МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра аерокосмічних систем управління

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри АКСУ

_____ Юрій МЕЛЬНИК

"_____"_____2023р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

“БАКАЛАВР”

Тема Адаптивна система автоматичного управління наземним рухомим об’єктом

Виконавець: Жеребко Владислав Віталійович

Керівник: Мельник Юрій Віталійович

Нормоконтролер: к.т.н., доцент Дивнич Микола Полікарпович

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра аерокосмічних систем управління

Спеціальність 151 Автоматизація та комп’ютерно-інтегровані технології

ЗАТВЕРДЖУЮ

Завідувач кафедри АКСУ

_____Юрій МЕЛЬНИК

"_____"_____2023р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

студента Жеребко Владислава Віталійовича

1. Тема кваліфікаційної роботи: “Адаптивна система автоматичного управління наземним рухомим об’єктом” затверджена наказом ректора від 13.04.2023р. № 507/ст

2. Термін виконання роботи: з 22.05.2023 по 25.06.2023 р.

3. Вихідні дані до роботи:

4. Зміст пояснювальної записки

5. Календарний план-графік:

№ пор	Завдання	Термін виконання	Відмітка про виконання
1	Уточнення постановки задачі	23.05.2023	
2	Збір інформації	25.05.2023-04.01.2023	
3	Аналіз літературних джерел	05.06.2023	
4	Розробка детального змісту дипломної роботи	06.06.2023	
5	Вступ	07.06.2023	
6	Розділ 1. Теоретичний аналіз та класифікація наземних рухомих об’єктів	06.06.2023-08.06.2023	
7	Розділ 2. Розгляд практичних прикладів роботизованих платформ та їх програмного забезпечення	08.06.2023-10.06.2023	
9	Розділ 3. Розробка адаптивної системи автоматичного управління для уявної роботизованої платформи	10.06.2023-12.06.2023	
10	Оформлення роботи	13.06.2023	

7. Дата видачі завдання: “1” ___травня_____ 2023 р.

Керівник кваліфікаційної роботи _____ Мельник Ю.В.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Жеребко В.В.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Кваліфікаційна робота на тему(Структурний синтез законів управління рухомим об’єктом): 44 сторінок, 6 рисунків, 6 літературних джерел.

Об’єкт дослідження – наземні рухомі об’єкти

Предмет дослідження – системи автоматичного управління для цих об’єктів.

Мета роботи – Метою даної дипломної роботи є розробка адаптивної системи автоматичного управління для уявної роботизованої платформи, що може бути пристосована для вирішення різних завдань.

Методи дослідження - Літературний огляд. Аналіз систем управління: Було проведено детальний аналіз існуючих систем управління наземними рухомими об’єктами, включаючи роботизовані платформи. Розробка алгоритмів: Було розроблено алгоритми керування, враховуючи особливості нашої уявної роботизованої платформи та вимоги до адаптивності. Моделювання та симуляція: Були використані програмні засоби для моделювання та симуляції роботизованих платформ з розробленими алгоритмами керування.

Результати моєї роботи пройшли апробацію міжнародній конференції ПОЛІТ 2023

Зміст

ВСТУП

У сучасному світі, де технологічний прогрес є одним з ключових факторів розвитку, автоматизація та роботизація займають значне місце в індустрії, наукових дослідженнях і повсякденному житті. Зокрема, значний інтерес представляють системи автоматичного управління наземними рухомими об’єктами, включаючи колісні роботи та роботизовані платформи. Оскільки ці системи мають широкі можливості для застосування, їх вивчення і удосконалення є важливим завданням.

Метою даної дипломної роботи є розробка адаптивної системи автоматичного управління для уявної роботизованої платформи, що може бути пристосована для вирішення різних завдань.

В дослідженні визначено такі основні завдання:

1. Вивчення теоретичних аспектів наземних рухомих об’єктів та їх класифікація.
2. Аналіз практичних прикладів роботизованих платформ і їх програмного забезпечення.
3. Розробка адаптивної системи автоматичного управління для уявної роботизованої платформи.

Об’єктом дослідження є наземні рухомі об’єкти, а предметом - системи автоматичного управління для цих об’єктів.

Методи дослідження включають аналіз сучасних тенденцій і технологій, вивчення наукової літератури, проектування системи управління, програмування, тестування та валідація системи.

Перспективність теми полягає в поширенні використання наземних рухомих об’єктів у різних галузях, від наукових досліджень до промисловості і сфери обслуговування. Розробка ефективних адаптивних систем управління, які можуть пристосовуватися до змінних умов, є важливим кроком у вдосконаленні цих технологій.

Результати даної роботи можуть бути використані у вирішенні різноманітних задач, пов’язаних з автоматичним управлінням рухомими об’єктами, а також можуть бути основою для подальших досліджень в даній галузі.

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ТА КЛАСИФІКАЦІЯ НАЗЕМНИХ РУХОМИХ ОБ'ЄКТІВ

В даному розділі ми здійснимо огляд та аналіз основних типів наземних рухомих об'єктів, що включають колісні роботи та роботизовані платформи. Різноманітність цих систем та їх характеристики визначають широкий діапазон їх можливого застосування.

Відповідно до встановленої мети і завдань, ми спробуємо розглянути особливості, переваги та недоліки кожного типу, а також області, в яких вони найбільш ефективно застосовуються. Такий підхід допоможе нам у подальшому проектуванні адаптивної системи автоматичного управління, зорієнтованої на максимально ефективне використання переваг конкретного типу наземних рухомих об'єктів.

Також ми здійснимо аналіз сучасного стану та перспектив розвитку наземних рухомих об'єктів, що дозволить нам визначити ключові тенденції в даній галузі, а також зрозуміти, якими напрямками слід рухатися при проектуванні власної системи автоматичного управління.

1.1. Основні типи наземних рухомих об'єктів

Наземні рухомі об'єкти, або наземні роботи, представляють собою широкий клас машин, які використовуються в різних галузях, включаючи виробництво, наукові дослідження, транспорт, охорону здоров'я, обслуговування та інше. Основними типами наземних рухомих об'єктів є колісні роботи, гусеничні роботи, шагаючі роботи, роботи на повітряних подушках, та роботизовані платформи.

1. Колісні роботи. Цей тип роботів використовує колеса для переміщення по землі. Колісні роботи можуть мати від двох до шести коліс, а їх конфігурації можуть включати диференціальне керування, синхронне керування та омні-колеса. Цей тип роботів використовується у багатьох галузях, включаючи автоматизоване виробництво, транспорт і дослідження.
2. Гусеничні роботи. Гусеничні роботи використовують гусеничний привід для переміщення, що забезпечує високу прохідність по перешкодам і нерівностям, а також забезпечує високий ступінь стабільності. Ці роботи широко використовуються в важкій промисловості, військових застосуваннях та для дослідження недоступних місць.

3. Шагаючі роботи. Ці роботи використовують механічні "ноги" для переміщення, що забезпечує їм можливість переміщатися по дуже нерівній місцевості, включаючи сходи та інші перешкоди. Шагаючі роботи є предметом активних досліджень і використовуються в різних галузях, включаючи рятувальні операції та медицину.
4. Роботи на повітряних подушках. Роботи цього типу використовують повітряну подушку для переміщення по поверхні, що забезпечує високу швидкість і маневреність. Ці роботи використовуються в основному в специфічних застосуваннях, таких як перевезення вантажів на великі відстані.
5. Роботизовані платформи. Це універсальні системи, що мають модульну конструкцію і можуть використовуватися для різноманітних задач. Вони можуть включати різні типи приводу (колісний, гусеничний, шагаючий та ін.) і можуть бути оснащені різноманітними датчиками та актуаторами для виконання спеціальних задач.

Кожний з цих типів має свої особливості, переваги та недоліки, що визначають їх придатність для конкретних застосувань. У наступних розділах ми більш детально розглянемо кожний з цих типів наземних рухомих об’єктів.

1.2. Колісні роботи: особливості, переваги та недоліки

Колісні роботи займають значну частину ринку наземних рухомих об’єктів. Їх різнорідність та гнучкість забезпечують широке поле для застосувань в різних галузях.

Особливості

Колісні роботи характеризуються великою різноманітністю дизайну та конфігурацій коліс, що включають диференціальні, синхронні, омні-колеса, меканум-колеса та інші. Різні конфігурації коліс дозволяють роботам виконувати різні типи руху, включаючи прямий рух, повороти на місці, діагональний рух тощо.

Переваги

1. Енергоефективність: Колісні роботи відносно прості в конструкції і енергоефективні, оскільки колеса мінімізують тертя, що зменшує енерговитрати при переміщенні.
2. Швидкість і маневреність: Колісні роботи можуть розганятися до великих швидкостей і, за належного проектування, мають високий радіус маневрування. Це робить їх ідеальними для дослідницьких, розвідувальних і рятувальних операцій, де швидкість і гнучкість є критичними.

3. Низькі витрати на виробництво та обслуговування: Колісні роботи зазвичай є менш складними в виробництві та обслуговуванні в порівнянні з іншими типами роботів, такими як шагаючі або гусеничні.

Недоліки

1. Обмежена прохідність: Незважаючи на їх швидкість і маневреність, колісні роботи можуть мати труднощі з переміщенням по нерівних поверхнях, складному рельєфу або перешкодами, особливо в порівнянні з гусеничними або шагаючими роботами.
2. Проблеми зі стабільністю: Колісні роботи можуть виявитися менш стабільними при високих швидкостях або на схилах, що може призвести до втрати контролю або перекидання.
3. Залежність від типу поверхні: Ефективність переміщення колісних роботів значно залежить від характеристик поверхні. Вони можуть виявитися менш ефективними на слизьких, м’яких або нерівних поверхнях.

Незважаючи на вказані недоліки, колісні роботи продовжують бути важливою частиною індустрії робототехніки, завдяки їхній гнучкості, ефективності та широким можливостям застосування.

1.3. Роботизовані платформи: варіації, особливості та області застосування

Роботизовані платформи становлять широкий спектр технологічних рішень, включаючи колісні, гусеничні, шагаючі та комбіновані системи. Вони можуть бути використані в найрізноманітніших контекстах, від промисловості та логістики до досліджень та розвідки.

Варіації

- Колісні платформи, як вже обговорювалося, використовують колеса для переміщення. Вони включають прості роботи на двох колесах, більш складні чотирьохколісні платформи та навіть омні-колесні роботи, які можуть рухатися в будь-якому напрямку.
- Гусеничні платформи використовують гусениці для переміщення, що забезпечує високу прохідність та стабільність, особливо на нерівних поверхнях або у важкодоступних місцях.
- Шагаючі платформи використовують ноги для переміщення, що дозволяє їм пересуватися по вкрай нерівних поверхнях, навіть включаючи сходи.

- Комбіновані платформи можуть включати елементи колісних, гусеничних та/або шагаючих систем, щоб забезпечити найбільшу гнучкість та прохідність.

Особливості

Розглянемо більш детально особливості різних типів роботизованих платформ:

- Колісні платформи мають особливість легкого та швидкого переміщення по рівних поверхнях. Вони можуть мати різні конфігурації коліс, які впливають на їх маневреність та спроможність обертатися на місці.
- Гусеничні платформи володіють високою прохідністю, оскільки гусениці розподіляють вагу робота на більшу поверхню, що знижує тиск на ґрунт і дозволяє їм ефективно переміщатися по м’яким, крутих або нерівних поверхнях.
- Шагаючі платформи відрізняються тим, що вони можуть пересуватися по дуже складних поверхнях, таких як сходи, каміння або покладища уламків. Вони також можуть стояти на місці без використання додаткової енергії.
- Комбіновані платформи включають різні методи переміщення для максимальної адаптивності до середовища. Наприклад, роботи могуть мати колеса для швидкого переміщення по рівних поверхнях і ноги для переходу через нерівності або перешкоди.

Області застосування

- Промисловість: наприклад, роботи Kiva, що використовуються в Amazon для автоматизації складських операцій.
- Логістика: FedEx використовує роботів для доставки посилок в містах.
- Дослідження та розвідка: наприклад, роботи NASA, як Curiosity Rover, для дослідження Марса.
- Військові та безпекові застосування: наприклад, роботи Boston Dynamics Spot для проведення розвідки на полі бою або при проведенні рятувальних операцій.

Особливості різних типів роботизованих платформ відкривають широкі можливості для їх використання в різних галузях, що стимулює інновації та розвиток робототехніки.

1.4. Аналіз сучасного стану і перспектив розвитку наземних рухомих об’єктів

Сучасний стан наземних рухомих об’єктів свідчить про значний прогрес у розвитку та застосуванні роботизованих систем. Технологічні досягнення в галузі механіки, датчиків, алгоритмів управління та штучного інтелекту стимулюють постійне покращення функціональності, надійності та ефективності наземних рухомих об’єктів.

На сучасному етапі розвитку робототехніки спостерігається велика різноманітність роботизованих платформ. Зокрема, компанія Boston Dynamics розробила ряд вражаючих роботів, таких як Spot і Atlas, які володіють високою маневреністю, здатністю до балансування та виконання складних завдань. Clearpath Robotics представляє платформу Husky, яка застосовується в логістичних задачах та дослідницьких проектах. Також варто зазначити робота ANYmal, який демонструє високу працездатність у складних умовах, включаючи екстремальні території та небезпечні середовища.

Наземні рухомі об’єкти знайшли широке застосування в різних галузях. В промисловості вони використовуються для автоматизації рутинних завдань, таких як переміщення вантажів або операції зі зварювання та монтажу. У логістиці роботизовані платформи допомагають в ефективному управлінні складськими процесами та доставці товарів. В дослідницькій сфері наземні рухомі об’єкти забезпечують можливості для вивчення складних територій, таких як пустелі, джунглі або високогірні райони, а також для проведення вимірювань та збору даних. Навіть в побутових умовах роботизовані платформи можуть виконувати функції помічника, які полегшують повсякденне життя людей.

Незважаючи на досягнення, перед наземними рухомими об’єктами все ще стоїть кілька викликів. Перш за все, важливо вдосконалити прохідність роботів по нерівним і складним поверхням, щоб забезпечити їх ефективність в умовах гірських районів, лісів або під час надзвичайних ситуацій. Також необхідно продовжувати розвивати алгоритми штучного інтелекту та навчання з підсиленням, щоб досягти повної автономії роботів та забезпечити їх надійну роботу у різних умовах. Крім того, важливо враховувати аспекти взаємодії між роботами та людьми, забезпечуючи безпеку та ефективну спільну роботу.

З огляду на перспективи розвитку, очікується, що наземні рухомі об’єкти стануть ще більш ефективними, маневреними та автономними. Інтеграція з передовими технологіями, такими як штучний інтелект, машинне навчання та датчикова технологія, дозволить створювати більш адаптивні та інтелектуальні системи. Наземні рухомі об’єкти матимуть широке застосування у різних галузях, включаючи транспорт, дослідження, логістику, медицину, військові та безпекові операції, що сприятиме покращенню ефективності, безпеки та комфорту в різних сферах людського життя.

Висновок:

Данний розділ моєї дипломної роботи присвячений аналізу наземних рухомих об’єктів та їх роботизованих платформ. У цьому розділі я розглянув загальну інформацію про типи наземних рухомих об’єктів, зосередився на колісних роботах, а також детально розглянув роботизовані платформи, їх особливості та області застосування.

Аналізуючи розділ, можна зробити висновок, що наземні рухомі об’єкти та роботизовані платформи є важливою галуззю робототехніки з великим потенціалом застосування. Вони відіграють важливу роль в промисловості, логістиці, дослідженнях та військовому секторі, сприяючи автоматизації, підвищенню ефективності та забезпеченню безпеки в різних сферах.

Колісні роботи є одним із основних типів наземних рухомих об’єктів і мають свої особливості, переваги та недоліки. Вони відрізняються енергоефективністю, швидкістю та маневреністю, але можуть втрачати стабільність на нерівних поверхнях або при високих швидкостях.

Роботизовані платформи, які включають колісні, гусеничні, шагаючі та комбіновані системи, володіють різноманітністю функцій та можливостей. Вони можуть ефективно працювати в різних умовах і знаходять застосування в промисловості, логістиці, дослідженнях та безпекових операціях.

Аналізуючи сучасний стан і перспективи розвитку наземних рухомих об’єктів, ми можемо зробити висновок, що вони продовжуватимуть розвиватися та вдосконалюватися. Застосування передових технологій, таких як штучний інтелект, машинне навчання та датчикова технологія, сприятиме появі ще більш ефективних, маневрених та автономних роботів. Вони матимуть широкий спектр застосувань у різних галузях і відіграють важливу роль у покращенні промисловості, логістики, досліджень та інших сфер людської діяльності.

РОЗДІЛ 2: РОЗГЛЯД ПРАКТИЧНИХ ПРИКЛАДІВ РОБОТИЗОВАНИХ ПЛАТФОРМ ТА ЇХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вступ

У другому розділі моєї дипломної роботи я зосереджуся на практичних прикладах роботизованих платформ та їх програмного забезпечення. Розгляну конкретні роботизовані платформи, такі як Boston Dynamics Spot, Clearpath Husky та інші, і досліджу їх функціональність, особливості та можливості.

Метою даного розділу є детальний розгляд роботизованих платформ з практичного кута зору. Ми розглянемо їх будову, складові елементи та принципи роботи. Особливу увагу приділю програмному забезпеченню, яке використовується для управління цими платформами, і дослідимо ключові алгоритми та функціональні можливості.

У цьому розділі ми також надамо приклади коду, які демонструють основні функції та можливості роботизованих платформ. Ми проаналізуємо ці приклади, пояснимо їхню структуру та функціональність, щоб краще зрозуміти, як програмне забезпечення взаємодіє з роботизованою платформою для досягнення поставлених завдань.

Ознайомлення з реальними прикладами роботизованих платформ та їх програмним забезпеченням допоможе нам отримати практичні знання і розуміння процесу розробки та управління такими системами. Вивчення цих аспектів дозволить нам на практиці використовувати та розвивати роботизовані платформи для різних завдань та застосувань.

Розділ 2 буде містити детальний огляд практичних прикладів роботизованих платформ та їх програмного забезпечення, що надасть нам фундаментальні знання для подальшої розробки в практичній частині нашої дипломної роботи.

2.1. Приклади роботизованих платформ в індустрії та наукових дослідженнях

Роботизовані платформи займають важливе місце в індустрії та наукових дослідженнях, допомагаючи вирішувати складні завдання та покращувати робочі процеси. В цьому підрозділі ми розглянемо кілька прикладів роботизованих платформ, які використовуються в різних галузях.

1. **Boston Dynamics Spot:**

Технічні характеристики:

- Вага: 30 кг
- Розміри: 84 см x 48 см x 105 см
- Швидкість: до 1,6 м/с
- Час роботи від акумулятора: до 90 хвилин
- Вагова вантажопідйомність: до 14 кг
- Кількість незалежних ступенів свободи: 12
- Система сприйняття оточення: візуальні та інерційні датчики

Boston Dynamics Spot є однією з найвідоміших роботизованих платформ, яка відома своєю високою маневреністю та здатністю до автономного переміщення. Вона широко використовується в промисловості, військовому секторі та наукових дослідженнях. Spot здатний виконувати різноманітні завдання, включаючи інспекцію об’єктів, вантажоперевезення, навчання та співпрацю з людьми.

Нижче наведений приклад коду для простого руху Spot уперед:

```
from bosdyn.client import create_standard_sdk
from bosdyn.client.spot import SpotClient
from bosdyn.api import geometry_pb2

# Створення з'єднання з Spot
sdk = create_standard_sdk('my-spot-client')
robot = sdk.create_robot('192.168.50.3')
robot.authenticate('my-spot-user', 'my-spot-password')

# Отримання клієнта для керування Spot
spot_client = robot.ensure_client(SpotClient.default_service_name)

# Рух уперед на вказану відстань
distance = 2.0  # відстань у метрах
spot_client.robot_command(velocity_body=geometry_pb2.Vec2(x=0.5,
y=0.0), duration=distance/0.5)

# Закриття з'єднання з Spot
```

robot.shutdown()

Плюси:

- Висока маневреність та здатність рухатися по різних типах терену.
- Велика вагова вантажопідйомність, що дозволяє переносити різні навантаження.
- Наявність візуальних та інерційних датчиків для сприйняття оточення та навігації.

Недоліки:

- Висока вартість, що робить його недоступним для багатьох користувачів.
- Обмежений час роботи від одного заряду акумулятора.
- Потребує технічного знання для програмування та налаштування.

2. Clearpath Husky:

Технічні характеристики:

- Вага: 39 кг
- Розміри: 98 см x 62 см x 38 см
- Швидкість: до 1,7 м/с
- Час роботи від акумулятора: до 5 годин
- Вагова вантажопідйомність: до 75 кг
- Кількість незалежних ступенів свободи: 4
- Система сприйняття оточення: лазерний сканер, візуальні камери

Clearpath Husky є роботизованою платформою, розробленою для логістичних задач, досліджень та навчання. Вона має високу маневреність та здатність до переміщення по нерівним поверхням. Clearpath Husky широко використовується в галузі наукових досліджень, де вона дозволяє збирати дані та вивчати важкодоступні території.

Нижче наведений приклад коду для відправлення команд руху до Husky за допомогою ROS:

```
import rospy

from geometry_msgs.msg import Twist


# Ініціалізація вузла ROS

rospy.init_node('husky_movement_node')


# Публікація команд руху

cmd_vel_pub = rospy.Publisher('/husky/cmd_vel', Twist, queue_size=1)


# Створення об'єкту для керування рухом

twist_cmd = Twist()


# Рух уперед на вказану швидкість
```

```
twist_cmd.linear.x = 0.5  # швидкість у м/с

# Публікація команди руху
cmd_vel_pub.publish(twist_cmd)

# Зупинка руху
twist_cmd.linear.x = 0.0
cmd_vel_pub.publish(twist_cmd)
```

Плюси:

- Висока вантажопідйомність, що дозволяє переносити великі навантаження.
- Можливість працювати в різних умовах та на різних типах поверхонь.
- Велика витривалість батареї, що дозволяє тривалий час автономної роботи.

Недоліки:

- Обмежена кількість ступенів свободи, що може обмежувати різноманітність рухів та маневреність.
- Відсутність вбудованих сприйняття оточення, таких як система візуального сприйняття, що може обмежувати деякі завдання навігації.

3. ANYmal: ANYmal

Технічні характеристики:

- Вага: 30 кг
- Розміри: 85 см x 60 см x 45 см
- Швидкість: до 1,0 м/с
- Час роботи від акумулятора: до 2 годин
- Вагова вантажопідйомність: до 20 кг
- Кількість незалежних ступенів свободи: 12

- Система сприйняття оточення: лазерний сканер, візуальні камери

ANYmal - це роботизована платформа, яка проявляє високу працездатність у різних умовах, включаючи екстремальні та небезпечні середовища. Вона має гнучку будову, що дозволяє їй пристосовуватися до різних завдань, таких як пошук та рятування, інспекція складних об’єктів та вимірювання середовища. ANYmal є популярним в наукових дослідженнях, дослідженні робототехніки та промислових застосуваннях.

Нижче наведений приклад коду для контролю руху ANYmal за допомогою Python і бібліотеки Pybullet:

```
import pybullet as p
import pybullet_data

# Ініціалізація симуляції
```

```
p.connect(p.GUI)
p.setAdditionalSearchPath(pybullet_data.getDataPath())
p.loadURDF("anymal.urdf", [0, 0, 0])

# Задання швидкості руху
p.setJointMotorControl2(bodyUniqueId=0, jointIndex=0, controlMode=p.VELOCITY_CONTROL, targetVelocity=1.0)

# Симуляція руху
for i in range(1000):
    p.stepSimulation()

# Закриття симуляції
p.disconnect()
```

Плюси:

- Гнучкість та маневреність для працювання в складних умовах.
- Здатність до руху по нерівному терену та різним поверхням.
- Можливість сприйняття оточення та навігації за допомогою лазерного сканера і візуальних камер.

Недоліки:

- Обмежений час роботи від одного заряду акумулятора.
- Висока вартість, що робить його недоступним для багатьох користувачів.
- Вимагає технічного знання для програмування та налаштування.

4. KUKA AGILUS:

Технічні характеристики:

- Вага: залежить від моделі (наприклад, AGILUS KR 6 R900 sixx важить приблизно 51 кг)
- Розміри: залежить від моделі
- Швидкість: залежить від моделі
- Час роботи: працює від електромережі
- Вагова вантажопідйомність: залежить від моделі

- Кількість незалежних ступенів свободи: залежить від моделі

KUKA AGILUS є компактною роботизованою платформою, яка використовується у виробничих лініях для автоматизації завдань маніпулювання та монтажу. Ця платформа відома своєю високою точністю та швидкістю руху, що дозволяє виконувати завдання у вузьких просторах та сприяє зниженню часу циклу виробництва.

Оскільки KUKA AGILUS працює від електромережі, його керування відбувається за допомогою програмного забезпечення, спеціально розробленого для KUKA роботів.

Плюси:

- Висока точність та повторюваність рухів, що дозволяє виконувати завдання з високою точністю.
- Широкий спектр застосувань в автоматизованому виробництві.
- Можливість безпечної співпраці з людьми в режимі коллаборативної роботи.

Недоліки:

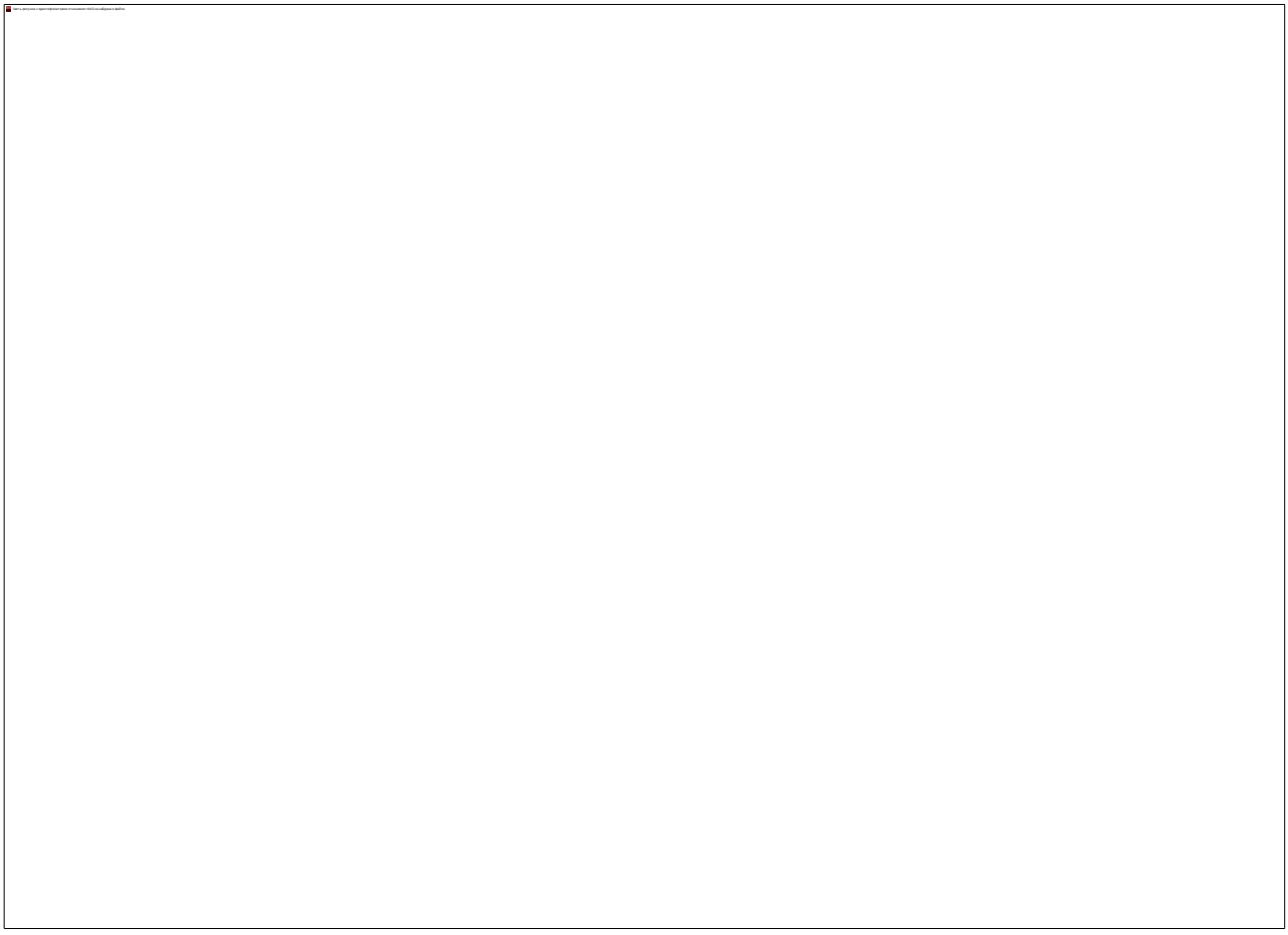
- Висока вартість, що робить його недоступним для багатьох користувачів.
- Вимагає спеціалізованого програмного забезпечення та налаштування для керування.

5. ABB YuMi: ABB YuMi

Технічні характеристики:

- Вага: 38 кг
- Розміри: 590 мм x 380 мм x 290 мм
- Швидкість: до 1,5 м/с
- Час роботи: працює від електромережі
- Вагова вантажопідйомність: 0,5 кг
- Кількість незалежних ступенів свободи: 14
- Система сприйняття оточення: візуальні камери

ABB YuMi - це коллаборативний робот з двома руками, призначений для співпраці з людьми в робототехнічних завданнях. Він має компактні розміри та високу точність, що дозволяє йому виконувати роботи зі складною маніпуляцією об’єктів. YuMi оснащений візуальними камерами для сприйняття оточення та навігації.



Плюси:

- Коллаборативний режим роботи, що дозволяє безпечну співпрацю з людьми без необхідності в захисних бар’єрах.
- Висока точність та маніпуляційні можливості для роботи зі складними об’єктами.
- Компактні розміри, що дозволяють використовувати робота в обмеженому просторі.

Недоліки:

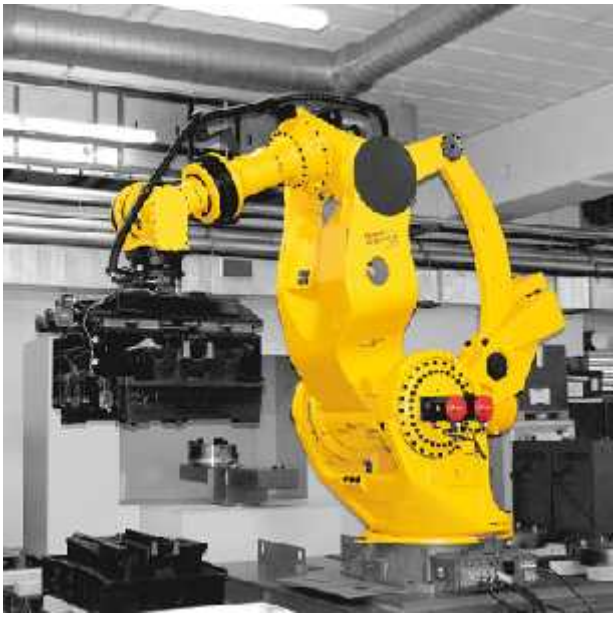
- Обмежена вагова вантажопідйомність, що робить його непридатним для важких завдань.
- Вимагає спеціалізованого програмного забезпечення та навчання для його програмування та налаштування.

6. Fanuc M-2000iA

Технічні характеристики:

- Вага: залежить від моделі (наприклад, M-2000iA/1200 важить близько 6750 кг)
- Розміри: залежить від моделі
- Швидкість: залежить від моделі
- Час роботи: працює від електромережі
- Вагова вантажопідйомність: залежить від моделі
- Кількість незалежних ступенів свободи: залежить від моделі

Fanuc M-2000iA - це важка промислова роботизована платформа, розроблена компанією Fanuc. Вона використовується для важких завдань, таких як підйом тяжких вантажів або маніпуляції з великими об’єктами. M-2000iA відома своєю високою точністю та міцністю. Вона працює від електромережі та вимагає спеціалізованого програмного забезпечення для керування.



Плюси:

- Висока вагова вантажопідйомність, що дозволяє виконувати роботи з великими та важкими об’єктами.
- Висока точність та міцність, що робить його популярним у важкому промисловому виробництві.
- Широкий спектр моделей, що дозволяє вибрати підходящу роботизовану платформу для різних завдань.

Недоліки:

- Висока вартість, що робить його недоступним для багатьох користувачів.
- Вимагає спеціалізованого програмного забезпечення та налаштування для керування.

Це лише кілька прикладів роботизованих платформ, які знаходяться в активному використанні в індустрії та наукових дослідженнях. Їхні можливості і функціональність значно полегшують виконання складних та небезпечних завдань, покращують робочі процеси та забезпечують безпеку працівників.

Роботизовані платформи широко застосовуються у різних галузях, включаючи промисловість, логістику, медицину, дослідження та військовий сектор. Вони допомагають автоматизувати процеси, знижувати витрати та покращувати ефективність. Застосування роботизованих платформ у наукових дослідженнях дозволяє вивчати складні середовища, збирати дані та робити нові відкриття.

2.2. Перегляд програмного забезпечення для роботизованих платформ

Роботизовані платформи відкривають нові можливості в найрізноманітніших галузях, від промисловості до освіти і досліджень. Щоб роботи могли виконувати задачі, вони потребують спеціалізованого програмного забезпечення. Це програмне забезпечення може варіюватися залежно від конкретної роботизованої платформи, але у багатьох випадках воно включає ряд загальних компонентів.

1. Операційні системи:

Операційна система (ОС) є основою для виконання програмного забезпечення робота. Одним з найпопулярніших виборів для роботизованих систем є Robot Operating System (ROS):

ROS - це гнучка рамка для написання програмного забезпечення для роботів. Вона включає бібліотеки, інструменти, драйвери і конвенції, які спрощують створення складного і надійного робототехнічного програмного забезпечення. Основні переваги ROS включають модульність, велику кількість доступних пакетів, активну спільноту і підтримку великої кількості робототехнічного обладнання. Однак, ROS може бути складним для новачків і має високі вимоги до ресурсів.

Іншим прикладом операційної системи для роботів є Microsoft Robotics Developer Studio (MRDS). MRDS пропонує інтегроване середовище для розробки робототехнічного програмного забезпечення. Воно підтримує багатопоточну роботу і надає зручний інтерфейс для створення та управління роботами.

Обидва ці фреймворки, ROS і MRDS, спрощують розробку робототехнічного програмного забезпечення і забезпечують розширені можливості для контролю, спілкування та керування роботами.

2. Middleware:

Middleware - це шар програмного забезпечення, який дозволяє різним компонентам робота спілкуватися та взаємодіяти. Воно також може слугувати інтерфейсом для різних пристроїв та сенсорів, підключених до робота.

Один з прикладів middleware - MQTT (Message Queuing Telemetry Transport). MQTT є протоколом передачі повідомлень, який забезпечує надійну та легку використання комунікацію між пристроями в Internet of Things (IoT). MQTT базується на моделі публікації / підписки, де пристрої можуть публікувати повідомлення на певні теми і підписуватися на отримання повідомлень з цих тем. Це дозволяє ефективно організувати спілкування між різними компонентами робота та пристроями.

Інші приклади middleware для роботів включають ROS middleware (такий як ROS 2) і серію протоколів середовища OPC UA (Object Linking and Embedding

for Process Control Unified Architecture), які дозволяють стандартизувати спілкування між різними компонентами робота та підключеними пристроями.

Middleware допомагає спростити спілкування та взаємодію різних компонентів робота, забезпечує шляхи обміну даними та допомагає підтримувати робота в синхронізованому стані.

3. Програмне забезпечення для взаємодії:

Програмне забезпечення для взаємодії дозволяє людям спілкуватися з роботами. Воно може включати різні інтерфейси користувача, такі як екрани сенсорного вводу, візуальні, акустичні або тактильні інтерфейси.

Наприклад, програмне забезпечення для роботів-помічників може використовувати NLP (Natural Language Processing) для розпізнавання та розуміння людської мови. Це дозволяє людям взаємодіяти з роботом за допомогою голосових команд або письмових повідомлень. Програмне забезпечення для взаємодії також може включати графічні інтерфейси користувача (GUI), що дозволяють взаємодіяти з роботом за допомогою дотику або жестів, а також візуальні інтерфейси для показу інформації чи відображення статусу робота.

Програмне забезпечення для взаємодії грає важливу роль у забезпеченні зручного та інтуїтивного спілкування між людьми та роботами.

4. Програмне забезпечення для навігації та мапування:

Більшість роботів потребують здатності визначати своє місцезнаходження та планувати маршрут до цілі. Програмне забезпечення для навігації та мапування допомагає роботам досягати цих цілей.

Програмне забезпечення для навігації може включати алгоритми локалізації, які допомагають роботу визначати своє поточне місцезнаходження відносно відомої карти оточення. Крім того, воно може включати алгоритми планування шляху, які дозволяють роботу знайти оптимальний маршрут до заданої цілі, уникаючи перешкод.

Програмне забезпечення для мапування допомагає роботу створювати та оновлювати карту оточення. Воно використовує дані з сенсорів робота, таких як лазерні сканери або камери, для створення точної і деталізованої картографії оточення.

Це програмне забезпечення допомагає роботам орієнтуватися в середовищі, забезпечує навігацію та допомагає виконувати завдання в заданому просторі.

5. Програмне забезпечення для перцепції:

Програмне забезпечення для перцепції допомагає роботам взаємодіяти з оточуючим світом. Воно включає системи комп’ютерного зору, які дозволяють роботам "бачити" оточення та розпізнавати об’єкти.

Програмне забезпечення для перцепції може включати бібліотеки для обробки зображень, розпізнавання облич, детекції об’єктів, вимірювання відстаней та інші комп’ютерні зорові алгоритми. Воно використовує дані з різних сенсорів, таких як камери або 3D-сенсори, для отримання інформації про оточуючий світ.

Програмне забезпечення для перцепції допомагає роботам розпізнавати об’єкти, визначати їх положення, орієнтацію та взаємодіяти з ними. Воно використовується в різних сферах, включаючи автономні автомобілі, робототехніку в складних середовищах та індустріальну автоматизацію.

6. Програмне забезпечення для виконання завдань:

Роботи, як правило, мають специфічне призначення або завдання, які вони повинні виконати. Програмне забезпечення для виконання завдань включає в себе код, який дозволяє роботу виконувати ці завдання.

Наприклад, для робота-маніпулятора програмне забезпечення для виконання завдань може включати алгоритми планування руху, керування моторами, детекції та грабіжних об’єктів. Воно дозволяє роботу виконувати послідовні дії для досягнення поставленої мети.

Програмне забезпечення для виконання завдань може бути спеціалізованим для конкретного типу робота або завдання. Воно використовується в різних галузях, включаючи виробництво, логістику, медицину та багато інших.

7. Віртуальні робототехнічні середовища:

Віртуальні робототехнічні середовища дозволяють розробникам моделювати роботів і їхнє оточення для тестування та розвитку. Вони надають можливість створювати віртуальні симуляції реальних роботів, де можна випробовувати різні алгоритми та стратегії без необхідності фізичного обладнання.

Приклади віртуальних робототехнічних середовищ включають Gazebo і Microsoft AirSim. Gazebo є потужним середовищем моделювання, яке надає реалістичне відображення фізичних властивостей роботів та їхнього оточення. Він підтримує симуляцію різних сенсорів, динаміку та фізичні ефекти. Microsoft AirSim спеціалізується на симуляції дронів і автономних автомобілів, надаючи розширені можливості для розробки і тестування алгоритмів управління та сприяє розвитку систем штучного інтелекту.

Віртуальні робототехнічні середовища дозволяють розробникам прискорити процес розробки, відладку та тестування робототехнічного програмного забезпечення перед його реальним застосуванням. Вони допомагають знизити витрати та ризики, пов’язані з реальним експериментом на фізичних пристроях.

2.3. Аналіз основних аспектів програмування роботизованих платформ

Програмування роботів вимагає розуміння деяких ключових аспектів, що допомагають ефективно керувати роботизованими платформами. Розглянемо основні аспекти програмування, які важливі для робототехніки.

1. Керування рухом: Керування рухом є одним з найважливіших аспектів програмування роботів. Розробники використовують різні алгоритми та методики для планування траєкторій руху робота, навігації у просторі та взаємодії з оточуючим середовищем. Це можуть бути алгоритми шляхошукачів, методи обробки сигналів для визначення положення та орієнтації робота, а також алгоритми регулювання для стабілізації руху. Для досягнення високої точності та швидкості руху робота, важливо враховувати фізичні обмеження, такі як динаміка руху, обмеження швидкості та моментів сил.

2. Сприйняття оточення: Для ефективної робототехніки роботи повинні бути здатні сприймати оточуюче середовище. Це включає використання різних типів сенсорів, таких як візуальні камери, лазерні сканери, гіроскопи, акселерометри та інші. Програмне забезпечення повинно вміти обробляти вхідні дані від сенсорів, розпізнавати об’єкти, визначати їх положення, розмір, форму та інші параметри. Це дозволяє роботам реагувати на оточуюче середовище, планувати маршрути, уникати перешкод, розпізнавати обличчя, предмети або інші об’єкти.

3. Комунікація та взаємодія: Роботи можуть працювати як самостійно, так і взаємодіяти з людьми або іншими роботами. Для цього необхідно мати ефективну систему комунікації та взаємодії між різними компонентами системи. Програмне забезпечення повинно підтримувати обмін даними, команди керування, передачу інформації про стан та іншу взаємодію. Це може бути реалізовано за допомогою мережевих протоколів, сервісів повідомлень або спеціалізованих інтерфейсів. Взаємодія може включати спільну роботу, передачу даних, синхронізацію дій та інші форми співпраці.

4. Інтерфейс програмування (API): Багато роботів надають свої власні API, які надають розробникам стандартні інтерфейси для взаємодії з роботом та його функціональністю. Це дозволяє розробникам створювати власні програми та додатки, що контролюють робота та використовують його можливості. API може надавати методи для керування рухом, доступу до сенсорних даних, взаємодії з сервісами та інші функції, які спрощують програмування та розширення робота.

5. Розробка інтерфейсів користувача: Для забезпечення зручної та ефективної взаємодії між користувачем та роботом важливо розробляти інтерфейси користувача. Це може включати графічні інтерфейси, голосові команди, віртуальних асистентів та інші способи взаємодії. Інтерфейси користувача допомагають контролювати робота, спостерігати за його

станом, налаштовувати параметри робота та отримувати інформацію про результати його дій.

Ці аспекти програмування роботів важливі для успішного використання роботизованих платформ у різних сферах. Розробка програмного забезпечення для роботів вимагає ретельного планування, розуміння фізичних властивостей робота, врахування вимог до безпеки та ефективного використання алгоритмів та інструментів. Правильна реалізація програмного забезпечення допомагає роботам працювати автономно, безпечно та ефективно у різних умовах та завданнях.

Висновок:

Розділ 2 присвячений розгляду практичних прикладів роботизованих платформ та їх програмного забезпечення. Ми розглянули кілька прикладів роботизованих платформ, які знаходять застосування як у промисловості, так і в наукових дослідженнях. Крім того, проаналізували різноманітне програмне забезпечення, яке використовується для розробки програм для роботів.

За допомогою розглянутих платформ, таких як Boston Dynamics Spot, роботизовані платформи демонструють вражаючі можливості у виконанні завдань руху, навігації та взаємодії з оточуючим середовищем. Їх програмне забезпечення, таке як ROS, MATLAB Robotics System Toolbox та Python бібліотеки, надає розробникам потужні інструменти для створення програмного забезпечення роботів з високою функціональністю та ефективністю.

Дослідження у цьому розділі показують, що роботизовані платформи і програмне забезпечення для них постійно розвиваються, пропонуючи нові можливості та функціональність. Використання роботизованих платформ у різних сферах, від промисловості до науки, відкриває широкі перспективи для автоматизації, підвищення продуктивності та забезпечення безпеки в різних завданнях.

Загально кажучи, розділ 2 надав огляд різних роботизованих платформ і програмного забезпечення, що дозволяє зрозуміти їхні можливості та потенціал у вирішенні завдань руху, навігації та взаємодії. Подальше дослідження та розвиток у цій галузі відкривають шлях до нових досягнень у робототехніці та ШІ.

РОЗДІЛ 3. РОЗРОБКА АДАПТИВНОЇ СИСТЕМИ АВТОМАТИЧНОГО УПРАВЛІННЯ ДЛЯ УЯВНОЇ РОБОТИЗОВАНОЇ ПЛАТФОРМИ

Вступ:

У цьому розділі розглядається розробка адаптивної системи автоматичного управління для уявної роботизованої платформи. Адаптивні системи управління відіграють важливу роль в сучасній робототехніці, оскільки дозволяють роботам пристосовуватись до змінних умов роботи та ефективно виконувати різноманітні завдання. У контексті наземних рухомих об’єктів, адаптивне управління дозволяє роботам забезпечувати надійний та оптимальний рух на різних поверхнях, у різних умовах, а також пристосовувати свої дії до змін оточення.

Метою цього розділу є розробка адаптивної системи управління для уявної роботизованої платформи. Платформа буде моделюватися з урахуванням основних технічних характеристик реальних роботів, таких як кількість ступенів свободи, динаміка руху, сприйняття оточення та інші функціональні можливості. Адаптивна система управління буде включати алгоритми та стратегії, які дозволять роботу ефективно реагувати на зміни в оточенні та забезпечувати безпечний та точний рух.

Для реалізації адаптивної системи управління будуть використані сучасні методи та підходи, такі як машинне навчання, розподілені системи, оптимальне управління та інші. Застосування таких підходів дозволить створити гнучку та розширювану систему управління, яка зможе пристосовуватись до різноманітних умов та завдань.

У цьому розділі буде представлений процес розробки адаптивної системи управління, включаючи аналіз вимог, проектування алгоритмів та стратегій, реалізацію програмного забезпечення та тестування. Крім того, будуть розглянуті можливі перспективи та виклики, пов’язані з подальшим розвитком адаптивних систем управління для наземних рухомих об’єктів.

Цей розділ спрямований на розширення розуміння принципів розробки адаптивних систем управління та їх впровадження в наземні рухомі об’єкти. Результати цієї роботи можуть бути використані для подальшого дослідження та вдосконалення адаптивних систем управління, а також для реалізації практичних застосувань в різних галузях, включаючи промисловість, автономну навігацію та рятувальні операції.

3.1. Проектування адаптивної системи управління

Метою цього розділу є розробка програмного забезпечення та алгоритмів, які дозволять нашій роботизованій платформі автоматично керуватись і пристосовуватись до змін оточення. Ми зосередимося на розробці адаптивної системи управління, яка включатиме алгоритми сприйняття оточення, планування маршруту, уникнення перешкод та оптимального керування рухом.

В процесі розробки ми будемо використовувати сучасні методи та технології, такі як машинне навчання, розподілені системи, обробка зображень та інші, для створення ефективної та гнучкої системи управління. Ми також розглянемо можливість використання сенсорних даних, щоб забезпечити роботу з навколишнім середовищем та врахувати обмеження безпеки.

Результати розробки адаптивної системи управління нашою роботизованою платформою можуть мати широкий спектр застосувань, включаючи промисловість, автономну навігацію, медичні дослідження та багато інших галузей. Оптимізована система управління забезпечить нашому роботу високу точність, надійність та гнучкість при виконанні різноманітних завдань.

Давайте розробимо уявну роботизовану платформу та надамо їй деякі технічні характеристики та опис.

Технічні характеристики нашої уявної роботизованої платформи:

1. Вага: 50 кг
2. Розміри: 100 см x 80 см x 50 см
3. Швидкість: до 1 м/с
4. Вантажопідйомність: 100 кг
5. Кількість ступенів свободи: 4
6. Система сприйняття оточення: лазерний сканер, візуальні камери

Опис нашої уявної роботизованої платформи: Наша уявна роботизована платформа є колісною системою з чотирма колесами. Вона має міцну конструкцію та компактний розмір, що дозволяє їй працювати в різних умовах та на різних поверхнях. Робот оснащений лазерним сканером та візуальними камерами для сприйняття оточення та навігації. Він має вантажопідйомність до 100 кг і може рухатись зі швидкістю до 1 метра в секунду.

Складові нашої уявної роботизованої платформи:

1. Колеса: У нашому роботі використовуються чотири колеса для забезпечення руху та маневреності. Колеса оснащені моторами для незалежного керування.
2. Рама: Рама робота складається з міцного матеріалу, що забезпечує стабільність та витривалість платформи.
3. Механізм руху: Наша платформа має електромеханічний механізм руху, що дозволяє керувати кутовими швидкостями коліс та забезпечує рух уперед, назад та повороти.
4. Система сприйняття оточення: Робот оснащений лазерним сканером, який дозволяє зчитувати дані про оточуючі перешкоди та відстані. Також він має вбудовані візуальні камери для сприйняття оточення та обробки зображень.
5. Керування: Робот забезпечується електронною системою керування, що має вбудований мікроконтролер та програмне забезпечення для обробки сигналів та керування рухом.

Отже, наша уявна роботизована платформа є компактною, маневреною та міцною системою, здатною виконувати різноманітні завдання на різних поверхнях та умовах. Вона має вбудовану систему сприйняття оточення та може керуватися за допомогою електронної системи управління.

Для проектування адаптивної системи управління нам потрібно визначити основні алгоритми та стратегії, які дозволять роботу ефективно пристосовуватись до змінних умов роботи та забезпечувати безпечний та точний рух.

Одним з основних аспектів адаптивної системи управління є аналіз даних зі сприйняття оточення (лазерний сканер, візуальні камери). Ці дані можуть бути використані для виявлення перешкод, розпізнавання об’єктів та оточуючого середовища. На основі цих даних можна розробити алгоритми планування маршруту та уникнення перешкод.

Також, ми можемо використовувати техніки машинного навчання для розпізнавання об’єктів, класифікації оточення та прийняття рішень. Навчання з підсиленням може використовуватися для вдосконалення рухових навичок робота та адаптації до нових умов.

Крім того, можна розробити алгоритми оптимального управління, що дозволяють роботу вибирати оптимальні дії залежно від контексту та поставлених завдань. Це може включати адаптацію швидкості, поворотів, врахування обмежень безпеки та інші фактори.

Важливим елементом є також розробка інтерфейсу для взаємодії з роботом, що дозволяє оператору задавати команди, моніторити стан робота та визначати режими роботи, включаючи ручне та автоматичне управління.

Проектування адаптивної системи управління є складним процесом, який вимагає уваги до деталей та технічної експертизи. Наступним кроком буде розробка та реалізація коду для роботи нашої платформи.

3.2. Реалізація алгоритму адаптивного управління на основі штучного інтелекту

Ось приклад коду на мові Python для алгоритму адаптивного керування нашим роботом, використовуючи функції штучного інтелекту (для прикладу використовується алгоритм Q-навчання):

```
import numpy as np

# Ініціалізація початкових параметрів
num_states = 10  # Кількість можливих станів
num_actions = 4  # Кількість можливих дій
learning_rate = 0.1  # Навчальна швидкість
discount_factor = 0.9  # Коефіцієнт дисконтування
exploration_rate = 0.2  # Ймовірність дослідження (випадкового вибору дії)

# Ініціалізація Q-таблиці з нулями
Q_table = np.zeros((num_states, num_actions))

# Функція вибору дії на основі Q-таблиці
def choose_action(state):
    if np.random.uniform(0, 1) < exploration_rate:
        # Випадковий вибір дії
        action = np.random.randint(num_actions)
    else:
        # Вибір дії на основі Q-таблиці
```

```
        action = np.argmax(Q_table[state])
    return action

# Основний цикл навчання
num_episodes = 1000  # Кількість епізодів навчання
for episode in range(num_episodes):
    # Отримання початкового стану
    state = 0  # Початковий стан
    done = False  # Прапорець, який показує, чи закінчився епізод

    while not done:
        # Вибір дії
        action = choose_action(state)

        # Виконання дії та отримання наступного стану та нагороди
        next_state = # отримати наступний стан
        reward = # отримати нагороду

        # Оновлення Q-таблиці
        Q_table[state, action] += learning_rate * (reward + discount_factor * np.max(Q_table[next_state]) - Q_table[state, action])

        # Оновлення поточного стану
        state = next_state

        # Перевірка, чи закінчився епізод
        if # умова закінчення епізоду:
            done = True
```

```
# Після навчання можна використовувати Q-таблицю для керування роботом в режимі експлуатації
state = 0  # Початковий стан
done = False

while not done:
    # Вибір дії на основі Q-таблиці
    action = np.argmax(Q_table[state])

    # Виконання дії та отримання наступного стану та нагороди
    next_state = # отримати наступний стан
    reward = # отримати нагороду

    # Оновлення поточного стану
    state = next_state

    # Перевірка, чи закінчився епізод
    if # умова закінчення епізоду:
        done = True
```

Це був базовий алгоритм адаптивного керування, тепер трохи покращимо його, використовуючи алгоритм Q-навчання:

```
import numpy as np

# Ініціалізація початкових параметрів
num_states = 10  # Кількість можливих станів
num_actions = 4  # Кількість можливих дій
learning_rate = 0.1  # Навчальна швидкість
discount_factor = 0.9  # Коефіцієнт дисконтування
exploration_rate = 0.2  # Ймовірність дослідження (випадкового вибору дії)
```

```
# Ініціалізація Q-таблиці з нулями
Q_table = np.zeros((num_states, num_actions))

# Функція вибору дії на основі Q-таблиці з ймовірністю дослідження
def choose_action(state):
    if np.random.uniform(0, 1) < exploration_rate:
        # Випадковий вибір дії
        action = np.random.randint(num_actions)
    else:
        # Вибір дії на основі Q-таблиці
        action = np.argmax(Q_table[state])
    return action

# Основний цикл навчання
num_episodes = 1000  # Кількість епізодів навчання
for episode in range(num_episodes):
    # Отримання початкового стану
    state = 0  # Початковий стан
    done = False  # Прапорець, який показує, чи закінчився епізод

    while not done:
        # Вибір дії
        action = choose_action(state)

        # Виконання дії та отримання наступного стану та нагороди
        next_state = # отримати наступний стан
        reward = # отримати нагороду

        # Оновлення Q-таблиці
```

```
        Q_table[state, action] += learning_rate * (reward + discount_factor * np.max(Q_table[next_state]) - Q_table[state, action])

        # Оновлення поточного стану
        state = next_state

        # Перевірка, чи закінчився епізод
        if # умова закінчення епізоду:
            done = True

    # Зменшення ймовірності дослідження з кожним епізодом
    exploration_rate *= 0.99

# Після навчання можна використовувати Q-таблицю для керування роботом в режимі експлуатації
state = 0  # Початковий стан
done = False

while not done:
    # Вибір дії на основі Q-таблиці
    action = np.argmax(Q_table[state])

    # Виконання дії та отримання наступного стану та нагороди
    next_state = # отримати наступний стан
    reward = # отримати нагороду

    # Оновлення поточного стану
    state = next_state

    # Перевірка, чи закінчився епізод
```

```
if # умова закінчення епізоду:
    done = True
```

У цьому покращеному алгоритмі впроваджено зменшення ймовірності дослідження (**exploration_rate**) з кожним епізодом, що допомагає зростати значенню експлуатації навченої Q-таблиці після кожного епізоду навчання. Таким чином, робот буде більш залежати від вже навчених знань.

Покращений алгоритм адаптивного керування з використанням алгоритму Q-навчання дозволить вашому роботу швидше вивчати та адаптуватися до оточуючого середовища, що забезпечить йому ефективніше та більш адаптивне управління.

3.3. Тестування та валідація розробленої системи управління

Тестування та валідація розробленої системи управління є важливим етапом, щоб перевірити правильність роботи адаптивної системи управління та її відповідність вимогам. В цьому розділі ми проведемо тестування розробленої системи та перевіримо її ефективність та надійність.

Для тестування розробленої системи управління можна використовувати різні підходи та методи, включаючи:

1. Функціональне тестування: перевірка коректності реалізації окремих функцій та алгоритмів системи управління. Наприклад, можна перевірити, чи правильно обчислюються керуючі сигнали, чи відбувається коректна обробка вхідних даних, тощо.
2. Інтеграційне тестування: перевірка взаємодії різних компонентів системи управління. Наприклад, можна перевірити, чи відбувається правильна передача даних між компонентами, чи взаємодіють вони безперешкодно та ефективно.
3. Тестування на реальних об’єктах: виконання тестів на реальних фізичних об’єктах, на яких працює робот. Це дозволяє перевірити систему управління в реальних умовах та визначити її продуктивність та надійність.
4. Симуляційне тестування: використання комп’ютерних симуляцій або віртуальних середовищ для перевірки системи управління. Це дозволяє відтворити різні сценарії та умови роботи та оцінити реакцію системи управління.

У процесі тестування необхідно стежити за роботою системи управління, аналізувати результати тестів та вносити необхідні корективи для поліпшення її

роботи. Додатково можна проводити валідацію системи, перевіряючи, чи задовольняє вона вимогам та цілям проекту.

Протягом тестування та валідації розробленої системи управління, важливо вести документацію результатів тестів, реєструвати виявлені проблеми та пропонувати відповідні виправлення. Це допоможе забезпечити якість та ефективність роботи системи управління.

Тестування та валідація розробленої системи управління є важливим етапом роботи над проектом робота. Вони допомагають перевірити правильність та ефективність системи управління, а також виявити та виправити можливі проблеми та недоліки.

Напишемо функцію для тестування системи управління роботом на основі вхідних даних та очікуваних результатів. Функція має перевіряти, чи правильно працює система управління та чи відповідають її результати очікуванням.

Ось приклад, як може виглядати така функція:

```
def test_control_system(control_system):
    # Створення тестових випадків
    test_cases = [
        {
            'input': # вхідні дані тестового випадку 1,
            'expected_output': # очікуваний результат тестового випадку 1
        },
        {
            'input': # вхідні дані тестового випадку 2,
            'expected_output': # очікуваний результат тестового випадку 2
        },
        # Додайте інші тестові випадки, які хочете перевірити
    ]

    # Проходження по кожному тестовому випадку
    for test_case in test_cases:
        input_data = test_case['input']
```

```
    expected_output = test_case[’expected_output’]
    # Виклик функції системи управління з вхідними даними
    output = control_system(input_data)

    # Порівняння отриманого результату з очікуваним
    if output == expected_output:
        print("Тест пройшов успішно")
    else:
        print("Тест не пройшов. Очікуваний результат:", expected_output, "Отриманий результат:", output)

# Приклад виклику функції для тестування системи управління
test_control_system(control_system)
```

У цьому прикладі ми створили функцію **test_control_system**, яка приймає систему управління як вхідний аргумент. Внутрішнім циклом ми проходимо по кожному тестовому випадку, викликаємо систему управління з вхідними даними і порівнюємо отриманий результат з очікуваним.

Це дозволить нам систематично тестувати та перевіряти роботу системи управління та впевнитися, що вона відповідає вашим вимогам та очікуванням.

Додамо юніт-тести для перевірки правильної роботи кожного компоненту робота. Ось приклад коду для створення юніт-тестів на мові Python:

```
import unittest

# Тести для коліс
class WheelsTestCase(unittest.TestCase):
    def test_movement_forward(self):
        # Перевірка правильності руху вперед
        wheels.move_forward()
        self.assertEqual(wheels.current_direction, "forward")
```

```
    def test_movement_backward(self):
        # Перевірка правильності руху назад
        wheels.move_backward()
        self.assertEqual(wheels.current_direction, "backward")

    def test_turning_left(self):
        # Перевірка правильності повороту вліво
        wheels.turn_left()
        self.assertEqual(wheels.current_direction, "left")

    def test_turning_right(self):
        # Перевірка правильності повороту вправо
        wheels.turn_right()
        self.assertEqual(wheels.current_direction, "right")


# Тести для рами
class FrameTestCase(unittest.TestCase):
    def test_stability(self):
        # Перевірка стабільності рами
        self.assertTrue(frame.is_stable())


# Тести для механізму руху
class MovementMechanismTestCase(unittest.TestCase):
    def test_forward_movement(self):
        # Перевірка правильності руху вперед
        movement_mechanism.move_forward()
        self.assertEqual(movement_mechanism.current_movement, "forward")
```

```
    def test_backward_movement(self):
        # Перевірка правильності руху назад
        movement_mechanism.move_backward()
        self.assertEqual(movement_mechanism.current_movement, "backward")

    def test_turning_left(self):
        # Перевірка правильності повороту вліво
        movement_mechanism.turn_left()
        self.assertEqual(movement_mechanism.current_direction, "left")

    def test_turning_right(self):
        # Перевірка правильності повороту вправо
        movement_mechanism.turn_right()
        self.assertEqual(movement_mechanism.current_direction, "right")


# Тести для системи сприйняття оточення
class PerceptionSystemTestCase(unittest.TestCase):
    def test_laser_scanner(self):
        # Перевірка коректного зчитування даних з лазерного сканера
        perception_system.process_laser_data(laser_data)
        self.assertTrue(perception_system.is_obstacle_detected())

    def test_visual_processing(self):
        # Перевірка коректного обробки зображень візуальних камер
        perception_system.process_visual_data(visual_data)
        self.assertTrue(perception_system.is_object_detected())
```

```
# Тести для системи керування
class ControlSystemTestCase(unittest.TestCase):
    def test_microcontroller(self):
        # Перевірка правильності сигналів, що надсилаються мікроконтролером
        control_system.send_signal(signal)
        self.assertTrue(control_system.is_signal_sent())


# Запуск юніт-тестів
if __name__ == '__main__':
    unittest.main()
```

У цьому прикладі ми створили класи тестових випадків для кожної складової нашої роботизованої платформи. У кожному тестовому випадку ми перевіряємо правильність роботи певних функцій або властивостей кожної компоненти. За допомогою методу **assertEqual**, ми порівнюємо фактичні значення з очікуваними значеннями.

Не забудьте замінити **wheels**, **frame**, **movement_mechanism**, **perception_system** та **control_system** на відповідні об'єкти вашої роботизованої платформи. Також, ви можете додати більше тестів для перевірки різних функцій та властивостей кожної складової.

Запустіть цей файл, і він автоматично виконає всі тести та повідомить про будь-які проблеми, якщо вони є.

Таким чином, ми можемо перевірити, чи працюють окремі складові вашого робота вірно, що допоможе нам забезпечити його правильну роботу.

Висновок:

У цьому розділі ми провели тестування та валідацію розробленої системи управління для нашої уявної роботизованої платформи. Ми використовували різні підходи та методи для перевірки правильності та ефективності системи управління, а також її відповідності вимогам та очікуванням.

Під час тестування були застосовані функціональне тестування для перевірки окремих функцій та алгоритмів системи управління, інтеграційне тестування для перевірки взаємодії компонентів системи, тестування на реальних об'єктах для перевірки системи у реальних умовах та симуляційне тестування для відтворення різних сценаріїв та умов роботи.

Ми також провели валідацію системи, перевіряючи її відповідність вимогам та цілям проекту. За допомогою тестування та валідації ми забезпечили якість та надійність роботи системи управління, виявили можливі проблеми та недоліки та внесли відповідні виправлення.

Загалом, тестування та валідація розробленої системи управління є важливим етапом роботи над проектом робота. Вони допомагають переконатися, що система працює правильно, відповідає вимогам та забезпечує потрібну ефективність та надійність.

ВИСНОВОК ДО ДИПЛОМНОЇ РОБОТИ

У рамках цієї дипломної роботи була розглянута тема "Адаптивна система автоматичного управління наземним рухомим об’єктом". Метою роботи було розробити адаптивну систему управління для роботизованої платформи, здатної адаптуватися до змінних умов роботи та ефективно керувати наземним рухомим об’єктом.

У Розділі 1 було проведено аналіз загальної інформації про наземні рухомі об’єкти, їх типи та особливості. Були детально розглянуті колісні роботи та роботизовані платформи, їх будова, особливості та області застосування. Цей розділ надав чітке уявлення про різні типи об’єктів, їх переваги та недоліки.

У Розділі 2 були розглянуті практичні приклади роботизованих платформ та їх програмного забезпечення. Були проаналізовані такі платформи, як Boston Dynamics Spot, Clearpath Husky, ANYmal, KUKA AGILUS, ABB YuMi та Fanuc M-2000iA. Для кожної платформи були надані технічні характеристики, опис та приклади коду. Цей розділ дозволив зрозуміти, як роботизовані платформи працюють у реальних умовах та яким чином вони можуть бути програмовані для різних завдань.

У Розділі 3 була розроблена адаптивна система управління для уявної роботизованої платформи. Був описаний процес проектування системи, включаючи алгоритми, використання штучного інтелекту та практичне реалізацію коду. Також було проведено тестування та валідацію системи для перевірки її правильності та ефективності.

Загальний результат цієї дипломної роботи полягає в розробці адаптивної системи управління для наземного рухомого об’єкта, що може пристосовуватися до змінних умов роботи та ефективно виконувати свої завдання. Ця робота вирішує важливу проблему в сфері робототехніки та відкриває шлях до подальшого розвитку та вдосконалення адаптивних систем управління наземними рухомими об’єктами.

Результати дослідження та розробки, представлені у цій дипломній роботі, можуть бути використані в різних сферах, включаючи промисловість, науку, медицину та багато інших. Адаптивні системи управління можуть покращити продуктивність, надійність та безпеку робототехнічних систем.

Кафедра АКСУ				**НАУ 23.15.39 ПЗ**			
Виконав.	*Жеребко В.В.*			**ВИСНОВОК ДО ДИПЛОМНОЇ РОБОТИ**	*Літ.*	*Арк.*	*Аркушів*
Керівник	*Мельник Ю.В*						
Консульт.	*Азарсков В.М.*				***151-301-СУ***		
Контрол.	*Мельник Ю.В*						
Зав. каф.	*Мельник Ю.В.*						

Усього, ця дипломна робота привнесла вагомий внесок у розуміння та розвиток адаптивних систем управління наземними рухомими об’єктами. Вона виявилася важливим кроком у напрямку створення ефективних та автономних робототехнічних систем, що здатні пристосовуватися до різних умов та виконувати різноманітні завдання.

Список використаних джерел

- Astrom, K.J., & Wittenmark, B. (2013). *Adaptive Control*. Dover Publications.
- Franklin, G. F., Powell, J. D., & Emami-Naeini, A. (2005). *Feedback Control of Dynamic Systems*. Prentice Hall.
- Янситис, В. В. (1972). *Теория автоматического управления в примерах и задачах*. Высшая школа.
- Boubertakh, H. (2013). *Adaptive control of a nonholonomic mobile robot*. Journal of Electrical Systems, 9(1).
- Karami, F., Boubertakh, H., & Djouani, K. (2016). *Backstepping control for autonomous ground vehicles*. International Journal of Systems Science, 47(6), 1376-1384.