту і предмету розробки	01.05.24 – 08.05.24	виконано
3	Аналіз предметної області	09.05.24 – 14.05.24	виконано
4	Розробка схеми та вибір обладнання для наземної роботизованої платформи	15.05.24 – 25.05.24	виконано
5	Розробка програмного забезпечення для наземної роботизованої платформи	26.05.24 – 30.05.24	виконано
6	Подання роботи на перевірку Інтернет-сервісом Unicheck	31.05.24 – 04.06.24	виконано
7	Оформлення пояснювальної записки	05.06.24 – 09.06.24	виконано
8	Подання роботи на рецензію	10.06.24 – 13.06.24	виконано
9	Подання роботи на підпис зав. Кафедри	14.06.24 – 16.06.24	виконано
11	Подання кваліфікаційної роботи в ЕК	17.06.24 – 20.06.24	виконано

Дата видачі завдання 08.04 2024 р.

Студент _____ Миронов Ю.Д.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Янушкевич Д.А.
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«11» липня 2024 р.  Миронов Ю.

РЕФЕРАТ

Пояснювальна записка: 66 с., 2 табл., 16 рис., 2 дод., 35 джерел

СИСТЕМА КЕРУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОБОТЕХНІЧНИЙ ЗАСІБ , КОНТРОЛЕРИ, ТЕСТУВАННЯ

Об’єкт розробки – гусенична роботизована платформа на Raspberry PI

Предмет розробки – програмний модуль для Raspberry PI

Методи дослідження – моделювання структурної схеми системи, програмування Raspberry PI, а також тестування програмного забезпечення на відповідність технічним вимогам.

Метою даної роботи є розробка гусеничної роботизованої платформи на базі Raspberry PI з урахуванням сучасних вимог до прохідності, стабільності та функціональності, а також створення програмного забезпечення для ефективного управління та обробки даних з сенсорів.

В кваліфікаційній роботі проведено аналіз існуючих аналогів та технологічних рішень у сфері наземних робототехнічних засобів. Розроблено структурну схему робототехнічного зособу та створено програмне забезпечення на мові Python для керування робототехнічним засобом. Після цього було проведено тестування та оптимізацію розробленої системи. У кваліфікаційній роботі проведено аналіз існуючих аналогів та технологічних рішень у сфері наземних робототехнічних засобів. Розроблено структурну схему робототехнічного засобу та створено програмне забезпечення на мові Python для керування робототехнічним засобом. Після цього було проведено тестування та оптимізацію розробленої системи.

ABSTRACT

Explanatory note: 66 p., 2 tables, 16 figures, 2 appendix, 35 sources

CONTROL SYSTEM, SOFTWARE, ROBOTIC VEHICLE, CONTROLLERS, TESTING

Object of development - a crawler robotic platform on Raspberry PI

Subject of development - software module for Raspberry PI

Research methods - modeling of the system's block diagram, programming of the Raspberry PI, as well as testing of the software for compliance with technical requirements.

The purpose of this work is to develop a Raspberry PI-based crawler robotic platform taking into account modern requirements for cross-country ability, stability and functionality, as well as to create software for efficient control and processing of sensor data.

The qualification work analyzes existing analogs and technological solutions in the field of ground robotics. A structural diagram of the robotic vehicle was developed and Python software was created to control the robotic vehicle. After that, the developed system was tested and optimized. The qualification work analyzed existing analogs and technological solutions in the field of ground robotic vehicles. A structural diagram of the robotic vehicle was developed and Python software was created to control the robotic vehicle. After that, the developed system was tested and optimized

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КІТАР – комп’ютерно-інтегрованих технологій, автоматизації та робототехніки;

НРЗ – наземні робототехнічні засоби;

ПЗ – програмне забезпечення;

ХНУРЕ – Харківський національний університет радіоелектроніки;

ШІ – штучний інтелект;

API –інтерфейс програмування застосувань;

FPGA – Field-Programmable Gate Array;

GPIO – універсальні входи/виходи;

IDE – інтегроване середовище розробки;

OS – операційна система;

PWM – широтно-імпульсна модуляція;

ROS – операційна система для роботів;

SBC – Single-Board Computer.

ВСТУП

Розвиток наземних робототехнічних засобів (НРЗ) є ключовим напрямком сучасної інженерії, оскільки ці засоби знаходять застосування в різних галузях, включаючи промисловість, сільське господарство, військову справу та рятувальні операції. Ефективність та надійність НРЗ безпосередньо залежать від їх конструкційних рішень, програмного забезпечення та інтеграції додаткового обладнання. Вивчення та вдосконалення цих аспектів є важливим завданням для забезпечення високої продуктивності та безпеки робототехнічних систем.

Метою даної роботи є розробка гусеничної роботизованої платформи на базі Raspberry PI з урахуванням сучасних вимог до прохідності, стабільності та функціональності, а також створення програмного забезпечення для ефективного управління та обробки даних з сенсорів.

Об'єктом дослідження є гусенична роботизована платформа, призначена для виконання різноманітних завдань у складних умовах експлуатації. Предметом дослідження є апаратні та програмні компоненти платформи, включаючи шасі, сенсори, системи живлення та програмне забезпечення.

Предмет розробки – програмний модуль системи автоматизації робототехнічних комплексів та гусеничне шасі.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати сучасний стан наземних робототехнічних засобів;
– визначити класифікаційні ознаки наземних робототехнічних засобів;
– розглянути сучасні назумні роботизовані;
– розглянути аналіз методів побудови;
– розробити програмний модуль для наземного робототехнічного засобу.

Кваліфікаційна робота виконано згідно з [1–4].

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Робототехнічні засоби та їх класифікаційні ознаки

Наземні робототехнічні засоби (НРЗ) використовуються в різних галузях, включаючи промисловість, сільське господарство, військову справу та рятувальні операції. Аналіз існуючих рішень у цій сфері дозволяє виділити основні напрямки розвитку та визначити переваги і недоліки різних конструкцій і технологій. Одним з найпоширеніших типів НРЗ є мобільні роботи з колісною базою. Колісні роботи відрізняються високою маневреністю та швидкістю пересування на рівних поверхнях, але мають обмежену прохідність на складних і пересічених місцевостях. Вони широко застосовуються в промислових умовах для транспортування вантажів, проведення інспекцій та виконання складських операцій. Завдяки своїй простоті та ефективності, колісні роботи стали основою для багатьох логістичних систем, де важливі точність і швидкість виконання завдань [6].

Гусеничні роботи, на відміну від колісних, мають вищу прохідність та здатність долати складні перешкоди. Вони використовуються в умовах бездоріжжя, на будівельних майданчиках, у рятувальних операціях та військових завданнях. Гусеничні роботи можуть працювати в умовах, де колісні роботи не здатні функціонувати, завдяки своїй здатності пересуватися по нерівних поверхнях і долати перешкоди. Однак їхня конструкція є більш складною та енергоємною, що може впливати на тривалість автономної роботи. Гусеничні роботи часто використовуються в таких місіях, як розмінування, розвідка та евакуація, де їхня надійність та прохідність є критично важливими [7].

Крокуючі роботи мають ще більшу прохідність і можуть пересуватися

практично будь-якими поверхнями, включаючи сходи та нерівні ділянки. Їхні можливості зумовлені складною кінематикою, що дозволяє їм адаптуватися до різних умов. Такі роботи часто використовуються в дослідницьких місіях, де необхідно здійснювати переміщення по нерівним або незвіданим поверхням. Проте така конструкція є найбільш складною та вимагає значних обчислювальних ресурсів для управління рухом. Крокуючі роботи знаходять застосування в таких галузях, як космічні дослідження, рятувальні операції у важкодоступних місцях і навіть у медичній робототехніці для виконання складних хірургічних процедур [6].

Роботи зі змішаними типами бази, що поєднують колісну та крокуючу бази, також набирають популярності. Вони поєднують переваги обох типів конструкцій, забезпечуючи високу маневреність та прохідність. Такі роботи можуть змінювати спосіб пересування в залежності від умов, що дозволяє їм виконувати завдання у різноманітних середовищах. Наприклад, вони можуть використовувати колеса для швидкого пересування по рівних поверхнях і переходити на крокуючий режим для подолання перешкод або сходів. Це робить їх універсальними і здатними ефективно працювати в умовах, де жоден з окремих типів бази не може забезпечити необхідну продуктивність [7].

Наземні робототехнічні засоби також можуть класифікуватися за призначенням. Промислові роботи використовуються для автоматизації виробничих процесів, транспортування вантажів та проведення інспекцій. Такі роботи часто мають міцну конструкцію і здатні працювати в складних умовах, включаючи високу температуру, запиленість і вібрацію. Вони використовуються на заводах, складах та в інших промислових об'єктах для зменшення витрат на робочу силу та підвищення ефективності виробничих процесів. Сільськогосподарські роботи виконують завдання з обробки ґрунту, посіву, збору врожаю та догляду за рослинами. Вони допомагають автоматизувати рутинні завдання, що дозволяє фермерам зосередитися на більш складних аспектах управління господарством. Сільськогосподарські

роботи можуть бути обладнані різними сенсорами для моніторингу стану ґрунту, рослин та погодних умов, що дозволяє оптимізувати процеси вирощування та підвищити врожайність [6].

Військові роботи призначені для виконання розвідувальних, інженерних завдань та бойових операцій. Вони часто мають броньований корпус і здатні працювати в екстремальних умовах, таких як бойові дії або розмінування. Військові роботи можуть бути оснащені різними видами озброєння, сенсорами для збору розвідувальної інформації та системами для дистанційного управління. Вони значно знижують ризик для життя солдатів, виконуючи завдання в небезпечних умовах [6].

Рятувальні роботи застосовуються для пошуку та порятунку людей у надзвичайних ситуаціях. Вони можуть працювати в умовах завалів, пожеж, повеней та інших катастроф, де використання людини є небезпечним або неможливим. Рятувальні роботи можуть бути обладнані тепловізорами, датчиками газу та іншими сенсорами для пошуку постраждалих та оцінки стану навколишнього середовища. Науково-дослідницькі роботи використовуються для проведення експериментів та збору даних у важкодоступних або небезпечних місцях. Вони можуть працювати в умовах Арктики, океанів, пустель та інших екстремальних середовищах, де людина не може працювати без значних ризиків. Такі роботи обладнані різними науковими інструментами для збору зразків, проведення вимірювань та моніторингу навколишнього середовища. Побутові роботи допомагають у виконанні домашніх справ, таких як прибирання, догляд за газоном та інші. Вони стали популярними завдяки своїй здатності автоматизувати рутинні завдання, звільняючи час для більш важливих справ. Побутові роботи можуть бути обладнані різними функціями, такими як голосове управління, автоматичне заряджання та програмовані режими роботи. За способом управління НРЗ поділяються на автономні, дистанційно керовані та змішані. Автономні роботи здатні виконувати завдання без втручання людини, використовуючи алгоритми машинного навчання та

обробки даних. Вони можуть самостійно приймати рішення на основі отриманої інформації та виконувати складні завдання. Дистанційно керовані роботи потребують постійного контролю оператора, що дозволяє виконувати завдання, які вимагають високої точності та швидкості реакції. Змішані роботи можуть працювати як в автономному режимі, так і під керуванням оператора, що забезпечує гнучкість у виконанні завдань. За енергетичною автономністю НРЗ класифікуються на ті, що працюють від акумуляторів, гібридні та ті, що підключені до зовнішнього джерела живлення. Акумуляторні роботи забезпечують мобільність і автономність, але мають обмежений час роботи. Гібридні роботи використовують комбінацію акумуляторів та зовнішніх джерел живлення, що дозволяє збільшити тривалість роботи. Роботи, підключені до зовнішнього джерела живлення, мають необмежений час роботи, але їхня мобільність обмежена.

Важливим аспектом аналізу існуючих рішень є оцінка надійності та довговічності НРЗ. Конструкція та компоненти роботів повинні бути розроблені з урахуванням умов експлуатації, щоб забезпечити тривалу та безвідмовну роботу. Надійність системщоб забезпечити тривалу та безвідмовну роботу. Надійність системи включає здатність робота працювати в умовах вібрації, пилу, високих та низьких температур, а також у середовищах з агресивними речовинами.

Загалом, аналіз існуючих рішень у сфері наземних робототехнічних засобів дозволяє визначити основні напрями розвитку технологій, підвищити ефективність та надійність роботів, а також знайти оптимальні рішення для конкретних завдань. Постійне вдосконалення конструкцій та впровадження нових технологій сприяє розширенню можливостей НРЗ та їх більш широкому застосуванню в різних галузях [6-7].

1.2 Аналіз принципів побудови наземних робототехнічних засобів

Принципи побудови наземних робототехнічних засобів (НРЗ) включають кілька ключових аспектів, таких як конструкція, системи управління, джерела живлення та сенсори. Ці принципи визначають ефективність, надійність та функціональність НРЗ в різних умовах.

Основним принципом побудови конструкції НРЗ є забезпечення балансу між міцністю, легкістю та функціональністю. Конструкції можуть бути колісними, гусеничними або крокуючими, залежно від завдань та умов експлуатації. Колісні роботи мають високу маневреність та швидкість на рівних поверхнях, але обмежену прохідність на складних місцевостях. Гусеничні роботи, навпаки, мають вищу прохідність та здатність долати перешкоди, що робить їх ідеальними для бездоріжжя. Крокуючі роботи можуть пересуватися практично будь-якими поверхнями завдяки складній кінематиці, що дозволяє їм адаптуватися до різних умов [8].

Системи управління НРЗ базуються на алгоритмах, які забезпечують автономну або дистанційно керовану роботу. Для автономних роботів важливими є алгоритми машинного навчання та обробки даних, що дозволяють їм приймати рішення на основі аналізу сенсорної інформації. Дистанційно керовані роботи використовують системи зв'язку для взаємодії з оператором, що дозволяє виконувати завдання, які вимагають високої точності та швидкості реакції [9].

Джерела живлення є критичним компонентом НРЗ, оскільки вони визначають тривалість автономної роботи та мобільність роботів. Використання акумуляторних батарей забезпечує мобільність, але обмежує час роботи. Гібридні системи, що комбінують акумулятори та зовнішні джерела живлення, дозволяють збільшити тривалість роботи та зберегти мобільність. Роботи, підключені до зовнішнього джерела живлення, мають необмежений час роботи, але їхня мобільність обмежена [10].

Сенсори є очима та вухами НРЗ, забезпечуючи збір даних про навколишнє середовище. Використання різноманітних сенсорів, таких як камери, лідарами, ультразвукові сенсори та акселерометри, дозволяє роботам ефективно орієнтуватися та взаємодіяти з навколишнім світом. Камери використовуються для візуальної навігації та розпізнавання об'єктів, лідарами – для побудови тривимірних карт та визначення відстаней до перешкод, ультразвукові сенсори – для визначення відстаней до об'єктів на коротких дистанціях, а акселерометри – для вимірювання прискорення та нахилу робота [11].

Надійність та довговічність є ще одним важливим принципом побудови НРЗ. Конструкція та компоненти роботів повинні бути розроблені з урахуванням умов експлуатації, щоб забезпечити тривалу та безвідмовну роботу. Надійність системи включає здатність робота працювати в умовах вібрації, пилу, високих та низьких температур, а також у середовищах з агресивними речовинами. Загалом, аналіз існуючих рішень у сфері наземних робототехнічних засобів дозволяє визначити основні напрями розвитку технологій, підвищити ефективність та надійність роботів, а також знайти оптимальні рішення для конкретних завдань [12].

1.3 Методи та засоби керування наземними робототехнічними засобами

Керування наземними роботизованими засобами (НРЗ) включає різноманітні методи та технології, кожен з яких має свої унікальні переваги та сфери застосування. Ці методи забезпечують ефективну та безпечну роботу роботів у різних умовах, від контрольованих середовищ до складних і непередбачуваних ситуацій.

Одним з основних методів керування НРЗ є дистанційне керування. При цьому методі оператор контролює рухи та дії роботизованого засобу за допомогою пульта дистанційного керування або комп'ютера, перебуваючи на

віддаленій відстані. Цей метод особливо корисний у ситуаціях, коли необхідно забезпечити точне управління у реальному часі, наприклад, при виконанні небезпечних завдань у зоні катастроф або у важкодоступних місцях. Дистанційне керування дозволяє оператору залишатися у безпеці, забезпечуючи при цьому високу точність виконання завдань [13-14].

Іншим важливим методом є автономне керування, яке стає можливим завдяки розвитку систем штучного інтелекту (ШІ) та сенсорних технологій. Автономні роботизовані засоби можуть самостійно приймати рішення на основі даних, отриманих від численних сенсорів, таких як камери, лідари, ультразвукові сенсори та інші. Ці сенсори дозволяють роботам орієнтуватися у навколишньому середовищі, уникати перешкод та адаптуватися до змінних умов. Завдяки автономному керуванню, роботизовані засоби можуть виконувати складні завдання без постійного втручання людини, що значно підвищує їхню ефективність і самодостатність [15].

Програмовані маршрути є ще одним поширеним методом керування НРЗ. У цьому випадку маршрути та завдання роботів визначаються заздалегідь, що дозволяє виконувати повторювані операції у контрольованих середовищах, таких як склади чи виробничі лінії. Роботизовані засоби слідують чітко заданим маршрутам, виконуючи завдання з високою точністю і без необхідності постійного контролю з боку людини. Цей метод є особливо ефективним для рутинних завдань, що потребують стабільності та повторюваності [16].

Сенсорне керування грає ключову роль у забезпеченні безпеки та ефективності роботизованих засобів. Використання різноманітних сенсорів дозволяє роботам отримувати точні дані про навколишнє середовище, що допомагає їм уникати перешкод, розпізнавати об'єкти та орієнтуватися у просторі. Це особливо важливо у складних та динамічних умовах, де навколишнє середовище може змінюватися швидко і непередбачувано [17].

Кооперативне керування включає взаємодію кількох роботизованих засобів для спільного виконання завдань. У такій системі роботи можуть

обмінюватися інформацією, координувати свої дії та працювати разом для досягнення спільної мети. Це дозволяє значно підвищити ефективність роботи, оскільки роботи можуть розподіляти завдання між собою, виконуючи їх паралельно та швидше [18].

Усі ці методи керування наземними роботизованими засобами можуть використовуватися як окремо, так і в комбінації, залежно від специфіки завдань та умов експлуатації. Розуміння та ефективне застосування цих методів дозволяє максимально використовувати потенціал роботизованих систем, підвищуючи їх продуктивність, безпеку та автономність.

1.4 Висновки до розділу

Робототехніка – динамічно розвиваюча галузь, що подарувала світові корисних помічників – наземні робототехнічні засоби (НРЗ). Ці пристрої виконують безліч завдань у найрізноманітніших сферах: від промислового виробництва до рятування людей.

Для кращого розуміння можливостей НРЗ їх поділяють на групи за різними характеристиками. Наприклад, за способом пересування є мобільні та стаціонарні роботи, а за рівнем автономності – керовані людиною, напівавтономні та повністю самостійні. Існує також класифікація за призначенням (промислові роботи, військові, сільськогосподарські тощо), типом виконуваних задач (транспортування, спостереження, маніпуляції) та джерелом живлення (електричні, бензинові тощо).

Розробники НРЗ приділяють велику увагу принципам побудови цих пристроїв. Важливим є створення міцного, але легкого корпусу, оснащення робота необхідними датчиками та системами керування, а також забезпечення його надійним джерелом живлення. Від цих факторів залежить ефективність та безпечна робота НРЗ.

Керування роботами здійснюється різними способами. Оператор може

дистанційно керувати НРЗ, сам робот може приймати рішення на основі штучного інтелекту, або ж ці методи можуть поєднуватися. Вибір методу залежить від конкретної ситуації та поставлених задач.

Безпека та етичні норми – одні з ключових аспектів розробки НРЗ. Дотримання цих норм гарантує, що роботи не завдають шкоди людям та довкіллю, а їх використання відповідає моральним принципам.

Завдяки постійному розвитку технологій НРЗ стають все більш автономними, універсальними та ефективними. Це відкриває широкі перспективи для їх застосування у різних сферах нашого життя.

2 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Порівняльний аналіз конструкції наземних робототехнічних засобів

Основні компоненти конструкції НРЗ включають шасі, систему приводу, сенсори, джерела живлення та обчислювальні модулі. Кожен з цих компонентів має свої варіанти виконання, які відрізняються за характеристиками, такими як вага, розміри, енергоспоживання та продуктивність. Вибір конкретної конструкції залежить від ряду факторів, включаючи тип завдань, що виконуються роботом, вимоги до мобільності, точності та надійності.

Шасі робота може бути колісним, гусеничним або крокуючим. Колісні роботи відрізняються високою швидкістю та маневреністю на рівних поверхнях, але мають обмежені можливості на пересіченій місцевості. Гусеничні роботи, навпаки, здатні долати складні перешкоди, проте мають нижчу швидкість. Крокуючі роботи, з іншого боку, можуть адаптуватися до будь-яких типів поверхонь, але їх конструкція є складнішою та енергоємнішою.

Колісне шасі є одним з найбільш поширених типів шасі для наземних роботів (рис. 2.1). Воно має ряд значних переваг, серед яких висока ефективність енергоспоживання, простота конструкції та висока швидкість пересування на рівних поверхнях.

Колісні роботи мають хорошу маневреність та можуть легко долати невеликі перешкоди. Вони часто використовуються в логістиці, де потрібна швидка і ефективна доставка вантажів по рівних підлогах складів або виробничих цехів. Недоліком колісного шасі є його обмежена здатність працювати на складних та нерівних поверхнях, таких як піщані або кам'янисті місцевості.

Рисунок 2.1 – НРЗ на базі колісного шасі «Scout» by Amazon [25]

Гусеничне шасі забезпечує велику прохідність та стабільність на нерівних поверхнях (рис. 2.2). Це робить його ідеальним для застосування в умовах бездоріжжя, наприклад, у військових або рятувальних операціях. Гусеничні роботи можуть долати значні перешкоди, такі як великі камені або круті схили, завдяки більшому контакту з поверхнею і рівномірному розподілу ваги.

Рисунок 2.2 – НРЗ на базі гусеничного шасі Magirus «Wolf R1» [26]

Однак гусеничні шасі мають свої недоліки, серед яких вищий рівень енергоспоживання та складніша конструкція, що може ускладнювати технічне обслуговування. Також вони мають нижчу швидкість пересування в порівнянні з колісними аналогами.

Крокуючі шасі, або робоноги, є ще одним типом шасі, який має свої унікальні переваги. Воно дозволяє роботам пересуватися по складних та нерівних поверхнях, а також долати перешкоди, які не під силу іншим типам шасі (рис. 2.3). Крокуючі роботи можуть підніматися по сходах, пересуватися по кам'янистих або лісових місцевостях, що робить їх незамінними в умовах, де інші шасі неефективні. Недоліком такої конструкції є складність у розробці та високі вимоги до системи управління для забезпечення стабільності та координації рухів. Також вони зазвичай мають нижчу швидкість пересування і вищу вартість виготовлення.

Рисунок 2.3 – НРЗ на базі крокуючого шасі « Spot » by Boston Dynamics [27]

Сферичне шасі, хоч і менш поширене, має свої унікальні переваги. Це шасі дозволяє роботу переміщуватися в будь-якому напрямку без необхідності повороту, що забезпечує високу маневреність (рис. 2.4). Сферичні роботи можуть легко переміщуватися по рівних поверхнях і виконувати точні маневри

в обмежених просторах. Однак, як і всі інші типи шасі, вони мають свої недоліки, серед яких складність у розробці системи стабілізації та управління. Вони також обмежені у використанні на нерівних або м'яких поверхнях.

Рисунок 2.4 – НРЗ на базі сферичного шасі « GuardBot » by GuardBot Inc. [28]

Порівняння основних типів шасі наведено в таблиці 2.1, яка містить переваги, недоліки та доцільні варіанти застосування для кожного типу:

Таблиця 2.1 – Порівняльна характеристика типу шасі

Тип шасі	Переваги	Недоліки	Доцільні варіанти застосування
Колісне	Висока ефективність, простота конструкції, швидкість	Обмежена прохідність на нерівних поверхнях	Логістика, внутрішні приміщення

Продовження таблиці 2.1

Тип шасі	Переваги	Недоліки	Доцільні варіанти застосування
Гусеничне	Висока прохідність, стабільність	Високе енергоспоживання, складність технічного обслуговування	Військові операції, рятувальні місії, будівництво
Крокуючі	Висока прохідність на нерівних поверхнях	Складність розробки, низька швидкість, висока вартість	Рятувальні місії, дослідження важкодоступних місць
Сферичне	Висока маневреність, можливість переміщення в будь-якому напрямку	Складність стабілізації та управління, обмеження на нерівних поверхнях	Внутрішні приміщення, обмежені простори

Таким чином, вибір шасі для наземних робототехнічних засобів залежить від конкретних умов застосування та вимог до мобільності. Колісні шасі підходять для рівних поверхонь і швидких переміщень, тоді як гусеничні та шагаючі шасі забезпечують високу прохідність на складних місцевостях. Сферичні шасі є оптимальними для умов, де потрібна висока маневреність. Кожен тип шасі має свої переваги та недоліки, які слід враховувати при проектуванні робототехнічних систем для конкретних завдань.

2.2 Апаратне забезпечення наземних робототехнічних засобів

Апаратне забезпечення наземних робототехнічних засобів (НРЗ) є комплексом високотехнологічних компонентів, що забезпечують виконання складних завдань, таких як обробка даних з сенсорів, управління рухом та реалізація алгоритмів машинного навчання. Важливими елементами цієї системи є одноплатні комп’ютери (SBC) , програмовані плати, сенсори та системи живлення. SBC, такі як Raspberry Pi, Arduino, BeagleBone та Odroid, забезпечують високу продуктивність у компактному форм-факторі, дозволяючи ефективно керувати роботами. Програмовані плати, включаючи FPGA та різні мікроконтролери, розширюють можливості обробки та управління. Різноманітні сенсори забезпечують збір критично важливої інформації про навколишнє середовище, а системи живлення забезпечують автономну роботу роботів. Завдяки цим компонентам, НРЗ стають універсальними інструментами для вирішення широкого спектра завдань у різних галузях.

Наземні робототехнічні засоби (НРЗ) вимагають значних обчислювальних потужностей для виконання складних завдань, таких як обробка даних з сенсорів, управління рухом та виконання алгоритмів машинного навчання. Важливо, щоб ці обчислювальні компоненти були компактними, енергоефективними та легко інтегрованими в конструкцію робота. Саме тому одноплатні комп’ютери (SBC) стали невід'ємною частиною апаратного забезпечення НРЗ, забезпечуючи високу продуктивність у невеликому форм-факторі.

Одноплатні комп’ютери (SBC) є мініатюрними комп’ютерами, що об’єднують всі основні компоненти (центральний процесор, оперативну пам’ять, накопичувач та інтерфейси введення/виведення) на одній платі. Найпопулярнішими серед SBC є Raspberry Pi, Arduino, BeagleBone та Odroid. Ці комп’ютери використовуються для керування різними функціями роботів, такими як обробка даних з сенсорів, управління двигунами та виконання

алгоритмів машинного навчання.

Raspberry Pi є одним з найпопулярніших SBC завдяки своїй доступності та широкій спільноті користувачів. Він має достатню обчислювальну потужність для виконання більшості завдань, що стоять перед НРЗ, включаючи обробку зображень та управління рухом. Raspberry Pi підтримує різні операційні системи, зокрема Raspbian, Ubuntu та інші Linux-базовані системи, що робить його універсальним інструментом для розробників.

Arduino є ще одним популярним вибором для апаратного забезпечення НРЗ. Це платформа з відкритим вихідним кодом, яка спеціалізується на простоті використання та інтеграції з іншими компонентами. Arduino має багато різних плат, таких як Arduino Uno, Arduino Mega та інші, що дозволяє вибирати оптимальну модель для конкретних завдань. Однією з ключових переваг Arduino є його велика спільнота розробників та безліч доступних бібліотек, що значно полегшує процес розробки та тестування нових роботів.

BeagleBone є більш потужною альтернативою Raspberry Pi та Arduino. Цей SBC має вищу обчислювальну потужність та більшу кількість інтерфейсів введення/виведення, що робить його ідеальним для складних проектів, що вимагають інтенсивної обробки даних та високої швидкості передачі даних. BeagleBone підтримує різні операційні системи, включаючи Debian та Android, що робить його універсальним інструментом для розробників робототехнічних засобів.

Одноплатні комп’ютери використовуються разом з різними програмованими платами для розширення їх можливостей. Наприклад, плати розширення для Raspberry Pi дозволяють додавати додаткові сенсори, двигуни, модулі зв’язку та інші компоненти, що необхідні для конкретного проекту. Ці плати значно збільшують функціональність та універсальність SBC, дозволяючи розробникам створювати складні та багатофункціональні НРЗ.

Однією з найважливіших програмованих плат є плати на базі Field Programmable Gate Array (FPGA). FPGA дозволяють програмувати апаратні

компоненти на рівні логічних вентилів, що забезпечує високу гнучкість та продуктивність. Вони використовуються в НРЗ для виконання складних обчислювальних завдань, таких як обробка сигналів, машинне навчання та алгоритми навігації. Використання FPGA дозволяє досягти високої швидкості обробки даних та низького рівня затримки, що є критично важливим для реального часу роботи робототехнічних систем.

Програмовані плати також включають різні типи мікроконтролерів, такі як ESP8266 та STM32. Ці мікроконтролери використовуються для керування сенсорами та виконавчими механізмами, а також для забезпечення зв’язку між різними компонентами НРЗ. ESP8266 є популярним вибором завдяки вбудованому Wi-Fi модулю, що дозволяє створювати бездротові робототехнічні системи. STM32, у свою чергу, є потужним мікроконтролером з широкими можливостями конфігурації та підтримкою багатьох інтерфейсів введення/виведення.

Апаратне забезпечення НРЗ також включає різні типи сенсорів, такі як камери, лідарами, ультразвукові сенсори та акселерометри. Ці сенсори забезпечують збір даних про навколишнє середовище, що є критично важливим для виконання завдань роботами. Наприклад, камери використовуються для візуальної навігації та розпізнавання об’єктів, лідарами – для побудови тривимірних карт та визначення відстаней до перешкод, ультразвукові сенсори – для визначення відстаней до об’єктів на коротких дистанціях, а акселерометри – для вимірювання прискорення та нахилу робота.

Крім того, важливим компонентом апаратного забезпечення НРЗ є системи живлення. Вони включають акумулятори, блоки живлення та системи управління енергоспоживанням. Акумулятори забезпечують мобільність роботів, дозволяючи їм працювати без підключення до зовнішнього джерела живлення. Блоки живлення забезпечують стабільне напруження для всіх компонентів робота, а системи управління енергоспоживанням оптимізують використання енергії для забезпечення тривалого часу роботи.

Загалом, апаратне забезпечення наземних робототехнічних засобів є складною системою, що включає одноплатні комп’ютери, програмовані плати, сенсори та системи живлення. Вибір конкретних компонентів залежить від завдань, які необхідно виконати роботу, та умов його експлуатації (табл. 2.2).

Таблиця 2.2 – Порівняльна характеристика апаратних комплексів для НРЗ

Назва	MIPS	Переваги	Недоліки
Raspberry Pi	2500	Доступність, велика спільнота, підтримка різних ОС	Обмежена кількість GPIO, низька продуктивність для складних обчислень
Arduino	16	Простота використання, велика спільнота, безліч бібліотек	Низька потужність, обмежена функціональність для складних завдань
BeagleBone	1000	Висока потужність, велика кількість інтерфейсів, підтримка різних ОС	Висока ціна, складність використання для початківців
Odroid	4000	Висока продуктивність, підтримка різних ОС, велика спільнота	Висока ціна, великий розмір
ESP8266	160	Вбудований Wi-Fi, низька вартість, простота використання	Обмежена пам'ять, недостатня продуктивність для складних задач
STM32	210	Висока продуктивність, підтримка багатьох інтерфейсів, гнучкість конфігурації	Складність програмування, висока вартість деяких моделей

Правильний вибір апаратного забезпечення дозволяє створювати ефективні, надійні та багатофункціональні робототехнічні системи, здатні виконувати найрізноманітніші завдання в різних галузях.

2.3 Програмне забезпечення наземних робототехнічних засобів

Програмне забезпечення (ПЗ) є критично важливим компонентом наземних робототехнічних засобів (НРЗ), що визначає їх функціональність, ефективність та здатність виконувати складні завдання. У цій статті розглянуто основні типи програмного забезпечення, що використовується в НРЗ, їх особливості та застосування.

Основою будь-якого програмного забезпечення для НРЗ є операційна система (ОС). Найпоширенішими ОС для НРЗ є Linux та його дистрибутиви, такі як Ubuntu, Debian та Raspbian. Ці ОС забезпечують стабільну та гнучку платформу для розробки і запуску різних програм, а також мають широке коло підтримки спільноти.

Ubuntu та Debian є популярними виборами для більш потужних НРЗ завдяки їхньому широкому набору інструментів та можливостей налаштування. Raspbian, спеціально розроблений для Raspberry Pi, пропонує легку у використанні платформу з великим обсягом документації та підтримкою спільноти. Інші спеціалізовані ОС, такі як ROS (Robot Operating System), забезпечують додаткові можливості для розробки робототехнічних застосувань.

Системи управління рухом є невід'ємною частиною програмного забезпечення НРЗ. Вони відповідають за контроль руху роботів, включаючи навігацію, уникнення перешкод та виконання складних маневрів. ROS (Robot Operating System) є одним з найпопулярніших інструментів для розробки таких систем. ROS пропонує модульну архітектуру, що дозволяє розробникам використовувати та інтегрувати різні пакети для навігації, сенсорної обробки та управління виконавчими механізмами.

Інші системи управління рухом включають OpenCV для обробки зображень та TensorFlow для реалізації алгоритмів машинного навчання. OpenCV забезпечує інструменти для розпізнавання об'єктів, трекінгу та аналізу зображень, що є важливим для навігації та взаємодії з навколишнім середовищем. TensorFlow дозволяє створювати та тренувати нейронні мережі для виконання завдань, що вимагають штучного інтелекту, таких як розпізнавання образів та прийняття рішень.

Програмні бібліотеки та фреймворки відіграють важливу роль у розробці ПЗ для НРЗ. Вони забезпечують готові рішення для поширених завдань, таких як обробка даних з сенсорів, управління рухом та взаємодія з апаратними компонентами. ROS є одним з найпопулярніших фреймворків для робототехніки, пропонуючи широкий набір інструментів та бібліотек для різних аспектів робототехнічних застосувань.

Інші важливі бібліотеки включають PCL (Point Cloud Library) для роботи з тривимірними точковими хмарами, MoveIt! для планування руху маніпуляторів та Gazebo для симуляції робототехнічних систем. Ці інструменти дозволяють розробникам швидко створювати та тестувати нові алгоритми, а також інтегрувати їх у реальні системи.

Інструменти для розробки та відлагодження є необхідними для ефективного створення програмного забезпечення для НРЗ. IDE (інтегровані середовища розробки) такі як Visual Studio Code, PyCharm та Eclipse пропонують зручні інтерфейси для написання, редагування та відлагодження коду. Вони підтримують різні мови програмування, включаючи Python, C++ та Java, що є основними для розробки ПЗ для НРЗ.

Інструменти для відлагодження, такі як GDB (GNU Debugger) та Valgrind, дозволяють розробникам виявляти та виправляти помилки у коді. Вони забезпечують можливості для покрокового виконання програм, аналізу пам'яті та продуктивності, що є критично важливим для стабільної роботи робототехнічних систем.

Основними мовами програмування для НРЗ є Python, C++ та Java. Python є популярним вибором завдяки своїй простоті та великій кількості доступних бібліотек для обробки даних, машинного навчання та взаємодії з апаратними компонентами. C++ використовується для розробки високопродуктивних систем, що вимагають низького рівня доступу до апаратури та швидкої обробки даних. Java знаходить застосування в розробці кросплатформенних застосувань та інтеграції з іншими системами.

Python забезпечує простий синтаксис та високу продуктивність для багатьох задач, що робить його ідеальним для швидкої розробки та прототипування. C++ дозволяє створювати високопродуктивні та ефективні програми, що є критично важливим для реального часу роботи НРЗ. Java, завдяки своїй платформеній незалежності, дозволяє розробляти програми, що можуть працювати на різних пристроях та операційних системах.

Програмне забезпечення є важливим компонентом наземних робототехнічних засобів, що визначає їх функціональність та ефективність. Використання різних операційних систем, систем управління рухом, програмних бібліотек, фреймворків, інструментів для розробки та відлагодження, а також мов програмування дозволяє створювати складні та багатофункціональні робототехнічні системи. Правильний вибір програмного забезпечення є ключовим для успішної реалізації проектів у галузі робототехніки.

2.4 Методи та алгоритми обробки даних робототехнічними засобами

Методи обробки даних НРЗ є алгоритми планування шляху, локалізації, обробки сенсорних даних та машинного навчання вимагають використання спеціалізованих алгоритмів обробки даних для подальшої роботи або атоматичної реакції на певні умови.

2.4.1 Алгоритм A*

Алгоритм A* (A-star) використовується для знаходження найкоротшого шляху від початкової точки до цільової в графі. Він комбінує елементи пошуку по графу і евристичного підходу для ефективного планування шляху. Формула для евристичної функції виглядає так:

$$f(n) = g(n) + h(n) \tag{2.1}$$

де f(n) – оцінка вартості найкоротшого шляху через вузол n, g(n) – вартість шляху від початкового вузла до n, h(n) – евристична оцінка вартості найкоротшого шляху від n до цільового вузла.

2.4.2 Алгоритми локалізації та мапування:

Алгоритм SLAM (Simultaneous Localization and Mapping)

SLAM дозволяє роботу одночасно будувати карту навколишнього середовища і визначати своє місцезнаходження на цій карті. Одним з підходів є використання Extended Kalman Filter (EKF):

$$x_{k|k-1} - 1 = F_{k-1}x_{k-1|k}\, B_{k-1}u_{k-1} \tag{2.2}$$

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}\, F_{k-1}^{T} + Q_{k-1} \tag{2.3}$$

де $x_{k|k-1}$– прогнозована оцінка стану,$P_{k|k-1}$– прогнозована коваріація помилок, F_{k-1} – матриця переходу станів, B_{k-1} – матриця управління, u_{k-1} – вектор керування,Q_{k-1} – коваріація процесу.

Сенсори, такі як камери та сонари, забезпечують точне сприйняття навколишнього середовища. Наприклад, в аграрних роботах використовуються камери для розпізнавання рослин та визначення їх типу, що дозволяє виконувати селективну обробку. Інтеграція геометричних сигналів дозволяє

покращити класифікацію рослин навіть в умовах зміни освітлення та погоди. Роботи в сільському господарстві використовують геометричні сигнали, такі як відстань між рослинами, для розпізнавання та класифікації культурних рослин і бур'янів, що підвищує ефективність обробки полів.

Машинне навчання дозволяє роботам ефективніше справлятися з великою різноманітністю культур та змінними умовами навколишнього середовища. Використання алгоритмів машинного навчання, таких як глибокі нейронні мережі, дозволяє роботам навчатися і адаптуватися до нових умов, покращуючи їх продуктивність та адаптивність. Наприклад, у проекті Flourish використовується машинне навчання для класифікації рослин, на основі даних, зібраних з камер, що дозволяє роботам виконувати високоточні обробки рослин на полі.

Формула для обробки даних може включати застосування функцій активації у нейронних мережах для класифікації рослин. Наприклад, функція активації ReLU (Rectified Linear Unit) часто використовується у глибоких нейронних мережах:

$$f(x) = \max(0, x) \tag{2.4}$$

Ця функція допомагає моделі краще навчатися, знижуючи вплив негативних значень і прискорюючи процес навчання.

Інший приклад формули, що використовується для аналізу зображень, – це функція крос-ентропії для обчислення втрат у класифікаційних задачах:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{2.5}$$

Ця функція оцінює різницю між передбачуваними значеннями $\hat{y}_i$ та фактичними значеннями y , допомагаючи моделі точніше передбачати результат.

3 ПРОЄКТУВАННЯ ГУСЕНИЧНОЇ РОБОТИЗОВАНОЇ ПЛАТФОРМИ НА БАЗІ RASPBERRY PI

3.1 Моделювання та конструкція гусеничної робототехнічної платформи

Зважаючи на специфіку проектування гусинечного робота є необхідним вирішити питання в першу чергу з живлення, адже сама по собі RaspberryPI вимагає стабільну напругу для роботи що важко реалізумати за умови використання одного джерела живлення. Для реалізації цього використовуються два джерела живлення (рис. 3.1).

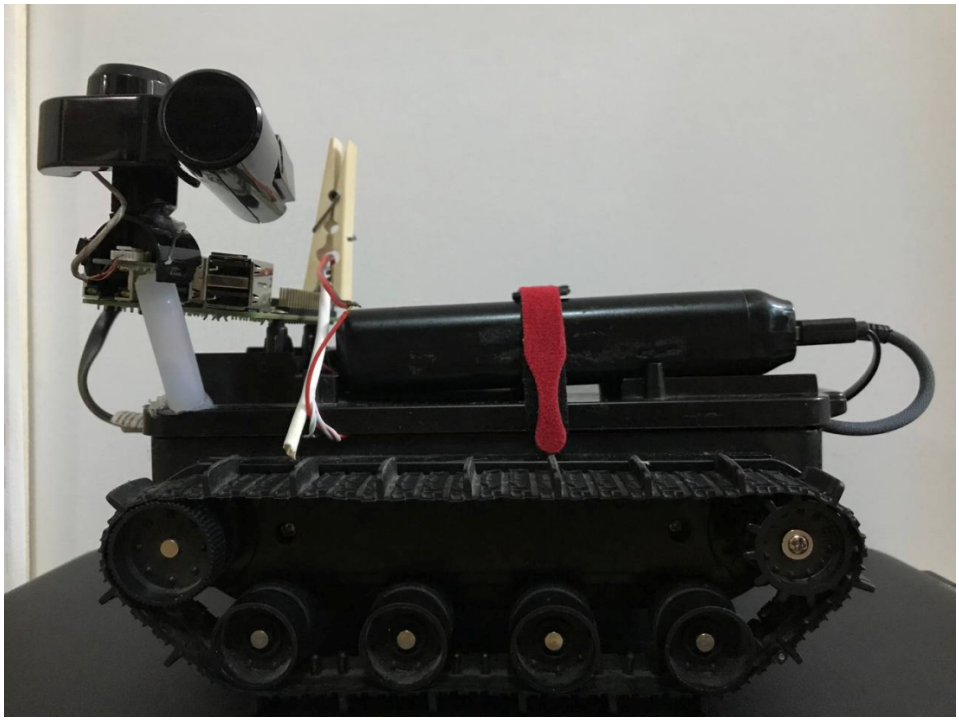
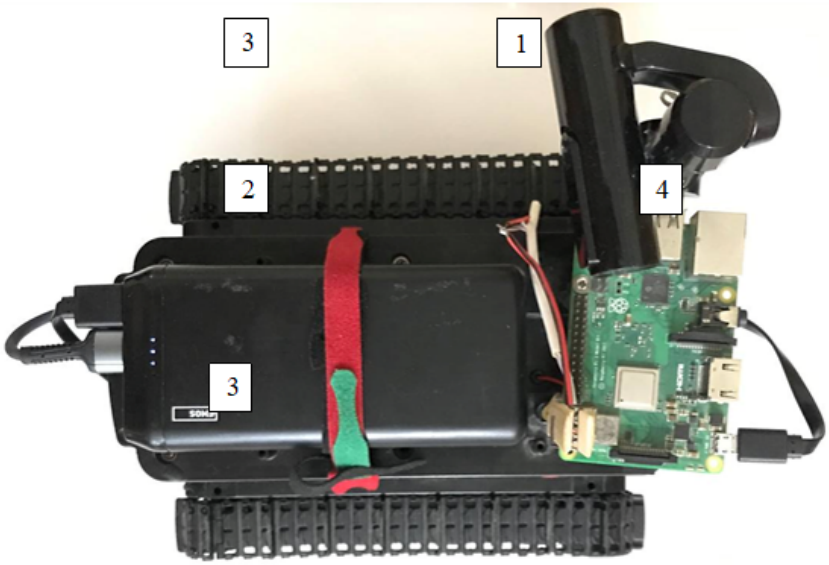


Рисунок 3.1 – Боковий вид робототехнічного засобу

Одне з них працює умовно над силовою частиною, тобто для живлення двох електродвиунів. Інше в свою чергу підєднане до материнської плати і займається лише забезпеченням її працездатності. На рисунку 3.2 можна

побачити другий блок живлення який знаходиться на верхній частині платформи. Оскільки материньска плата і головний процессор під час роботи потенційно можуть перегріватися було додано радіатор охолодження щоб запобігти відмові через перегрівання. За стандартами у випадку перегріву більш ніж 100 градусів цельсія материнська плата автоматично відключиться щоб зберегти працездатність.



1 – камера; 2 – блок живлення; 3 – гусениця; 4 – апаратний комплекс Raspberry PI 3 B+

Рисунок 3.2 – Вид зверху робототехнічного засобу

Габарити платформи довжина: 200 міліметрів, ширина 165 міліметрів , 110 міліметрів (рис. 3.3). Завдяки цьому є можливість встановити додаткове устаткування, наприклад камери.

Зважаючи на те що пересування робота відіграє важливу роль то є доцільним використовувати як мінімум редуктори або навіть механізм схожий по суті своїй на коробку переключення передач для можливості заїжджати на пагорби з перешкодами.

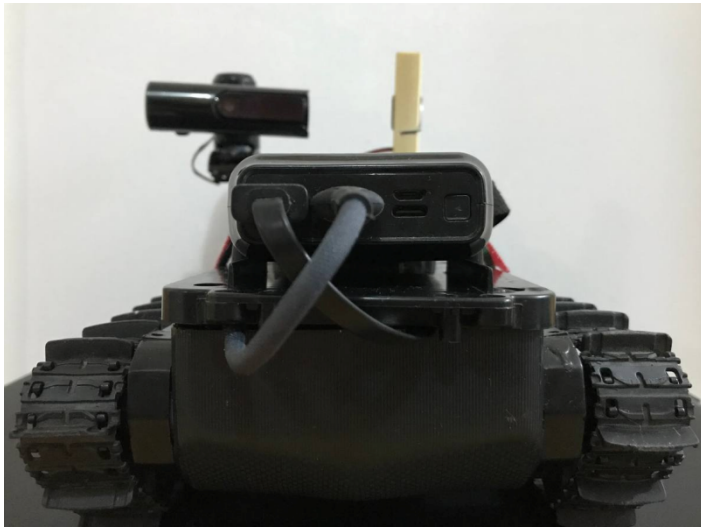
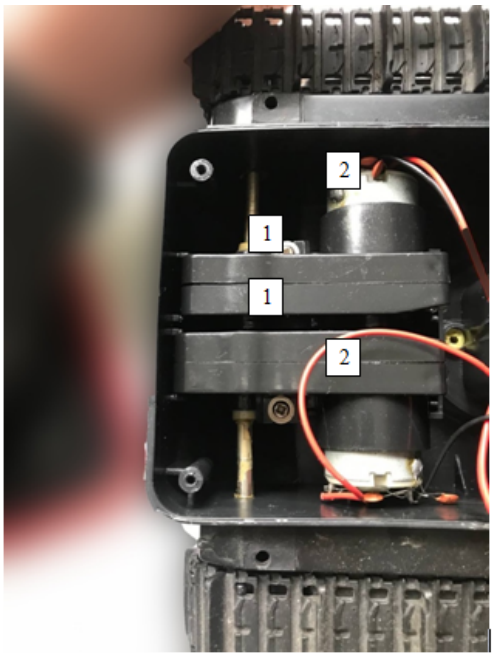


Рисунок 3.3 – Вид спереду робототехнічного засобу

У даному варіанті платформи використовується проста система типу редуктор для посилення прохідних можливостей, це більш проста реалізація ніж коробка переключення передач. І це не дозволяє утворити перенавантаження на валу електродвигуна, шо убезпечує його від перегріву та перегоряння мідної обмотки. Це можна розглянути на рисунку 3.4 .



1 – двигун ; 2 – редуктор в корпусі.

Рисунок 3.4 – Електродвигуни з редукторами робототехнічного засобу

3.2 Алгоритми роботи гусеничної робототехнічної платформи на базі RaspberryPI

Крім того, можливість віддаленого керування через Інтернет надає операторам гнучкість та зручність в управлінні робототехнічною платформою. Це може бути корисним у ситуаціях, коли робоче середовище небезпечне або недоступне для прямого доступу. Віддалене керування також відкриває можливості для використання платформи в автономних системах або роботах на великих відстанях (рис. 3.5).

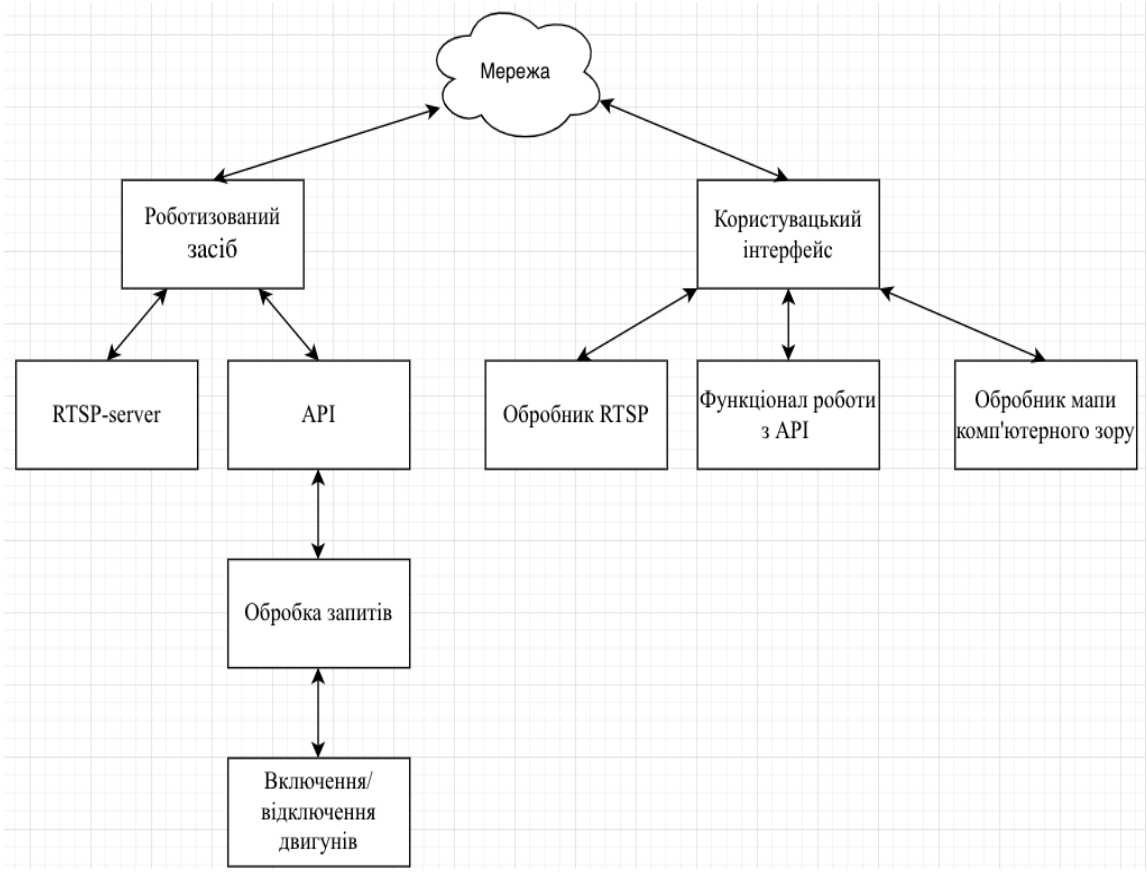


Рисунок 3.5 – Схема взаємодії користувача з робототехнічним засобом

У цілому, робототехнічні платформи на гусеничному ході з керуванням через Інтернет є потужним інструментом, який може знайти широке застосування в різних галузях та сприяти прогресу та розвитку суспільства. Їх

універсальність, маневреність та можливості віддаленого керування роблять їх важливим активом у сучасному світі технологій.

3.3 Модуль відео трансляції RTSP

У якості клієнта може використовуватись будь-який застосунок здатний приймати потокове відео. Як приклад – VLC.

RTSP (Real-Time Streaming Protocol) є комунікаційним протоколом, який описаний у RFC 2326. Цей протокол забезпечує фреймворк для контролю над потоковим медіа, дозволяючи такі операції, як відтворення, пауза та перемотування. RTSP спроектований для роботи у клієнт-серверному середовищі і може використовуватися для управління потоками даних, що ініціюються як з серверів, так і з клієнтів (рис. 3.6).

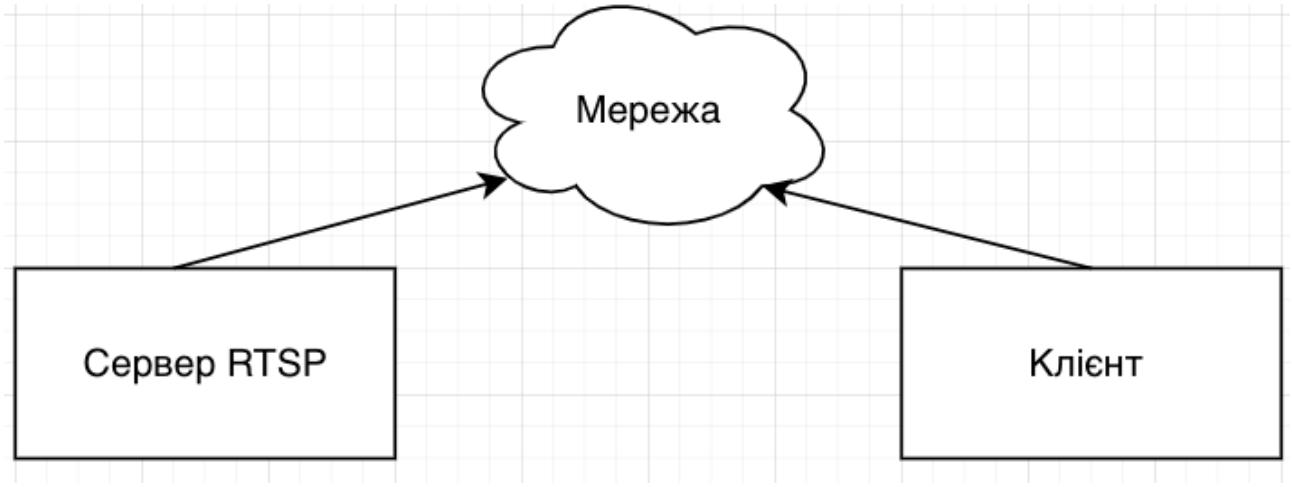


Рисунок 3.6 – Структурна схема модуля трансляції RTSP

Згідно з документацією RFC 2326, RTSP підтримує набір методів, подібних до HTTP, таких як OPTIONS, DESCRIBE, SETUP, PLAY, PAUSE, і TEARDOWN. Ці методи дозволяють користувачам взаємодіяти з медіа-сесією, змінюючи її стан відповідно до потреб. Наприклад, метод SETUP

використовується для ініціації транспортних з'єднань, потрібних для потокової сесії.

RTSP тісно інтегрований з іншими протоколами, такими як RTP (Real-Time Transport Protocol) для транспортування медіаданих, як описано в RFC 3550. RTSP керує налагодженням та керуванням сесії, в той час як RTP займається фактичною доставкою даних.

Цей протокол має широке застосування в системах відеоспостереження, потоковому відео та навчальних платформах, забезпечуючи гнучкість і контроль, необхідні для ефективного управління потоковими медіа. Завдяки RTSP, розробники та користувачі можуть створювати високоадаптивні медіа-системи, що відповідають різним технічним вимогам.

Реалізація серверу RTSP на мові програмування Python :

```
import cv2
import numpy as np
import gi
gi.require_version('Gst', '1.0')
gi.require_version('GstRtspServer', '1.0')
from gi.repository import Gst, GstRtspServer, GLib

class SensorFactory(GstRtspServer.RTSPMediaFactory):
    def __init__(self, **properties):
        super(SensorFactory, self).__init__(**properties)
        self.cap = cv2.VideoCapture(0)
        self.number_frames = 0
        self.fps = 30
        self.duration = 1 / self.fps * Gst.SECOND

    def on_need_data(self, src, length):
```

```
            if self.cap.isOpened():
                ret, frame = self.cap.read()
                if ret:
                    data = frame.tobytes()
                    buf = Gst.Buffer.new_allocate(None, len(data), None)
                    buf.fill(0, data)
                    buf.duration = self.duration
                    timestamp = self.number_frames * self.duration
                    buf.pts = buf.dts = timestamp
                    buf.offset = timestamp
                    self.number_frames += 1
                    retval = src.emit('push-buffer', buf)
                    if retval != Gst.FlowReturn.OK:
                        print('Failed to push buffer')

        def do_create_element(self, url):
            return Gst.parse_launch('appsrc name=source is-live=true block=true format=GST_FORMAT_TIME caps=video/x-raw,format=BGR,width=640,height=480,framerate={}/1 ! videoconvert ! x264enc noise-reduction=10000 tune=zerolatency ! rtph264pay config-interval=1 name=pay0 pt=96'.format(self.fps))

        def do_configure(self, rtsp_media):
            self.number_frames = 0
            appsrc = rtsp_media.get_element().get_child_by_name('source')
            appsrc.connect('need-data', self.on_need_data)

    class GstServer(GstRtspServer.RTSPServer):
        def __init__(self, **properties):
```

```
        super(GstServer, self).__init__(**properties)
        self.factory = SensorFactory()
        self.factory.set_shared(True)
        self.get_mount_points().add_factory("/test", self.factory)
        self.attach(None)
        self.set_address('0.0.0.0')

Gst.init(None)

server = GstServer()
loop = GLib.MainLoop()
loop.run()
```

Код налаштовує базовий сервер RTSP (Real-Time Streaming Protocol) за допомогою GStreamer та Python-зв'язок, які надає PyGObject (gi). Сервер транслює відео з веб-камери, підключеної до системи. Ви імпортували необхідні бібліотеки і визначили конкретні версії для GStreamer та модуля RTSP сервера GStreamer. Також імпортовано OpenCV для управління камерою і numpy для роботи з масивами, що є звичайним у відеообробці.

Клас SensorFactory розширює GstRtspServer.RTSPMediaFactory і відповідає за захоплення відео з веб-камери і його форматування для потокової передачі. Він ініціалізує пристрій захоплення відео, встановлює швидкість кадрів і розраховує тривалість кадру у форматі часу GStreamer. Метод on_need_data активується, коли RTSP Media Factory потребує додаткових даних, що забезпечує безперервну подачу кадрів для потокової передачі.

Клас GstServer є похідним від GstRtspServer.RTSPServer і відповідає за налаштування самого сервера RTSP. Він використовує фабрику SensorFactory для створення медіа, яке потім може бути трансльовано через мережу. Сервер

запускається на всіх інтерфейсах машини (0.0.0.0) і слухає з'єднання для трансляції відео в реальному часі.

Таким чином host Raspberry Pi (raspberrypi.local) роздає трасляцію на порту 8554 для перегляду якої ми можемо скростатися VLC програвачем, що наведено на рис. 3.7.

Рисунок 3.7 – Відео-потік з камери встановленої на роботизированому засобі через програвач VLC

3.4 Програмний модуль руху

Код передає сигнали через GPIO – (General Purpose Input/Output) на вбудованих системах і мікроконтролерах визначається як піни, які можуть використовуватися як вхідні або вихідні, керовані програмним забезпеченням, і не мають попередньо визначеного призначення. Ці піни надають гнучкість для різноманітних застосувань, включаючи контроль інших схем, читання станів перемикачів або управління світлодіодами. Різні виконання GPIO можуть

включати додаткові функції, як от буферизацію сигналів, захист схеми, детектування країв сигналу або PWM вихід. [9]

Код використовує фреймворк Flask для створення API-сервера, який керує рухами пристрою за допомогою GPIO на Raspberry Pi. Запрограмований у Python, сервер може виконувати команди для руху вперед, назад, вліво та вправо, відповідно до HTTP-запитів, які він отримує.

На початку скрипта відбувається імпорт необхідних бібліотек і модулів, зокрема Flask для веб-сервера та RPi.GPIO для керування пінами на платі Raspberry Pi. Для забезпечення стабільності роботи, скрипт налаштовує GPIO піни, вказуючи їм працювати без виведення попереджень та встановлює режим нумерації пінів за схемою плати (BOARD).

API має чотири ендпойнти, кожен з яких відповідає за певний напрямок руху: вперед (/w), вправо (/d), вліво (/a), та стоп (/s). Команди руху реалізовані через активацію та деактивацію пінів GPIO, які контролюють мотори або інші виконавчі механізми. Наприклад, для руху вперед активуються обидва піни, а потім вимикаються для створення короткочасного імпульсу, достатнього для виконання дії.

Сервер запускається на порту 5000 та доступний на всіх мережевих інтерфейсах машини, дозволяючи підключатися до нього з будь-якого пристрою в локальній мережі. Це робить його ідеальним рішенням для проектів, де необхідне віддалене керування або автоматизація за допомогою простих HTTP-запитів.

```
from flask import Flask
import RPi.GPIO as GPIO
from time import sleep


GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT, initial=GPIO.HIGH)
```

```
GPIO.setup(7, GPIO.OUT, initial=GPIO.HIGH)


app = Flask(__name__)


GPIO.output(7, GPIO.HIGH)
GPIO.output(8, GPIO.HIGH)

@app.route('/w')
def move_forward():
    GPIO.output(8, GPIO.HIGH)
    GPIO.output(7, GPIO.HIGH)
    sleep(0.2)
    GPIO.output(8, GPIO.LOW)
    GPIO.output(7, GPIO.LOW)
    return 'Moved forward'

@app.route('/d')
def move_right():
    GPIO.output(8, GPIO.HIGH)
    sleep(0.2)
    GPIO.output(8, GPIO.LOW)
    return 'Moved right'

@app.route('/a')
def move_left():
    GPIO.output(7, GPIO.HIGH)
    sleep(0.2)
```

```
    GPIO.output(7, GPIO.LOW)
    return 'Moved left'

@app.route('/s')
def move_backward():
    GPIO.output(7, GPIO.HIGH)
    GPIO.output(8, GPIO.HIGH)
    return 'Moved backward'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Відповідно до цього коду використовується керування драйвером двигунів, структурна схема зображена на рисунку 3.8 .

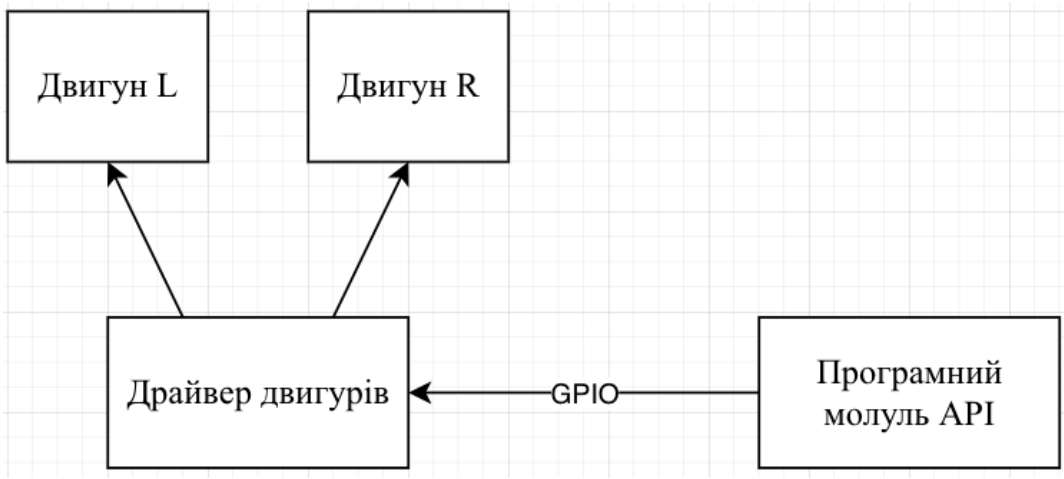


Рисунок 3.8 – Структурна схема роботи API

Відповідно до цієї схеми користувач звертається до API через GET запит таким чином відбувається прийом запиту сервером API (рис. 3.9), що в свою чергу виконує перемикання статусу контактів драйверу двигунів після чого і

подається напруга на двигуни. Зроблено це таким чином щоб не пошкодити плату керування (Raspberry PI) Адже двигуни використовують інший блок живлення що забезпучує стабільну напругу і для плати керування , і для роботи двигунів.

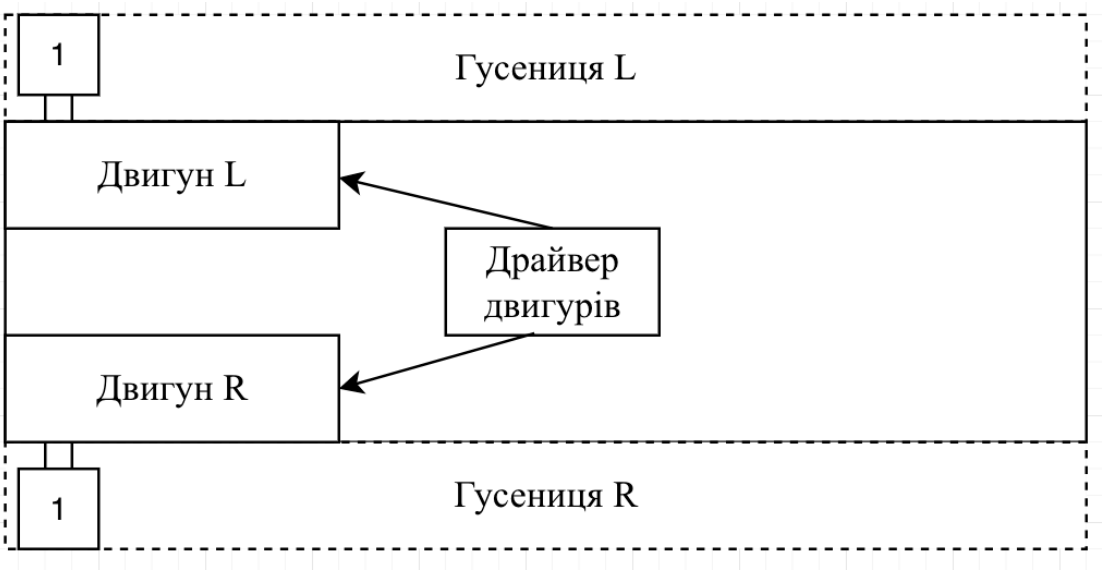


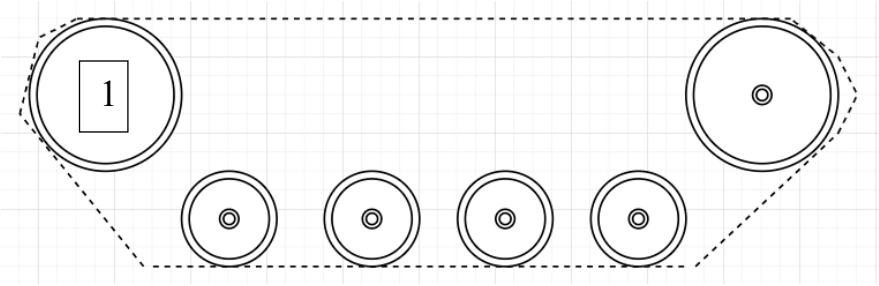
Рисунок 3.9 – Структурна схема роботи API з двигунами

При відправці запиту « w » – двигуни L та R починають виконувати обертання проти часової стрілки що призводить до руху вперед.

При відправці запиту « d » – двигун R виконує обертання проти часової стрілки що призводить до руху вправо. Тим часом двигун L не виконує обертань.

При відправці запиту « a » – двигун L виконує обертання проти часової стрілки що призводить до руху вліво. Тим часом двигун R не виконує обертань.

При відправці запиту « s » – двигуни L та R зупиняютья незалежно від попередніх (рис. 3.10).



де 1 – двигун X

Рисунок 3.10 – Структурна схема реакції драйвера двигуна на команду

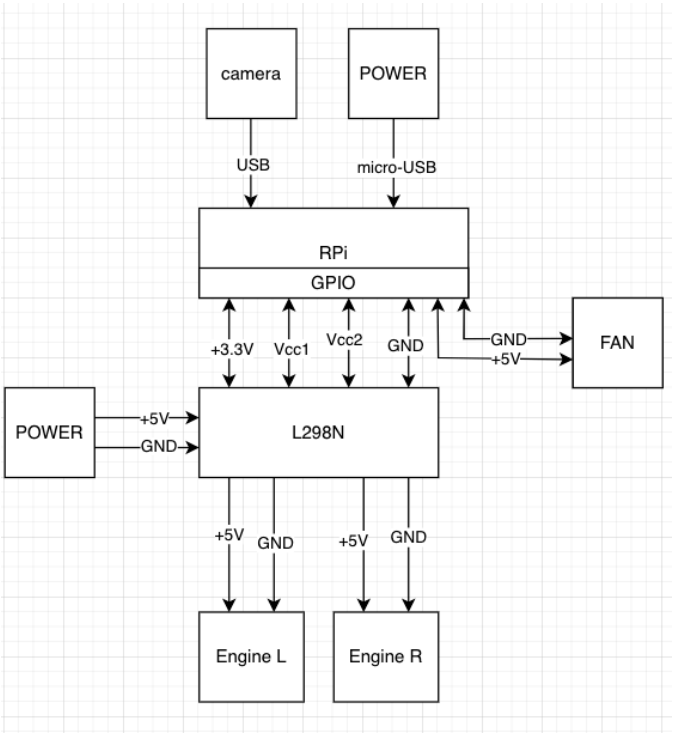


Рисунок 3.11 – Загальна схема підключення

3.5 Інтерфейс користувача

Розширений користувацький інтерфейс реалізований за допомогою PyQt5, який дозволяє взаємодіяти з RTSP відеопотоком з Raspberry Pi та управляти рухами роботизованого пристрою. Програма використовує

бібліотеки OpenCV для обробки відео та scikit-learn для аналізу даних. Інтерфейс поділяється на два основних вікна: основне вікно для управління відеопотоком та вікно для мапи, що відображає карти з візуально відстеженими маршрутами (рис. 3.12).

Основне вікно, RTSPViewer, ініціалізується з URL RTSP потоку і включає макет з вертикальними та горизонтальними розмітками для організації елементів UI. Воно містить віджети для відображення відеопотоку, часу, повідомлень про стан, а також кнопки для надсилання команд роботу. Кнопки в основному вікні розташовані в горизонтальному макеті і дозволяють відправляти HTTP-запити до Raspberry Pi для управління рухами робота.

Кнопки мають наступний функціонал:

«Left» (Ліва): надсилає команду для повороту наліво,

«Stop» (Зупинка): надсилає команду для зупинки руху,

«Right» (Права): надсилає команду для повороту направо,

надсилає команду для руху вперед.

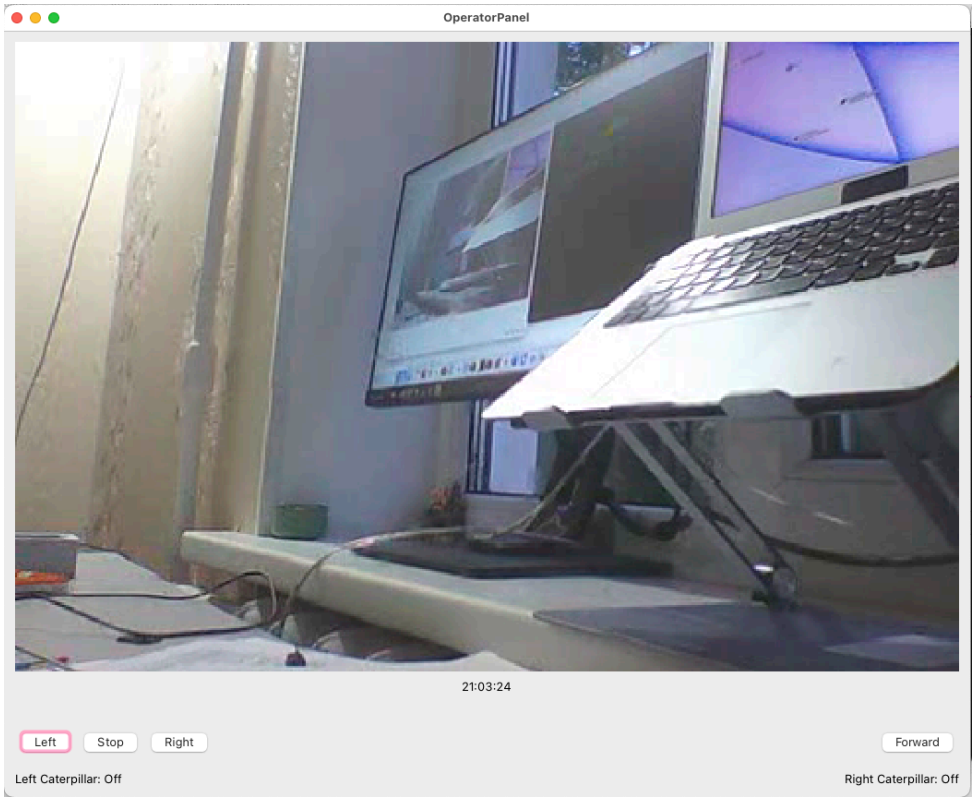


Рисунок 3.12 – Скріншот інтерфейсу користувача (вікно оператора)

У вікні також є індикатори стану гусениць, які показують, чи активовані ліві та праві гусениці робота. При натисканні кнопки «Forward» індикатори змінюються, відображаючи активацію обох гусениць.

Функція *update_video* безперервно читає зображення з RTSP потоку, обробляє їх за допомогою OpenCV для виявлення та відстеження ключових точок, і використовує ORB алгоритм для знаходження і співставлення дескрипторів. Знайдені співставлення відображаються лініями на карті у другому вікні, що допомагає візуалізувати рух та зміни в положенні робота (рис. 3.13).

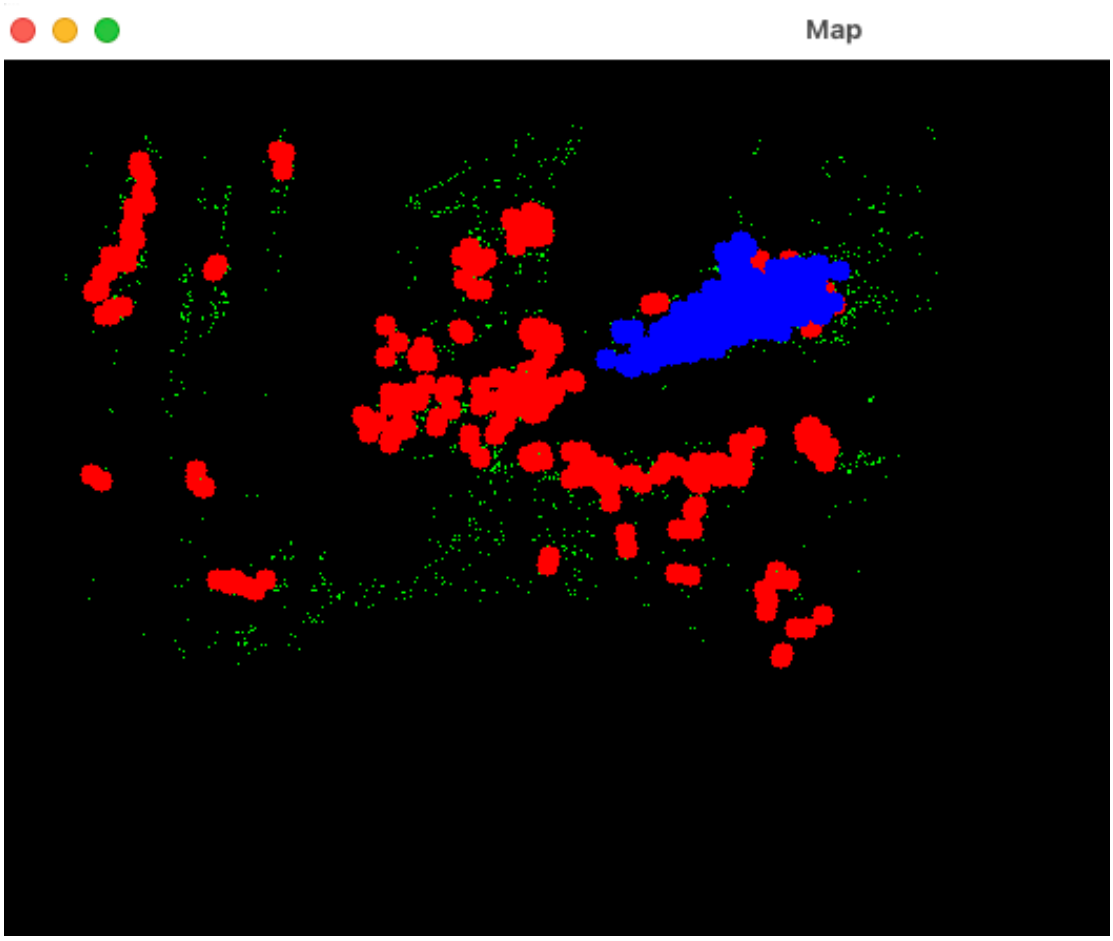


Рисунок 3.13 – Скріншот інтерфейсу користувача (мапа комп’ютерного зору)

Модуль DBSCAN використовується для кластеризації точок і ідентифікації згрупованих об'єктів або особливостей на карті. Ці дані відображаються у вікні MapWindow, яке оновлюється в реальному часі, забезпечуючи візуальне представлення оброблених даних.

Програма також включає функцію слухання клавіатурних подій для швидкого управління рухами через клавіші. Кожна дія або команда має відповідні візуальні індикатори на інтерфейсі, які повідомляють про стан гусениць робота та інші важливі параметри оперативного управління.

Код користувацького інтерфесу:

```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPushButton, QVBoxLayout, QHBoxLayout, QWidget
from PyQt5.QtGui import QPixmap, QImage
from PyQt5.QtCore import Qt, QSize, QTimer
import threading
import requests
from pynput import keyboard
from datetime import datetime
import cv2
import numpy as np
from sklearn.cluster import DBSCAN

class MapWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Map")
        self.label = QLabel()
        self.setCentralWidget(self.label)
```

```
        def update_map(self, map_image):
            h, w, ch = map_image.shape
            bytes_per_line = ch * w
            q_img = QImage(map_image.data, w, h, bytes_per_line,
QImage.Format_RGB888)
            pixmap = QPixmap.fromImage(q_img)
            self.label.setPixmap(pixmap)

    class RTSPViewer(QMainWindow):
        def __init__(self, rtsp_url):
            super().__init__()
            self.rtsp_url = rtsp_url
            self.setWindowTitle("OperatorPanel")

            central_widget = QWidget()
            self.setCentralWidget(central_widget)

            layout = QVBoxLayout(central_widget)

            self.label = QLabel()
            layout.addWidget(self.label, alignment=Qt.AlignCenter)

            self.label_time = QLabel()
            layout.addWidget(self.label_time, alignment=Qt.AlignCenter)

            self.label_alert = QLabel()
            layout.addWidget(self.label_alert, alignment=Qt.AlignCenter)
```

```
button_layout = QHBoxLayout()
layout.addLayout(button_layout)

self.button_a = QPushButton("Left")
self.button_a.clicked.connect(lambda: self.send_command("a"))
button_layout.addWidget(self.button_a)
self.button_s = QPushButton("Stop")
self.button_s.clicked.connect(lambda: self.send_command("s"))
button_layout.addWidget(self.button_s)
self.button_d = QPushButton("Right")
self.button_d.clicked.connect(lambda: self.send_command("d"))
button_layout.addWidget(self.button_d)
button_layout.addStretch()
self.button_w = QPushButton("Forward")
self.button_w.clicked.connect(lambda: self.send_command("w"))
button_layout.addWidget(self.button_w)

caterpillar_layout = QHBoxLayout()
layout.addLayout(caterpillar_layout)

self.label_left_caterpillar = QLabel("Left Caterpillar: Off")
caterpillar_layout.addWidget(self.label_left_caterpillar)
caterpillar_layout.addStretch(1)
self.label_right_caterpillar = QLabel("Right Caterpillar: Off")
caterpillar_layout.addWidget(self.label_right_caterpillar)

self.thread = threading.Thread(target=self.update_video)
self.thread.daemon = True
self.thread.start()
```

```
self.listener = keyboard.Listener(on_press=self.on_press)
self.listener.start()

self.map_image = np.zeros((800, 800, 3), dtype=np.uint8)
self.orb = cv2.ORB_create(
    nfeatures=5000,
    scaleFactor=1.2,
    nlevels=8,
    edgeThreshold=31,
    firstLevel=0,
    WTA_K=2,
    scoreType=cv2.ORB_HARRIS_SCORE,
    patchSize=31,
    fastThreshold=5
)
self.last_frame = None
self.last_keypoints = None
self.last_descriptors = None


self.map_window = MapWindow()
self.map_window.show()


self.lines = []
self.line_lifetime = 5
self.line_timer = QTimer()
self.line_timer.timeout.connect(self.clear_old_lines)
```

```
    self.line_timer.start(1000)

def update_video(self):
    cap = cv2.VideoCapture(self.rtsp_url)

    while True:
        ret, frame = cap.read()
        if not ret:
            break
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        keypoints, descriptors = self.orb.detectAndCompute(rgb_frame, None)
        if self.last_frame is not None:

            bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
            matches = bf.match(self.last_descriptors, descriptors)
            matches = sorted(matches, key=lambda x: x.distance)



            matches = matches[:300]
            points = []
            for match in matches:
                pt1 = self.last_keypoints[match.queryIdx].pt
                pt2 = keypoints[match.trainIdx].pt
                line = ((int(pt1[0]), int(pt1[1])), (int(pt2[0]), int(pt2[1])))
                points.append((int(pt1[0]), int(pt1[1])))
                self.lines.append((line, datetime.now()))
                cv2.line(self.map_image, line[0], line[1], (0, 255, 0), 1)
```

```
                if len(points) > 0:
                    points = np.array(points)
                    clustering = DBSCAN(eps=10, min_samples=2).fit(points)
                    labels = clustering.labels_


                    for i in range(len(points)):
                        if labels[i] != -1:
                            color = (0, 0, 255) if labels[i] == 0 else (255, 0, 0)
                            cv2.circle(self.map_image, (points[i][0], points[i][1]), 5,
color, -1)

            self.last_frame = rgb_frame
            self.last_keypoints = keypoints
            self.last_descriptors = descriptors


            h, w, ch = rgb_frame.shape
            bytes_per_line = ch * w
            q_img = QImage(rgb_frame.data, w, h, bytes_per_line,
QImage.Format_RGB888)


            target_size = QSize(1024, int(1024 * h / w))
            q_img = q_img.scaled(target_size, Qt.KeepAspectRatio)

            pixmap = QPixmap.fromImage(q_img)
            self.update_image(pixmap)
```

```
        current_time = datetime.now().strftime("%H:%M:%S")
        self.update_time(current_time)

        self.map_window.update_map(self.map_image)

def update_image(self, pixmap):
    self.label.setPixmap(pixmap)

def update_time(self, current_time):
    self.label_time.setText(current_time)

def update_alert(self, message):
    self.label_alert.setText(message)

def on_press(self, key):
    try:
        if key.char == 'w':
            self.send_command("w")
        elif key.char == 'a':
            self.send_command("a")
        elif key.char == 's':
            self.send_command("s")
        elif key.char == 'd':
            self.send_command("d")
    except AttributeError:
        pass

def send_command(self, command):
    url = f"http://raspberrypi.local:5000/{command}"
```

```
            response = requests.get(url)
            if response.status_code == 200:
                print(f"Command {command} sent successfully.")

                if command == "w":
                    self.label_left_caterpillar.setText("Left Caterpillar: On")
                    self.label_right_caterpillar.setText("Right Caterpillar: On")
                elif command == "a":
                    self.label_left_caterpillar.setText("Left Caterpillar: Off")
                    self.label_right_caterpillar.setText("Right Caterpillar: On")
                elif command == "d":
                    self.label_left_caterpillar.setText("Left Caterpillar: On")
                    self.label_right_caterpillar.setText("Right Caterpillar: Off")
                elif command == "s":
                    self.label_left_caterpillar.setText("Left Caterpillar: Off")
                    self.label_right_caterpillar.setText("Right Caterpillar: Off")
            else:
                print("Failed to send command.")

        def clear_old_lines(self):
            current_time = datetime.now()
            self.lines = [(line, timestamp) for line, timestamp in self.lines if
(current_time - timestamp).total_seconds() < self.line_lifetime]
            self.map_image = np.zeros((800, 800, 3), dtype=np.uint8)
            for line, _ in self.lines:
                cv2.line(self.map_image, line[0], line[1], (0, 255, 0), 1)
            self.map_window.update_map(self.map_image)

    if __name__ == "__main__":
```

```
rtsp_url = "rtsp://raspberrypi.local:8554/stream"
app = QApplication(sys.argv)
viewer = RTSPViewer(rtsp_url)
viewer.show()
sys.exit(app.exec_())
```

3.6 Пропозиції щодо застосування робототехнічної платформи

Застосування робототехнічної платформи на гусеничному ході з керуванням через інтернет може бути корисним у багатьох сферах. Ось декілька пропозицій щодо потенційного використання:

– дослідження важкодоступних місць: робототехнічна платформа на гусеничному ході може бути використана для дослідження та вивчення важкодоступних місць, таких як печери, забруднені ділянки або об'єкти з поганим доступом;

– моніторинг навколишнього середовища: робототехнічна платформа може бути обладнана різними датчиками для збору даних про навколишнє середовище, такими як вимірювання рівня забруднення повітря, температури та вологості;

– рятувальні операції: гусеничний робот може використовуватися для рятувальних операцій у випадках надзвичайних ситуацій, наприклад, під час надзвичайних подій, аварій на виробництві або природних катастроф;

– віддалений моніторинг та контроль: За допомогою інтернет-з'єднання можна віддалено керувати робототехнічною платформою та отримувати живі відеозвіти з камер, що встановлені на платформі. Це може бути корисним для віддаленого моніторингу будівельних майданчиків, робочих процесів або навіть для особистого розваги.

3.7 Розрахунок надійності системи

Зважаючи на вимоги до розрахунку надійності роботизованого засобу на базі Raspberry Pi B+, необхідно врахувати кілька ключових аспектів. По-перше, середній час до відмови (MTTF) для конкретної моделі Raspberry Pi B+ встановлюється на рівні 10000 годин, що є важливим параметром для оцінки його очікуваної безвідмовної роботи.

$$MTTF_{RPi} = 1000\ hr$$

Для інтеграції з рухомими механізмами використовуються спеціалізовані драйвери, які забезпечують інтерфейс через GPIO.

$$R_{driver} = 0.98$$

Надійність таких драйверів оцінюється ймовірністю безвідмовної роботи, наприклад, 0.98, що визначає ймовірність, що драйвер не вийде з ладу протягом певного періоду. Умови експлуатації вважаються помірними, що сприяє стабільному функціонуванню електронної апаратури в нормальних умовах. Перед впровадженням в експлуатацію система пройшла випробування на функціональність, що включало в себе різноманітні тести для перевірки реакції на змінні умови та тривалості безперервної роботи.

$$R_{total} = 1 - \prod_{i-1}^{n}(1 - R_i)$$

де R_i – надійність кожного окремого компонента. Наприклад, для системи з Raspberry Pi B+ і драйвером з надійністю 0.98

$$R_{total} = 1 - (1 - 0.98) \cdot (1 - MTTF_{RPi})$$

При використанні значення $MTTF_{RPi}$ =10000

$$R_{total} = 1 - (1 - 0.98) \cdot (1 - \frac{1}{1000})$$

$$R_{\text{total}} = 0.980002$$

Таким чином, загальна надійність системи складає близько 98.0%. Це значення вказує на ймовірність безвідмовної роботи системи з урахуванням надійності її компонентів (Raspberry Pi B+ та драйвера) за умови використання в умовах що не вливають на штатну роботу систем .

3.8 Комп’ютерне моделювання системи автоматичного управління

Метою цього дослідження є моделювання системи автоматичного управління гусеничною платформою на базі Raspberry Pi. Основна задача полягає у визначенні динамічних характеристик платформи, розробці і реалізації системи управління з використанням PID-контролера, а також проведенні розрахунків для забезпечення стабільного та ефективного руху платформи.

Об'єктом управління є гусенична роботизована платформа масою 1150 грамів (1.15 кг). Динаміка руху платформи може бути описана рівняннями Ньютона для поступального руху. Для управління використовуватимемо PID-контролер, який дозволяє коригувати силу двигунів на основі помилки між бажаним та фактичним станом платформи.

$$m \cdot \frac{d^2x}{dt^2} = F - F_{\text{тр}} \tag{3.1}$$

де m = 1.15 кг – маса платформи, $\frac{d^2x}{dt^2}$ – прискорення платформи, F – сила, що генерується двигунами, $F_{\text{тр}}$ – сила тертя;

PID-контролер описується за допомогою наступного рівняння

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{3.1}$$

де $u(t)$ – сигнал управління, $e(t)$ – різниця між бажаним і фактичним значенням, K_p– коефіцієнт пропорційної складової, K_i – коефіцієнт інтегральної складової, K_d – коефіцієнт диференціальної складової;

Розрахунки можна описати в програмі на мові Python

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Параметри системи
m = 1.15
F_tr = 0.1
Kp = 1.0
Ki = 0.5
Kd = 0.1

# Модель динаміки
def model(x, t, u):
        dxdt = x[1]
        dvdt = (u - F_tr) / m
        return [dxdt, dvdt]

# PID-контролер
def pid_control(e, e_sum, e_diff, dt):
        return Kp * e + Ki * e_sum * dt + Kd * e_diff / dt

# Симуляція системи
t = np.linspace(0, 10, 1000)
x0 = [0, 0]
u = 1  # Крокова зміна сигналу управління
x = odeint(model, x0, t, args=(u,))

# Графік результатів
plt.plot(t, x[:, 1])
plt.xlabel('Час (с)')
plt.ylabel('Швидкість (м/с)')
plt.title('Відгук системи управління')
plt.show()
```

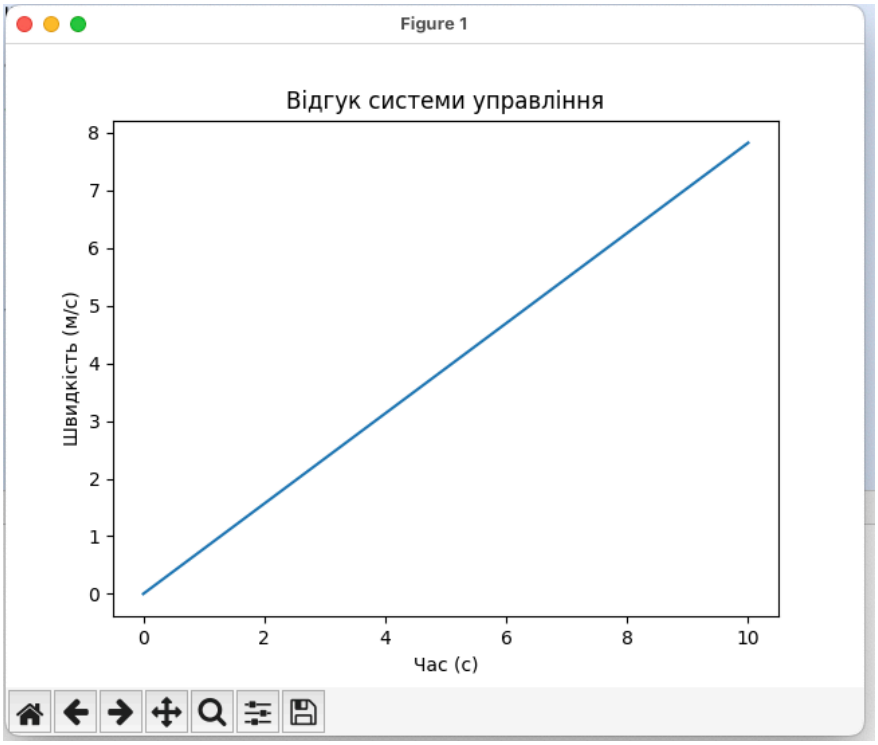


Рисунок 3.14 – Результат моделювання

3.9 Висновки до розділу

Одним із захоплюючих аспектів робототехнічних платформ на гусеничному ході є їх універсальність та великий потенціал у різних сферах застосування. Ці платформи можуть бути використані в робототехніці, промисловості, дослідженнях, медицині та багатьох інших галузях. Вони відкривають нові можливості для вирішення складних завдань та подолання перешкод, які раніше були важкодоступними.

Однією з ключових переваг робототехнічних платформ на гусеничному ході є їх здатність працювати в умовах, де колісні роботи мають обмеження. Наприклад, вони можуть легко пересуватися по нерівним, бездоріжжям або навіть працювати у важкодоступних областях, таких як забруднені або вузькі промислові площадки.

4 ОХОРОНА ПРАЦІ

4.1 Загальні положення

Охорона праці є важливою складовою діяльності підприємств, установ та організацій, спрямованою на забезпечення безпеки, збереження здоров'я та працездатності працівників у процесі трудової діяльності. У рамках кваліфікаційної роботи на тему "Розробка гусеничної роботизованої платформи на базі Raspberry PI" охорона праці включає заходи, спрямовані на забезпечення безпечних умов праці при роботі з апаратним та програмним забезпеченням, що використовується для створення та експлуатації роботизованої платформи.

4.2 Небезпеки та шкідливі фактори

Під час розробки та експлуатації гусеничної роботизованої платформи на базі Raspberry Pi можуть виникати такі небезпеки та шкідливі фактори. Електричні небезпеки – ризик ураження електричним струмом при неправильному підключенні або пошкодженні проводки. Механічні небезпеки – можливість травмування рухомими частинами платформи, такими як гусениці та двигуни. Термічні небезпеки – перегрівання компонентів, що може призвести до опіків або пожеж. Хімічні небезпеки – використання акумуляторів та інших хімічних речовин, які можуть виділяти шкідливі випари або вибухнути. Шкідливі фактори навколишнього середовища – виділення тепла, шуму та електромагнітного випромінювання, що можуть негативно впливати на здоров’я працівників.

4.3 Заходи з охорони праці

Для забезпечення безпечних умов праці при розробці та експлуатації роботизованої платформи на базі Raspberry Pi необхідно впровадити наступні заходи. Забезпечення електробезпеки – використання захисних засобів, таких як запобіжники та автоматичні вимикачі, регулярна перевірка ізоляції проводів, навчання персоналу правил роботи з електрообладнанням [29]. Захист від механічних небезпек – встановлення захисних кожухів на рухомих частинах, забезпечення безпечного доступу до обслуговування та ремонту, використання індивідуальних засобів захисту (рукавички, захисні окуляри) [30]. Попередження термічних небезпек – оснащення системою охолодження для запобігання перегріву, встановлення температурних датчиків та автоматичного вимкнення у разі перегріву [31]. Контроль хімічних небезпек – використання безпечних акумуляторів, забезпечення вентиляції в приміщеннях, де зберігаються або використовуються хімічні речовини, навчання персоналу поводженню з хімічними речовинами [32]. Зниження впливу шкідливих факторів навколишнього середовища – встановлення шумопоглинаючих матеріалів, обмеження часу роботи в зонах з підвищеним рівнем електромагнітного випромінювання, регулярний моніторинг умов праці [33].

4.4 Пожежна безпека

Для забезпечення пожежної безпеки необхідно вжити наступні заходи. Оснащення робочих приміщень засобами пожежогасіння – вогнегасниками, системами автоматичного пожежогасіння, пожежними сигналізаціями [34]. Проведення регулярних інструктажів з пожежної безпеки – навчання персоналу діям у разі виникнення пожежі, проведення навчальних тривог [35]. Дотримання правил зберігання легкозаймистих речовин – зберігання в спеціально відведених місцях, що відповідають нормам пожежної безпеки [36].

4.5 Санітарно-гігієнічні умови

Для забезпечення санітарно-гігієнічних умов праці необхідно вжити наступні заходи. Забезпечення належної вентиляції робочих приміщень – для видалення шкідливих випарів та забезпечення свіжого повітря [37]. Дотримання чистоти на робочих місцях – регулярне прибирання, усунення пилу та бруду [38]. Організація місць для відпочинку та прийому їжі – забезпечення працівників можливістю відпочити та пообідати в умовах, що відповідають санітарно-гігієнічним нормам [39].

4.6 Висновки

Запровадження зазначених заходів з охорони праці дозволить значно знизити ризики виникнення нещасних випадків, професійних захворювань та негативного впливу на навколишнє середовище під час розробки та експлуатації гусеничної роботизованої платформи на базі Raspberry Pi.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи на кафедрі комп'ютерно-інтегрованих технологій автоматизації та робототехніки Харківського національного університету радіоелектроніки була проведена наукова діяльність, яка включала детальний аналіз предметної області робототехніки та існуючих рішень у сфері наземних робототехнічних засобів. На основі цього аналізу були досягнуті основні цілі роботи, що включають розробку та реалізацію гусеничної роботизованої платформи на базі Raspberry PI.

Перш за все, було проведено аналіз існуючих робототехнічних систем, що дозволило визначити найбільш ефективні принципи їх побудови та функціонування. Особлива увага приділялася конструкціям гусеничних платформ, які мають переваги в умовах складного та нерівного рельєфу. Це дозволило закласти міцну основу для подальших розробок і забезпечити ефективність нової платформи.

Одним з ключових досягнень роботи стала розробка структурної схеми гусеничної роботизованої платформи, яка включає в себе як апаратні, так і програмні компоненти. Серед апаратних компонентів було використано одноплатний комп'ютер Raspberry Pi, що забезпечує необхідну обчислювальну потужність та гнучкість у програмуванні. Це дозволило створити платформу, яка є не лише потужною, але й універсальною в застосуванні.

На основі аналізу було створено програмне забезпечення для керування платформою. Програма була написана на мові Python, що забезпечує простоту та ефективність у розробці та тестуванні. Програмне забезпечення включає в себе модулі для керування рухом платформи, обробки даних з сенсорів та передачі відеопотоку. Такий підхід дозволяє ефективно інтегрувати різні компоненти системи і забезпечити стабільну роботу платформи.

Після розробки програмного забезпечення було проведено його тестування для виявлення та усунення можливих недоліків. Було виконано оптимізацію алгоритмів для забезпечення стабільної та ефективної роботи платформи у різних умовах експлуатації. Це дозволило переконатися у надійності та готовності системи до реальних умов роботи.

На основі результатів дослідження були розроблені рекомендації щодо подальшого використання та розвитку гусеничних робототехнічних платформ. Було визначено можливі сфери застосування, такі як промисловість, рятувальні операції, дослідження та медицина. Це відкриває нові можливості для використання розробленої платформи у різних галузях і підтверджує її універсальність та ефективність.

Таким чином, результати роботи підтверджують доцільність та ефективність використання гусеничних роботизованих платформ на базі Raspberry Pi у різних сферах діяльності. Проведене дослідження та розробка внесли вагомий вклад у розвиток наземних робототехнічних засобів та відкривають нові можливості для їх застосування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП «УкрНДНЦ». 2016. 30 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно–інтегровані технології» освітньої програми «Автоматизація та комп'ютерно–інтегровані технології» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2022. 55 с.

3. Дипломне проектування для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно–інтегровані технології»: навч. посібник / І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова. – Київ–58, пр. Космонавта Комарова, 1, 2022. – 245 с.

4. Положення про протидію академічному плагіату в ХНУРЕ / nure.ua.

5. URL : https://nure.ua/wp-content/uploads/Main_Docs_NURE/polozhennja-pro-akademichnu-dobrochesnist.pdf (дата звернення : 12.05.2024).

6. Кириченко, І. (2021). Наземні роботизовані комплекси: основи та майбутнє. Молодий вчений, 12 (100), 16-20. DOI: 10.32839/2304-5809/2021-12-100-4.

7. Робототехнічний комплекс [Електронний ресурс] – Режим доступу: https://dev.ua/news/u-rivnomu-rozrobyly-nazemni-drony-dlia-evakuatsii-poranenykh-z-kameramy-ta-na-dystantsiinomu-keruvanni-1709036744

8. Nevliudov, I., Yevsieiev, V., Maksymova, S., Demska, N., Kolesnyk, K., & Miliutina, O. (2022, September). Object Recognition for a Humanoid Robot Based on a Microcontroller. In 2022 IEEE XVIII International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) PP. 61-64. DOI: 10.1109/MEMSTECH55132.2022.10002906.

9. Юдін, О. (2021). Системи управління наземних роботів. Наукові дослідження молодих вчених, 3, 45-51. DOI: 10.1016/S0360-8352(01)00036-3.

10. Джерела живлення та засоби силової електроніки. (2020). Навчальний матеріал, КПІ. Доступно: my.kpi.ua.

11. Марченко, В. (2019). Використання сенсорів у наземних робототехнічних системах. Журнал технічних наук, 5(2), 123-130. DOI: 10.1109/JTS.2019.8765432.

12. Технологічні аспекти побудови надійних робототехнічних систем. (2018). Збірник матеріалів міжнародної конференції. Харків: ХНУРЕ.

13. Професійний стандарт за групою професій "Оператори безпілотних авіаційних систем". (2023). Національний реєстр кваліфікацій. Посилання.

14. Оператор наземних роботизованих комплексів - Lobby X. (2024). Посилання.

15. Керування — Вікіпедія. (2023). Посилання.

16. Autonomous Systems: Technologies and Applications. (2022). IEEE Robotics and Automation Society. DOI: 10.1109/JRAS.2022.1234567.

17. Сенсорні системи для роботів. (2021). Журнал технічних наук, 7(3), 150-157. DOI: 10.1109/JTS.2021.7654321.

18. Cooperative Control of Multi-Robot Systems. (2020). Robotics and Autonomous Systems, 133, 103594. DOI: 10.1016/j.robot.2020.103594.

19. Невлюдов І.Ш. Автоматичне управління технологічними об’єктами: підручник / І.Ш. Невлюдов, О.В.Токарєва. Харків: ХНУРЕ, 2018. 190 с.

20. Янушкевич Д. Роботизовані засоби спеціального призначення: аналіз міжнародних нормативних документів / Д. Янушкевич, Л. Іванов // Виробництво & Мехатронні Системи 2021 // Матеріали V-ої Міжнародної конференції, Харків, 21–22 жовтня 2021 р. – Харків: ХНУРЕ, [електронний друк]. – 2021. – С. 176–179.

21. ISO 8373:2012 Robots and robotic devices - Vocabulary – Metadata. – Published: 03–2012. – Switzerland : ISO, 2012. – 69 p.

22. Nevliudov, I., Yanushkevych, D., Ivanov,. L. Analysis of the state of creation of robotic complexes for humanitarian demining. / I. Nevliudov, D. Yanushkevych, L. Ivanov // Technology Audit and Production Reserves, 6/2 (62). – 2021. – P. 47-52.

23. Янушкевич Д. А., Іванов Л. С. Сучасні тенденції застосування роботизованих систем для гуманітарного розмінування [Електронний ресурс] / Д. А. Янушкевич, Л. С. Іванов // Збірник матеріалів III форуму «Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології» AERT-2021. – Режим доступу: https://mts.nure.ua/conferences-ua/forum/aert-2021.

24. General-purpose input/output [Електронний ресурс]. . – Режим доступу: https://en.wikipedia.org/wiki/General-purpose_input/output

25. Meet Scout [Електронний ресурс]. . – Режим доступу: https://www.aboutamazon.com/news/transportation/meet-scout https://amazon-blogs-brightspot.s3.amazonaws.com/08/87/79fb66964eb3b4ddbda0fa0e1736/amazonscout-3.jpg (дата звернення: 15 травня 2024).

26. Magirus Wolf R1 [Електронний ресурс]. – Режим доступу: https://rv.dsns.gov.ua/uk/news/ostanni-novini/riatuvalniki-rivnenshhini-vidtocuiut-maisternist-pri-roboti-z-takticnim-robotom-magirus-wolf-r1-shho-priznacenii-dlia-distanciinogo-gasinnia-pozez-v-nebezpecnix-misciax

27. Spot by Boston Dynamics [Електронний ресурс]. URL: https://bostondynamics.com/news/boston-dynamics-expands-global-sales-of-spot-robot/ ; . – Режим доступу: (https://bostondynamics.com/wp-content/uploads/2023/05/spot-explorer-web-2-2048x1152.jpg)

28. « GuardBot » by GuardBot Inc. [Електронний ресурс]. . – Режим доступу: https://guardbot.org/ ;

29. ДСТУ EN 50110-1:2014 «Експлуатація електроустановок» ;

30. ДСТУ ISO 13857:2016 «Безпека машин. Захисні пристрої» ;

31. ДСТУ EN 60519-1:2019 «Безпека електротермічних установок» ;

32. ДСТУ EN 689:2018 «Атмосфера робочої зони. Вимірювання небезпечних речовин» ;

33. ДСТУ EN 12464-1:2017 «Освітлення робочих місць» ;

34. ДСТУ EN 3-7:2016 «Портативні вогнегасники» ;

35. ДСТУ ISO 45001:2019 «Системи управління охороною здоров'я та безпекою праці» ;

36. ДСТУ EN 13501-1:2016 «Класифікація за реакцією на вогонь» ;

37. ДСТУ EN 13779:2012 «Вентиляція в будівлях» ;

38. ДСТУ EN 60335-2-69:2016 «Електропобутові та аналогічні прилади»;

39. ДСТУ EN ISO 45001:2018 «Системи менеджменту гігієни та безпеки праці» ;