Хмельницький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

КВАЛІФІКАЦІЙНА РОБОТА

магістр
Освітній рівень

Удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів
Назва теми

КвРАКІТР.2023173.01.02.ПЗ

Рівень вищої освіти магістр

Галузь знань 17 «Електроніка, автоматизація та електронні комунікації»
Шифр, назва

Спеціальність 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Шифр, назва

Освітня програма «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Назва

Виконав:

студент II курсу, група АКІТРм-23-1 _____ Михайло КОВАЛЬ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник _____ Андрій СЕЛЬСЬКИЙ
Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер _____ Людмила КОРЕЦЬКА
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
зав. кафедри АКІТтаР _____ Валерій МАРТИНЮК
Підпис Ім'я, ПРІЗВИЩЕ

«17» грудня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *автоматизації, комп'ютерно-інтегрованих технологій та робототехніки*
Рівень вищої освіти *другий (магістерський)*
Галузь знань *17 – Електроніка, автоматизація та електронні комунікації*
Спеціальність *174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка*
Освітня програма *Автоматизація, комп'ютерно-інтегровані технології та робототехніка*

ЗАТВЕРДЖУЮ
Завідувач кафедри АКІТтаР
Валерій МАРТИНЮК
01 вересня 2024р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ковалю Михайлу Володимировичу
Прізвище, ім'я, по батькові студента

1 Тема роботи Удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів
Керівник роботи Сельський Андрій Анатолійович, к.ф.-м.н., доцент
Прізвище, ім'я, по батькові, науковий ступінь, учене звання
Затверджено наказом ректора університету від 26.08.2024 р. №60
2 Строк подання студентом роботи на кафедру 02.12.2024р.
3 Вихідні дані до роботи
Мета роботи: розробка удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів
Предмет дослідження: удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів
4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. Огляд літературних джерел та патентних даних. Математична модель процесу керування роботом-маніпулятором на основі операційної системи роботів. Імітаційна модель удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів. Експериментальне дослідження удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів. Висновки.
5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
презентаційні матеріали (слайди)
6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Антиплагіат	Федула М.В., доцент кафедри АКІТтаР		
Нормоконтроль	Корецька Л.О., доцент кафедри АКІТтаР		

7 Дата видачі завдання 01 вересня 2024р.

КАЛЕНДАРНИЙ ПЛАН

Назва розділу кваліфікаційної роботи	Строк виконання	Примітка
1. Вступ	10.09.2024р.	Виконано
2. Огляд літературних джерел та патентних даних	25.09.2024р.	Виконано
3. Математична модель процесу керування роботом-маніпулятором на основі операційної системи роботів	15.10.2024р.	Виконано
4. Імітаційна модель удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів	30.10.2024р.	Виконано
5. Експериментальне дослідження удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів	10.11.2024р.	Виконано
6. Висновки	15.11.2024р.	Виконано
7. Оформлення пояснювальної записки	20.11.2024р.	Виконано
8. Оформлення презентаційних матеріалів	1.12.2024р.	Виконано

Студент _____ Михайло КОВАЛЬ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Андрій СЕЛЬСЬКИЙ
Підпис Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів».

Автор роботи: Коваль Михайло Володимирович.

Керівник роботи: Сельський Андрій Анатолійович.

Пояснювальна записка: 89 с., 68 рис., 8 табл., 1 дод., 80 джерел.

Графічна частина: 12 презентаційних слайдів.

УДОСКОНАЛЕНИЙ МЕТОД КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ, ОПЕРАЦІЙНА СИСТЕМА РОБОТІВ, РОБОЧИЙ ПРОСТІР.

Мета роботи: розробка удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів. Розроблено удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів, особливістю якого є використання симуляції для перевірки правильності роботи контролерів. В процесі симуляції використовувалося керування шарнірами, в яких були проаналізовані реакції кожного шарніра в положенні, швидкості і зусиллі, а також вплив кожного шарніра на інші.

Підпис студента

02.12.24

Дата

ЗМІСТ

ВСТУП 4
1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ ТА ПАТЕНТНИХ ДАНИХ 6
1.1 Особливості операційної системи роботів 6
1.2 Огляд роботів, які підтримує операційна система роботів 10
1.3 Особливості моделі робота в операційній системі роботів 15
1.4 Висновки до першого розділу 18
2 МАТЕМАТИЧНА МОДЕЛЬ ПРОЦЕСУ КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ НА ОСНОВІ ОПЕРАЦІЙНОЇ СИСТЕМИ РОБОТІВ 20
2.1 Математична модель робота-маніпулятора 20
2.2 Динамічна модель робота-маніпулятора 25
2.3 Висновки до другого розділу 38
3 ІМІТАЦІЙНА МОДЕЛЬ УДОСКОНАЛЕНОГО МЕТОДУ КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ НА ОСНОВІ ОПЕРАЦІЙНОЇ СИСТЕМИ РОБОТІВ 39
3.1 Особливості імітаційної моделі удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів 39
3.2 Декартове керування роботом-маніпулятором 46
3.3 Висновки до третього розділу 53
4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ УДОСКОНАЛЕНОГО МЕТОДУ КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ НА ОСНОВІ ОПЕРАЦІЙНОЇ СИСТЕМИ РОБОТІВ 55
4.1 Особливості експериментального дослідження удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів 55
4.2 Експериментальне дослідження керування роботом-маніпулятором в декартовому просторі 63
4.3 Висновки до четвертого розділу 74
ВИСНОВКИ 76
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 77

Додаток А Стаття у фаховому журналі (подана до редакції журналу «Вимірювальна та обчислювальна техніка в технологічних процесах»)..... 85

ВСТУП

Актуальність теми. Робототехніка - одна з галузей техніки, яка вивчає процес, що виконується для проектування та подальшого конструювання машини чи робота. Машини або роботи призначені для виконання повторюваних завдань з використанням маніпуляторів. Маніпулятори - це роботизовані руки, які виконують певні завдання, такі як зварювання, фарбування, компонування деталей.

Операційна система роботів є відкритою платформою, яка дозволяє використовувати модулі (пакети) для розв'язання широкого спектра завдань, пов'язаних із робототехнікою. Це особливо актуально для складних маніпуляторів, де потрібно інтегрувати сенсори, планувальники рухів та виконавчі системи.

Операційна система роботів має вбудовані засоби для симуляції, такі як Gazebo, які дозволяють тестувати алгоритми керування віртуально перед їх використанням у реальному обладнанні. Це знижує ризики і витрати, пов'язані з експериментами.

Метою роботи є розробка удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів.

Відповідно до поставленої мети необхідно вирішити **завдання:**

- виконати огляд літературних джерел та патентних даних про методи керування роботом-маніпулятором на основі операційної системи роботів;
- розробити математичну модель процесу керування роботом-маніпулятором на основі операційної системи роботів;
- розробити алгоритм та програмну реалізацію удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів;
- розробити імітаційну модель удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів.

Об’єктом дослідження є процес керування роботом-маніпулятором на основі операційної системи роботів.

Предметом дослідження є удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів.

Методи досліджень. При вирішенні поставлених завдань у роботі були використані методи фізики, теорії автоматичного керування, методи обчислювальної математики, а також методи алгоритмізації та програмування.

Наукова новизна отриманих результатів:

У результаті проведеного дослідження розроблено удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ ТА ПАТЕНТНИХ ДАНИХ

1.1 Особливості операційної системи роботів

Операційна система роботів (ОСР) - це фреймворк для написання програмного забезпечення для роботів [1]. До складу операційної системи роботів входить набір інструментів, бібліотек і угод, спрямованих на спрощення завдання створення поведінки для складних і надійних роботів на різноманітних роботизованих платформах.

До основних особливостей операційної системи роботів відносяться наступні характеристики [2].

1. Великий набір контролерів, які дозволяють зчитувати дані з датчиків і надсилати команди двигунам та іншим виконавчим механізмам в абстрактному і чітко визначеному форматі. Широке розмаїття популярного апаратного забезпечення, включаючи зростаючу кількість комерційно доступних робототехнічних систем.

2. Великий набір фундаментальних робототехнічних алгоритмів для побудови карт світів, навігації по ним, представлення та інтерпретації сенсорних даних, планування рухів, маніпулювання об'єктами та багато іншого.

3. Потужна обчислювальна інфраструктура, яка дозволяє переміщувати дані, з'єднувати різні компоненти складної роботизованої системи та впроваджувати власні алгоритми. ОСР за своєю суттю є розподіленою і дозволяє без проблем розподіляти робоче навантаження між кількома комп'ютерами.

4. Великий набір інструментів, які дозволяють легко візуалізувати стан робота та алгоритму, налагоджувати помилкову поведінку та записувати дані з датчиків.

5. Операційна система роботів містить у собі великий набір ресурсів, таких як документація з багатьох аспектів фреймворку, сайт питань і відповідей, де

можна звернутися за допомогою і поділитися своїми знаннями, а також спільноту користувачів і розробників.

Архітектура операційної системи роботів була розроблена і розділена на три розділи або рівні [3].

1. Рівень файлової системи.

2. Рівень комп'ютерної графіки.

3. Рівень спільноти.

На рівні файлової системи пояснюється, як ОСР формується зсередини через структуру папок, а також мінімум файлів, необхідних для роботи. На рівні комп'ютерної графіки, де відбувається комунікація між процесами та системами. На цьому рівні базуються концепції та системи, які ОСР має налаштувати для управління процесами та спілкування з більш ніж одним комп'ютером.

На рівні спільноти використовуються інструменти та концепції для обміну знаннями, алгоритмами та кодом від будь-якого розробника. Цей рівень є важливим, оскільки ОСР може швидко зростати за умови сильної підтримки спільноти.

Подібно до операційної системи, рівень файлової системи ОСР-програми поділено на папки, а ці папки містять пакунки, які описують наступні функції.

1. Стек - об’єднання пакунків з деякими функціями з різним призначенням, наприклад, стек навігації.

2. Маніфест стеку - надає дані про стек, включаючи інформацію про його ліцензування та залежності від інших стеків.

3. Пакунок – формує атомарний рівень ОСР. Пакунок має мінімальну структуру та вміст для створення програми в ОСР. Він може містити процеси (вузли) виконання ОСР та конфігураційні файли.

4. Маніфест - надає інформацію про пакунок, ліцензію, відомості, залежності, параметри компілятора тощо. Керуються за допомогою файлу з назвою "manifests.xml".

5. Повідомлення (msg) - це інформація, яку процес надсилає іншим процесам. ОСР має велику кількість стандартних типів повідомлень. Описи повідомлень зберігаються у файлі "my_package/msg/MyMessageType.msg".

6. Сервіс (srv) - зберігаються у файлі "my_package/srv/MyServiceType.srv", визначають структури даних запиту та відповіді для сервісів ОСР.

7. Код - введення інструкцій допускається мовою програмування. Код використовується для введення інформації в ОСР, як правило, використовується в середовищі Phyton.

8. Файли довідки - генеруються при компіляції та виконанні різних програм.

На рівні комп'ютерної графіки ОСР створює мережу, до якої підключені всі процеси. Будь-який вузол системи може отримати доступ до цієї мережі, взаємодіяти з іншими вузлами, переглядати інформацію, яку він надсилає, і передавати дані в мережу.

Основними поняттями на рівні комп'ютерної графіки є вузол, майстер, сервер параметрів, повідомлення, сервіс, тема та обмін.

1. Вузол - це процес, в якому виконуються обчислення. Якщо потрібно мати процес, який може взаємодіяти з іншими вузлами, то необхідно створити вузол з цим процесом, щоб підключити його до мережі ОСР. Зазвичай, система має багато вузлів для управління різними функціями. Краще мати багато вузлів, які забезпечують єдину функціональність, ніж один великий вузол, який робить все в системі. Вузли програмуються за допомогою клієнтської бібліотеки ОСР, наприклад: ros.cpp або ros.py.

2. Майстер - забезпечує реєстрацію імен та їх пошук для всіх інших вузлів. Якщо в системі немає майстра, то немає можливості спілкуватися з вузлами, сервісами, повідомленнями тощо.

3. Сервер параметрів - дає можливість зберігати дані за допомогою ключів у центральному місці. За допомогою цього параметра можна конфігурувати вузли під час роботи або змінювати роботу вузлів.

4. Повідомлення - за допомогою повідомлень вузли спілкуються один з одним. Повідомлення містить дані, які надсилають інформацію іншим вузлам.

ОСР має багато типів повідомлень, а також є можливість розробити власний тип повідомлення, використовуючи стандартні повідомлення.

5. Тема - кожне повідомлення повинно мати назву для того, щоб пройти через мережу ОСР. Коли вузол надсилає дані, вважається, що він публікує тему. Вузли можуть отримувати теми від інших вузлів, просто підписавшись на тему. Вузол може підписатися на тему, а вузол, який публікує цю тему, не обов'язково повинен існувати. Це дозволяє відокремити виробництво від споживання. Важливо, щоб назва теми була унікальною, щоб уникнути проблем і плутанини між темами з однаковими назвами.

6. Сервіси - коли публікується тема, то надсилаються дані за принципом "багато до багатьох", але коли потрібно отримати запит або відповідь від вузла, це не можливо зробити за допомогою тем. Сервіси дають можливість взаємодіяти з вузлами і повинні мати унікальне ім'я. Коли вузол має сервіс, всі вузли можуть взаємодіяти з ним завдяки клієнтським бібліотекам ОСР.

7. Мішок - це формат для зберігання та відтворення даних ОСР-повідомлень. Він є важливим механізмом для зберігання даних, таких як дані про датчики, які може бути важко зібрати, але вони необхідні для розробки та тестування алгоритмів.

Концепція ОСР на рівні спільноти - це ресурси ОСР, які дозволяють окремим спільнотам обмінюватися програмним забезпеченням та знаннями. Існують наступні ресурси.

1. Дистрибутиви ОСР - це набори версійних стеків, які ожна встановити. Дистрибутиви ОСР відіграють подібну роль до дистрибутивів Linux. Вони полегшують встановлення набору програмного забезпечення, а також підтримують узгодженість версій у наборі програмного забезпечення.

2. Репозиторії - ОСР базується на об'єднаній мережі репозиторіїв коду, де різні установи можуть розробляти та випускати власні програмні компоненти для роботів.

3. OCP Wiki - це основний форум для документування інформації про OCP. Будь-хто може зареєструвати обліковий запис і внести свій внесок, надавати виправлення чи оновлення, писати підручники та багато іншого.

4. Список розсилки користувачів OCP - це основний канал зв'язку щодо нових оновлень OCP, а також форумом, де можна поставити запитання про нове програмне забезпечення OCP.

1.2 Огляд роботів, які підтримує операційна система роботів

В даний час відомий широкий спектр роботів у різних категоріях, що використовують фреймворк OCP. Операційна система роботів підтримує повітряні роботи (Gapter), наземні роботи (Turtlebot і Pepper), роботи-маніпулятори (Fanuc і ABB), і морські роботи (Clearpath Heron USV).

Також операційна система роботів підтримує велику кількість компонентів, а саме маніпулятори, камери, 3D-сенсори та лазери, які входять до складу роботизованих систем.

В процесі розвитку робототехніки виникли певні труднощі з написанням сумісного програмного забезпечення для різних типів роботів. Якщо апаратне забезпечення значно відрізняється, повторне використання коду не є тривіальним.

Зіткнувшись з цією тенденцією, OCP надає бібліотеки та інструменти, які допомагають розробникам програмного забезпечення створювати робототехнічні програми. Основними завданнями OCP є абстрагування низькорівневого управління апаратними пристроями, реалізація часто використовуваних функцій, передача повідомлень між процесами та управління пакетами.

Операційна система роботів надає всі частини програмної системи робота, які в іншому випадку потрібно розробляти самостійно. Це дозволяє зосередитися на тих частинах системи, які мають важливе значення, не турбуючись про ті частини, які є відомими.

За останні десятиліття мобільна робототехніка досягла неймовірних успіхів. Такі теми, як автономна навігація, сприйняття, реактивність, картографування, уникнення перешкод і самолокалізація, були глибоко вивчені в останні роки і знайшли застосування в промисловості, транспорті, військовій сфері та сфері безпеки. Крім того, деякі мобільні роботи стали споживчими товарами для розваг або для виконання повсякденних побутових завдань, наприклад для прибирання підлоги.

Мобільний робот [4] - це автоматична машина, здатна переміщатися в будь-якому середовищі. Мобільні роботи мають можливість переміщатися в навколишньому середовищі і не прив'язані до певного місця.

Мобільні роботи можуть бути автономними, тобто здатними орієнтуватися в неконтрольованому середовищі і приймати власні рішення автоматично, без втручання людини. Для цього вони складаються з трьох інструментів: датчиків, процесорів і виконавчих механізмів.

Сприйняття передбачає використання датчиків - пристроїв, здатних фіксувати фізичні величини (відстань, температуру, звук, світло) і є джерелом інформації для робота. За допомогою цих приладів машина може визначити, де вона знаходиться, в яких умовах, і вони є основою процесора.

Процесор - це те, що дозволяє роботу приймати рішення, а актуатори дозволяють виконувати рішення, прийняті процесором. Завдяки актуаторам робот повертає в певний момент, якщо він занадто близько підходить до стіни.

Платформа робот-черепаха (англійською мовою Turtlebot) [5] - це недорогий мобільний робот з відкритим програмним забезпеченням, який зображений на рисунку 1.1.

Рисунок 1.1 - Платформа робот-черепаха [5]

Це базова платформа для роботи з алгоритмами OCP і просунутими алгоритмами одночасної локалізації та відображення (ОЛВ) - англійською мовою Simultaneous Localization and Mapping (SLAM), а також комп'ютерним зором, RGB-D камерами і хмарами точок.

Робот складається з мобільної бази Kobuki, яка містить систему диференціального руху з датчиками та електронним гіроскопом. Робот-черепаха, який використовується, також має деякі додаткові пристрої, зокрема 2D лазерний сканер Hokuyo і 3D камеру Astra, за допомогою яких він збирає інформацію з навколишнього простору. І все це інтегровано з комп'ютера-нетбука через фреймворк OCP.

Мобільна база Kobuki [6] - це робот, який зображений на рисунку 1.2 і схожий на Irobot Roomba від компанії Irobot.

Рисунок 1.2 - Мобільна база Kobuki [6]

Основною функцією мобільної бази Kobuki є прибирання та ведення домашнього господарства. Kobuki має інфрачервоні датчики для виявлення сходів, бампер, який допомагає йому прийняти удар і відреагувати на подію, гіроскоп для контролю одометрії, спеціальні колеса для подолання перешкод і пересування в будь-якому напрямку.

Лазер Hokuyo UST-10LX [7] - це невеликий, точний, високошвидкісний 2D лазерний сканер, який зображений на рисунку 1.3 і призначений для виявлення перешкод.

Рисунок 1.3 - Лазер Hokuyo UST-10LX [7]

Лазер Hokuyo UST-10LX ідеально підходить для застосування в робототехніці. Він може сканувати з апертурою до 270 градусів на відстані до 10 метрів. Hokuyo є більш точним, ніж датчик Kinect, і буде більш корисним для gmapping та SLAM додатків.

Astra [8] - це потужна і надійна 3D-камера, яка містить власний мікрочіп Orbbec з підтримкою 3D і VGA і зображена на рисунку 1.4.

Рисунок 1.4 - Камера Astra [8]

Камера Astra була розроблена для забезпечення високої сумісності з додатками, розробленими за допомогою OpenNI, що робить її ідеальною для вже існуючих додатків, які були розроблені за допомогою OpenNI. Дальність виявлення камери становить від 0,4 до 8 метрів.

Незважаючи на свою назву, OCP не є операційною системою як такою, фактично, вона працює поверх Linux. Насправді OCP надає стандартні послуги операційної системи, такі як апаратна абстракція, низькорівневе керування пристроями, реалізація часто використовуваної функціональності, міжпроцесорна передача повідомлень та керування пакетами, залишаючи обробку, керування пам'яттю, графічним інтерфейсом тощо на власний розсуд. Вона також надає інструменти та бібліотеки для отримання, побудови, написання та виконання коду на декількох комп'ютерах.

Операційна система роботів працює як однорангова мережа процесів (вузлів), так що кожен вузол може отримувати повідомлення про стан від інших. Ці вузли вільно пов'язані між собою за допомогою комунікаційної інфраструктури OCP.

Кожен вузол згрупований у пакети, які можна використовувати на інших OCP-системах. Крім того, вузол не залежить від мови, оскільки може бути реалізований на будь-якій сучасній мові програмування, будучи вже реалізованим на Python, C++ та Lisp. З іншого боку, вузол повністю сумісний з OpenCV, що дає можливість реалізовувати програми, розроблені в OCP.

Система керування роботом зазвичай складається з багатьох вузлів. Наприклад, один вузол керує лазером, інший - колісними двигунами, третій - локалізацією, четвертий - плануванням шляху, п'ятий - графічним відображенням системи тощо. Вузол - це клієнт OCP, написаний за допомогою бібліотеки C++ (roscpp) або Python (rospy).

Наприклад, для керування лазерним детектором відстані Hokuyo запускається драйвер hokuyo_node, який зв'язується з лазером і публікує повідомлення sensor_msgs/LaserScan у темі сканера (/scan). Для обробки цих даних використовується вузол з використанням laser_filters, який підписується на

повідомлення у темі /scan. Після підписки фільтр автоматично почне отримувати повідомлення від лазера.

Все, що робить hokuyo_node - це публікує свої знімки, не знаючи, чи хтось на них підписаний. Все, що робить фільтр - це підписується на /scan, не знаючи, чи хтось їх публікує. Обидві вузли можна запускати, вимикати і перезапускати у будь- якому порядку, не викликаючи при цьому помилок.

1.3 Особливості моделі робота в операційній системі роботів

Операційна система роботів містить різноманітні пакети з різними функціями та дві клієнтські бібліотеки: roscpp (для C++) та rospy (для phyton). У бібліотеці roscpp розміщені інтерфейси прикладного програмування (англійською мовою - Application Programming Interface (API)).

Інтерфейс прикладного програмування (ІПП) - це набір функцій, класів, процедур або протоколів, які розробники можуть використовувати для взаємодії з певною програмою, бібліотекою чи операційною системою.

Прикладом (ІПП) для C++ є robot_model - стек, який містить набір пакетів для кінематичного та динамічного моделювання роботів, а також набір інструментів для їх перевірки та конвертації в різні формати.

Центральний пакет стеку називається urdf, який має інтерпретатор, що дозволяє вказати модель робота у форматі XML. Цей формат опису робота називається уніфікованим форматом опису робота (УФОР) (англійською мовою Unified Robot Description Format (URDF)) і сумісний як із програмою перегляду Rviz, так і з симулятором Gazebo.

Назва програми перегляду Rviz [9] утворена від двох слів робот візуалізація (англійською мовою Robot Vizualization). Rviz - це інструмент 3D візуалізації для ОСР. Rviz забезпечує перегляд змодельованої моделі робота, дозволяє збирати інформацію з датчиків робота та відтворювати отримані дані з датчиків.

Візуалізуючи те, що бачить, думає і робить робот, можна налагодити робототехнічну програму, починаючи з входів датчиків і закінчуючи запланованими (або незапланованими) діями.

Rviz відображає дані 3D-сенсорів зі стереокамер, лазерів, Kinect та інших 3D-пристроїв у вигляді хмар точок або зображень глибини. Дані 2D-сенсорів з веб-камер, RGB-камер і 2D-лазерних далекомірів можна переглядати в Rviz як дані зображень.

Симулятор Gazebo (українською мовою - альтанка) [10] - це 3D, кінематичний, динамічний і мультиробот-симулятор, який дозволяє моделювати шарнірних роботів у складних, внутрішніх або зовнішніх, реалістичних і тривимірних середовищах з високим ступенем точності: прискорення, гравітаційні сили, інерція, маси тощо. На додаток до цього, Gazebo також може імітувати велику кількість датчиків.

Цей симулятор ідеально підходить для симуляції роботів, оскільки він має зв'язок з ОСР. Gazebo реалізовано в ОСР за допомогою пакетів, які називаються gazebo_ros_pkgs. Ці пакети містять плагіни, які взаємодіють з будь-яким об'єктом у сцені симулятора та надають необхідний інтерфейс для використання ОСР у поєднанні з Gazebo. Gazebo - це 3D-симулятор, а ОСР слугує інтерфейсом до робота. Поєднання обох програм утворює потужний симулятор робота.

Gazebo розглядається як вузол для ОСР, що дозволяє запускати його за допомогою launch-файлу. Операційна система роботів дозволяє запускати декілька вузлів одночасно, і одним з них може бути Gazebo.

Це значно полегшує комунікацію між ними, тому що Gazebo є вузлом для ОСР і може публікувати і підписуватися на будь-яку тему в системі. Наприклад, швидкість робота, яка публікується в темі ОСР, завдяки цьому може бути застосована до віртуальних двигунів робота в Gazebo.

За допомогою цієї платформи Gazebo можна досягти точної симуляції поведінки робота, так що його поведінка у віртуальному світі Gazebo буде схожою на поведінку в реальному світі.

Формат опису робота URDF складається з серії тегів XML, які дозволяють вказати серію параметрів для кожного посилання та кожного з’єднання робота. Хоча режим опису є практично загальним для всіх типів роботів, від БПЛА до маніпулятора, існують певні обмеження, які унеможливлюють опис певних роботів.

Основне обмеження полягає в тому, що роботи повинні бути послідовними роботами (деревоподібна структура). В URDF неможливо описати паралельні роботи або іншими словами замкнуті кінематичні ланки.

Крім того, передбачається, що робот складається з жорстких ланок, з’єднаних шарнірами, тобто гнучкі ланки неможливі. Незважаючи на це, формат URDF є досить цікавим, тому що за допомогою URDF можна створити три головні моделі робота.

1. Кінематичний та динамічний опис робота (ланки та суглоби).

2. Візуальне представлення робота (кожної ланки).

3. Модель зіткнення для кожної ланки робота.

У форматі URDF робот розглядається як набір ланок або тіл, об’єднаних набором з’єднань (суглобів), у відповідності до рисунку 1.5.

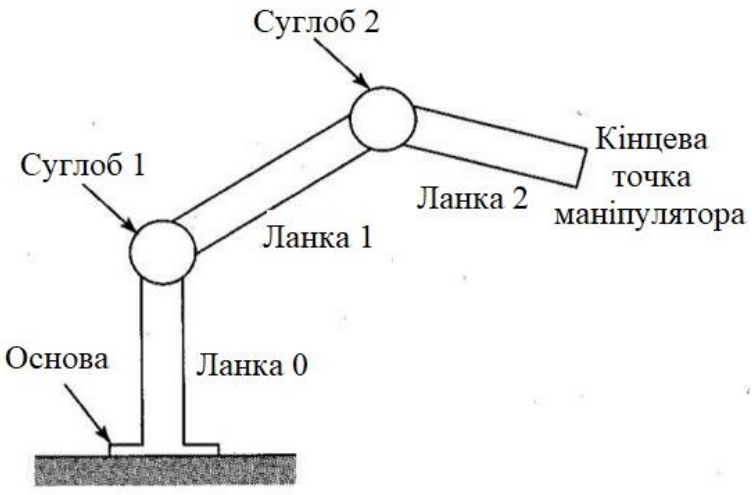


Рисунок 1.5 – Складові робота-маніпулятора

Кожний шарнірний або ланковий елемент описує тверде тіло з точки зору його інерції, його візуальної форми та форми зіткнення, тому він буде складатися з трьох блоків, незалежних один від одного.

Кожний блок має свою систему відліку відносно системи відліку попереднього з'єднання, крім випадку нульової ланки, системи відліку якої буде відносно системи відліку основи.

Якщо програма виконується лише за допомогою URDF, виникає кілька обмежень, наприклад неможливість імітувати робота закритим способом, відсутність міток визначення датчика або розширення чи модифікація робота.

Модифікація має великий сенс, оскільки якщо подовжити руку робота або додамо інший елемент, це може бути небезпечно, оскільки зроблені рухи та обчислення зміняться, а рухи, які були раніше запрограмовані, можуть спричинити зіткнення. Залишається симуляція з нереалістичними елементами.

Усі ці обмеження усуваються за допомогою власної системи ROS, що складається з фреймворків під назвою XACRO, яка виконує всі наступні функції в програмі для спрощення та покращення коду в URDF.

1.4 Висновки до першого розділу

1. Операційна система роботів - це фреймворк для написання програмного забезпечення для роботів. До складу операційної системи роботів входить набір інструментів, бібліотек і угод, спрямованих на спрощення завдання створення поведінки для складних і надійних роботів на різноманітних роботизованих платформах.

2. Операційна система роботів надає бібліотеки та інструменти, які допомагають розробникам програмного забезпечення створювати робототехнічні програми. Основними завданнями операційної системи роботів є абстрагування низькорівневого управління апаратними пристроями, реалізація часто

використовуваних функцій, передача повідомлень між процесами та управління пакетами.

3. Операційна система роботів працює як однорангова мережа процесів (вузлів), так що кожен вузол може отримувати повідомлення про стан від інших. Ці вузли вільно пов'язані між собою за допомогою комунікаційної інфраструктури операційної системи роботів.

4. Формат опису робота URDF складається з серії тегів XML, які дозволяють вказати серію параметрів для кожного посилання та кожного з’єднання робота. У форматі URDF робот розглядається як набір ланок або тіл, об’єднаних набором з’єднань (суглобів).

5. Кожний шарнірний або ланковий елемент описує тверде тіло з точки зору його інерції, його візуальної форми та форми зіткнення, тому він буде складатися з трьох блоків, незалежних один від одного.

2 МАТЕМАТИЧНА МОДЕЛЬ ПРОЦЕСУ КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ НА ОСНОВІ ОПЕРАЦІЙНОЇ СИСТЕМИ РОБОТІВ

2.1 Математична модель робота-маніпулятора

Механічно робот-маніпулятор - це кінематичне коло, яке складається з ланок, з'єднаних шарнірами з одним або декількома ступенями свободи. У більшості роботів-маніпуляторів це кінематичне коло є розімкненим, починаючи з першої ланки або нерухомого базового корпусу і закінчуючи іншою, яка називається маніпулятором або маніпуляторним кінцем.

На рисунку 2.1 зображена кінематична схема робота-маніпулятора.

Рисунок 2.1 – Кінематична схема робота-маніпулятора

Ступінь свободи з'єднання (ССЗ) - це кількість незалежних відносних рухів, які ланка може здійснювати відносно попередньої ланки, з якою її з'єднує з'єднання. Кількість ступенів свободи робота-маніпулятора - це сума ступенів свободи всіх з'єднань, за винятком кінцевого ефектора, що, в свою чергу, є кількістю незалежних параметрів, необхідних для визначення просторового положення кінцівки робота.

У випадку робота-маніпулятора суглоби мають тільки один ступінь свободи і всі вони є поворотними, як показано на рисунку 2.2. З цією механічною структурою пов'язані актуатори, тобто елементи, які створюють необхідні зусилля для переміщення суглобів, які у випадку робота-маніпулятора будуть електродвигунами.

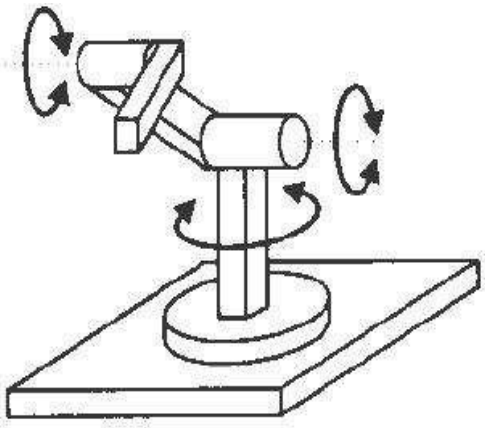


Рисунок 2.2 – Робота-маніпулятор суглобового типу

Залежно від їх розташування розрізняють два типи конфігурацій:

1. Конфігурація з розподіленими двигунами, в якій двигуни працюють безпосередньо на шарнірах, що спрощує механізм, але зменшує вагу.

2. Конфігурація з зосередженими двигунами, в якій двигуни розташовані по всій основі маніпулятора, центруючи і зменшуючи навантаження та знижуючи моменти інерції. Однак необхідно додати систему ременів для передачі моментів на відповідні шарніри, що робить цей механізм більш складним.

На рисунку 2.3 зображена конфігурація двигунів робота-маніпулятора.

а) б)

Рисунок 2.3 – Конфігурація двигунів робота-маніпулятора:

а) конфігурація з розподіленими двигунами;

б) конфігурація з зосередженими двигунами

Насправді в роботах-маніпуляторах зазвичай використовують не прості електродвигуни, а серводвигуни. Серводвигуни - це теж електродвигуни, але з можливістю керування, тобто підтримання потрібного кута і з певним датчиком для зчитування поточного кута, наприклад, енкодером.

Ще одним елементом базової конструкції робота-маніпулятора є трансмісія, яка відповідає за передачу руху від від приводів до шарнірів. Трансмісія - це набір шестерень і ременів, які передають круговий рух від позиції, де знаходиться привід, до позиції шарніра.

Існує ще один можливий елемент, пов'язаний з трансмісією - редуктор. Завдання редуктора - збільшити вихідний крутний момент за рахунок зменшення швидкості обертання. Виконаємо моделювання робота-маніпулятора без редукторів, тобто з передавальним числом, що дорівнює одиниці. Такий тип роботів-маніпуляторів називається прямим приводом і має переваги швидшого і точнішого позиціонування, зменшення тертя і кращої керованості.

Що стосується системи керування, то оскільки це робот-маніпулятор, змодельований у Gazebo, то паралельно працюватиме інша програма, яка розраховуватиме та надсилатиме необхідні моменти на двигуни, щоб розмістити ефектор у потрібному положенні.

Робочий простір або об'єм робота-маніпулятора визначається по відношенню до основи робота-маніпулятора як сукупність точок, доступних від ефектора робота, як правило, за винятком захвату.

Модифікуємо і розширено робочий простір або об'єм робота-маніпулятора з урахуванням того, що ми маємо справу з роботом-маніпулятором, маніпулятор якого є інструментом для захоплення і відпускання об'єктів і який з'єднаний з іншим роботом, з яким він не може зіткнутися.

Робочий простір повинен бути визначений для центральної точки інструменту. Це точка на роботі-маніпуляторі, яку потрібно розташувати в точці, де буде знаходитися об'єкт, який потрібно взяти, або в точці, де його потрібно відпустити.

Будуть позиції або зони, які, хоча і можливі, повинні вважатися недоступними для робота-маніпулятора з міркувань безпеки (можливі зіткнення з об'єктом або іншим робот-маніпулятор).

На рисунку 2.4 зображена робоча область робота-маніпулятора.

Рисунок 2.4 – Робоча область робота-маніпулятора

Розглянемо взаємозв'язок між розташуванням вигину руки робота-маніпулятора відносно системи відліку, розташованої на нерухомій основі, і кутами, які приймають суглоби. Існує дві еквівалентні задачі або способи підходу до кінематичної моделі робота-маніпулятора, але з різним застосуванням.

Першою і найпростішою є пряма кінематична модель (ПКМ), в якій розташування ефектора (положення і орієнтація) відносно нерухомої системи відліку бази маніпулятора визначається як функція кутів або значень шарнірів.

Система рівнянь (2.1) описує пряму кінематичну модель в загальному вигляді, де орієнтація представлена, кутами Ейлера (крен, тангаж, рискання), але еквівалентно може бути представлена матрицею обертання R.

$$\begin{cases} X = f_1(q_1, \dots, q_n) \\ Y = f_2(q_1, \dots, q_n) \\ Z = f_3(q_1, \dots, q_n) \\ r = f_4(q_1, \dots, q_n) \\ p = f_5(q_1, \dots, q_n) \\ y = f_6(q_1, \dots, q_n) \end{cases} \quad (2.1)$$

На рисунку 2.5 зображена пряма кінематична модель робота-маніпулятора.

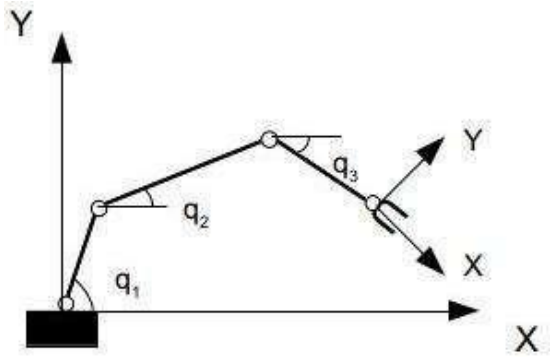


Рисунок 2.5 – Пряма кінематична модель робота-маніпулятора

Пряма кінематична модель робота-маніпулятора дозволяє непрямим способом визначити точне місця розташування ефектора за координатами суглоба, які зазвичай відомі в будь-який час за допомогою датчика типу енкодера, пов'язаного з кожним суглобом.

Зазвичай пряму кінематичну модель отримують за допомогою процедури Денавіта-Хартенберга (Д-Х), яка базується на виконанні серії базових змін з використанням матриць однорідного перетворення (МОП) від базової системи координат маніпулятора до кінцевої системи координат, що проходить через усі шарніри, окрім самого ефектора [10].

З іншого боку, існує також обернена кінематична модель (ОКМ), яка забезпечує зворотний зв'язок з прямою кінематичною моделлю, а координати суглоба залежать від місця розташування ефектора у відповідності до системи рівнянь (2.2).

$$\begin{cases} q_1 = g_1(X, Y, Z, r, p, y) \\ \dots \\ q_n = g_n(X, Y, Z, r, p, y) \end{cases}. \quad (2.2)$$

Обернена кінематична модель є більш складною, ніж пряма кінематична модель, тому що її важче отримати аналітично, вона може бути нелінійною і мати

декілька розв'язків. З цієї причини для розв'язання ОКМ зазвичай застосовують ітераційний метод, який дозволяє отримати єдиний розв'язок простим способом, без необхідності отримувати аналітичний розв'язок оберненої кінематичної моделі.

Корисність оберненої кінематичної моделі полягає в тому, що вона дозволяє керувати положенням ефектора. Система декартового позиційного керування забезпечується бажаним розташуванням ефектора, але кожному окремому шарнірному регулятору, як буде видно з прийнятої схеми керування, повинен бути забезпечений бажаний кут в його шарнірній координаті. Використання оберненої кінематичної моделі необхідне для кожної з координат суглоба.

2.2 Динамічна модель робота-маніпулятора

Динамічна модель робота-маніпулятора - це взаємозв'язок між положенням робота-маніпулятора, заданим координатами суглобів або розташуванням ефектора, і силами, які прикладеними до суглобів (τ):

$$\tau(t) = f(q(t)). \tag{2.3}$$

Для однієї пари ланка-шарнір виразу (2.3) буде описуватися виразом (2.4) [11]:

$$\tau = I\,\ddot{\theta} + M\,g\,L\,cos\theta, \tag{2.4}$$

де M - загальна маса ланки, що знаходиться в центрі мас;

L - відстань від центра мас ланки до початку з'єднання;

I - інерційність ланки.

На рисунку 2.6 зображена модель динамічного зв'язку ланки робота-маніпулятора.

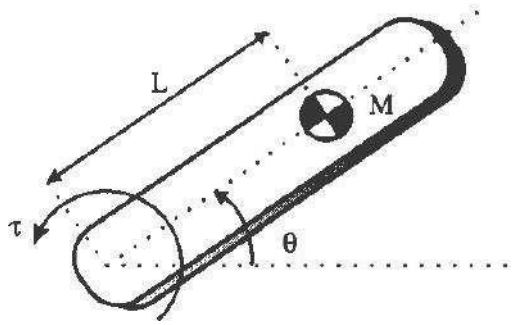


Рисунок 2.6 – Модель динамічного зв'язку ланки робота-маніпулятора

Аналізуючи вираз (2.4) можна зробити висновок, що розробити динамічну модель маніпулятора можна на основі геометрії робота-маніпулятора та ряду параметрів: загальної маси, матриці інерції та розташування центрів мас кожної ланки.

Для створення динамічної моделі робота-маніпулятора, за допомогою якої буде проводитися моделювання в Gazebo, достатньо розрахувати цей ряд параметрів для кожної ланки, для якої буде використовуватися програмне забезпечення Solid Edge.

Далі необхідно створити URDF-модель, яка копіює геометрію і конфігурацію маніпулятора (ланки і шарніри, еквівалентні ланці і шарніру) і в якій цей ряд динамічних параметрів задається для кожного тіла або ланки.

Початкова модель робота-маніпулятора зображена на рисунку 2.7.

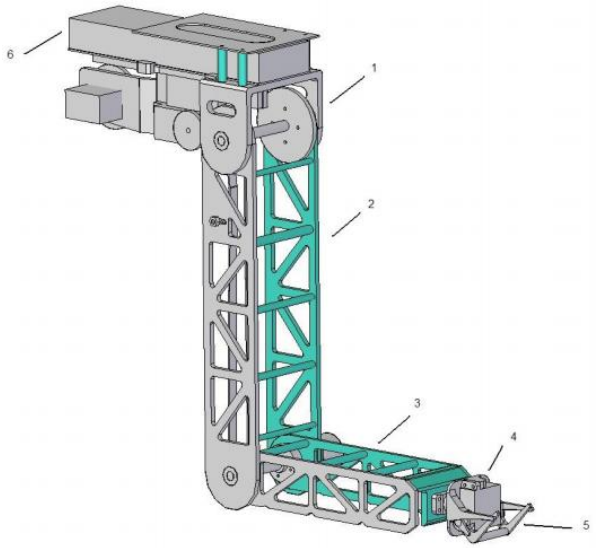


Рисунок 2.7 – Початкова модель робота-маніпулятора

Початкова модель робота-маніпулятора складається з наступних налок.

1. Основа.

2. Ланка 1.

3. Ланка 2.

4. Шасі ефектора.

5. Гачок ефектора.

6. Батарея.

Основа, ланка 1, ланка 2 та шасі ефектора виготовлені з жорсткого поліуретану або поліуретану високої щільності (густиною 1200 кг/м^3), жорсткого, але легкого матеріалу, який використовується в автомобілях і яхтах.

Основа, ланка 1, ланка 2 та шасі ефектора з'єднані трьома обертовими шарнірами, що робить його роботом-маніпулятором шарнірного типу.

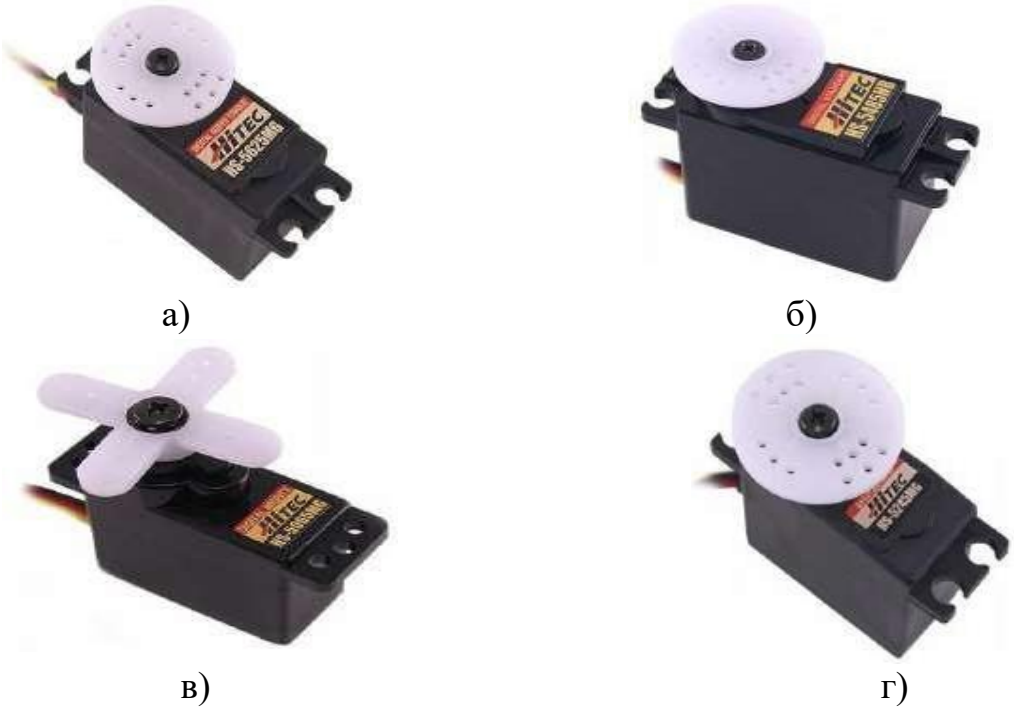
Вісь обертання шарніра 1 паралельна осі X, вісь обертання шарніра 2 також паралельна осі X, а вісь обертання шарніра 3 паралельна осі Y системи координат, пов'язаної з ефектором.

Крім того, є гачок-ефектора 5, який з'єднаний з корпусом за допомогою шарніра, що обертається, для відкривання та закривання корпусу. Захоплення відбувається за рахунок рухомого гака, який притискає об'єкт до нерухомого шасі.

Враховуючи цю конфігурацію, робот-маніпулятор має три ступені свободи, по одному для кожного шарніра, які дозволяють розміщувати ефектор в потрібному положенні (Y, Z), завжди в межах робочого простору, і з певною орієнтацією за висотою ефектора.

Робот-маніпулятор має конфігурацію двигунів, які розміщені в його основі, а це зменшує вагу та інерційність ланок, але для цього необхідно додати систему ременів і шестерень. Єдині двигуни, які знаходяться безпосередньо на суглобі - це третій двигун, що встановлений у ланці 2, а також двигун самого ефектора.

Двигун самого ефектора майже у всіх роботів-маніпуляторів розташований безпосередньо на ефекторі. Оберемо моделі та запишемо технічні характеристик серводвигунів, які будуть використані для моделювання. В якості серводвигунів оберемо серводвигуни компанії Hitec [8], які зображені на рисунку 2.8.



а) б)

в) г)

Рисунок 2.8 – Серводвигуни компанії Hitec для робота-маніпулятора:
а) HS-5625MG; б) HS-5485HB; в) HS-5065M; г) HS-5245MG

Для першого шарніра оберемо серводвигун моделі HS-5625MG, який зображено на рисунку 2.8, а) із наступними технічними характеристиками:

- вага: 60 г;
- кутовий діапазон: 180°;
- максимальна швидкість: 0,15с/60 (6,98 рад/с);
- максимальний крутний момент: 9,4 кг см (0,94 Н м).

Для другого шарніра оберемо серводвигун моделі HS-5485HB, який зображено на рисунку 2.8, б) із наступними технічними характеристиками:

- вага: 45 г;
- кутовий діапазон: 180°;
- максимальна швидкість: 0,17с/60 (6,16 рад/с);
- максимальний крутний момент: 6,4 кг см (0,64 Н м).

Для третього шарніра оберемо серводвигун моделі HS-5065M, який зображено на рисунку 2.8, в) із наступними технічними характеристиками:

- вага: 11,9 г;

- кутовий діапазон: 120°;

- максимальна швидкість: 0,11с/60 (9,52 рад/с);

- максимальний крутний момент: 2,2 кг см (0,22 Н м).

Для зміщення лотка акумулятора оберемо серводвигун моделі HS-5245MG, який зображено на рисунку 2.8, г) із наступними технічними характеристиками:

- вага: 32 г;

- кутовий діапазон: 180°;

- максимальна швидкість: 0,12с/60 (6,16 рад/с);

- максимальний крутний момент: 5,5 кг см (0,55 Н м).

Для шарніра ефектора використовується той самий серводвигун, що і для шарніра 3, але з меншим діапазоном повороту (90°). Останнім основним елементом маніпулятора є акумуляторна батарея, яка забезпечує енергією двигуни.

Акумуляторна батарея встановлена на лотку в корпусі основи, щоб можна було зрушити його назовні для можливої компенсації навантаження. Останній не враховуватиметься під час кервання, хоча буде змодельований, щоб залишити відкритою можливість включення його в подальшу роботу.

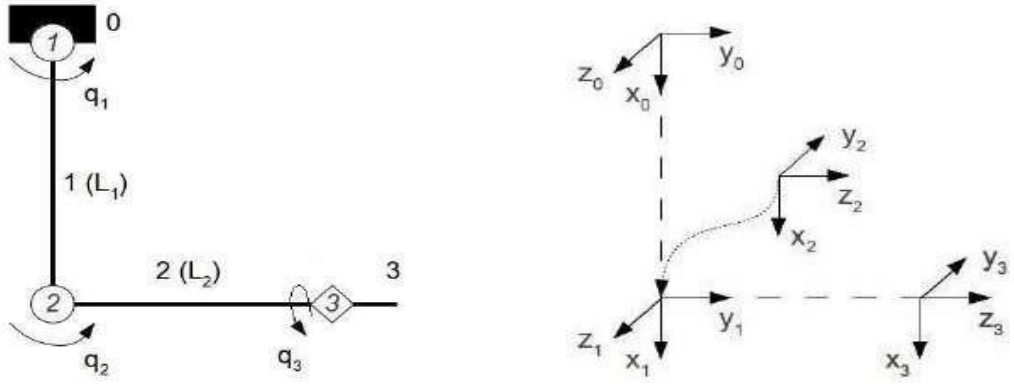
В таблиці 2.1 вказані граничні кути кожного з’єднання, виміряні відносно початкового положення рівноваги, відповідно до фізичних обмежень робота-маніпулятора і кутових діапазонів двигунів.

Таблиця 2.1 - Граничні кути з'єднання робота-маніпулятора

Шарніри	Нижня межа [рад]	Верхня межа [рад]
1	-1.08	1.57
2	-2.89	0.25
3	-1.047	1.047
Ефектор	-1	0.6

Для знаходження прямої кінематичної моделі використовується метод Д-Х. Згідно з процедурою, встановлюються системи координат, пов'язані з кожною

ланкою, відповідно до специфікацій методу, і розраховується ряд параметрів (θ, d, a, α), пов'язаних з перетворенням кожної з систем відносно попередньої, за допомогою яких знаходять МОП, віднесені до цих систем. На рисунку 2.9 зображено схему робота-маніпулятора та його систему координат Д-Х.



а) б)

Рисунок 2.9 – Схема робота-маніпулятора та його систему координат Д-Х:

а) схема робота-маніпулятора;

б) система координат Д-Х

Параметри Д-Х для робота-маніпулятора наведені в таблиці 2.2.

Таблиця 2.2 - Параметри Д-Х для робота-маніпулятора

Шарнір	θ	d	a	α
1	q_1	0	L_1	0
2	q_2	0	0	-90
3	q_3	L_2	0	0

Для отримання матриць однорідних перетворень (англійською мовою - omogeneous transformation matrices) використовується програма direct_kinematic_model.m, написана на мові Matlab і зображена на рисунку 2.10.

У цій програмі використовується функція HTM.m (англійською мовою- Homogeneous Transformation Tatrices), яка повертає МОП Д-Х для певних значень параметрів і зображена на рисунку 2.11.

```
direct_kinematic_model.m | HTM.m
1  clc
2  clear
3  syms q1 q2 q3 L1 L2 real;
4  pi1=sym(pi);
5  disp('Homogeneous transformation matrices:')
6  A01=HTM(q1, 0, L1,0);
7  A12=HTM(q2, 0, 0, -pi1/2);
8  A23=HTM(q3, L2,0, 0);
9
10 disp('Base Matrix to Effector')
11
12 A03=simplify(A01 * A12 * A23)
```

Рисунок 2.10 – Програма direct_kinematic_model.m, написана на мові Matlab

```
direct_kinematic_model.m | HTM.m
1  function A=HTM(teta,d,a,alfa)
2  A=[cos(teta) -cos(alfa)*sin(teta) sin(alfa)*sin(teta) a*cos(teta);
3      sin(teta) cos(alfa)*cos(teta) -sin(alfa)*cos(teta) a*sin(teta);
4      0              sin(alfa)             cos(alfa)       d        ;
5      0                  0                     0           1        ];
6  end
```

Рисунок 2.11 – Функція HTM.m

В результаті виконання програми direct_kinematic_model.m отримуємо базову матрицю перетворення A03 від основи до ефектора, яка зображена на рисунку 2.12.

```
Command Window
Homogeneous transformation matrices:
Base Matrix to Effector

A03 =

[cos(q1 + q2)*cos(q3), -cos(q1 + q2)*sin(q3), -sin(q1 + q2), L1*cos(q1) - L2*sin(q1 + q2)]
[sin(q1 + q2)*cos(q3), -sin(q1 + q2)*sin(q3),  cos(q1 + q2), L2*cos(q1 + q2) + L1*sin(q1)]
[           -sin(q3),              -cos(q3),             0,                            0]
[                  0,                     0,             0,                            1]

fx >>
```

Рисунок 2.12 – Базова матриця перетворення A03 від основи до ефектора

З компонентів базової матриці перетворення A03 від основи до ефектора отримують кінематичну модель робота-маніпулятора у вигляді системи рівнянь, що описуютькоорданати ефектора (2.5).

$$\begin{cases} x = L_1 cosq_1 - L_2 \sin(q_1 + q_2) \\ y = L_1 sinq_1 + L_2 \cos(q_1 + q_2) \\ z = 0 \end{cases}. \quad (2.5)$$

Також отримуємо матрицю орієнтації, яка описується виразом (2.6).

$$R = \begin{pmatrix} \frac{\cos(q_1+q_2+q_3)+cos(q_1+q_2+q_3}{2}) & \frac{-sin(q_1+q_2+q_3)+sin(q_1+q_2+q_3)}{2} & -sin(q_1+q_2) \\ \frac{sin(q_1+q_2+q_3)+sin(q_1+q_2+q_3)}{2} & \frac{-cos(q_1+q_2+q_3)+cos(q_1+q_2+q_3)}{2} & \cos(q_1+q_2) \\ -\sin(q_3) & \cos(q_3) & 0 \end{pmatrix}. \quad (2.6)$$

Для розрахунку фізичних властивостей маніпулятора його необхідно розділити на вузли, які належать до однієї ланки або корпусу. Ланки повинні відповідати ланкам в URDF-описі моделі робота-маніпулятора. В результаті отримаємо наступні ланки або вузли разом з деталями, що входять до кожної з них.

1. Збірка основи - сама конструкція основи, верхня опора батареї разом з розпірками, зосереджені двигуни шарнірів 1 і 2 та двигун батареї.

2. Батарея.

3. Кронштейн 1 - сама конструкція кронштейна, вали шарнірів 1 і 2, система ременів і шестерень для другого шарніра, потенціометр і кронштейн потенціометра.

4. Кронштейн 2 - сама конструкція та розподілений двигун суглоба 3.

5. Шасі ефектора - сама конструкція і розподілений двигун ефектора.

6. Гачок ефектора.

Батарея може бути додана до вузла основи, але вона спроектована окремо для можливого перепроектування, в якому вона буде мобільною або буде використовуватися інший тип батареї.

Далі кожна з наведених вище підмножин буде показана разом з їхніми параметрами, які використовуються в моделі у форматі URDF разом із повною масою, розташуванням центра мас та матрицею інерції.

1. Збірка основи:

- маса = 0,215 кг;

- центр маси відносно візуальної системи відліку заданий матрицею (2.7):

$$(X, Y, Z) = (3{,}94, -23{,}54, -40{,}54)\ мм; \tag{2.7}$$

- моменти інерції відносно системи відліку заданий матрицею (2.8):

$$L_{\text{ЦМ}} = \begin{pmatrix} 3618{,}30 & -549{,}48 & -122{,}19 \\ -- & 1151{,}36 & 470{,}42 \\ -- & -- & 3685{,}77 \end{pmatrix} g\ cm^2. \tag{2.8}$$

Цей вузол включає в себе деякі деталі, які не описані на кресленні маніпулятора: чотири розпірки масою 0,001 кг кожна, верхній кронштейн масою 0,012 кг, шестерні сервоприводів шарнірів 1 і 2, обидві масою 0,003 кг, і шестерня акумулятора масою 0,009 кг. Всі вони виготовлені з алюмінію марки 1060.

2. Батарея:

- маса = 0,232 кг;

- центр маси відносно візуальної системи відліку заданий матрицею (2.9):

$$(X, Y, Z) = (0, 0, 0)\ мм; \tag{2.9}$$

- моменти інерції відносно системи відліку заданий матрицею (2.10):

$$L_{\text{ЦМ}} = \begin{pmatrix} 4416{,}44 & 0{,}0 & 0{,}0 \\ -- & 420{,}48 & 0{,}0 \\ -- & -- & 4711{,}53 \end{pmatrix} g\ cm^2. \tag{2.10}$$

3. Кронштейн 1:

- маса = 0,102 кг;

- центр маси відносно візуальної системи відліку заданий матрицею (2.11):

$$(X, Y, Z) = (-5{,}65,\ 0{,}39,\ -102{,}06)\ мм; \tag{2.11}$$

- моменти інерції відносно системи відліку заданий матрицею (2.12):

$$L_{\text{ЦМ}} = \begin{pmatrix} 7595{,}15 & -2{,}24 & 312{,}68 \\ -- & 8157{,}22 & -7{,}95 \\ -- & -- & 776{,}25 \end{pmatrix} g\ см^2. \tag{2.12}$$

Цей вузол також включає деякі деталі, які відображуються в площині маніпулятора. До них відносяться вали шарнірів 1 і 2, обидва виготовлені з алюмінію і мають маси 5 г. і 4 г. відповідно.

Решта деталей є частиною системи передачі від шарніра 2 до шарніра 1, всі вони виготовлені з алюмінію, і їхня маса враховується при розрахунку вищезгаданих властивостей. Однак вони не відображені на загальному плані маніпулятора, оскільки механіка цієї передачі не є об'єктом моделювання.

4. Кронштейн 2:

- маса = 0,036 кг;

- центр маси відносно візуальної системи відліку заданий матрицею (2.13):

$$(X, Y, Z) = (-1{,}88,\ 94{,}38,\ 1{,}44)\ мм; \tag{2.13}$$

- моменти інерції відносно системи відліку заданий матрицею (2.14):

$$L_{\text{ЦМ}} = \begin{pmatrix} 1041{,}69 & -31{,}59 & 1{,}18 \\ -- & 143{,}04 & 0{,}43 \\ -- & -- & 1126{,}69 \end{pmatrix} g\ см^2. \tag{2.14}$$

5. Шасі ефектора:

- маса = 0,017 кг;

- центр маси відносно візуальної системи відліку заданий матрицею (2.15):

$$(X, Y, Z) = (-4{,}53, 8{,}19, -0{,}90)\ мм; \quad (2.15)$$

- моменти інерції відносно системи відліку заданий матрицею (2.16):

$$L_{\text{ЦМ}} = \begin{pmatrix} 14{,}33 & -0{,}36 & -0{,}59 \\ -- & 21{,}99 & -0{,}15 \\ -- & -- & 15{,}21 \end{pmatrix} g\ cm^2. \quad (2.16)$$

6. Гачок ефектора:

- маса = 0,002 кг;

- центр маси відносно візуальної системи відліку заданий матрицею (2.17):

$$(X, Y, Z) = 0{,}81, 8{,}63, -3{,}79)\ мм; \quad (2.17)$$

- моменти інерції відносно системи відліку заданий матрицею (2.18):

$$L_{\text{ЦМ}} = \begin{pmatrix} 1{,}43 & 0{,}01 & -0{,}01 \\ -- & 3{,}11 & -0{,}01 \\ -- & -- & 3{,}72 \end{pmatrix} g\ cm^2. \quad (2.18)$$

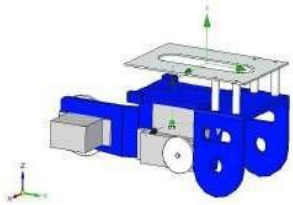
Модель робота-маніпулятора у форматі URDF містить повний кінематичний і динамічний опис робота, а також візуальні сітки і сітки зіткнень, так що при створенні моделі робота-маніпулятора в цьому форматі кінематична і динамічна моделі маніпулятора стають доступними безпосередньо.

Перед тим, як перейти до розгляду моделі URDF, слід зробити кілька зауважень.

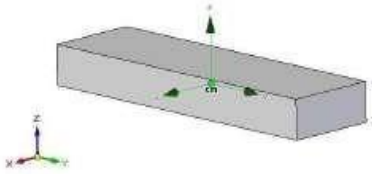
1. Центральна точка інструменту (ЦТІ) визначається як "фантомна" ланка, тобто вона не матиме ні сітки зіткнень, ні маси. Ця ланка буде визначена в центрі відкритого захвату і буде точкою маніпулятора, яку контролер повинен позиціонувати в (X, Y, Z) в декартовому просторі.

2. Фантомна ланка визначена відносно руки 2, а не відносно корпусу ефектора, оскільки поворот шарніра 3 впливатиме на позиціонування ефектора і постійно порушуватиме керування в декартовому просторі. Ці два моменти будуть більш детально розглянуті в розділі, присвяченому управлінню маніпулятором в декартовому просторі.

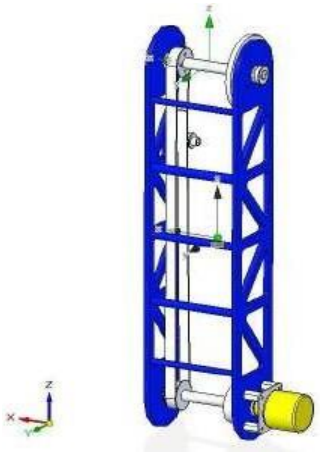
На рисунку 2.13 показані вузли робота-маніпулятора із системами координат в цетрі мас.



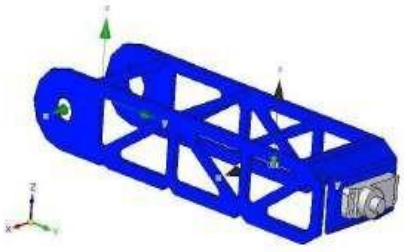
а)



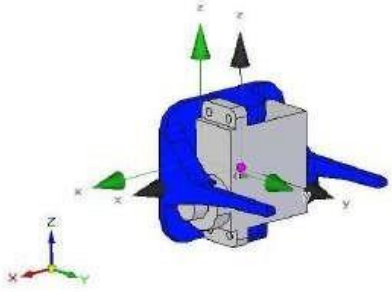
б)



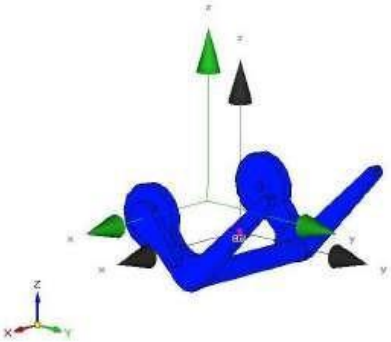
в)



г)



д)



е)

Рисунок 2.13 – Вузли робота-маніпулятора із системами координат в цетрі мас

а) збірка основи; б) батарея; в) кронштейн 1;

г) кронштейн 2; д) шасі ефектора; е) гачок ефектора

На рисунку 2.14 показана загальна модель робота-маніпулятора, від'єднаного від будь-якої опори. Як видно на схемі, вона складається з 6 ланок і 5 шарнірів, не враховуючи шарнір і фантомну ланку, що використовується для ЦТІ.

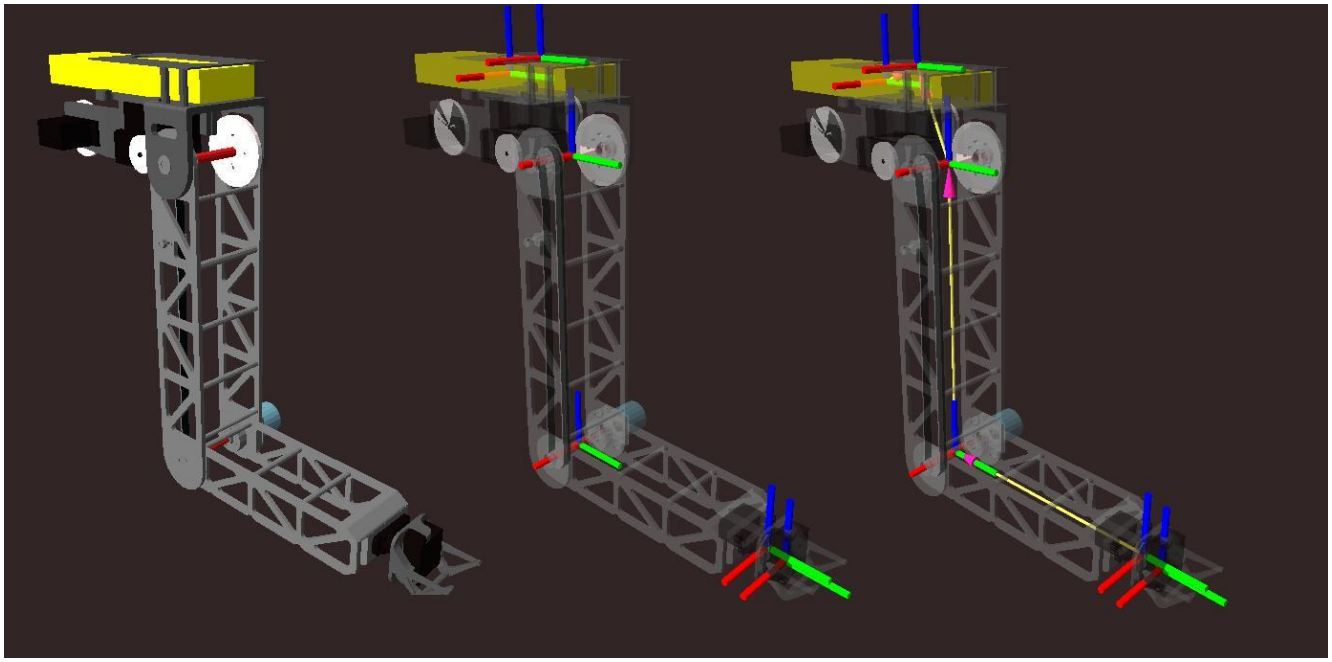


Рисунок 2.14 – Загальна модель робота-маніпулятора, від'єднаного від будь-якої опори

Потрібно також зазначити, що шарнір балкового лотка задано як призматичний, що залишає відкритою можливість проведення контролю та моделювання з компенсацією навантаження на основі цієї моделі.

На рисунку 2.15 центральна точка інструменту (ЦТІ) у вигляді "фантомної" ланки у відкритому ефекторі.

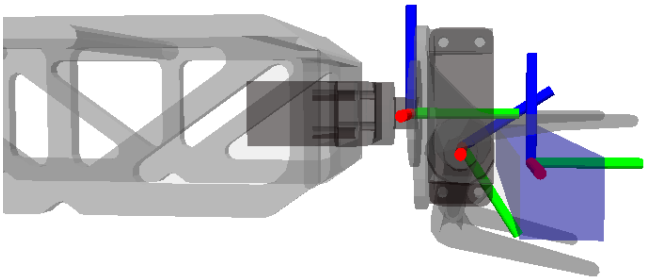


Рисунок 2.15 – Центральна точка інструменту у вигляді "фантомної" ланки у відкритому ефекторі

2.3 Висновки до другого розділу

1. Механічно робот-маніпулятор - це кінематичне коло, яке складається з ланок, з'єднаних шарнірами з одним або декількома ступенями свободи. У більшості роботів-маніпуляторів це кінематичне коло є розімкненим, починаючи з першої ланки або нерухомого базового корпусу і закінчуючи іншою, яка називається маніпулятором або маніпуляторним кінцем.

2. Пряма кінематична модель робота-маніпулятора дозволяє непрямим способом визначити точне місця розташування ефектора за координатами суглоба, які зазвичай відомі в будь-який час за допомогою датчика типу енкодера, пов'язаного з кожним суглобом.

3. Обернена кінематична модель забезпечує зворотний зв'язок з прямою кінематичною моделлю, а координати суглоба залежать від місця розташування ефектора. Корисність оберненої кінематичної моделі полягає в тому, що вона дозволяє керувати положенням ефектора. Система декартового позиційного керування забезпечується бажаним розташуванням ефектора, але кожному окремому шарнірному регулятору, як буде видно з прийнятої схеми керування, повинен бути забезпечений бажаний кут в його шарнірній координаті.

4. Динамічна модель робота-маніпулятора - це взаємозв'язок між положенням робота-маніпулятора, заданим координатами суглобів або розташуванням ефектора, та силами, які прикладені до суглобів. Робот-маніпулятор має три ступені свободи, по одному для кожного шарніра, які дозволяють розміщувати ефектор в потрібному положенні (Y, Z), завжди в межах робочого простору, і з певною орієнтацією за висотою ефектора.

5. Робот-маніпулятор має конфігурацію двигунів, які розміщені в його основі, а це зменшує вагу та інерційність ланок, але для цього необхідно додати систему ременів і шестерень. Єдині двигуни, які знаходяться безпосередньо на суглобі - це третій двигун, що встановлений у ланці 2, а також двигун самого ефектора.

3 ІМІТАЦІЙНА МОДЕЛЬ УДОСКОНАЛЕНОГО МЕТОДУ КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ НА ОСНОВІ ОПЕРАЦІЙНОЇ СИСТЕМИ РОБОТІВ

3.1 Особливості імітаційної моделі удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів

Існує два типи керування роботом-маніпулятором. Перший тип називається керуванням шарнірами, при якому кутове положення кожного шарніра контролюється безпосередньо незалежно або відокремлено від решти шарнірів, що дозволяє позиціонувати кожен шарнір під потрібним кутом і є дуже корисним при незалежному налаштуванні пов'язаних з ним контролерів.

Другий тип керування роботом-маніпулятором - це так зване декартове керування, в якому розташування ефектора в декартовому просторі контролюється за допомогою попереднього керування шарніром і кінематичною моделлю робота-маніпулятора.

Для кожного типу керування в ОСР створюється окремий пакет. Так, програми та класи, що відповідають шарнірному керуванню, знаходяться в пакеті control_manipulator, а ті, що відповідають декартовому управлінню - в пакеті control_cartesian_manipulator.

Обидва типу керування в ОСР використовують інфраструктуру керування pr2_mechanism, визначаючи для кожного окремий клас керування цього типу, і обидва мають усі необхідні файли для програмування, конфігурації та запуску.

Незалежне керування шарнірами 1, 2 і 3 відбувається за базовою схемою керування, показаною на рисунку 3.1. Базова схема керування складається з першого блоку, в якому перевіряється, що бажане положення шарніра знаходиться в межах шарніра, описаного в таблиці 2.1, в іншому випадку воно не змінюється, після чого обчислюється похибка положення шарніра, обчислюється зусилля, яким повинен керувати ПІД-регулятор, і надсилається на симулятор.

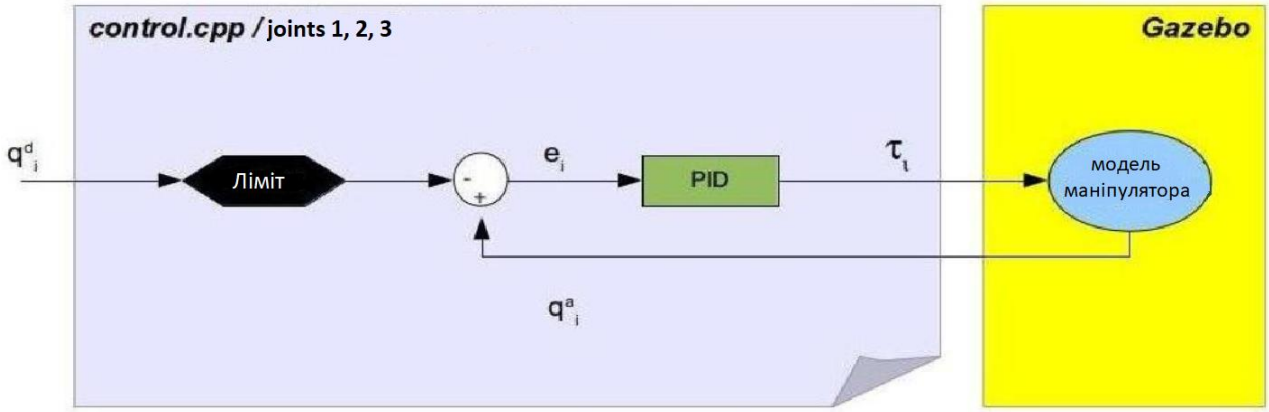


Рисунок 3.1 – Базова схема керування шарнірами 1, 2 і 3

Керування відкриттям і закриттям ефектора здійснюється за допомогою аналогічної структури керування, в якій блок перевірки меж з'єднання замінено на блок під назвою "вибір", показаний на рисунку 3.2.

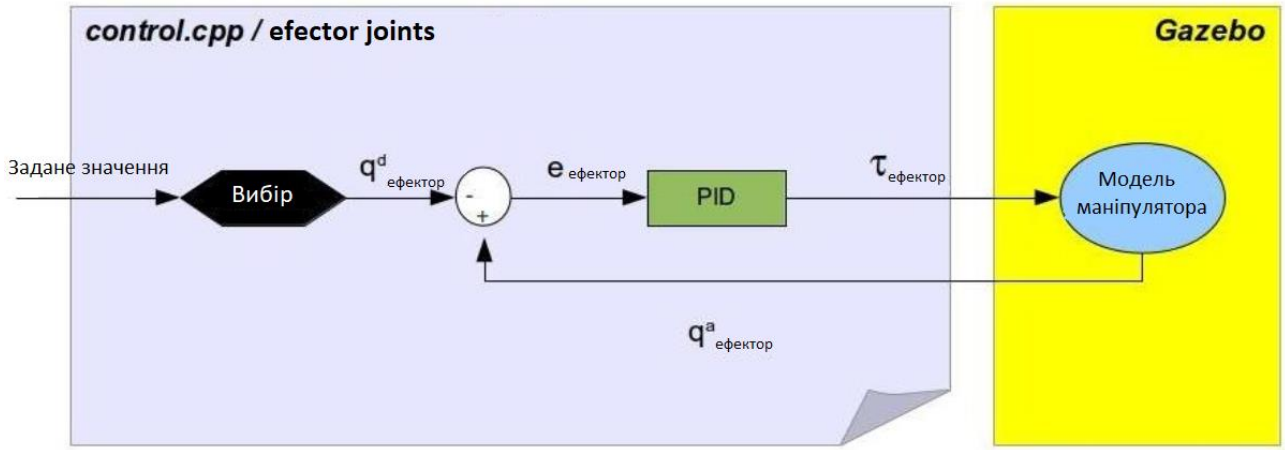


Рисунок 3.2 – Базова схема керування шарнірами ефектора

Цей блок отримує уставку з дискретним значенням 1, 0 або -1, призначаючи бажаному положенню вихідного з'єднання кут, необхідний для закриття (0,45°), повернення у вихідне положення (0°) або відкриття захвату (-0,8°) відповідно.

Для отримання бажаних положень шарнірів та уставки ефектора створені 4 сервіси з назвою setpoint_change_i (з i = 1, 3, 3 і ефектор), в яких змінюється значення змінних, що представляють ці позиції, setpoint_i, та де запрограмована перевірка меж зчленувань і блок "вибору" у випадку ефектора.

Функція ControlClass, яка виконується при виклику сервісу першого шарніра, має наступний вигляд.

```
bool ControlClass::change_setpoint_1(control_manipulator::
Change_Setpoint::Request& req, control_manipulator:: Change_Setpoint::Response&
resp)
{
if((req.setpoint<=1.57)&&(req.setpoint>=-1.08)) // Обмеження суглоба 1
{
setpoint_1 = req.setpoint;
}
else
{
ROS_ERROR("Бажане значення для шарніра 1 поза діапазоном");
resp.setpoint = setpoint_1; // Повертає значення без змін
return false;
}
resp.setpoint = setpoint_1; // Повертає змінене значення
return true;
}
```

При перевищенні значення обмеження, величина не змінюється від заданого значення. Це умова, яку повинні прийняти програми, що використовують спільний контроль вищого рівня, але це не означає, що це єдине рішення.

Для отримання поточних положень шарнірів та надсилання значення сил шарнірів, розрахованих ПІД-регуляторів, кожному шарніру присвоюється об'єкт JointState. Об'єкт JointState називається joint_state_i та використовує функції-члени, які йому належать, у відповідності до рисунку 3.3.

Крім того, для кожного шарніра створюється тема Read_data_i, через яку будуть надсилатися повідомлення типу Read_data.msg з інформацією про положення, швидкості, сили та моменти прикладання. Тема Read_data_i створюється в тому ж пакеті, що і контролер.

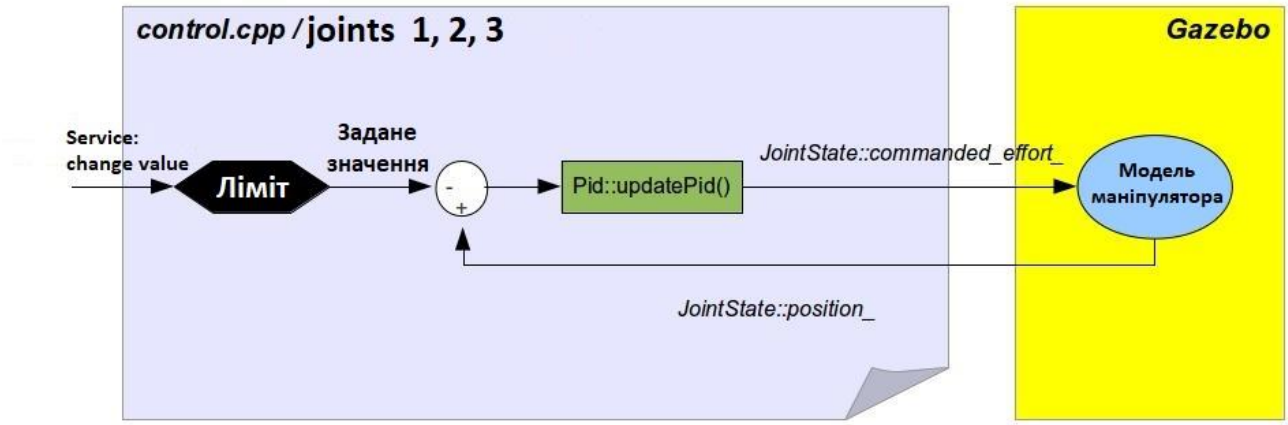


Рисунок 3.3 – Об'єкти та функції спільного контролю

Програмування є досить простим, оскільки необхідно лише перезаповнити функції-члени класу контролера, а диспетчер контролерів подбає про виконання кожної з них. Перш ніж перейти до програмування кожної з цих функцій, необхідно подивитися на файл параметрів controllers.yaml, який завантажується для цього контролера і який буде використовуватися в них.

```
controllers:
type: control_manipulator/ControlManipulatorPlugin
joint_name_1: Joint_1
joint_name_2: Joint_2
joint_name_3: Joint_3
joint_name_effector: Joint_effector
pid_parameters_1:
p: 10
i: 10
d: 0.5
i_clamp: 1000
pid_parameters_2:
p: 10
i: 10
d: 0.5
i_clamp: 1000
```

pid_parameters_3:

p: 0.07

i: 0.05

d: 0.005

i_clamp: 1000

pid_parameters_efector:

p: 0.001

i: 0.0005

d: 0.0001

i_clamp: 1000

Файл параметрів controllers.yaml визначає параметр type для реєстрації плагіна, параметри joint_name_i з іменами кожного шарніра в моделі URDF і параметри pid_parameters_i з коефіцієнтами ПІД-регуляторів кожного шарніра. Все це буде прочитано з контролера у першій з його функцій-членів:

Переходячи до програмування класу, заголовний файл control.h містить оголошення класу ControlClass, який буде спільним контролером, що успадковує форму класу Controller.

Він також визначає більшість об'єктів (JointState, RobotState, Pid і ті, що стосуються служб і тем), які будуть використані у визначенні класу control.cpp. Функції-члени класу control.cpp, які відноситься до шарніру, мають вигляд.

Функція init() контролера виконує наступні дії.

1. Отримує від сервера параметрів імена шарнірів, якими потрібно керувати, із отриманого NodeHandle (n).

n.getParam("joint_name_1", joint_name_1);

2. Отримує об'єкти JointState, пов'язані з кожним шарніром, з отриманого об'єкта RobotState (робот) та отриманого імені шарніра.

joint _state_1 = robot->getJointState(joint_name_1);

3. Відкриває сервіси для бажаної зміни позиції та теми для візуалізації даних.

srv_setpoint_1 = n.advertiseService("change_setpoint_1", &ClaseControl::

```
change_setpoint_1, this);
topic_pub_1 = n.advertise<control_manipulator::
Reading_data>("Reading_data_1", 10000);
```

4. Ініціалізує змінні заданих значень, які будуть містити бажані позиції та зберігає їх в об'єкті RobotState, маніпулятор, класу отриманий об'єкт того ж типу, що і робот.

5. Будує ПІД-регулятор на основі значень, завантажених у сервер параметрів.

```
pid_1.init(ros::NodeHandle(n, "pid_parameters_1")
```

Функція starting() записує значення початкових позицій і початкового моменту кінця циклу для розрахунку приросту часу ПІД-регулятора, а також для скидання ПІД-регулятора.

```
ros_initial_1 = join_state_1->position_;
end_cycle = manipulator->getTime();
pid_1.reset();
```

Функція update() виконується циклічно в реальному часі та має наступні кроки.

1. Зберігає бажані положення шарнірів (поточне значення змінної setpoint_i) в інших внутрішніх змінних функції, щоб вони не змінювалися в цьому циклі, а значення поточних положень шарнірів отримуємо з об'єктів JointState. Для першого шарніра фрагмент коду має вигляд.

```
double ros_des_1 = setpoint_1;
double ros_join_1 =join_state_1->position_;
```

Важливо відзначити, що при зміні бажаного положення, коли відбувається виклик одного з сервісів setpoint_change_i, функція сервісу виконується паралельно з функцією update(), змінюючи значення setpoint_i, не впливаючи на цикл керування і забезпечуючи його роботу в реальному часі.

2. Отримує приріст часу відносно останньої керуючої дії та оновлює значення моменту кінця циклу (останньої керуючої дії).

```
ros::Duration dt = manipulator->getTime() - end_cycle;
```

end_cycle = manipulator->getTime();

3. Обчислює зусилля, які потрібно докласти, використовуючи функцію update-Pid() об'єктів Pid, передаючи їм помилки в положенні та приріст часу, і надсилає їх роботу-маніпуляторц за допомогою об'єктів JointState. Для першого шарніра фрагмент коду має вигляд.

join_state_1->commanded_effort_ = pid_1.updatePid(pos_act_1-pos_des_1, dt);

4. Зберігає в об'єктах Data_i, типу control_manipulator::Read_data, всю інформацію про кожний шарнір і передає її через теми Read_data_i. Для першого шарніра фрагмент коду має вигляд.

Data_1.time = ros::Time::now().toSec();

Data_1.dt = dt.toSec();

Data_1.position = joint_state_1->position_;

Data_1.desired_position = ros_des_1;

Data_1.velocity = joint_state_1->velocity_;

Data_1.commanded_effort = joint_state_1->commanded_effort_;

Data_1.measured_effort = joint_state_1->measured_effort_;

topic_pub_1.publish(Data_1);

Функція stopping() залишена порожньою, тобто при зупинці або завершенні роботи контролера нічого не повинно виконуватися/

Налаштування ПІД-регуляторів виконується по черзі і безпосередньо в Gazebo. Процес налаштування кожного регулятора здійснюється шляхом фіксації шарнірів, починаючи від першого шарніра та закінчуючи ефектором. Вручному режимі (методом спроб і помилок) налаштуються коефіцієнти ПІД-регулятора для кожного шарніра, поки не буде досягнуто хорошої перехідної характеристики.

У таблиці 3.1 наведенні отримані коефіцієнти, які співпадають із параметрами у файлі controllers.yaml.

Таблиця 3.1 - Налаштування ПІД-регуляторів робота-маніпулятора

Шарнір	K_p	T_i	T_d
1	10	10	0.5
2	10	10	0.5
3	0,07	0,05	0,005
Ефектор	0,001	0,0005	0,0001

3.2 Декартове керування роботом-маніпулятором

Декартове керування роботом-маніпулятором використовує попередні спільні контролери разом з кінематичною моделлю робота-маніпулятора для керування положенням ефектора в декартовому просторі.

Як і в попередньому випадку, створюється успадкований клас Controller і модифікуються його функції-члени для виконання бажаного типу керування маніпулятором, використовуючи інфраструктуру pr2_mechanism.

У цьому випадку також використовується пакет kdl для розв'язання прямої та оберненої кінематичних моделей робота-маніпулятора у вигляді кінематичної схеми.

З огляду на конфігурацію робота-маніпулятора, було зроблено висновок, що незалежними змінними з точки зору розташування ефектора є його положення (Y, Z), яке завжди дорівнює нулю в X, та його орієнтація навколо осі Y ефектора, надалі - крок, оскільки орієнтація відносно двох інших осей є функцією положення. Таким чином, декартове керування роботом-маніпулятором поділяється на наступні три частини.

1. Розміщення ефектора у потрібних точках по Y та Z відносно системи координат основи робота-маніпулятора в робочій зоні маніпулятора.

2. Орієнтування ефектора на потрібну висоту відносно системи координат самого ефектора.

3. Керування відкриттям і закриттям ефектора.

Спочатку можна було передбачити, що орієнтація ефектора буде по відношенню до системи координат основи, однак, було вирішено посилатися на системи координат ефектора так, щоб бажана орієнтація була отримана по відношенню до основи за допомогою якогось датчика (наприклад, камери), розташованої в ефекторі.

З іншого боку, слід пам'ятати, що насправді контролер позиціонує роботом-маніпулятором у центральній точці інструменту. З точки зору моделі URDF центральна точка інструменту - це фантомна ланка, тому, коли ми говоримо про ефектор у цьому сенсі, ми завжди будемо мати на цю фантомну ланку.

На рисунку 3.4 показано змінні робота-маніпулятора, якими потрібно керувати в декартововому просторі.

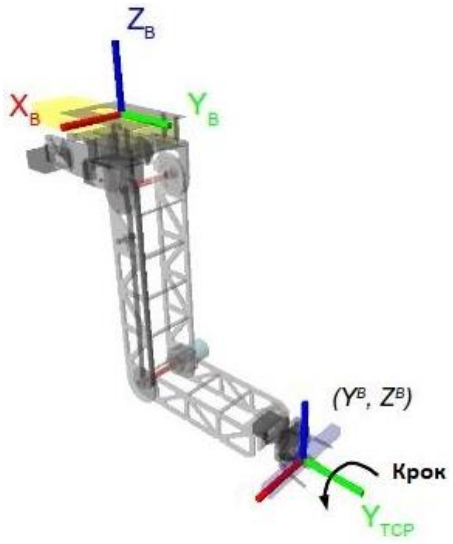


Рисунок 3.4 – Змінні робота-маніпулятора, якими потрібно керувати в декартововому просторі

Структура керування робота-маніпулятора, яку потрібно запрограмувати, зображена на рисунку 3.5. Завдяки конфігурації робота-маніпулятора, керування кроком ефектора безпосередньо пов'язане з керуванням шарніром q3.

Керування положенням (Y, Z) складається з трьох кроків. Спочатку виконується перевірка на приналежність бажаної позиції робочій області маніпулятора, визначеній системою рівнянь (2.6).

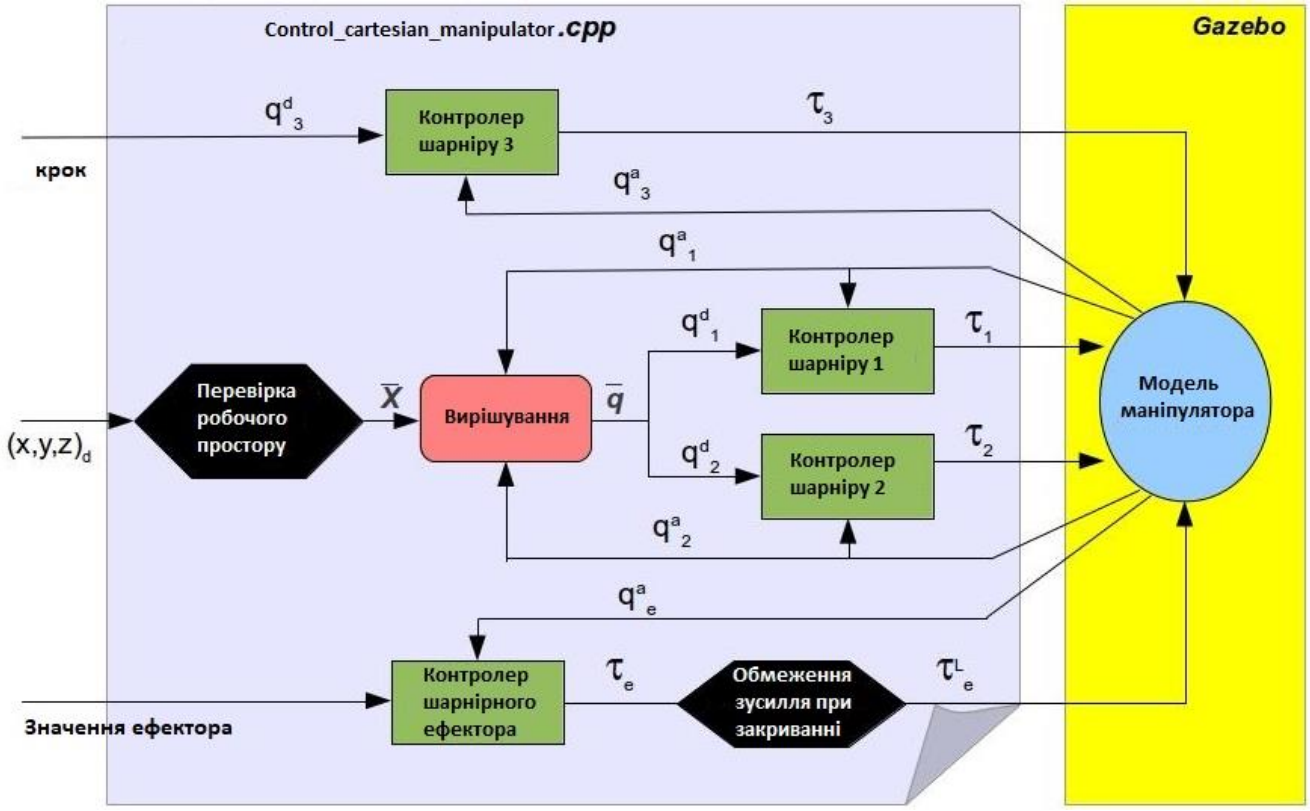


Рисунок 3.5 – Декартова структура керування робота-маніпулятора

У цьому випадку розв'язується оберненана кінематична задача, яка полягає в отриманні значень положень першого q1 і другого q2 шарнірів для цього положення ефектора. Вводячи ці необхідні або бажані значення положень шарнірів у відповідні контролери шарнірів, симулятору надсилаються необхідні зусилля для отримання цих q1 і q2 або, іншими словами, для позиціонування ефектора в бажаних координатах Y і Z.

Керування відкриттям і закриттям ефектора відбувається так само, як і при спільному керуванні, за винятком того, що накладається обмеження на максимальне зусилля для закриття.

Обмеження на максимальне зусилля для закриття ефектора призначене для усунення невизначеного збільшення інтегральної похибки ПІД-регулятора через те, що об'єкт, який захоплюється, може не досягти положення закриття шарніра (0,45рад), якщо його розміри більші, ніж передбачені та встановлені для цього положення закриття.

Бажані позиції будуть зчитуватися сервісом, а поточні позиції шарнірів будуть зчитуватися об'єктами JointState в еквівалентний спосіб ніж при спільному керуванні, хоча їх також можна отримати за допомогою кінематичної схеми.

Для використання розв'язувача оберненої кінематики з пакета kdl необхідно передати кінематичну схему у вигляді об'єкта KDL::Chain, бажану декартову позицію у вигляді об'єкта KDL::Frame і, аналогічно, зібрати необхідні штучні позиції в об'єкті KDL::JntArray.

На рисунку 3.6 показано основні об'єкти та функції, що використовуються у контролері, а також прокоментовані сервіси.

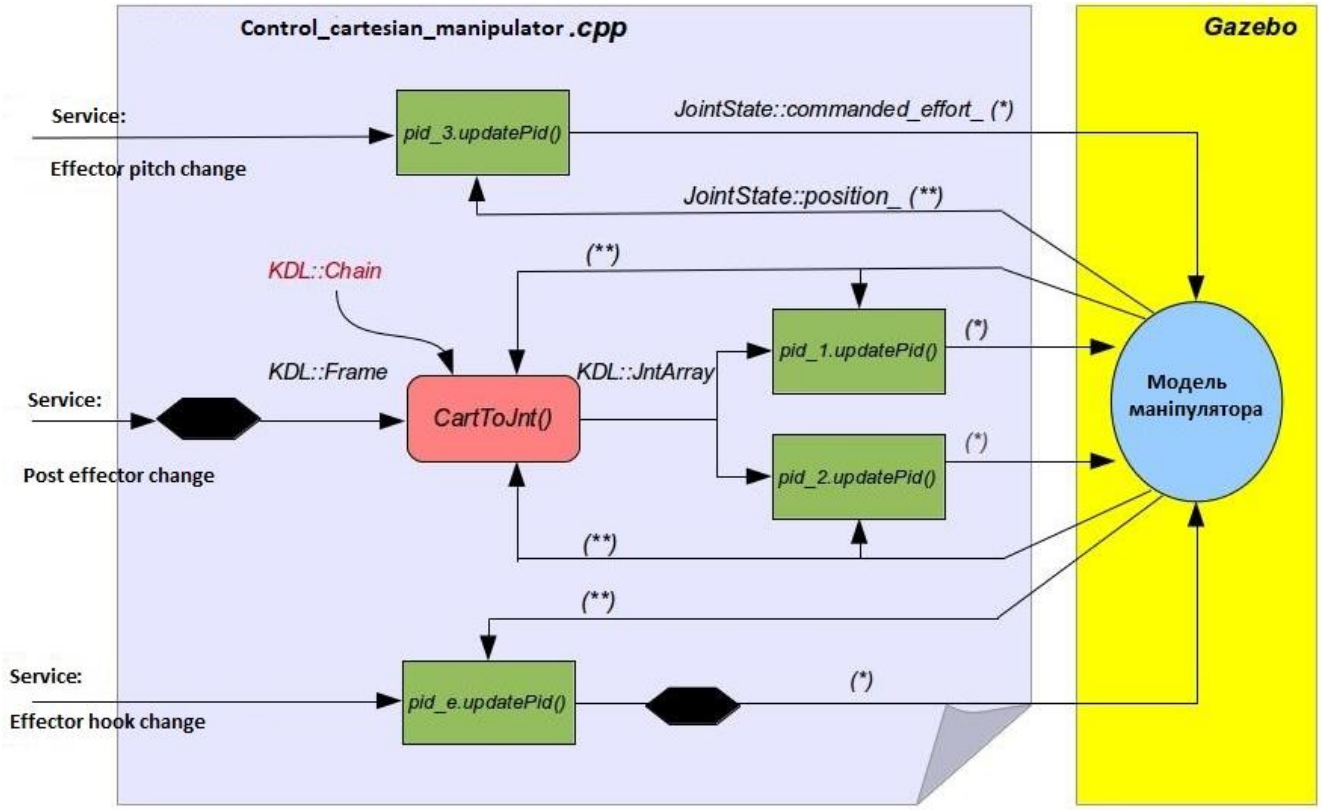


Рисунок 3.6 – Основні об'єкти та функції, що використовуються у декартовій структурі керування роботом-маніпулятором

Сервісна функція (і пов'язані з нею типи повідомлень) для зміни уставки ефектора така ж, як і для спільного керування, а сервісна функція для зміни кроку така ж, як і для з'єднання 3 того ж контролера.

Однак, щоб змінити бажану декартову позицію центральної точки інструменту, необхідна абсолютно нова функція, у якій ця позиція отримується відносно системи координат основи робота-маніпулятора.

Отримане положення відносно системи координат основи маніпулятора переноситься на позицію ланки 1 і перевіряється, чи отримане положення знаходиться в межах робочого простору робота-маніпулятора.

```
bool CartesianControlClass::change_effector_pos(
cartesian_control_manipulator::Off_effector_pos::Request&
req, cartesian_control_manipulator::Off_effector_pos::
Response& resp)
{
// Переносимо бажані позиції відносно основи до системи відліку плеча 1,
де визначено робочий простір, щоб побачити, чи знаходиться вона у межах
float y_off_ref_1= req.position.y - 0.026;
float z_off_ref_1= req.position.z + 0.055;
if((sqrt((y_des_ref_1*y_des_ref_1)+(z_des_ref_1*
z_des_ref_1))<=0.3780)&&(sqrt((y_des_ref_1*
y_des_ref_1)+(z_des_ref_1*z_des_ref_1))>=0.2745))
{
// Якщо позиція знаходиться всередині робочої області
xd.p(1)=req.position.y; // Присвоюємо y бажане значення
xd.p(2)=req.position.z; // Присвоюємо z бажане значення
resp.confirmation=true;
}
else
{
// Якщо позиція знаходиться поза робочої області
resp.confirmation=false; // Вказувати на те, що програма, яка викликала службу, вибрана неправильно
```

```
ROS_ERROR("Бажана позиція ефектора за межами поточної робочої області");
return false;
}
return true;
// Інструкція з використання терміналу:
// $ rosservice call /cartesian_controller/
change_effector_position '{position: {x: 0.0, y: 0.204, z: -0.264}}
}
```

Кінематична схема, яка використовується для розв'язання оберненої кінематичної задачі, буде утворена шарнірами 1, 2 і уявною ланкою та буде переміщуватися від місця кріплення до уявної ланки, як показано на рисунку 3.7.

Рисунок 3.7 – Кінематична схема робота-маніпулятора

Шарніри 1 і 2 відповідають за позиціонування центральної точки інструменту за осями Y і Z, а уявною ланка поглинає крен (орієнтацію за віссю X) у цьому положенні. Кореневі та кінцеві ланки кінематичної схеми завантажуються як параметри у файл control_cartesian_manipulator.yaml разом зі створеним для цього контролера плагіном, який в іншому ідентичний створеному для спільного керування.

```
cartesian_controller:
type: cartesian_control_manipulator/
CartesianControlManipulatorPlugin
root_name: Base_set
tip_name: phantom
```

Клас, розроблений для декартового керування роботом-маніпулятором, називається ControlCartesianControlClass, він є спадкоємцем класу Controller, оголошеного у заголовному файлі control_cartesian_manipulator.h та визначеного у файлі control_cartesian_manipulator.cpp. У цьому заголовному файлі оголошуються всі необхідні об'єкти, які будуть використовуватися у попередньому класі та які були показані на рисунку 3.6.

У функції init() із сервера параметрів зчитуються шарніри, якими потрібно керувати, а також параметри ПІД-регуляторів. Далі створюються об'єкти JointState для кожного зчитуваного шарніра, зберігається отриманий об'єкт RobotState, відкриваються 4 теми для надсилання інформації про моделювання кожного шарніра.

Також створюються три сервіси для зміни тих самих заданих параметрів, як це було зроблено для спільного керування. На додаток до цих дій додаються дії, пов'язані з кінематичною схемою і вирішувачем оберненої кінематики.

1. Зчитування із сервера параметрів початкової та кінцевої ланок кінематичної схеми.

```
std::string root_name, tip_name; n.getParam("root_name", root_name) n.getParam("tip_name", tip_name)
```

2. Створення рядка pr2_mechanism_model::Chain для робота-маніпулятора, початкового та кінцевого зчитування шарнірів та його перетворення до типу KDL::Chain.

```
string.init(robot, root_name, type_name)string.toKDL(string_kdl);
```

3. Після того, як ми знаємо розмір кінематичної схеми (у випадку 3-шарнірного маніпулятора), ми надаємо відповідний розмір змінним JntArray, які представляють бажані та фактичні положення шарнірів.

```
q.resize(string_kdl.getNrOfJoints());
qd.resize(string_kdl.getNrOfJoints());
```

4. Для розв'язувача оберненої кінематики необхідно мати розв'язувач прямої кінематики та розв'язувач оберненої кінематики за швидкістю. Їх можна створити наступним чином з рядка kdl.

```
solver_CD.reset(new KDL::
    ChainFkSolverPos_recursive(chain_kdl));
solver_CI_Vel.reset(new KDL ::
    ChainIkSolverVel_pinv(chain_kdl));
solver_CI.reset(newKDL::ChainIkSolverPos_NR(chain_kdl,*solver_CD,
*solver_CI_Vel));
```

Функція starting(), окрім отримання початкового часового кроку і перенацілювання об'єктів Pid, в цьому випадку присвоює бажаній декартовій позиції початкову позицію ефектора, використовуючи розв'язувач прямої кінематики.

```
// Початкові позиції шарнірів
string.getPositions(q0);
// Початкові декартові позиції
solver_CD->JntToCart(q0, x0);
// Бажана початкова позиція дорівнює початковій xd=x0;
```

Функція update() модифікується лише в тому сенсі, що бажані спільні позиції тепер більше не отримуються сервісом і більше не мають не зберігаються в жодній змінній на початку циклу.

Бажані спільні позиції тепер отримуються з бажаної декартової позиції, яка є тією, що надходить в роботу, та шляхом розв'язання оберненої кінематичної задачі за допомогою відповідного розв'язувача. В інших аспектах ця функція майже еквівалентна функції спільного контролю.

```
solver_CI->CartToJnt(q, xd, qd);
double pos_des_1 = qd(0);
double pos_des_2 = qd(1)
```

Функція stopping() знову залишається порожньою.

3.3 Висновки до третього розділу

1. Існує два типи керування роботом-маніпулятором. Перший тип називається керуванням шарнірами, при якому кутове положення кожного

шарніра контролюється безпосередньо незалежно або відокремлено від решти шарнірів, що дозволяє позиціонувати кожен шарнір під потрібним кутом і є дуже корисним при незалежному налаштуванні пов'язаних з ним контролерів.

2. Другий тип керування роботом-маніпулятором - це так зване декартове керування, в якому розташування ефектора в декартовому просторі контролюється за допомогою попереднього керування шарніром і кінематичною моделлю робота-маніпулятора.

3. Незалежне керування шарнірами 1, 2 і 3 відбувається за базовою схемою керування, що складається з блоку, в якому перевіряється, чи бажане положення шарніра знаходиться в межах шарніра чи ні. Далі обчислюється похибка положення шарніра, обчислюється зусилля, яким повинен керувати ПІД-регулятор і надсилається на симулятор.

4. Налаштування ПІД-регуляторів виконується по черзі і безпосередньо в Gazebo. Процес налаштування кожного регулятора здійснюється шляхом фіксації шарнірів, починаючи від першого шарніра та закінчуючи ефектором. Вручному режимі (методом спроб і помилок) налаштуються коефіцієнти ПІД-регулятора для кожного шарніра, поки не буде досягнуто хорошої перехідної характеристики.

5. Декартове керування роботом-маніпулятором використовує попередні спільні контролери разом з кінематичною моделлю робота-маніпулятора для керування положенням ефектора в декартовому просторі.

6. Декартове керування роботом-маніпулятором поділяється на наступні три частини:

- розміщення ефектора у потрібних точках по Y та Z відносно системи координат основи робота-маніпулятора в робочій зоні маніпулятора;

- орієнтування ефектора на потрібну висоту відносно системи координат самого ефектора;

- керування відкриттям і закриттям ефектора.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ УДОСКОНАЛЕНОГО МЕТОДУ КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ НА ОСНОВІ ОПЕРАЦІЙНОЇ СИСТЕМИ РОБОТІВ

4.1 Особливості експериментального дослідження удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів

Особливістю експериментального дослідження удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів є використання симуляції для перевірки правильності роботи контролерів.

В процесі симуляції використовувалося керування шарнірами, в яких будуть проаналізовані реакції кожного шарніра в положенні, швидкості і зусиллі, а також вплив кожного шарніра на інші.

Виконано симуляцію керування в декартовому просторі, де проаналізуємо реакції по Y, Z, а також перевіримо, чи дотримується робочий простір.

Усі симуляції виконувалися для моделі man_mesa.urdf ізапускатимуться з пакета manipulator, де знаходиться модель робота-маніпулятора. Для запуску кожного типу контролера достатньо включити до його пакету створений запускник.

Для цих симуляцій створено запускник man_mesa_gazebo_-control_join.launch. У ньому запрограмовано наступне.

1. Запуск Gazebo з конфігурацією empty.world та увімкнення параметру використання часу симуляції.

2. Завантаження робота-маніпулятора man_mesa.urdf в Gazebo.

3. Виконання команди controllers.launch з пакета control_manipulator (спільний контролер), в який завантажується файл параметрів (controllers.yaml) і запуск диспетчера контролерів через вузол spawner пакета pr2_controller_manager під ім'ям controller_spawner.

В цьому випадку відбудеться передача завантаженого файлу параметрів (тобто плагіна, який він повинен використовувати для управління контролером).

У результаті з'являється вузол під назвою controller_spawner, який буде виконувати функції-члени нашого класу шарнірного контролера належним чином і в реальному часі.

4. Запуск декількох вузлів rxplot, щоб отримати графіки з положенням, швидкістю, розрахованим зусиллям і переданим зусиллям (обмеженим максимальним зусиллям, заявленим в URDF) для всіх з'єднань з вихідних тем Reading_data_i.

Щоб запустити всю цю систему, в терміналі виконується наступна команда.

$roslaunch manipulator man_mesa_gazebo_control_join.launch

Граф системи в ОСР для моделювання робота-маніпулятора зобажено на рисунку 4.1.

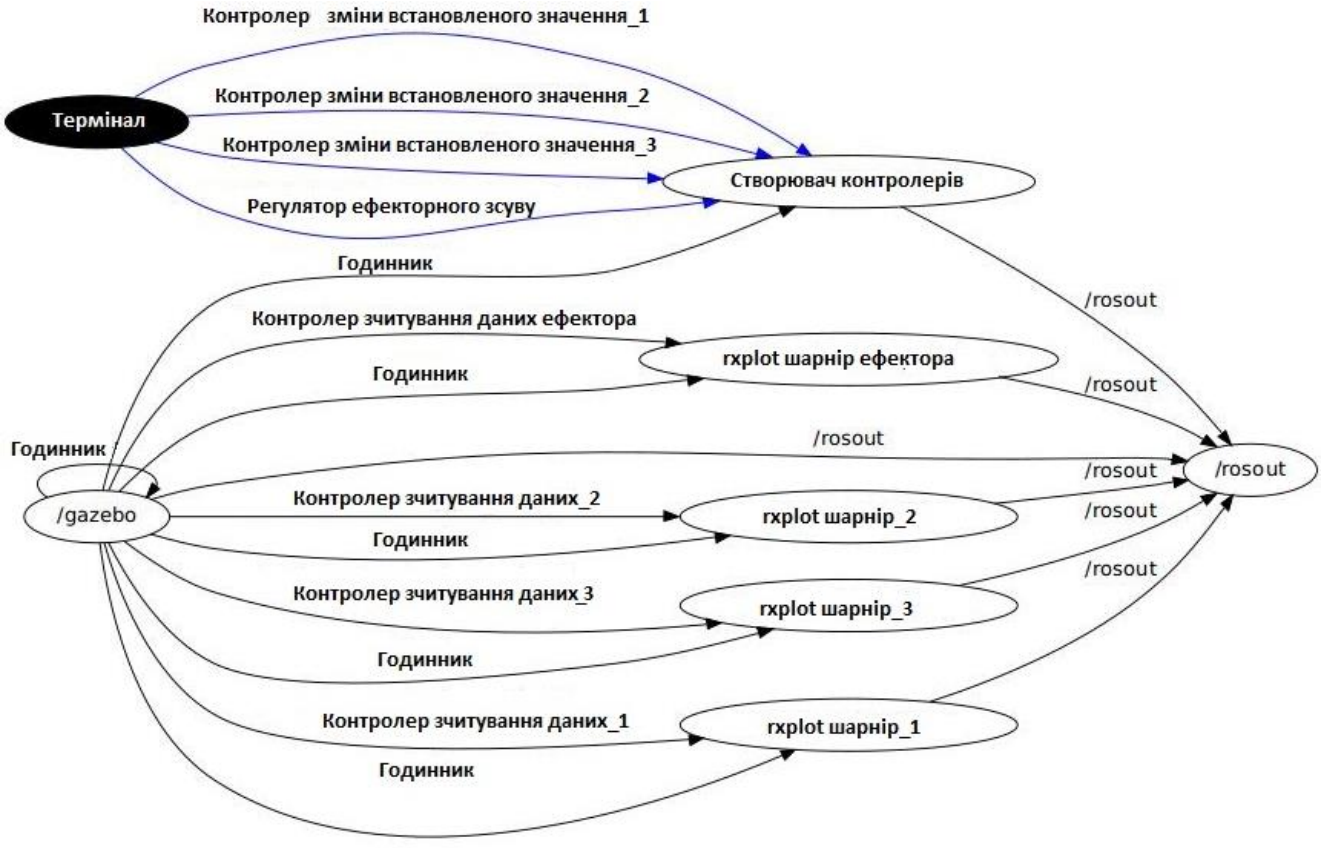


Рисунок 4.1 – Граф системи в ОСР для моделювання робота-маніпулятора

Граф системи в ОСР для моделювання робота-маніпулятора отриманий за допомогою rxgraph з додаванням сервісів для зміни бажаних положень шарнірів.

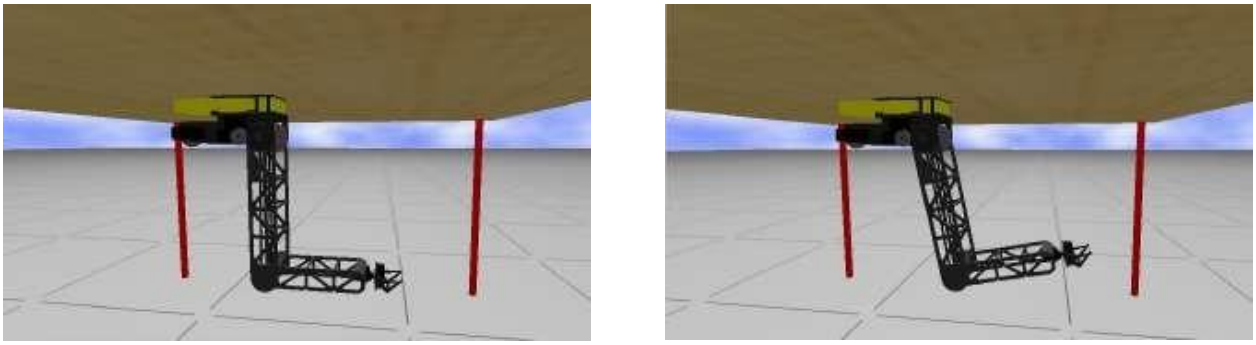
Крім того, додано вузол "TERMINAL", щоб підкреслити, що виклики цих сервісів будуть здійснюватися, в даному випадку, з терміналу Ubuntu, що в даному випадку еквівалентно додатковому вузлу.

1. Експериментальне дослідження шарніру 1.

Змінимо кут повороту першого шарніру на 0,2 рад.

$rosservice call/controllers/set_change_1 0.2.

На рисунку 4.2 зображено положення першого шарніру при 0рад та при 0,2 рад.



а) б)

Рисунок 4.2 – Положення першого шарніру:

а) при 0рад;

б) при 0,2 рад

Відгук першого шарніра показано на рисунку 4.3. Перехідний процес реакції положення має такі характеристики:

- час наростання 0,15 секунди;

- перерегулювання 2,75 %;

- час стабілізації 1,2 секунди (99 %).

Що стосується швидкості, то для цього кута повороту вона набула максимального значення близько 0,3 рад/с, що становить менше половини максимальної швидкості серводвигуна.

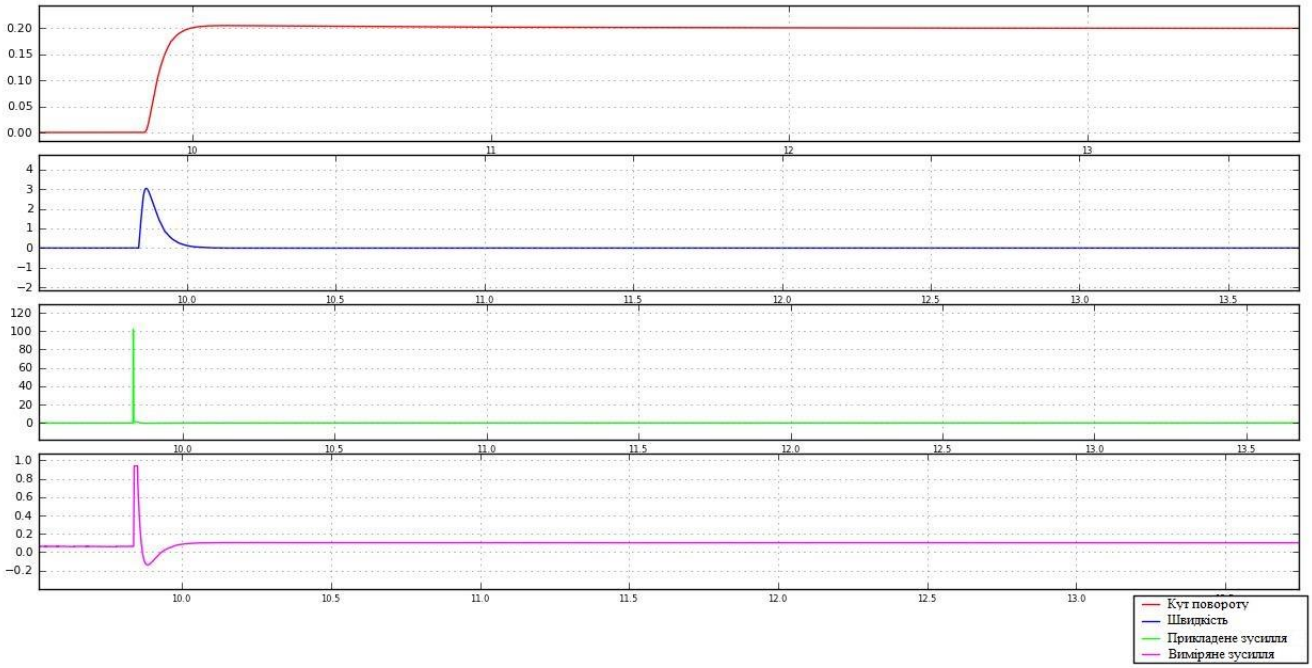


Рисунок 4.3 – Відгук першого шарніра

Що стосується зусилля, то було досягнуто максимального (рекомендованого) значення серводвигуна, близько 0,94 Нм, але без його перевищення, оскільки контролер обмежує зусилля до його можливого максимуму.

Зеленим кольором показано задане напруження, тобто напруження, розраховане ПІД-регулятором, а фіолетовим - те саме напруження після застосування обмежень, визначених для контролерів в URDF для кожного з'єднання. Що стосується постійного напруження, то воно змінюється від 0,05 Нм до 0,15 Нм.

Щодо впливу першого шарніра на інші шарніри, то варто виділити вплив на ефекторний шарнір, який є найбільшим, оскільки він знаходиться на найбільшій відстані від осі обертання, але таке моделювання не зовсім коректне, оскільки, як правило, ці шарніри мають гальмо на серводвигуні, який не моделюється. Вплив на інші шарніри є досить незначним. На рисунку 4.4 зображено вплив першого шарніра на інші шарніри.

2. Експериментальне дослідження шарніру 2.

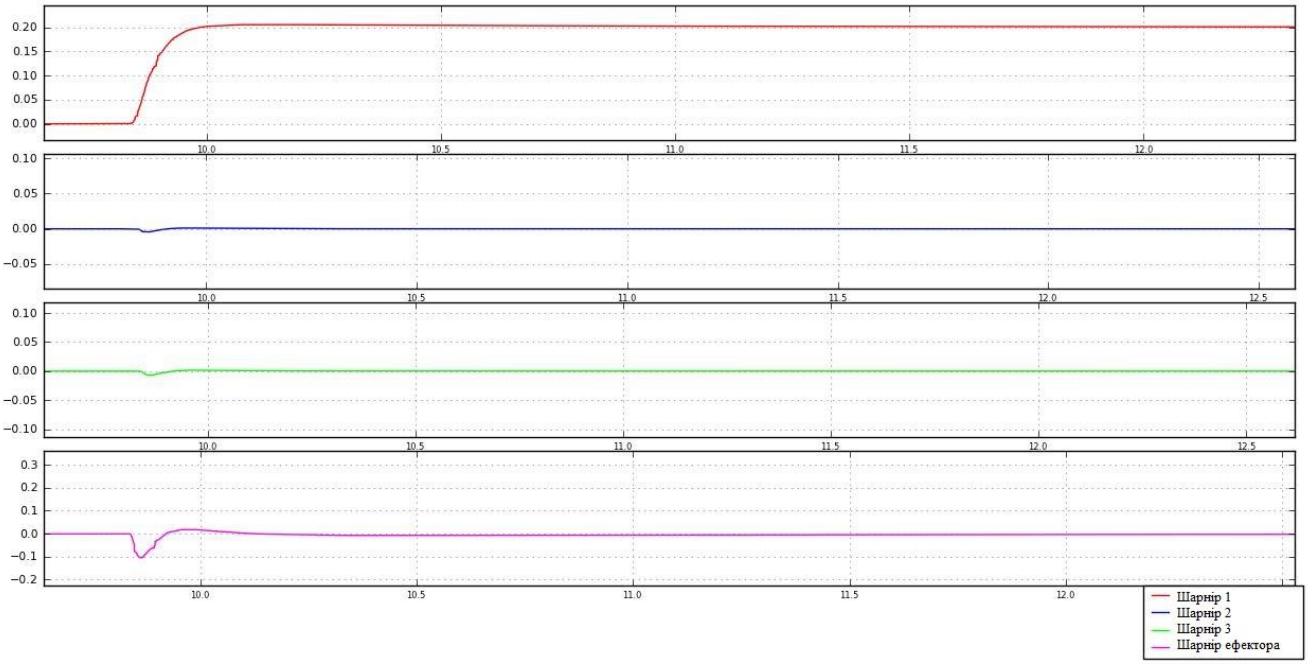
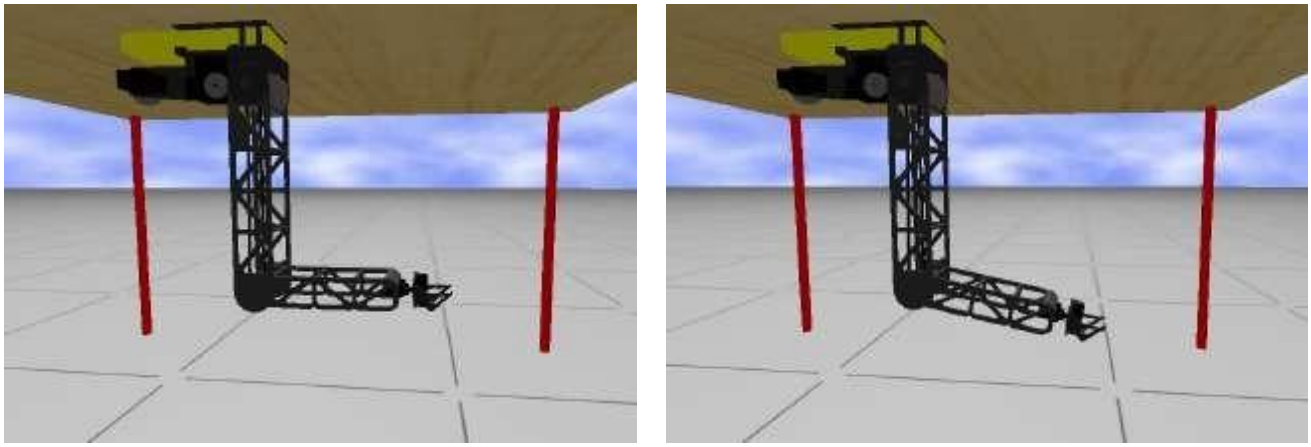


Рисунок 4.4 – Вплив першого шарніра на інші шарніри

Змінимо кут повороту другого шарніру на -0,2 рад, в той час як сигнали інших шарнірів залишаються на нульових значеннях.

$rosservice call/controllers/set_change_2 -0.2.

На рисунку 4.5 зображено положення першого шарніру при -0,2рад та при 0 рад.



а) б)

Рисунок 4.5 – Положення першого шарніру:

а) при 0рад;

б) при -0,2 рад

Відгук другого шарніра показано на рисунку 4.6. У цьому випадку позиційна характеристика показує під час перехідного процесу час наростання 0,16 секунди, перерегулювання 4 % і час стабілізації (99 %) 1,7 с.

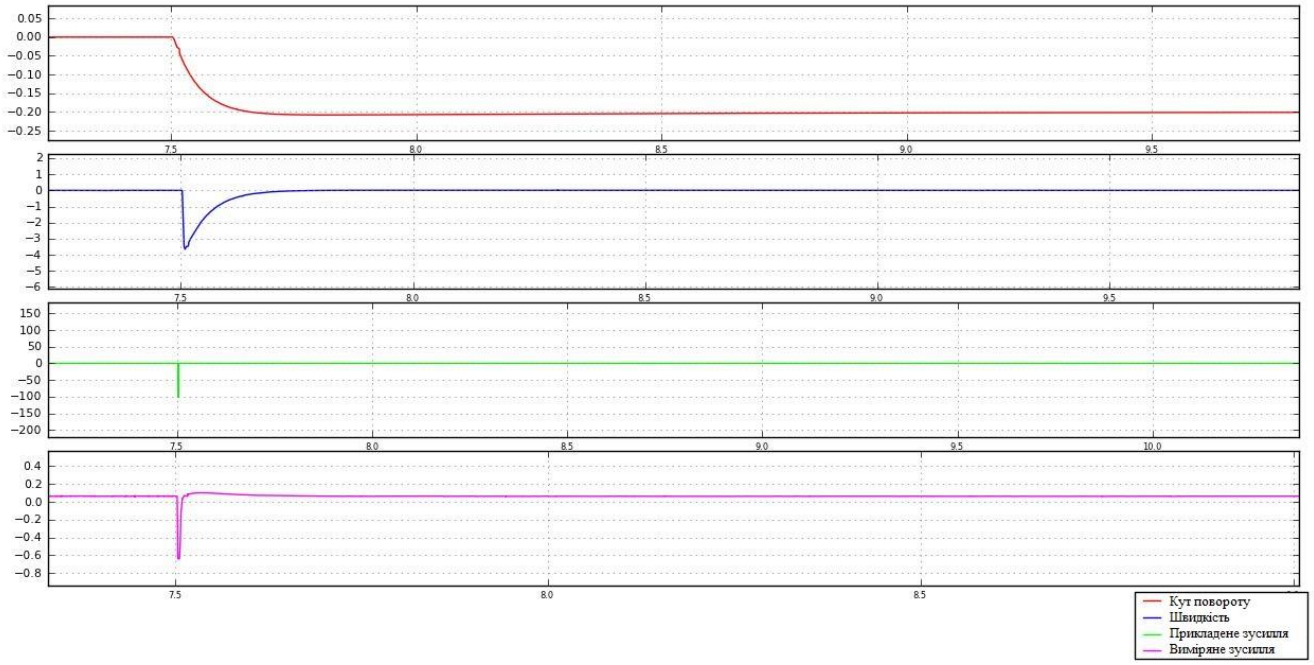


Рисунок 4.6 – Відгук другого шарніра

На рисунку 4.7 зображено вплив другого шарніра на інші шарніри.

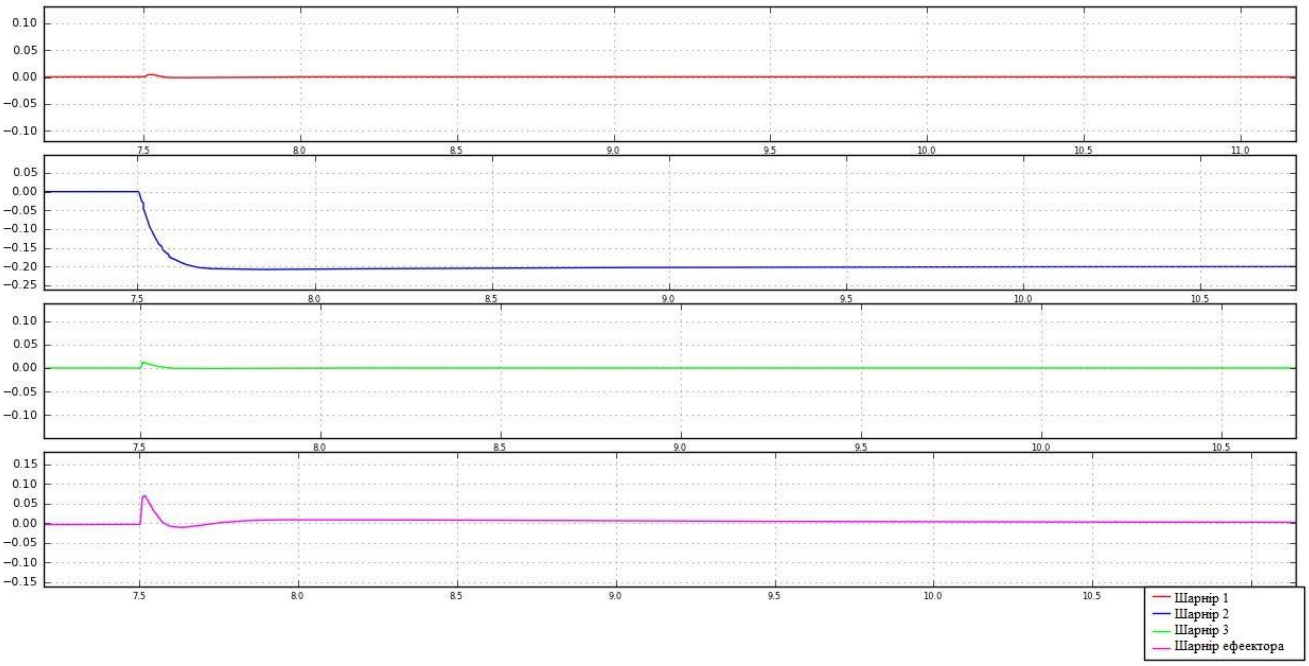


Рисунок 4.7 – Вплив другого шарніра на інші шарніри

Вплив другого шарніра на перший і третій шарнір практично незначний, але вплив на перший і третій суглоби більший, ніж вплив на третій суглоб. Причини цього такі ж, як і для першого суглоба.

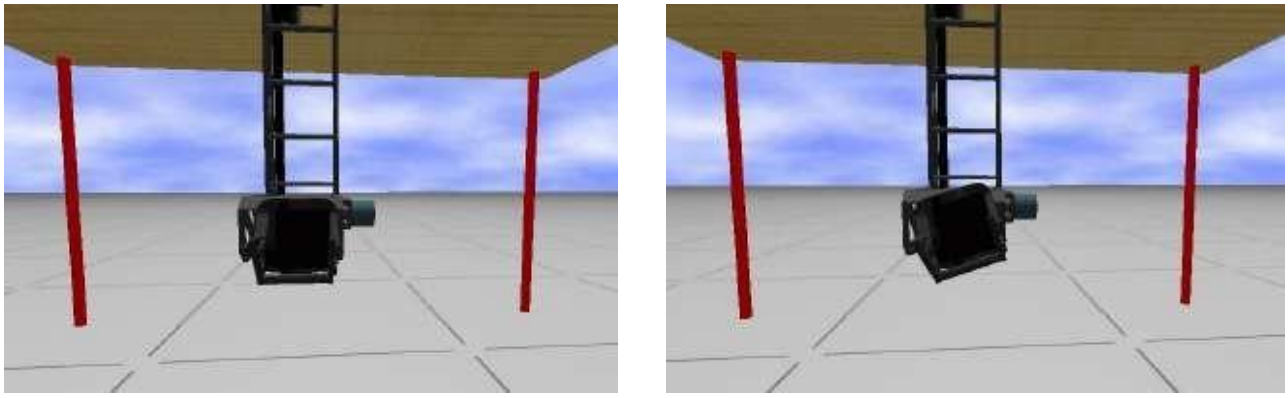
3. Експериментальне дослідження шарніру 3.

Змінимо кут повороту першого шарніру на 0,3 рад.

$rosservice call/controllers/set_change_1 0.3.

На рисунку 4.8 зображено положення першого шарніру при 0рад та при 0,3 рад.

$rosservice call/controllers/set_change_3 0.3



а) б)

Рисунок 4.8 – Положення першого шарніру:

а) при 0рад;

б) при 0,3 рад

Для такого підвищення позиції, час підйому від 0,22 секунди, перерегулювання 3,8 % і час встановлення (99 %) 2,31 секунди. У цьому випадку межа не досягається ні за швидкістю, ні за напруженням, однак у вимірах обох величин є багато шуму, як показано на рисунку 4.9.

Що стосується впливу на інші шарніри, то третій шарнір у своїй модифікації не впливає на жоден з інших шарнірів, що пояснюється його низькою інерційністю і перпендикулярністю осі обертання по відношенню до інших шарнірів.

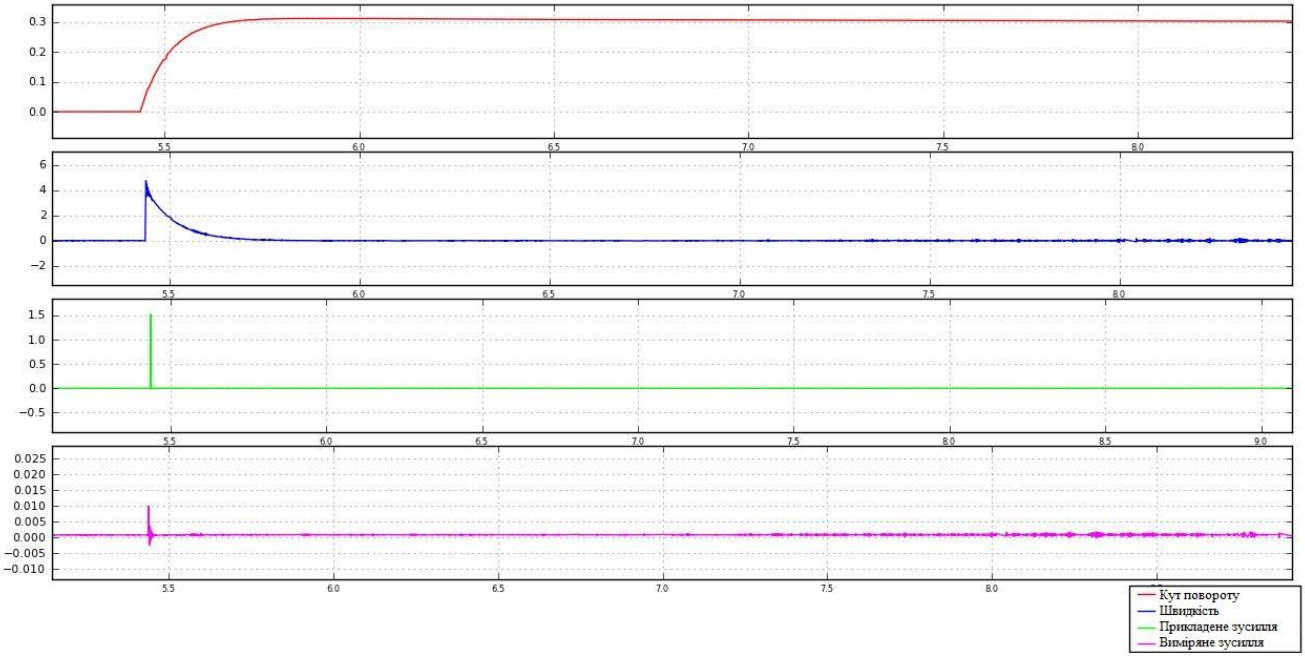


Рисунок 4.9 – Відгук третього шарніра

4. Експериментальне дослідження ефекторного шарніру.

Для ефектора повинні бути змодельовані випадки відкриття та закриття гачка

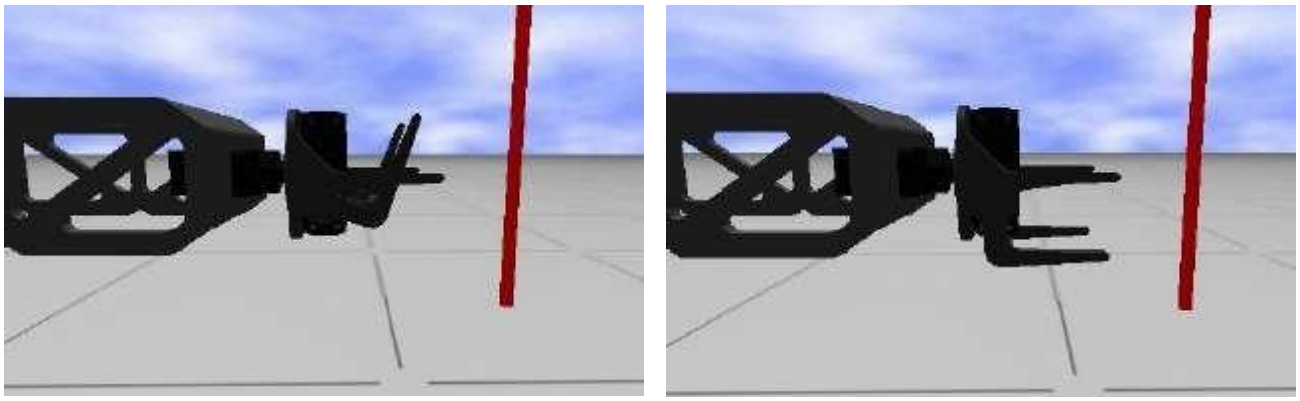
$rosservice call/controllers/set_change_efector - - - 1

На рисунку 4.10 зображено положення ефекторного шарніру.

В даному випадку, хоча бажані кути відкриття та закриття ефектора отримані, як видно з рисунку 4.11, говорити про характеристики реакції не має сенсу, оскільки при обертанні шарніра спостерігається непропорційна швидкість, яка не була зафіксована.

Можна було б подумати, що це пов'язано з тим, що, як зазначалося, для цього з'єднання не використовуються параметри URDF "управління безпекою", однак у разі їх використання реакція в з'єднанні стає нестійкою для всіх протестованих ПІД-регуляторів.

OCP попереджає про таку можливість для об'єктів з малою інерційністю, тому для нашого ефектора ми не зможемо отримати достовірну інформацію з моделювання, хоча за зусиллям він досить малий і здається надійним.



а) б)

Рисунок 4.10 – Положення ефекторного шарніру:

а) закриття гачка;

б) відкриття гачка

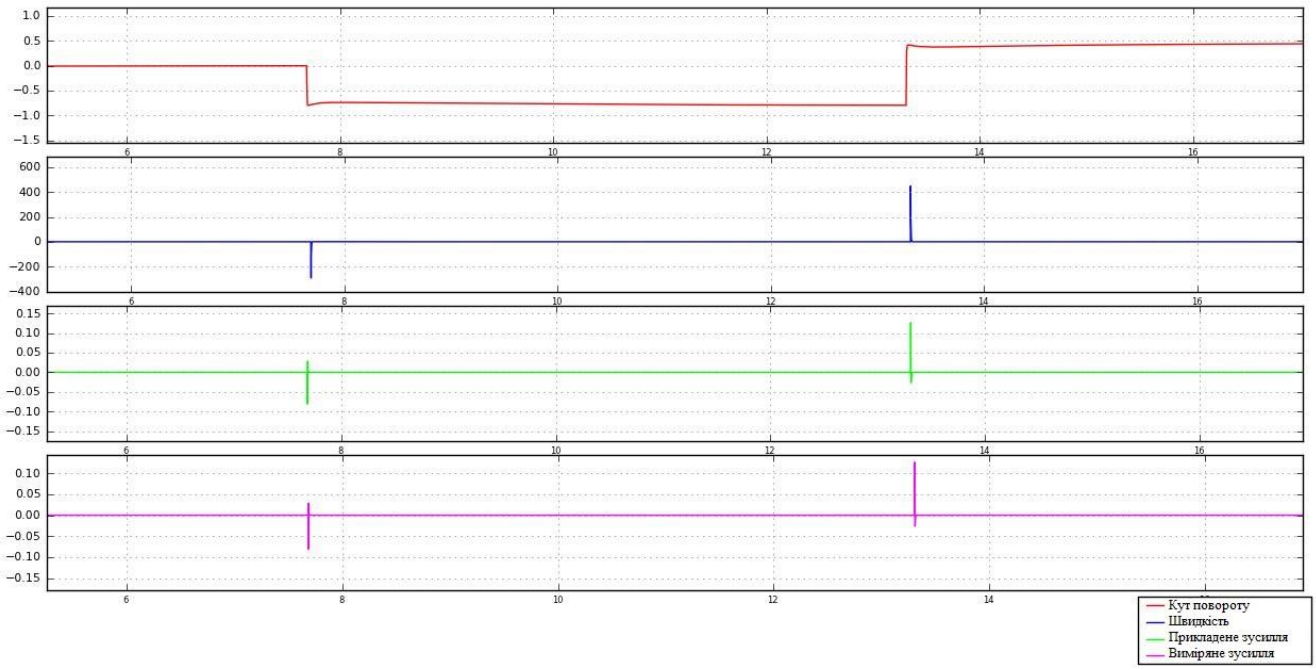


Рисунок 4.11 – Відгук ефекторного шарніра

4.2 Експериментальне дослідження керування роботом-маніпулятором в декартовому просторі

Експериментальне дослідження керування робота-маніпулятора в декартовому просторі здійснюється шляхом створення ще одного завантажувальник.

$roslaunch manipuladtor man_mesa_gazebo_control_cartesian.launch

У цьому завантажувальнику використовуються ті ж вузли, як і у випадку з шарнірним керуванням, з тією різницею, що тепер здійснюємо запуск декартового контролера control_cartesian_manipulator.launch.

Крім того, використовується rxplot для отримання графіків положення центру кінцевого інструменту від розташованого на ньому датчика положення, і консоль (rxconsole) для отримання інформації про виконання алгоритму, запрограмованого в контролері.

На рисунку 4.12 зображено граф системи в OCP для моделювання керування роботом-маніпулятором в декартовому просторі.

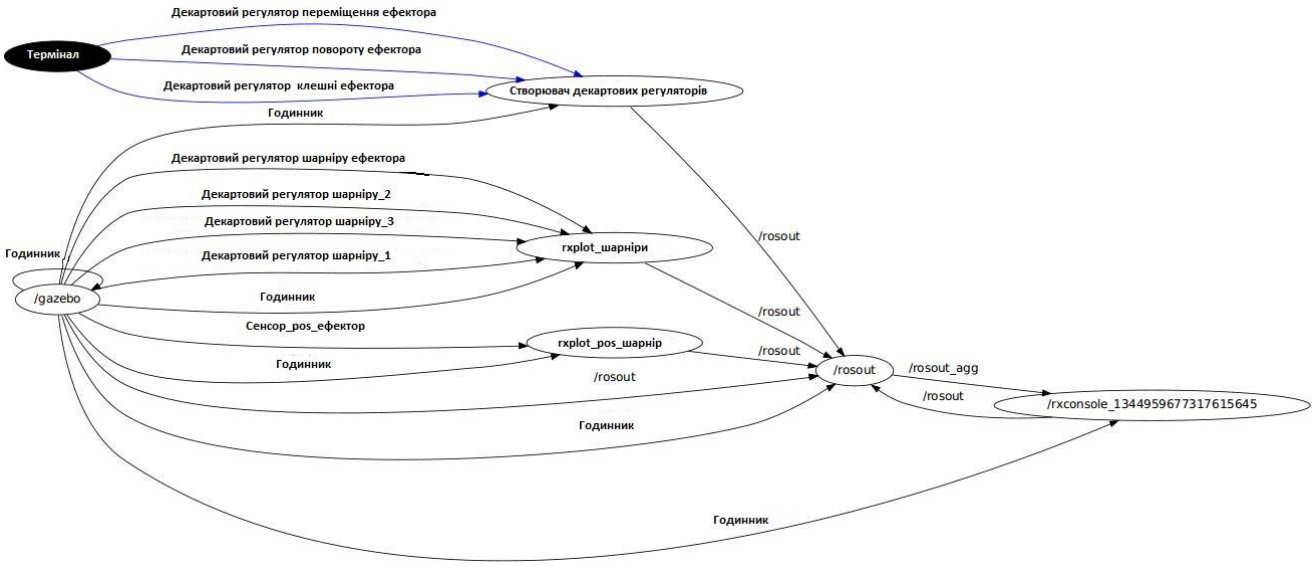


Рисунок 4.12 – Граф системи в OCP для моделювання керування роботом-маніпулятором в декартовому просторі

Для експериментального дослідження керування робота-маніпулятора в декартовому просторі запускається вищезгадана система і до неї надсилаються різні бажані позиції робота-маніпулятора за допомогою сервісу з терміналу.

$rosservice call /cartesian_controller/change_pos_efector '{position: x: 0.0, y: 0.19, z: -0.25}'

$ rosservice call/cartesian_controller/change_pos_efector '{position: x: 0.0, y: 0.25, z: -0.28}'

$rosservice call/cartesian_controller/change_pos_efector '{position: x: 0.0, y: 0.26, z: -0.28}'

$rosservice call/cartesian_controller/change_pos_efector '{position: x: 0.0, y: 0.26, z: -0.27}'

$rosservice call/cartesian_controller/change_pos_efector '{position: x: 0.0, y: 0.29, z: -0.30}'

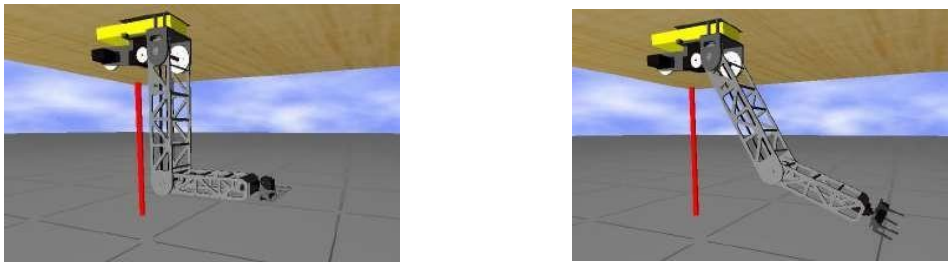
$rosservice call /cartesian_controller/change_pos_efector '{position: x: 0.0, y: 0.31, z: -0.31}'

$ rosservice call cartesian_controller/change_pitch_efector 0.6

$ rosservice call cartesian_controller/effector_hook_change - -1

Експериментальне дослідження керування робота-маніпулятора в декартовому просторі здійснюється за допомогою сервісів зміни положення, орієнтації, відкриття та закриття ефектора, приділяючи особливу увагу внутрішнім перевіркам робочого простору, отримання специфікацій відгуку по Y та Z.

Також здійснюється перевірка залежностей Y та Z змінних від штучних положень та перевірка правильності розв'язку оберненої кінематичної задачі. На рисунку 4.13 показано початкове (y: 0.204, z: -0.264, крок: 0, гак у початковому положенні) і кінцеве (y: 0.29, z: -0.30, крок: 0.6, гак відкритий) положення робота-маніпулятора в симуляторі Gazebo.



а) б)

Рисунок 4.13 – Експериментального дослідження керування робота-маніпулятора в декартовому просторі:

а) початкове положення; б) кінцеве положення

На рисунку 4.14 показані зміни координат Y та Z робота-маніпулятора в декартовому просторі, а на рисунку 4.15 зображено криві відгуку шарнірів 1, 2, 3 та ефекторного шарніру робота-маніпулятора в процесі експериментального дослідження керування робота-маніпулятора в декартовому просторі.

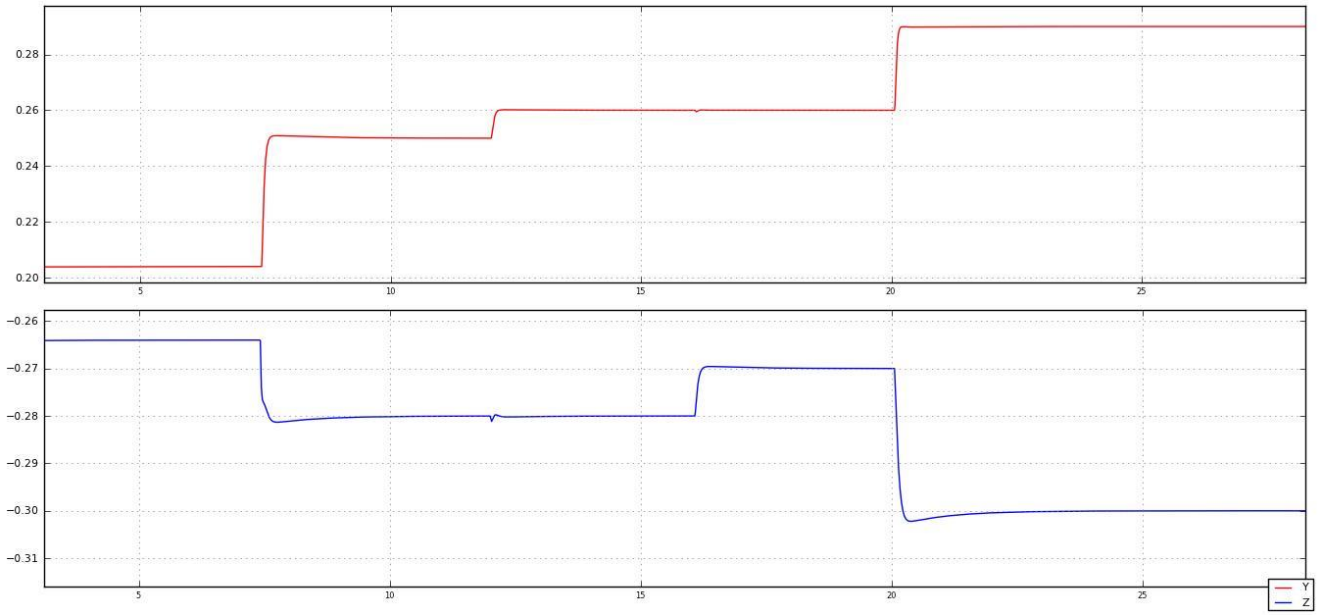


Рисунок 4.14 – Зміни координат Y та Z робота-маніпулятора в декартовому просторі

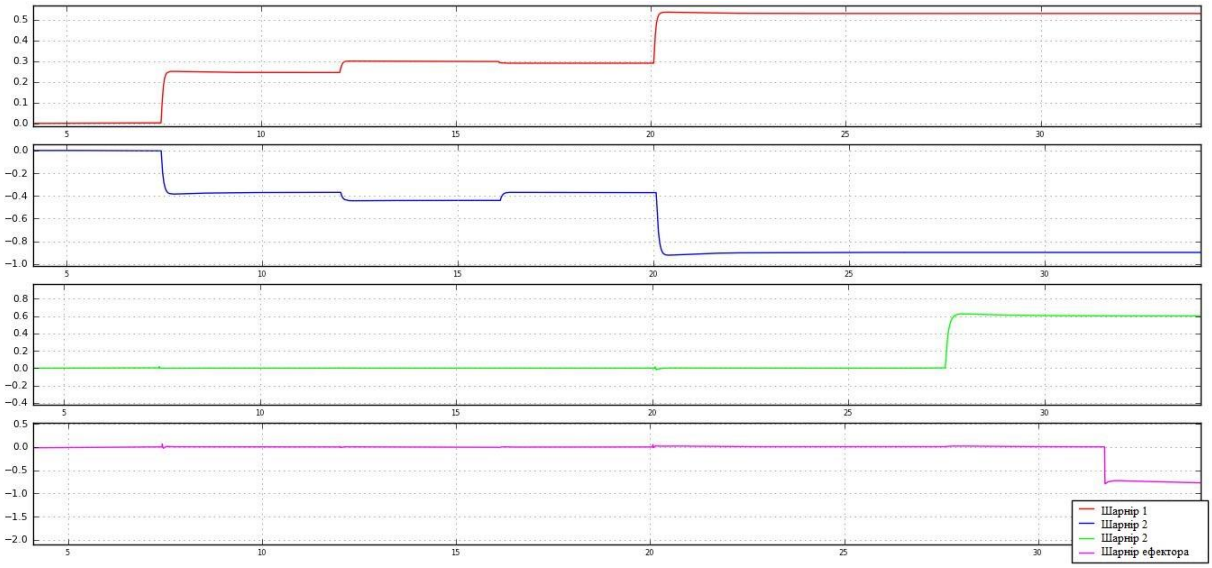


Рисунок 4.15 – Криві відгуку шарнірів 1, 2, 3 та ефекторного шарніру робота-маніпулятора в процесі експериментального дослідження керування робота-маніпулятора в декартовому просторі

Перехідний процес реакції має такі характеристики:

- час наростання 0,16 с.;

- перерегулювання 1,95 %;

- час стабілізації 1,2 с.

Однак це не так для координати Z, тому що через силу тяжіння зменшення Z пов'язане з великим перерегулюванням: 7,30 % вниз і 3,5 % вгору. Також час підйому на 1 см складає 0,16 секунди, а час опускання на 1 см складає 1,80 с.

Під час планування траєкторій маніпулятора ці характеристики враховуються таким чином, що при наближенні до об'єкта, який потрібно взяти, робот-маніпулятор буде рухатися з позиції наближення з невеликим кроком, щоб обмежити ці перерегулювання.

З огляду на положення шарнірів, які були прийняті для отримання бажаних положень по Y і Z, слід підкреслити, хоча і в цьому моделюванні, більший вплив на Y першого шарніра і на Z другого шарніра, тому що ці шарніри є залежними від інших шарнірів, більший вплив можна побачити з точки зору напрямку зміщення і його величини в цих співвідношеннях.

На рисунку 4.16 зображена схема планування руху робота-маніпулятора.

Рисунок 4.16 – Схема планування руху робота-маніпулятора

Планування руху робота-маніпулятора здійснюється блоком, який отримує завдання підняти або опустити об'єкт в певній позиції. Цей блок розрахує траєкторії та команди, які потрібно виконати, а далі використає декартовий контролер та його служби, щоб виконати симуляцію робота-маніпулятора в Gazebo для виконання завдання.

Служба блоку планування /target_service отримує задану позицію та орієнтацію, а тип завдання, який може бути одним з трьох рядків: go, catch і drop.

Тип завдання go еквівалентний використанню служб декартового контролера безпосередньо, без будь-якого планування, надсилаючи центральну точку інструменту в задану позицію та орієнтацію.

Завдання типу pick розраховує траєкторію в декартовому просторі для захоплення об'єкта в заданому положенні та орієнтації так, щоб робот-маніпулятор не зіткнувся з ним, а апертура ефектора була зорієнтована на об'єкт під час захоплення. Падіння типу цілі повинно розраховувати

Завдання типу drop розраховує траєкторію, на якій потрібно залишити об’єкт, щоб, відділяючись від об’єкта, робити це у відповідному напрямку, щоб уникнути зіткнення з ним.

Для обох типів завдань розраховуються траєкторії «від точки до точки». В цих завданнях обчислюватимуться точки, які надсилатимуться одна за одною контролеру, щоб перемістити центральну точку інструменту в задану позицію.

Визначимо два поняття, які використовуються для планування руху робота-маніпулятора.

1. Безпечний об'єм об’єкту.

Якщо об’єкт знаходиться в певному положенні, навколо нього буде створено об'єм, у який ефектор повинен входити тільки орієнтовано. Враховуючи, що робот-маніпулятор не може рухатися по осі X, це буде фактично коло радіусом 30 см. Такого кола радіусом 30 см більш ніж достатньо для того, щоб ефектор не зіткнувся з об’єктом, коли буде знаходитися навколо нього.

2. Позиція орієнтованого наближення (ПОН): це точка в межах безпечного об'єму, в якій напрямок наближення орієнтований на об'єкт, тобто об'єкт перпендикулярний до напрямку наближення, вирівняного з віссю ланки 2.

На рисунку 4.17 показано обидві позиції, а також графічні умови для знаходження цього положення (зображено зіркою).

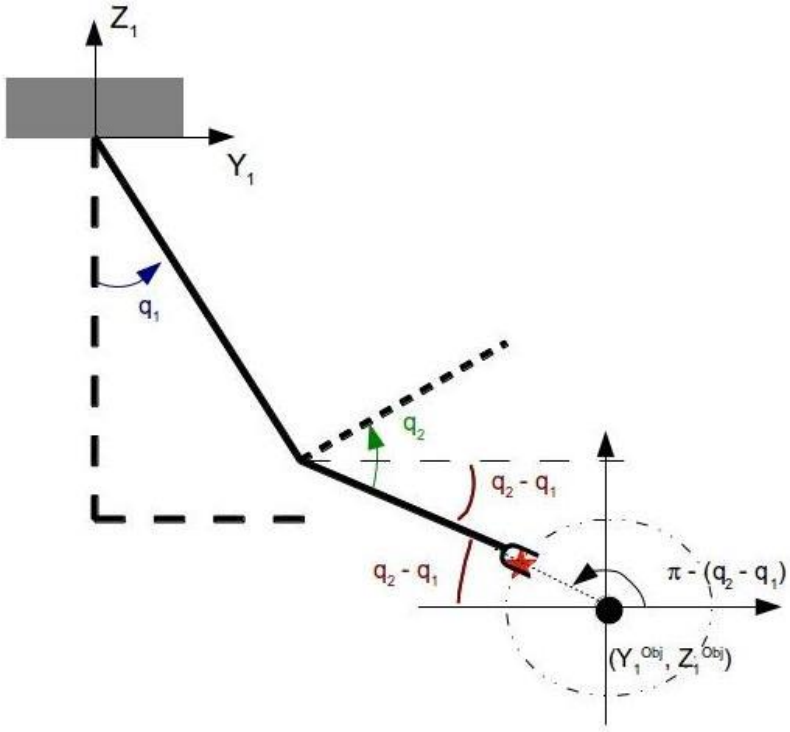


Рисунок 4.17 – Позиція орієнтованого наближення та безпечний об’єм

Якщо точка належить колу і плечу робота-маніпулятора, а також виконується рівняння прямої кінематичної моделі одночасно, то для цих спільних координат вісь ланки 2 утворює з горизонталлю кут, що дорівнює q2 –q1. Математично це еквівалентно системі рівнянь:

$$\begin{cases} Y_1^{\text{ПОН}} = L_1 \sin(q_1) + L_2 \cos(q_2 - q_1) \\ Z_1^{\text{ПОН}} = -L_1 \cos(q_1) + L_2 \sin(q_2 - q_1) \\ Y_1^{\text{ПОН}} = Y_1^{\text{Об'єкта}} + Rcos(\pi + q_1 - q_2) \\ Z_1^{\text{ПОН}} = Z_1^{\text{Об'єкта}} + Rsin(\pi + q_1 - q_2) \end{cases}. \quad (4.1)$$

Перші два рівняння є прямою кінематичною моделлю робота-маніпулятора, а інші два – є рівняннями в параметричних координатах кола для кута q2 – q1.

Для завдання типу захоплення об’єкта існують дві можливі ситуації, але об’єкт завжди буде знаходитися нижче ніж ефектор.

Перша ситуація.

ПОН випереджає ефектор (Y <Y). У цьому випадку передбачається, що немає небезпеки зіткнення будь-якої частини маніпулятора з об'єктом, тому ми можемо спрямувати його безпосередньо до ПОН.

Друга ситуація.

ПОН позаду ефектора (Y > Y). У цьому випадку існує небезпека зіткнення, і у великому відсотку випадків це станеться, якщо ефектор буде направлений прямо на ПОН. Щоб уникнути цього, ефектор проходить через іншу точку, яка називається позицією дотичного зближення (ПДЗ), перед ПОН. Ця позиція визначається як перетин вертикалі, що проходить через ПОН, з дотичною до безпечного об'єму, що проходить через поточну позицію центральної точки інструменту, як показано на рисунку 4.18.

Кут нахилу не відомий з попередньої системи рівнянь. Для його знаходження використовується рівняння дотичної, заданої ЦТІ і точкою дотику, рівняння перпендикуляра до дотичної, що проходить через центр кола безпеки (положення об’єкту) і рівняння самого кола.

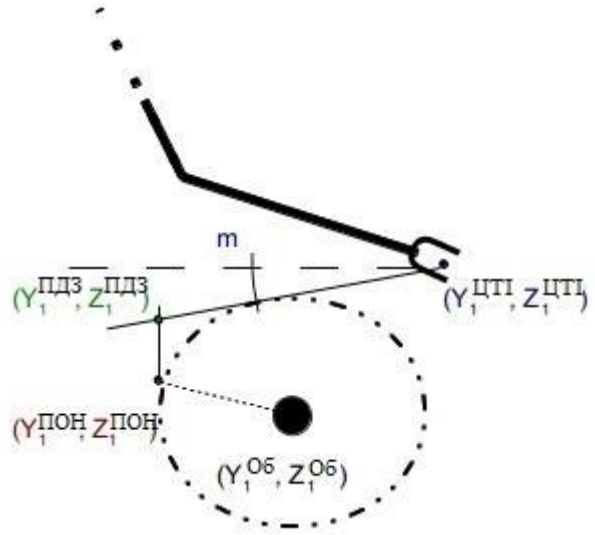


Рисунок 4.18 – Траєкторія для другої ситуації

$$\begin{cases} Z_1^G = m\left(Y_G - Y_{\text{ЦТІ}}\right) + Z_{\text{ЦТІ}} \\ Z_1^G = \frac{1}{m}\left(Y_G - Y_{\text{Об}}\right) + Z_{Obj} \\ \left(Y_G - Y_{\text{Об}}\right)^2 + \left(Z_G - Z_{\text{Об}}\right)^2 = R \end{cases} \quad (4.2)$$

Якщо будь-яка з отриманих або розрахованих позицій виходить за межі робочої області, планування переривається і надсилається повідомлення про помилку.

У цьому випадку передбачається, що брусок був раніше захоплений попередньою дією, і тепер завдання полягає в тому, щоб відпустити його в потрібному положенні та орієнтації.

Єдиним обмеженням, окрім очевидних, що стосуються приналежності позицій до робочого простору, є те, що ефектор повинен залишати робочий простір орієнтованим чином, щоб не зіткнутися з ним при русі від нього, так само, як він повинен увійти в нього, щоб взяти об'єкт.

Тому в цьому типі мішеней знову будуть використовуватися поняття безпечного об'єму навколо цільової позиції і орієнтованої позиції наближення, яка в даному випадку буде позицією, в яку слід відправити ефектор, щоб відійти від об'єкта.

Була проведена симуляція, в якій маніпулятору було запропоновано взяти брусок в одній позиції, повернутися в початкову позицію і, нарешті, відпустити брусок в іншій позиції з ненульовою орієнтацією.

Запуск бруска діаметром 8 мм (модель bar.urdf, створена в пакеті маніпулятора) в цільову позицію, в Gazebo, додається до програми планування для перевірки можливих зіткнень маніпулятора з ним на всьому шляху його руху і захоплення його ефектором. Крім того, створено сервіс /reset, який можна викликати в будь-який момент, без аргументів, щоб повернути маніпулятор у початковий стан і видалити всі запущені планки.

Для запуску Gazebo разом з моделлю маніпулятора, декартовим вузлом керування, вузлом планування, rxplot для зчитування TCP- позиції та вузлом image_view для камери, встановленої на ефекторі, запускається наступний лаунчер:

$roslaunch man_table_gazebo_planning.launch

Під час симуляції були отримані різні дані по кожному терміналу з розрахованими позиціями, правильними перевірками робочого простору і

викликами відповідних служб декартового контролера, так що коректна робота планувальника перевірялася постійно.

Також були проведені додаткові симуляції, щоб показати правильність функціонування планувальника в робочому просторі маніпулятора, а також коригування положення ТКП (до описаного положення ТКП по всій пам'яті) з метою забезпечення беззіткненого кінцевого переміщення навколо цілі.

Процес планування руху робота-маніпулятора, який включає симуляцію в середовищі Gazebo та використання декартового контролера, можна розділити на кілька ключових етапів:

1. Формулювання завдання.

Завдання для робота-маніпулятора зазвичай задається у вигляді кінцевої мети: наприклад, підняти об'єкт з певної позиції (початкової точки) та перемістити його в іншу позицію (кінцеву точку). Ці точки визначаються в декартовій системі координат (XYZ і орієнтація у вигляді Roll, Pitch, Yaw).

Завдання може бути сформульоване користувачем або отримане через зовнішній сенсорний пристрій, наприклад камеру.

2. Розрахунок траєкторії (планування руху).

Для планування руху використовується бібліотека MoveIt, яка інтегрується з OCP. Основні етапи цього процесу.

1. Побудова моделі робота.

Робот описується за допомогою файлів URDF або SRDF, які містять інформацію про кінематику, динаміку та геометрію маніпулятора.

2. Генерація траєкторії.

MoveIt розраховує плавну траєкторію для переходу від початкової до кінцевої точки. Використовуються різні алгоритми планування (наприклад, RRT або PRM).

Алгоритми враховують кінематичні обмеження (швидкість, прискорення), а також можливі перешкоди в робочому просторі.

3. Інтеграція декартового контролера.

Декартовий контролер забезпечує точний рух маніпулятора уздовж запланованої траєкторії в декартовій системі координат.

Контролер працює за наступним принципом:

1. Отримання команд траєкторії.

Контролер отримує траєкторію у вигляді послідовності точок, які включають позицію та орієнтацію робочого органа (end effector).

2. Розрахунок інверсної кінематики.

Контролер розраховує необхідні кути суглобів, щоб досягти кожної точки траєкторії. Інверсна кінематика враховує обмеження суглобів і уникає сингулярностей.

3. Формування сигналів управління.

Генеруються команди для приводів маніпулятора для виконання плавного руху.

4. Симуляція в Gazebo.

Симуляція виконується для перевірки та відлагодження руху робота віртуально перед його застосуванням на фізичному обладнанні.

Основні кроки.

1. Імпорт моделі робота.

URDF-файл імпортується у середовище Gazebo.

2. Ініціалізація фізики.

Задаються параметри фізичного моделювання, такі як гравітація, тертя та інерція.

3. Запуск контролера.

Декартовий контролер підключається до симуляції через OCP-топіки та служби.

4. Моніторинг та валідація.

Візуально та за допомогою інструментів Gazebo перевіряється, чи виконується рух коректно.Аналізуються можливі проблеми: наприклад, зіткнення з об'єктами чи неузгодженість траєкторії.

5. Виконання руху на фізичному роботі.

Після успішної симуляції траєкторія передається на реальний маніпулятор. Враховуються параметри калібрування робота, які можуть відрізнятися від моделі в симуляції. Декартовий контролер надсилає команди апаратним приводам через інтерфейси низького рівня.

6. Зворотний зв’язок і корекція.

У процесі виконання руху контролер отримує зворотний зв'язок від сенсорів (наприклад, енкодерів або камер). У випадку відхилень від траєкторії виконується корекція руху в реальному часі.

Цей підхід дозволяє не тільки забезпечити високу точність та надійність керування, але й мінімізувати ризики для обладнання та середовища завдяки попередньому тестуванню в симуляції.

4.3 Висновки до четвертого розділу

1. Особливістю експериментального дослідження удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів є використання симуляції для перевірки правильності роботи контролерів.

2. В процесі симуляції використовувалоя керування шарнірами, в яких будуть проаналізовані реакції кожного шарніра в положенні, швидкості і зусиллі, а також вплив кожного шарніра на інші.

3. Виконано симуляцію керування в декартовому просторі, де проаналізуємо реакції по Y, Z, а також перевіримо, чи дотримується робочий простір.

4. Усі симуляції виконувалися для моделі man_mesa.urdf ізапускатимуться з пакета manipulator, де знаходиться модель робота-маніпулятора. Для запуску кожного типу контролера достатньо включити до його пакету створений запускник.

5. Щодо впливу першого шарніра на інші шарніри, то варто виділити вплив на ефекторний шарнір, який є найбільшим, оскільки він знаходиться на

найбільшій відстані від осі обертання, але таке моделювання не зовсім коректне, оскільки, як правило, ці шарніри мають гальмо на серводвигуні, який не моделюється. Вплив на інші шарніри є досить незначним.

6. Експериментальне дослідження керування робота-маніпулятора в декартовому просторі здійснюється за допомогою сервісів зміни положення, орієнтації, відкриття та закриття ефектора, приділяючи особливу увагу внутрішнім перевіркам робочого простору, отримання специфікацій відгуку по Y та Z.

ВИСНОВКИ

1. Операційна система роботів - це фреймворк для написання програмного забезпечення для роботів. До складу операційної системи роботів входить набір інструментів, бібліотек і угод, спрямованих на спрощення завдання створення поведінки для складних і надійних роботів на різноманітних роботизованих платформах.

2. Механічно робот-маніпулятор - це кінематичне коло, яке складається з ланок, з'єднаних шарнірами з одним або декількома ступенями свободи. У більшості роботів-маніпуляторів це кінематичне коло є розімкненим, починаючи з першої ланки або нерухомого базового корпусу і закінчуючи іншою, яка називається маніпулятором або маніпуляторним кінцем.

3. Незалежне керування шарнірами 1, 2 і 3 відбувається за базовою схемою керування, що складається з блоку, в якому перевіряється, чи бажане положення шарніра знаходиться в межах шарніра чи ні. Далі обчислюється похибка положення шарніра, обчислюється зусилля, яким повинен керувати ПІД-регулятор і надсилається на симулятор.

4. Експериментальне дослідження керування робота-маніпулятора в декартовому просторі здійснюється за допомогою сервісів зміни положення, орієнтації, відкриття та закриття ефектора, приділяючи особливу увагу внутрішнім перевіркам робочого простору, отримання специфікацій відгуку по Y та Z.

5. Планування руху робота-маніпулятора здійснюється блоком, який отримує завдання підняти або опустити об'єкт в певній позиції. Цей блок розрахує траєкторії та команди, які потрібно виконати, а далі використає декартовий контролер та його служби, щоб виконати симуляцію робота-маніпулятора в Gazebo для виконання завдання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ОСР - Операційна система роботів. Режим доступу: https://ros.org.

2. Programming Robots with ROS: A Practical Introduction to the Robot Operating System / Morgan Quigley, Brian Gerkey, William D. Smart. - O'Reilly Media, Inc., 2015. – 448 с.

3. Learning ROS for Robotics Programming / Aaron Martinez, Aaron Romero, Enrique Fernández. - Packt Publishing, 2013. – 332 с.

4 A review of mobile robots: Concepts, methods, theoretical framework, and applications / Francisco Rubio, Francisco Valero and Carlos Llopis-Albert. - International Journal of Advanced Robotic Systems, March-April 2019, P. 1-22.

5. Робот-черепаха. Режим доступу: https://clearpathrobotics.com/turtlebot-4/.

6. Робот-прибиральник. Режим доступу: https://robots.ros.org/kobuki/.

7. Лазер Hokuyo UST-10LX. Режим доступу: https://www.hokuyo-aut.jp/search/single.php?serial=167.

8. Камера Astra. Режим доступу: https://www.orbbec.com/products/structured-light-camera/astra-series/.

9. RViz: a toolkit for real domain data visualization / H.R. Kam, S.H. Lee, T. Park. - Telecommunication Systems, 2015, P. 337–345.

10. Forward kinematics algorithm in dual quaternion space based on Denavit-Hartenberg convention / N. Zivkovic, J. Vidakovic, M. Lazarevic. - Applied Engineering Letters, 2023, Vol.8, No. 2, P. 52-59.

11. Derivation and analysis of a dynamic model of a robotic manipulator on a moving base / C.M. Wronka, M.W. Dunnigan. - Robotics and Autonomous Systems, 2011, Volume 59, Issue 10, P. 758-769.

12. Технічні характеристики серводвигуни компанії Hitec. Режим доступу: https://servodatabase.com/servos/hitec

13. Towards MRI-based autonomous robotic US acquisitions: a first feasibility study. / Christoph Hennersperger et al. - In: IEEE transactions on medical imaging 36.2 (2017), P. 538–548.

14., Dynamic modelling simulation of a four legged jumping robot with compliant legs / Ganesh Kumar K. Pushparaj Mani Pathak. - Robotics and Autonomous Systems Volume 61, Issue 3, March 2013, P. 221-228.

15. Real-time Model Predictive Control for Versatile Dynamic Motions in Quadrupedal Robots / Yanran Ding, Abhishek Pandala, and Hae-Won Park. - International Conference on Robotics and Automation (ICRA), Palais des congres de Montreal, Montreal, Canada, May 2019, P. 20-24.

16. Analysis of a machining industrial robot / bele, E., Kulok, M., & Weigold, M. 10th International Scientific Conference on Production Engineering, II, Lumbarda, Croatia, 2019, P. 1-11.

17. Industrial robotics: Opportunities for manufacturers of end effectors / Aljarboua, Z., Santhanam, N., Teulieres, M., Thomsen, J., & Tilley, J. - Industrial robotics: Opportunities for manufacturers of end effectors. McKinsey & Company. Retrieved November 19, 2019.

18. Regulatory challenges of robotics: some guidelines for addressing legal and ethical issues / Leenes, R., Palmerini, E., Koops, B.-J., Bertolini, A., Salvini, P., & Lucivero, F. - Law, Innovation and Technology, 9(1), 2017, P. 1-44.

19. Reconfigurable robotic machining system controlled and programmed in a machine tool manner / Milutinovic, D., Glavonjic, M., Slavkovic, N., Dimic, Z., Zivanovic, S., Kokotovic, B., & Tanovic, L. - International Journal of Advanced Manufacturing Technology 53, P. 1217-1229.

20. Off-line programming of an industrial robot for manufacturing / Mitsi, S., Bouzakis, K.-D., Mansour, G., Sagris, D., & Maliaris, G. - International Journal of Advanced Manufacturing Technology, 26(3), P. 262-267.

21. Direct off-line robot programming via a common CAD package / Neto, P., & Mendes, N. - Robotics and Autonomous Systems, 61(8), P. 896-910.

22. Augmented reality based teaching pendant for industria lrobot / Abbas S.M., Hassan S. and Yun J. - International Conferenceon Control, Automation and Systems. 2012, P. 2210-2213.

23. A Student Project using Robotic Operating System (ROS) for Undergraduate Research / S. A. Wilkerson, J. Forsyth, C. Sperbeck, M. Jones, and P. D. Lynn, - ASEE Annual Conference & Exposition, Columbus, Ohio, June 2017.

24. Low-Cost Robot Arms for the Robotic Operating System (ROS) and MoveIt / A. Yousuf, W. Lehman, M. A. Mustafa, and M. M. Hayder. - ASEE Annual Conference & Exposition, New Orleans, Louisiana, June 2016/

25. ROS-based Control of a Manipulator Arm for Balancing a Ball on a Plate / K.A. Khan and J. Ryu. - ASEE Annual Conference & Exposition, Columbus, Ohio, June 2017.

26. Teaching Robotics with ROS / S. Schiffer, et al (ed.). - European Robotics Forum 2018 Workshop Proceedings of the Workshop on Teaching Robotics with ROS (held at ERF 2018), Tampere, Finland, March 15th, 2018.

27. Introductory Mobile Robotics and Computer Vision Laboratories Using ROS and MATLAB / R. L. Avanzato and C. G. Wilcox. - ASEE Annual Conference & Exposition, Salt Lake City, Utah. June 2018.

28. Generalized Matlab/ROS/Robotic Platform Framework for Teaching Robotics / Rosillo, N. Montés, J. P. Alves, and N. M. - Robotics in Education. Advances in Intelligent Systems and Computing, vol 1023. Springer, Cham.

29. The State of Robotics Education: Proposed Goals for Positively Transforming Robotics Education at Postsecondary Institutions / J. M. Esposito. - IEEE Robotics &Automation Magazine, vol. 24, no. 3, 2017, P. 157-164.

30. ROS: an open-source Robot Operating System / Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew N. - IEEE International Conference on Robotics and Automation Workshop on Open Source Software, 2009.

31. ROS control: A generic and simple control framework for ROS / Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep,Adolfo Rodr´ıguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Ludtke. - Journal of Open Source Software, 2(20):456, 2017.

32. Robust navigation in an indoor office environment / Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. - IEEE International Conference on Robotics and Automation, pages 300–307, 2010.

33. Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study / David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. - Journal of Software Engineering for Robotics, 5(1):3–16, 2014.

34. Deterministic, asynchronous message driven task execution with ros / Brian Cairl. - Open Robotics, September 2018.

35. Steve Macenski, Francisco Mart´ın, Ruffin White, and Jonatan Gines Clavero. The Marathon 2: A Navigation System. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020.

36. A robust layered control system for a mobile robot / Rodney Brooks. - IEEE journal on robotics and automation, 2(1):14–23, 2020.

37. Lightweight Communications and Marshalling / Albert S. Huang, Edwin Olson, and David C. Moore. IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4057–4062, 2010.

38. Procedurally provisioned access control for robotic systems / Ruffin White, Gianluca Caiazza, Henrik Christensen, and Agostino Cortesi. - IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018.

39. Dumlu, Ahmet, et al. “A comparative study of two model-based control techniques for the industrial manipulator.” Robotica 35.10 (2017): 2036-2055.

40. Durmus, Burhanettin, et al. “A study on industrial robotic manipulator model using model based predictive controls.” Journal of intelligent manufacturing 20 (2009): 233-241.

41. Fan, Feng-Lei, et al. “On interpretability of artificial neural networks: A survey.”IEEE Transactions on Radiation and Plasma Medical Sciences, (2021): 741-760.

42. Schmidt, Michael, and Hod Lipson. “Distilling free-form natural laws from experimental data.”science324.5923 (2009): 81-85.

43. Chapelle, Frederic, and Philippe Bidaud. “Closed form solutions for inverse kinematics approximation of general 6R manipulators.”Mechanism and machine theory (2004): 323-338.

44. Zhang, Zhixin, and Zhiyong Chen. “Modeling and Control of Robotic Manipulators Based on Symbolic Regression.”IEEE Transactions on Neural Networks and Learning Systems (2021).

45. Danai, Kourosh, and William G. La Cava. “Controller design by symbolic regression.”Mechanical Systems and Signal Processing (2021): 107348.

46. Brunton, Steven L., Joshua L. Proctor, and J. Nathan Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems.”Proceedings of the national academy of sciences (2016): 3932-3937.

47. Huang, Zhilong, et al. “Data-driven automated discovery of variational laws hidden in physical systems.”Journal of the Mechanics and Physics of Solids137 (2020): 103871.

48. Chu, Hoang K., and Mitsuhiro Hayashibe. “Discovering interpretable dynamics by sparsity promotion on energy and the Lagrangian.”IEEE Robotics and Automation Letters (2020): 2154-2160.

49. Askarinejad, S. Emad, et al. “Data-driven identification of the Jacobian matrix of a 2-DoF spherical parallel manipulator.”2019 7th International Conference on Robotics and Mechatronics (ICRoM). IEEE, 2019.

50. Yu, Chenglong, et al. “A fast robotic arm gravity compensation updating approach for industrial application using sparse selection and reconstruction.”Robotics and Autonomous Systems (2022).

51. Kaheman, Kadierdan, J. Nathan Kutz, and Steven L. Brunton. “SINDyPI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics.”Proceedings of the Royal Society (2020).

52. Abdulrahman A.A.Emhemed, Rosbi Bin Mamat, Modelling and Simulation for Industrial DC Motor Using Intelligent Control, Procedia Engineering 41 (2012) 420-425.

53.. Amir Khazaee, Hosein Abootorabi Zarchi, Gholamreza Arab Markadeh, Loss model based efficiency optimized control of brushless DC motor drive, ISA Transactions 86 (2019) 238-248.

54. F. S. Ahmed, S. Laghrouche, M. El Bagdouri, Pancake Type DC Limited Angle Torque Motor: Modeling and Identification, Application to Automobile Air Path Actuators, 6th IFAC Symposium Advances in Automotive Control Munich, Germany, July 12-14, 2010, 431-436.

55. Fayez S. Ahmed, Salah Laghrouche, Mohammed El Bagdouri, Analysis, modeling, identification and control of pancake DC torque motors: Application to automobile air path actuators, Mechatronics 22 (2012), 195-212.

56. M. Ruderman, J. Krettek, F. Hoffmann, T. Bertram, Optimal State Space Control of DC Motor, Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea, July 6-11, 2008, IFAC Proceedings, Volume 41, Issue 2, 2008, Pages 5796-5801.

57. Masayoshi Tomizuka, Robust digital motion controllers for mechanical systems, Robotics and Autonomous Systems, 1996 Elsevier, 19 (1996) 143-149.

58. Mikhov M., Mathematical Modeling and Dynamic Simulation of a Class of Drive Systems with Permanent Magnet Synchronous Motors, Applied and Computational Mechanics, Vol. 3, No. 2, pp. 331-338, Czech Republic, 2009.

59. Radoslav Vasilev, Dimitar Dimitrov, Autonomous Mobile Robot for Research and Development of a Perceptual Anchoring System, Proceedings of the Technical University - Sofia, v. 62, book 2, 2012, pp. 313-322.

60. Guergov S., Beevski L., Study of the Capacity of a Fanuc M430i-A/4FH Robot to Perform Technological Operations, In Proceedings of Scientific XII International Congress "Machines, Technologies, Materials", Volume 3 (2015), pp.12-14.

61. West, C., Montazeri, A., Monk, S., and Taylor, C. A genetic algorithm approach for parameter optimization of a 7DOF robotic manipulator", IFAC-PapersOnLine, 49 (12), pp. 1261-1266 (2016).

62. A. M. Mustafa and A. Al-Saif, “Modeling, Simulation and Control of 2-R Robot,” Global Journal of Researchers in Engineering, vol. 14, pp. 49–54, 2014.

63. J. Ohri, D. R. Vyas, P. N. Topno, “Comparison of Robustness of PID Control and Sliding Mode Control of Robotic Manipulator” International Journal of Computer Applications, pp. 5–10, 2011.

64. M. Gopal, Digital Control and State Variable Methods, 3rd edition, Tata McGrawhill, pp. 575–581, 603-607, 2009.

65. Bandyopadhyay, S. Janardhanan, V. Sreeram and E. Abera, “Sliding Mode Control Design via Reduced Order Model Approach,” International Journal on Automation and Computing, pp. 329–334, October 2007.

66. O. O. Obadina, M. Thaha, K. Althoefer, and M. H. Shaheed, “A Modified Computed Torque Control Approach for a Master-Slave Robot Manipulator System,” in Towards Autonomous Robotic Systems. TAROS 2018. Lecture Notes in Computer Science, vol. vol 10965, M. Giuliani, T. Assaf, and M. Giannaccini, Eds. Springer, Cham, 2018, pp. 28–39.

67. John J. Craig, introduction to robotics Mechanics and Control, Pearson Education International, Third Edition, 2005, pp. 67.

68. Abdel-Nasser Sharkawy, Panagiotis N. Koustoumpardis “Dynamics and Computed-Torque Control of a 2- DOF manipulator: Mathematical Analysis” International Journal of Advanced Science and Technology Vol. 28, No. 12, (2019), pp. 201–212.

69. Kamal M.H. Raheem, Ameer Najm Najaf, “Simulation 3-DOF RRR Robotic Manipulator under PID Controller”, Journal of Engineering and Applied Sciences 15 (2): 410–414, 2020.

70. Islam and X. P. Liu, “Robust Sliding Mode Control for Robotic Manipulators,” IEEE transaction on Industrial Electronics, vol. 58, no. 6, pp. 2444–2453, June 2011.

71. S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. H. Santos, and E. Pekkeriet,“Agricultural Robotics for Field Operations,” Sensors (Basel)., vol. 20, no. 9, May 2020.

72. H. Zangl, L. M. Faller, and W. Granig, “Optimal design of angular position sensors,” COMPEL - Int. J. Comput. Math. Electr. Electron. Eng., vol. 36, no. 5, pp. 1372–1385, 2017.

73. S. S. Mehta and T. F. Burks, “Multi-camera Fruit Localization in Robotic Harvesting,” IFAC-PapersOnLine, vol. 49, no. 16, pp. 90–95, Jan. 2016.

74. J. Li, M. Karkee, Q. Zhang, K. Xiao, and T. Feng, “Characterizing apple picking patterns for robotic harvesting,” Comput. Electron. Agric., vol. 127, pp. 633–640, Sep. 2016.

75. M. R. Woodside, J. Fischer, P. Bazzoli, D. A. Bristow, and R. G. Landers, “A kinematic error controller for real-time kinematic error correction of industrial robots,” Procedia Manuf., vol. 53, pp. 705–715, 2021.

Додаток А

Стаття у фаховому журналі (подана до редакції журналу «Вимірювальна та обчислювальна техніка в технологічних процесах»)

УДК 681.5
DOI:

МАРТИНЮК Валерій
Хмельницький національний університет
ORCID ID: 0000-0001-5758-4244
e-mail: martynyuk.valeriy@gmail.com
СЕЛЬСЬКИЙ Андрій
Хмельницький національний університет
ORCID ID: 0000-0002-7373-0472
КОВАЛЬ Михайло
Хмельницький національний університет
e-mail: mops85270@gmail.com

УДОСКОНАЛЕНИЙ МЕТОД КЕРУВАННЯ РОБОТОМ-МАНІПУЛЯТОРОМ НА ОСНОВІ ОПЕРАЦІЙНОЇ СИСТЕМИ РОБОТІВ

У статті розроблено імітаційну модель робота маніпулятора з трьома ступенями свободи. Отримані графіки, які відображають вектор переміщення кінцевого суглоба робота маніпулятора можна зробити висновок, що координата z змінюється у часі. Це пояснюється тим, що використовується робот маніпулятор з трьома ступенями свободи, який може рухатися по трьох координатах x, y та z.

Проведено 3-D візуалізацію руху робота маніпулятора з трьома ступенями свободи дозволяє проводити дослідження руху суглобів робота маніпулятора, визначати їх координати, а також налаштовувати ПД-регулятори для кожного суглоба. Це дозволяє проводити динамічне моделювання руху робота маніпулятора для досягнення максимально можливої швидкості руху суглобів і максимально можливої точності позиціонування робочого органу робота маніпулятора.

Ключові слова: *робот-маніпулятор, імітаційна модель, 3-D візуалізацію руху робота маніпулятора з трьома ступенями свободи.*

MARTYNYUK Valeriy, SELSKYI Andrii, KOVAL Mykhailo
Khmelnitsky national university, Ukraine

AN IMPROVED METHOD FOR CONTROLLING A ROBOT MANIPULATOR BASED ON A ROBOT OPERATING SYSTEM

The article developed a simulation model of a robot manipulator with three degrees of freedom. From the obtained graphs, which display the vector of movement of the end joint of the manipulator robot, it can be concluded that the z coordinate changes over time. This is explained by the fact that a robot manipulator with three degrees of freedom is used, which can move along three coordinates x, y and z.

A 3-D visualization of the movement of the robot manipulator with three degrees of freedom has been carried out, which allows you to study the movement of the joints of the manipulator robot, determine their coordinates, as well as adjust the PD controllers for each joint. This allows dynamic modeling of the movement of the manipulator robot to achieve the maximum possible speed of movement of the joints and the maximum possible accuracy of positioning of the working organ of the manipulator robot.

Keywords: *robot manipulator, simulation model, 3-D visualization of the motion of a robot manipulator with three degrees of freedom.*

Постановка проблеми

Операційна система роботів (ОСР) - це фреймворк для написання програмного забезпечення для роботів [1]. До складу операційної системи роботів входить набір інструментів, бібліотек і угод, спрямованих на спрощення завдання створення поведінки для складних і надійних роботів на різноманітних роботизованих платформах.

До основних особливостей операційної системи роботів відносяться наступні характеристики [2].

1. Великий набір контролерів, які дозволяють зчитувати дані з датчиків і надсилати команди двигунам та іншим виконавчим механізмам в абстрактному і чітко визначеному форматі. Широке розмаїття популярного апаратного забезпечення, включаючи зростаючу кількість комерційно доступних робототехнічних систем.

2. Великий набір фундаментальних робототехнічних алгоритмів для побудови карт світів, навігації по ним, представлення та інтерпретації сенсорних даних, планування рухів, маніпулювання об'єктами та багато іншого.

3. Потужна обчислювальна інфраструктура, яка дозволяє переміщувати дані, з'єднувати різні компоненти складної роботизованої системи та впроваджувати власні алгоритми. ОСР за своєю суттю є розподіленою і дозволяє без проблем розподіляти робоче навантаження між кількома комп'ютерами.

4. Великий набір інструментів, які дозволяють легко візуалізувати стан робота та алгоритму, налагоджувати помилкову поведінку та записувати дані з датчиків.

5. Операційна система роботів містить у собі великий набір ресурсів, таких як документація з багатьох аспектів фреймворку, сайт питань і відповідей, де можна звернутися за допомогою і поділитися своїми знаннями, а також спільноту користувачів і розробників.

Архітектура операційної системи роботів була розроблена і розділена на три розділи або рівні [3].

1. Рівень файлової системи.
2. Рівень комп'ютерної графіки.
3. Рівень спільноти.

Операційна система роботів є відкритою платформою, яка дозволяє використовувати модулі (пакети) для розв'язання широкого спектра завдань, пов'язаних із робототехнікою. Це особливо актуально для складних маніпуляторів, де потрібно інтегрувати сенсори, планувальники рухів та виконавчі системи.

Аналіз останніх джерел

В даний час відомий широкий спектр роботів у різних категоріях, що використовують фреймворк OCP. Операційна система роботів підтримує повітряні роботи (Gapter), наземні роботи (Turtlebot і Pepper), роботи-маніпулятори (Fanuc і ABB), і морські роботи (Clearpath Heron USV).

Також операційна система роботів підтримує велику кількість компонентів, а саме маніпулятори, камери, 3D-сенсори та лазери, які входять до складу роботизованих систем.

В процесі розвитку робототехніки виникли певні труднощі з написанням сумісного програмного забезпечення для різних типів роботів. Якщо апаратне забезпечення значно відрізняється, повторне використання коду не є тривіальним.

Зіткнувшись з цією тенденцією, OCP надає бібліотеки та інструменти, які допомагають розробникам програмного забезпечення створювати робототехнічні програми. Основними завданнями OCP є абстрагування низькорівневого управління апаратними пристроями, реалізація часто використовуваних функцій, передача повідомлень між процесами та управління пакетами.

Операційна система роботів надає всі частини програмної системи робота, які в іншому випадку потрібно розробляти самостійно. Це дозволяє зосередитися на тих частинах системи, які мають важливе значення, не турбуючись про ті частини, які є відомими.

За останні десятиліття мобільна робототехніка досягла неймовірних успіхів. Такі теми, як автономна навігація, сприйняття, реактивність, картографування, уникнення перешкод і самолокалізація, були глибоко вивчені в останні роки і знайшли застосування в промисловості, транспорті, військовій сфері та сфері безпеки. Крім того, деякі мобільні роботи стали споживчими товарами для розваг або для виконання повсякденних побутових завдань, наприклад для прибирання підлоги.

Мобільний робот [4] - це автоматична машина, здатна переміщатися в будь-якому середовищі. Мобільні роботи мають можливість переміщатися в навколишньому середовищі і не прив'язані до певного місця. Платформа робот-черепаха (англійською мовою Turtlebot) [5] - це недорогий мобільний робот з відкритим програмним забезпеченням

Виклад основного матеріалу

На кафедрі автоматизації, комп'ютерно-інтегрованих технологій та робототехніки Хмельницького національного університету в лабораторії 4-319 розміщено робот-маніпулятор Dobot Magician, який зображено на рис. 1.

Рис. 1. Робот-маніпулятор Dobot Magician (Добот Чарівник)

Робот-маніпулятор Dobot Magician - це комерційний робот-маніпулятор із 3 ступенями свободи (СС), який має численні привабливі функції завдяки гібридній послідовній і паралельній конфігурації.

Однією з його головних характеристик робота-маніпулятора Dobot Magician є точність, завдяки тому, що він використовує зв'язок із чотирма балками для активації обертання кожної ланки.

Крім того, робот-маніпулятор Dobot Magician використовує механізм паралелограма, щоб гарантувати постійну орієнтацію кінцевого ефектора. Тривимірна модель робота-маніпулятора Dobot Magician представлена на рис. 2.

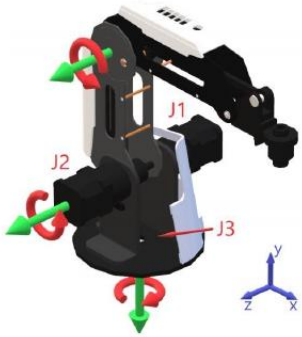


Рис. 2. Тривимірна модель робота-маніпулятора Dobot Magician

Як видно, шарніри J1 та J2 (q1 і q2) працюють в одному напрямку, отже, спрацьовування цих з'єднань і їхній вплив на кінематику робота можуть бути проаналізовані за допомогою плоскої моделі.

З іншого боку, шарнір J3 обертає робота по осі y, що змінює орієнтацію робота-маніпулятора. Оскільки це обертання є перпендикулярним до інших активованих шарнірів, це с обертання не враховується в процесі аналізу. На рис. 3 у розрізі показано механізм всередині корпусу робота-маніпулятора Dobot Magician.

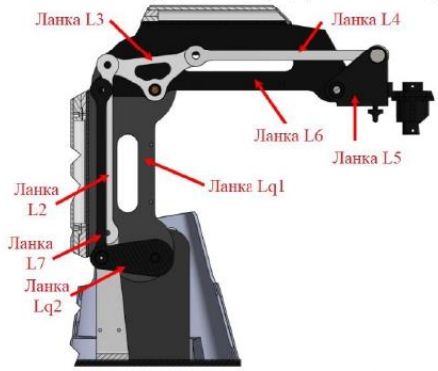


Рис. 3. Механізм всередині у розрізі корпусу робота-маніпулятора Dobot Magician

На рис. 4. зображено 3D імітаційну модель робота-маніпулятора Dobot Magician.

Рис. 4. 3D імітаційна модель робота-маніпулятора Dobot Magician

Створення 3D-моделі робота-маніпулятора Dobot Magician в OCP здійснюється за допомогою використання URDF фалу (уніфікований формат опису робота), який є форматом XML, що дозволяє описувати фізичні властивості робота. Для візуалізації робота-маніпулятора Dobot Magician будемо використовувати інструмент OCP Rviz. Rviz дозволяє в реальному часі візуалізувати на 3D-сцені всі компоненти робототехнічної системи - системи координат, частини, що рухаються, значення вимірювальних сигналів з датчиків, зображення з камер.

Для побудови 3D моделі робота-маніпулятора Dobot Magician будемо використовувати основа та чотири ланки: тулуб, плече, передпліччя та кисть, які з'єднуються між собою чотирма суглобами: тазостегновим, плечовим, ліктьовим та зап'ястковим.

Для побудови 3D моделі робота-маніпулятора Dobot Magician будемо використовувати уніфікований формат опису робота (URDF) - це формат файлу XML, який використовується в ОСР для опису всіх елементів робота. URDF файл робота-маніпулятора Dobot Magician розпочинається першим тегом, який задає ім'я робота.

Наступні теги у URDF файлі робота-маніпулятора Dobot Magician визначають кольори, які використовуються для 3D моделі робота-маніпулятора Dobot Magician. Створення 3D-моделі робота-маніпулятора Dobot Magician почнемо із основи, яка розміщується на робочій поверхні і зображена на рис. 5

Рис. 5. 3D модель основи робота-маніпулятора Dobot Magician

Перша ланка робота-маніпулятора Dobot Magician після основи – це тулуб, який повертає робота з боку в бік і зображений на рис. 6.

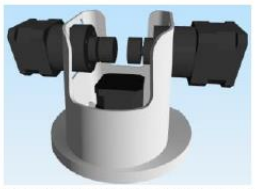


Рис. 6. 3D модель першої ланки робота-маніпулятора Dobot Magician

Друга ланка робота-маніпулятора Dobot Magician після тулуба – це плече, яке повертає руку до верху та до низу і зображене на рис. 7.

Рис. 7. 3D модель другої ланки робота-маніпулятора Dobot Magician

Третя ланка робота-маніпулятора Dobot Magician після плеча – це передпліччя, яке з'єднує плече із наступною ланкою – кистю і зображене на рис. 8.

Рис. 8. 3D модель третьої ланки робота-маніпулятора Dobot Magician

Четверта ланка робота-маніпулятора Dobot Magician після передпліччя – це кисть, яка з'єднує передпліччя із кінцевим ефектором і зображена на рис. 9.

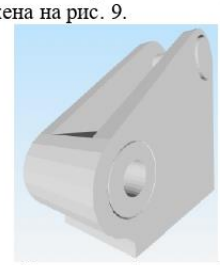


Рис. 9. 3D модель четвертої ланки робота-маніпулятора Dobot Magician

Висновки

Найбільш часто використовуваним типом роботизованих рук у виробництві є серійні маніпулятори. Такі маніпулятори складаються з серії ланок, як правило, твердих тіл, з'єднаних разом і приводених в дію двигунами, що розміщені на з'єднаннях.

Виконано моделювання робота-маніпулятора Dobot Magician виконано за допомогою операційної системи роботів. Створено 3D-модель робота-маніпулятора Dobot Magician в ОСР за допомогою URDF фалу (уніфікований формат опису робота), який є форматом XML, що дозволяє описувати фізичні властивості робота.

Програмне забезпечення Dobot Studio працює на основі графічного інтерфейсу користувача (ГІК). Графічний інтерфейс користувача має такі функції, як надання команд Dobot Magician для виконання таких операцій, як написання тексту та малювання, лазерне гравірування, 3D-друк і керування роботом за допомогою жестів рукою.

Література:

1. ОСР - Операційна система роботів. Режим доступу: https://ros.org.

2. Programming Robots with ROS: A Practical Introduction to the Robot Operating System / Morgan Quigley, Brian Gerkey, William D. Smart. - O'Reilly Media, Inc., 2015. – 448 с.

3. Learning ROS for Robotics Programming / Aaron Martinez, Aaron Romero, Enrique Fernández. - Packt Publishing, 2013. – 332 с.

4 A review of mobile robots: Concepts, methods, theoretical framework, and applications / Francisco Rubio, Francisco Valero and Carlos Llopis-Albert. - International Journal of Advanced Robotic Systems, March-April 2019, P. 1-22.

5. Робот-черепаха. Режим доступу: https://clearpathrobotics.com/turtlebot-4/.

6. Lynch, Kevin M. Park, Frank C. Modern Robotics: Mechanics, planning, and Control. Cambridge, UK: Cambridge University Press, 2017.

7. M. R. Islam, M. A. Rahaman, M. A. U. Zaman, and M. Habibur, "Cartesian trajectory based control of dobot robot," in Proc. Int. Conf. Ind. Eng. Operations Manage, pp. 1507-1517, 2019.

References

1. ROS - Robot Operating System. Access mode: https://ros.org.

2. Programming Robots with ROS: A Practical Introduction to the Robot Operating System / Morgan Quigley, Brian Gerkey, William D. Smart. - O'Reilly Media, Inc., 2015. – 448 p.

3. Learning ROS for Robotics Programming / Aaron Martinez, Aaron Romero, Enrique Fernández. - Packt Publishing, 2013. – 332 p.

4 A review of mobile robots: Concepts, methods, theoretical framework, and applications / Francisco Rubio, Francisco Valero and Carlos Llopis-Albert. - International Journal of Advanced Robotic Systems, March-April 2019, P. 1-22.

5. Turtle robot. Access mode: https://clearpathrobotics.com/turtlebot-4/.

6. Lynch, Kevin M. Park, Frank C. Modern Robotics: Mechanics, planning, and Control. Cambridge, UK: Cambridge University Press, 2017.

7. M. R. Islam, M. A. Rahaman, M. A. U. Zaman, and M. Habibur, "Cartesian trajectory based control of dobot robot," in Proc. Int. Conf. Ind. Eng. Operations Manage, pp. 1507-1517, 2019.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник Коваль Михайло Володимирович

Тема: Удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів

Спеціальність: 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Обсяг кваліфікаційної роботи:

Кількість сторінок записки 89

1. Короткий зміст роботи та прийнятих рішень: Метою роботи є розробка розробка удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі розглянуто особливості моделі робота в операційній системі роботів. У другому розділі удосконалено математичну модель робота-маніпулятора в операційній системі роботів. У третьому розділі розроблено імітаційну модель удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів. У четвертому розділі виконано експериментальні дослідження удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів

4. Позитивні сторони роботи: Забезпечення високої точності та надійності керування та мінімізація ризиків для обладнання та середовища завдяки попередньому тестуванню в симуляції. У цьому полягає суттєва перевага удосконаленого методу керування роботом-маніпулятором на основі операційної системи роботів.

5. Негативні сторони роботи: не виконано порівняння удосконаленого методу керування зарядною станцією електромобіля із відомим методами керування зарядними станціями електромобілів.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: відсутні

9. Оцінка дипломної роботи: добре (4,00/С)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Чесун Віктор Миколайович
канд. техн. наук, доцент кафедри кібербезпеки

"12" 12 2024 р. _____ (підпис)

Завідувачу кафедри АКІТтаР
д-ру техн.наук, проф. Мартинюку В.В.

Коваля М.В.
ПІБ здобувача вищої освіти

ФІТ, 2 курс, групи АКІТРм-23-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (StrikePlagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.12.24
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 160472 Назва: МКР Удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів Додано в БД: 2024-12-17 Автора: Михайло КОВАЛЬ Керівники: Андрій СЕЛЬСЬКИЙ Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	92205	710	1053 (1%)	14 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Михайло КОВАЛЬ

Співавтор:

Назва: МКР Коваль

Науковий керівник: Андрій СЕЛЬСЬКИЙ

Підрозділ: Кафедра автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

Коефіцієнт подібності 1:1.5%

Коефіцієнт подібності 2:0.2%

Мікропробіли: 10

Заміна букв: 4

Інтервали: 0

Білі знаки: 1

Дата створення звіту: 2024-12-17 12:02:34.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2024-12-17 М. В. Федула

Дата експерт

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ АВТОМАТИЗАЦІЇ, КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: Удосконалений метод керування роботом-маніпулятором на основі операційної системи роботів
Автор: Коваль Михайло Володимирович
Спеціальність: 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма: Освітньо-професійна програма «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Науковий керівник: Сельський Андрій Анатолійович, кандидат фізико-математичних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:
Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:
1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій у переліку джерел посилання;
2) усі запозичення є фрагментарними або мають належним чином оформленні посилання;
3) виявлені модифікації тексту не впливають на відсоток схожості.
Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 1,5% і адресується до 56 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Завідувач кафедри _____ Валерій МАРТИНЮК
Гарант освітньої програми _____ Валерій МАРТИНЮК
Керівник кваліфікаційної роботи _____ Андрій СЕЛЬСЬКИЙ