Національний університет кораблебудування
імені адмірала Макарова

Навчально-науковий інститут автоматики та електротехніки
(назва інституту, факультету)

кафедра електричної інженерії суднових та роботизованих комплексів
(назва кафедри)

Допущений до захисту

зав. кафедри ЕІС та РК
(скорочена назва кафедри)

(підпис) Костенко Д.В.
(прізвище та ініціали

" 16 " 12 2025 р.

Кваліфікаційна робота

на здобуття ступеня вищої освіти

магістр

на тему Розробка засобів віртуального прототипування робототехнічних систем

Виконав: студент групи 6361м

Спеціальності

141 «Електроенергетика, електротехніка та електромеханіка»

спеціалізації «Електричні системи і комплекси транспортних засобів»

Студент Рибак О.Б.
(прізвище та ініціали) (підпис)

Керівник к.т.н., доцент Бабкін Г.В.
(прізвище та ініціали) (підпис)

Рецензент к.т.н., доцент Клочков О.П.
(прізвище та ініціали) (підпис)

м. Миколаїв – 2025 р.

Міністерство освіти і науки України
Національний університет кораблебудування
імені адмірала Макарова

Інститут, факультет Навчально-науковий інститут автоматики та електротехніки
Кафедра електричної інженерії суднових та роботизованих комплексів
Рівень вищої освіти магістр
Галузь знань 14 «Електрична інженерія»
(шифр і назва)
Спеціальність 141 «Електроенергетика, електротехніка та електромеханіка»
(шифр і назва)
Освітньо-професійна програма «Електричні системи і комплекси транспортних засобів»
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
(підпис) Костенко Д.В.
(прізвище та ініціали)
"01" 10 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу
на здобуття ступеня вищої освіти магістр

Рибак Олександр Борисович
(прізвище, ім'я, по батькові студента)
1. Тема кваліфікаційної роботи Розробка засобів віртуального прототипування робототехнічних систем

Керівник кваліфікаційної роботи Бабкін Георгій Володимирович, канд. техн. наук, доцент НУК
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «14» 10 2025 року №1217уч
2. Строк подання студентом кваліфікаційної роботи
3. Вихідні дані кваліфікаційної роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Електронна презентація

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
ОП	Бабкін Г. В.		
ОНС	Бабкін Г. В.		

7. Дата видачі завдання на кваліфікаційну роботу «_15_» вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	*Видача завдання. Розроблення змісту роботи. Узгодження завдання по спец. розділам*		
2	*Розробка розділу 1. Розробка розділу 2.*		
3	*Розробка розділу 3. Розробка розділу 4.*		
4	*Розробка розділу 5. Затвердження додатків.*		
5	*Розробка висновків та переліку посилань. Розробка реферату та вступу.*		
6	*Затвердження висновків, переліку посилань, реферату та вступу. Оформлення ПЗ та презентації.*		
7	*Підготовка доповіді. Захист МР.*		

Студент _____ (підпис) *Рибак О. Б.* (прізвище та ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Бабкін Г. В.* (прізвище та ініціали)

АНОТАЦІЯ

У роботі виконано аналіз методів та засобів проектування робототехнічних систем. Розроблено бібліотеку інтелектуальних модульних компонентів робототехнічних засобів у форматі URDF для сучасних середовищ моделювання. Описано процес розробки бібліотеки універсальних моделей під керуванням операційної системи ROS та їхньої натурної реалізації. Виконано натурну реалізацію прототипу симетричної крокуючої робототехнічної платформи на основі розроблених інтелектуальних модульних компонентів бібліотеки готових рішень.

Ключові слова: віртуальне прототипування; проектування робототехнічних ситем; модульний принцип; модульні компоненти; моделювання

ABSTRACT

The robot has analyzed the methods and features of the design of robotic systems. A library of intelligent modular components of robotic techniques has been developed in the URDF format for modern modeling environments. The process of developing a library of universal models under the control of the ROS operating system and its full-scale implementation is described. A full-scale implementation of a prototype of a symmetrical robotic platform based on separate intelligent modular components of a library of ready-made solutions has been completed.

Keywords: virtual prototyping; robotic system design; modular principle; modular components; modeling

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

РТС – робототехнічних систем

dVRK – дослідницький комплекс da Vinci

COLLADA – COLLAborative Design Activity

URDF – Universal Robot Description Format

ВСТУП

Робототехніка є однією з областей техніки, що найбільш динамічно розвиваються. Це з тим, що є безліч процесів, потребують повної чи часткової автоматизації. Але розробка та конструювання робототехнічних засобів - довгий, трудомісткий та дорогий процес. Зниження трудомісткості, вартості та часу розробки таких систем можна досягти шляхом використання бібліотек готових рішень, що містять підсистеми управління робототехнічними компонентами. Такий підхід дозволяє знизити вимоги до рівня підготовки розробників.

Мета роботи: зниження трудомісткості проектування робототехнічних систем шляхом застосування засобів віртуального прототипування робототехнічних компонентів

Завдання, які вирішуються у роботі:

– створення бібліотеки готових рішень модульних компонент робототехнічних систем (РТС);

– розробка моделі модульного крокуючого робота;

– розробка алгоритму керування модульними компонентами;

– розробка алгоритму управління модульною роботизованою системою;

– натурна реалізація прототипу модульного крокуючого робота на основі моделі з бібліотеки готових рішень;

– виконання експериментальних досліджень одержаного прототипу.

РОЗДІЛ 1
ОГЛЯД МЕТОДІВ І ЗАСОБІВ РОЗРОБКИ РОБОТОТЕХНІЧНИХ СИСТЕМ. ПОСТАНОВКА ЗАВДАННЯ

1.1 Існуючі підходи до розробки робототехнічних засобів

Процес розробки робототехнічних засобів поділяється такі этапы [1]:

– Концепція формування. Визначення цілей створення об'єкта. Опис функціональних особливостей об'єкта та принципів роботи об'єкта.

– Науково-дослідна робота. Розробка моделі об'єкта та середовища застосування. Коригування вихідного уявлення про об'єкт з урахуванням результату моделювання.

– Реалізація. Втілення моделі у натурну реалізацію системи.

– Дослідження. Вивчення втіленого об'єкта, виявлення переваг та недоліків. При виявленні останніх необхідно повернутися на один із попередніх етапів та провести модернізацію системи (рис.1.1).

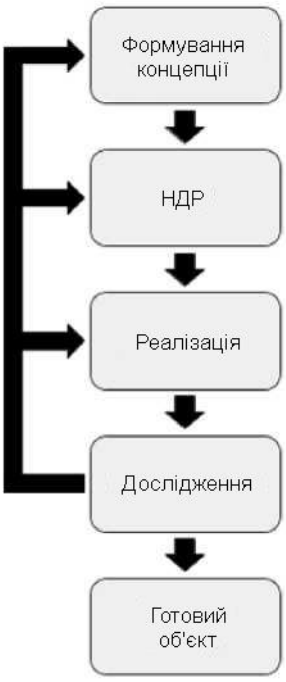

Рисунок 1.1 – Схематичне зображення процесу розробки РТС

Конструювання робототехнічної системи – це технічно складне завдання. Тому етап реалізації – довгий і трудомісткий процес. Це пов'язано з тим. що необхідно поєднати різні програмні та апаратні компоненти в єдину систему. Одним із підходів до реалізації робототехнічних засобів є інтеграційний підхід. Інтеграційний підхід включає у собі такі этапы [21]:

– купівля деталей;

– інтеграція. Об'єднання деталей у сполучену систему. Найбільш трудомісткий процес;

– побудова робота;

– програмування алгоритмів керування;

– тестування та адаптація робототехнічної системи;

– отримання прототипу.

Розробники можуть мати проблеми вже на етапі інтеграції через несумісність компонентів. Крім того, кожна модифікація системи вимагає подальшої інтеграції, а використання модулів, що вийшли, повторно дорогий і трудомісткий процес.

Рисунок 1.2 – Інтеграційний підхід

Модульний підхід до розробки (рис.1.3) або модульність, ділить систему більш дрібні і схожі частини, звані модулями. які можна налаштовувати, модернізувати, ремонтувати, масштабувати та повторно використовувати. Концепція модульності не нова, і її можна простежити до Другої світової війни, коли Отто Меркер досліджував як модульність може бути реалізована при конструюванні німецьких підводних човнів. IBM запустила модульну архітектуру для персональних комп'ютерів у 1980 році, і сьогодні модульність є широко відомим терміном у світі робототехніки.

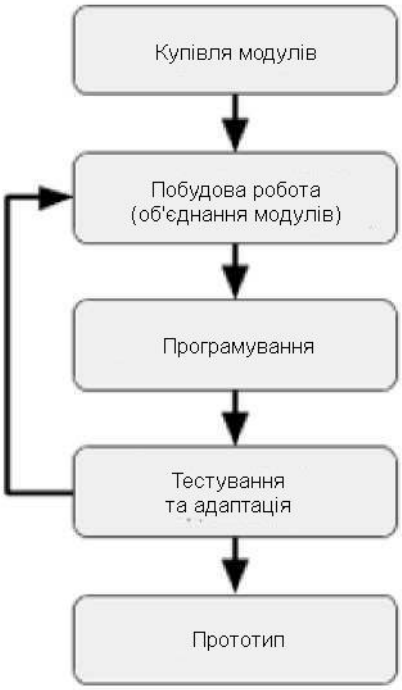

Рисунок 1.3 – Модульний підхід

Модульність є одним із головних наріжних каменів у майбутньому робототехніки. Створення легко сумісних компонентів різних виробників – це лише один приклад того, як потенціал галузі може бути значною мірою розкритий завдяки модульності. Фактично, інтеграція є одним із найбільш трудомістких та ресурсомістких вузьких місць у прикладній робототехніці. Інтеграція частин робота обходиться набагато дорожче, ніж їх виготовлення чи програмування.

Розглянемо кілька прикладів застосування засобів міжплатформного програмного забезпечення та модульної архітектури РТС. У 2017 році Редмондом Р. Шамширі та командою дослідників з університету Флориди було розроблено роботизовану систему для збору врожаю [25] за допомогою засобів віртуального прототипування. Розробники використовували зв'язку V-REP – ROS – MATLAB для симуляції, а потім реалізації РТС. Першим етапом було створення моделі комбайна у пакеті моделювання. Потім було розроблено алгоритми управління та розпізнавання об'єктів з використанням міжплатформного програмного забезпечення (рис. 1.4).

Рисунок 1.4 – Віртуальний прототип системи розпізнавання

ROS дозволив розробити універсальне програмне забезпечення. підходить як для віртуального прототипу, так і для натурної реалізації, тому розробки можна було випробовувати і модернізувати систему віртуально до того часу. поки не будуть досягнуті задовільні результати, а лише потім переходити до реальних випробувань (рис.1.5).

Рисунок 1.5 – Зображення з камери прототипу

Такий підхід дозволив знизити час розробки системи та трудомісткість і навіть підвищити якість отриманого РТС.

Компанія із Швейцарії з виробництва модульних маніпуляторів (рис.1.6) Acutronic Robotics [20] використовує ROS2 для уніфікації програмного забезпечення для різних компонентів у зв'язці з пакетом Gazebo. Розробники використовують відкритий інструмент навчання інтелектуальних пристроїв Open AI Gym. Такий підхід дозволяє без реальних компонентів розробляти алгоритми управління для віртуальної моделі, та був використовувати отримані програмні засоби натурної реалізації. Це дозволяє уникнути псування дорогих компонентів РТС при невдалих експериментах із софтом, що дозволяє знизити вартість розробки.

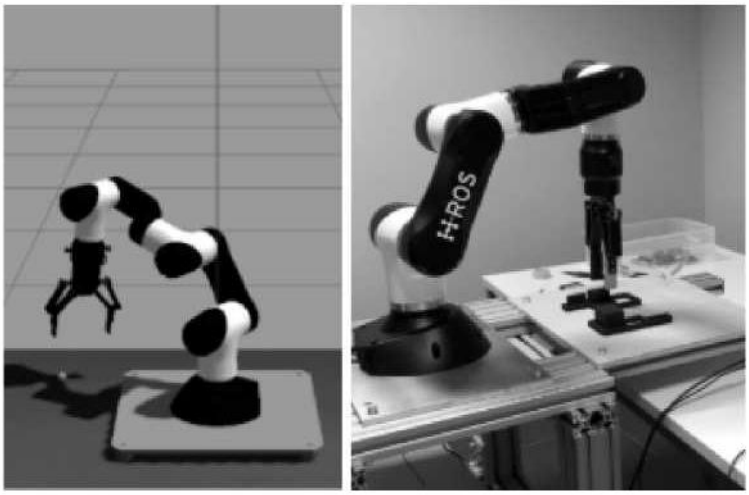

Рисунок 1.6 – Віртуальний прототип маніпулятора та натурна реалізація

Технології, що динамічно розвиваються, вимагають свіжих рішень, тому багато великих компаній влаштовують змагання для робототехнічної спільноти метою яких є розвиток нових підходів до розробки робототехнічних систем.

Управління перспективними дослідницькими проектами DARPA влаштовує щорічні випробування для спільноти робототехніків. Нещодавно одна з команд продемонструвала підхід до створення робототехнічних систем, що

залучає інфраструктуру з використанням засобів віртуального моделювання та міжплатформного програмного забезпечення для прискорення та підвищення якості розробки [26]. Метою було створення гуманоїдного робота (рис.1.7). Команда створила віртуальний прототип системи у пакеті Gazebo, а систему управління реалізувала за допомогою міжплатформного програмного забезпечення. Тим самим стало можливим протестувати поведінку системи в пакеті моделювання, модернізувати з урахуванням отриманих результатів, а потім завантажувати програмне забезпечення на реальний об'єкт.

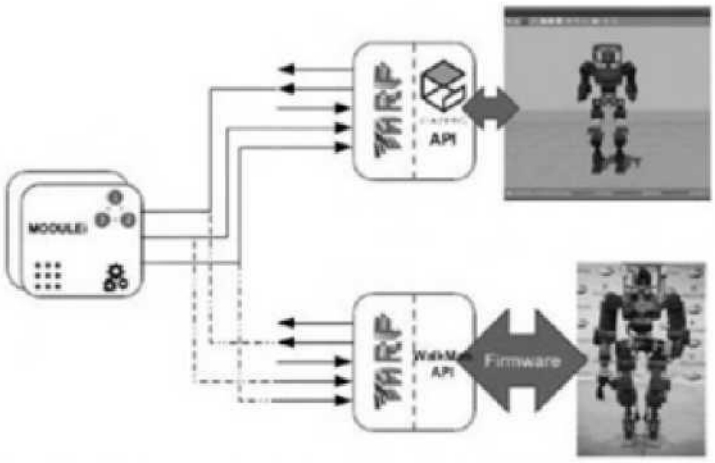

Рисунок 1.7 – Схема реалізації крокуючого робота

1.2 Огляд міжплатформного програмного забезпечення для робототехнічних систем

Автономні роботи – це складні системи, що потребують взаємодії між численними компонентами. Через ускладнення роботизованих систем та різноманітності апаратного забезпечення міжплатформне програмне забезпечення роботів застосовується для розробки та інтеграції компонентів апаратного та програмного забезпечення. Такі системи спрощують інтеграцію нових компонентів, розробку програмного забезпечення, що дозволяє абстрагуватися від низькорівневого зв'язку. Використання міжплатформного програмного забезпечення (рис.1.8) у робототехніці підвищує якість програмного

забезпечення, уможливлює його

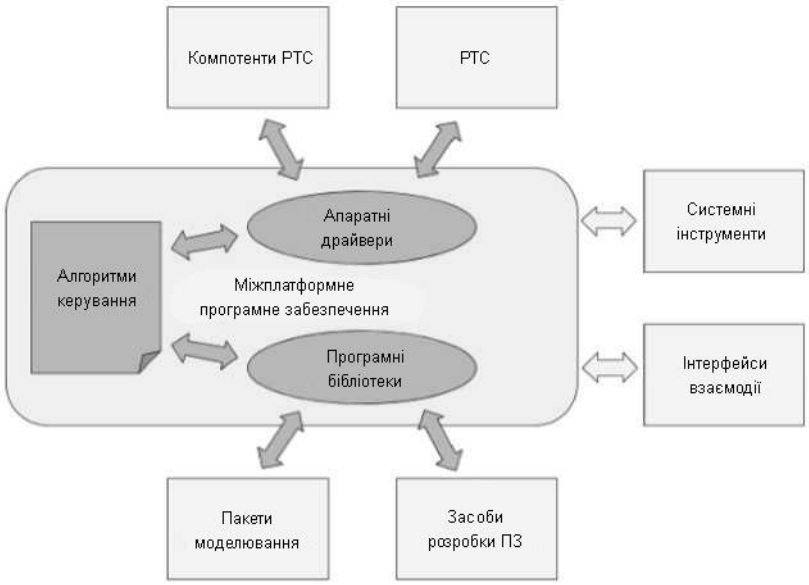

Рисунок 1.8 – Узагальнена структура міжплатформного програмного забезпечення

В даний час існує велика кількість міжплатформного програмного забезпечення, що має свої особливості. Оскільки сучасні тенденції вимагають швидкого розвитку програмного забезпечення, розглянемо найпопулярніші системи, останні версії яких було випущено протягом останніх п'яти років.

RT-middleware

RT-middloware – програмна платформа для компонентно-орієнтованої розробки робототехнічних систем. Це платформа. заснована на технології розподілених об'єктів. Особливість RT-middlowaro це конструювання різних мережевих роботизованих систем шляхом інтеграції мережевих роботизованих компонентів. званих RT-Components. Такий підхід дозволяє будувати децентралізовані системи із уніфікованими інтерфейсами взаємодії компонентів.

Система була розроблена японською компанією, тому більшість співтовариства цієї системи спілкуються японською мовою. Це накладає низку обмежень для користувачів спільноти, що не володіють японською мовою.

Проект VirCA[29]. розроблений спільно з Австралійським інститутом комп'ютерних наук та управління, використовує систему RT-middlowaro для

реалізації віртуального простору для створення та тестування моделей робототехнічних засобів. Система дозволяє співпрацювати розробникам, які знаходяться віддалено. Крім того, система дозволяє взаємодіяти з моделями у віртуальній реальності (рис.1.9).

Рисунок 1.9 – Модель у віртуальній реальності

Rock

Rock[22] – це програмний фреймворк розробки роботизованих систем. Базова компонентна модель базується на Orocos RTT (Real Time Toolkit). Rock надає інструменти, необхідні для налаштування та запуску високопродуктивних та надійних роботизованих систем для застосування у наукових дослідженнях та промисловості.

Особливість системи – це наявність драйверів для готових модулів. а також підтримка реального часу. Така система підійде для роботизованих компонентів, для яких основний критерій - це дотримання жорсткого реального часу.

В університеті Сан-Паулу [30] інтегрували Rock з Gazebo для симуляції автономного безлюдного підводного апарату в реальному часі (рис.1.10). Такий підхід дозволив не лише створити віртуальний підводний човен у популярному пакеті моделювання, а й протестувати програмне забезпечення в умовах реального часу.

Однак система підтримує лише ОС Linux та одну мову програмування C++.

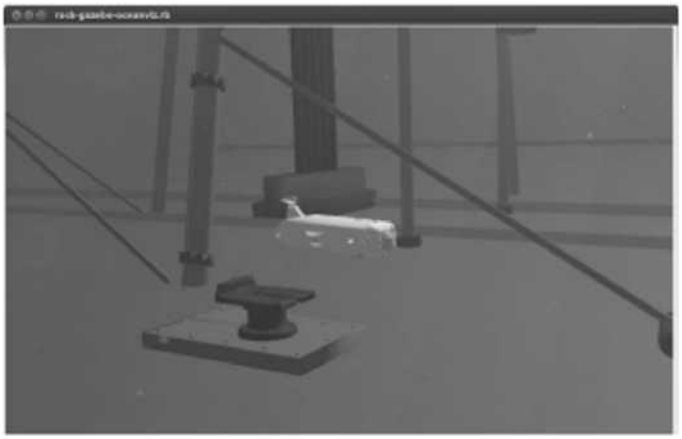

Рисунок 1.10 – Симуляція автономного підводного апарату

ROS

ROS (Robot Operating System) надає бібліотеки та інструменти, що допомагають розробникам програмного забезпечення створювати програми для роботів.Система забезпечує апаратну абстракцію, надає драйвери для пристроїв, бібліотеки, візуалізатори, передачу повідомлень, управління пакетами та багато іншого.ROS ліцензується з відкритим вихідним кодом.

ROS працює за принципом розсилок та підписок. Всі вузли в оточенні ROS спілкуються стандартизованими повідомленнями через Торті.

ROS є одним із найпопулярніших засобів серед спільноти розробників, тому існує безліч сумісних із нею компонентів. У статті [8] описується процес створення автономного робота із готових компонентів, що підтримуються ROS (рис.1.11). Використання міжплатформного програмного забезпечення дозволило знизити загальну вартість РТС, що дозволить використовувати такий підхід для навчання робототехніки в університеті Карлоса III у Мадриді.

Однак для роботи ROS потрібна наявність активної служби ROS core. Через це один із компонентів системи повинен бути одноплатним комп'ютером із запущеним майстром.

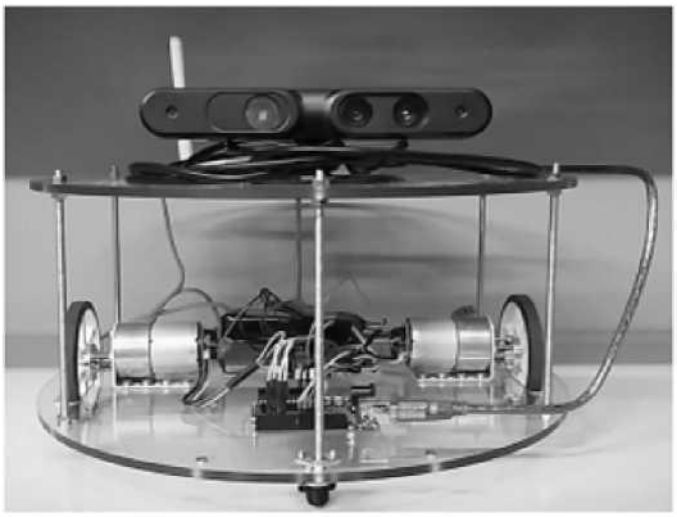

Рисунок 1.11 – Автономний робот на основі готових компонентів

YARP

YARP [31] є міжплатформним програмним забезпеченням для робототехніки з більш ніж десятирічним досвідом використання на різних робототехнічних системах. YARP був розроблений. щоб допомогти програмному забезпеченню пережити модернізацію та інтеграцію з іншими системами.

Перевага YARP – це можливість інтеграції з іншими міжплатформними програмними забезпеченнями. Це відкриває можливості зі збирання РТС із різних компонентів різних виробників.

З використанням цієї системи було розроблено безліч різних РТС. Один із них це автономний робот для обстеження тунелів метро щодо аномалій (рис.1.12). розроблений науковою групою із Чехії. Цей робот сканує стіни тунелю та виявляє відхилення від норми, а потім будує 3D модель обстеженого простору. Цей засіб тестувалося в Лондонському метрополітену [16].

Але YARP не є системою реального часу та не підходить для побудови РТС реального часу.

ROS2

ROS2fl8] відрізняється від ROS використанням служби розповсюдження даних (DDS). DDS підходить для розподілених вбудованих систем реального часу завдяки суворим часовим рамкам, відмовостійкості та масштабованості. ROS2 не є продовженням ROS. а розвивається паралельно, оскільки важливо

підтримувати незалежно дві різні концепції.

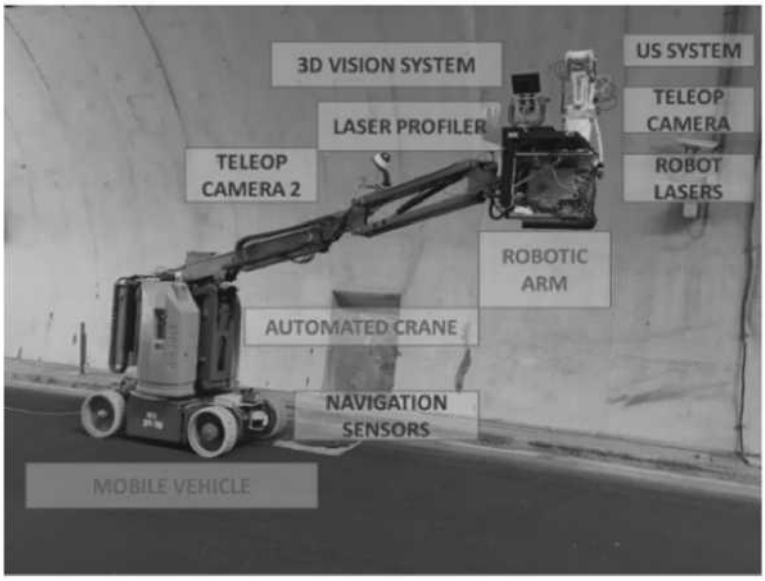

Рисунок 1.12 – Автономна система обстеження тунелів

Проте. ROS2 ще не підтримується багатьма пакетами моделювання. на відміну від ROS. Це накладає низку обмежень на можливість віртуального прототипування.

У статті [7] описується процес створення роїв управління роботами в університеті Клагенфурт в Австрії з використанням ROS2. і навіть дослідження взаємодії РТС друг з одним на вирішення поставлених завдань (рис.1.13).

Orocos

Orocos [l4] є абревіатурою від Open Robot Control Software. Метою проекту є розробка універсального, безкоштовного програмного забезпечення та модульної структури для управління роботизованими об'єктами. При використанні Orocos. можна використовувати готові компоненти, надані спільнотою, або створити власний компонент, використовуючи інструментарій Orocos Real-Time Toolkit. Orocos насамперед націлений на підтримку систем реального часу.

Обмеження використання цієї системи накладає наявність лише одного пакета віртуального прототипування. Отже. спільнота робототехніків не зможе використовувати отримані компоненти у звичних для себе інструментах.

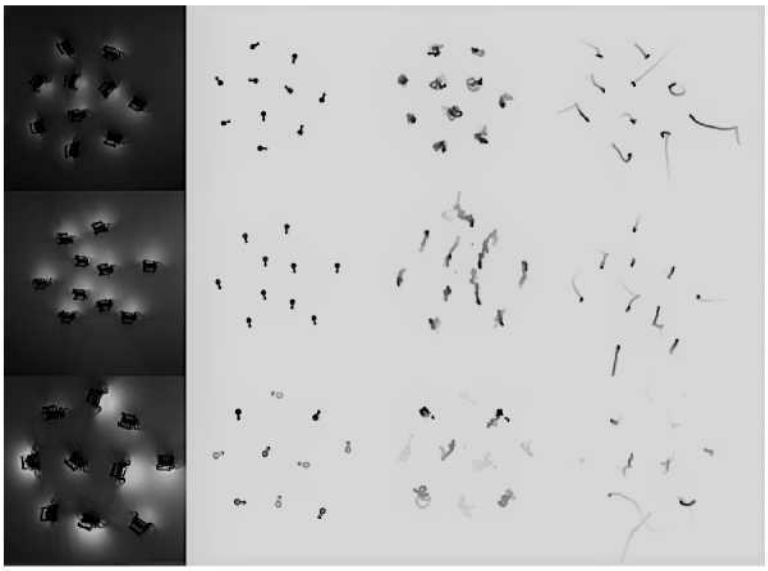

Рисунок 1.13 – Приклади роевого управління

Команда розробників із Китаю створила робототехнічний маніпулятор (рис.1.13) з допомогою реального часу з допомогою даного межплатформенного ПО [3]. Маніпулятор має 6 ступенів свободи.

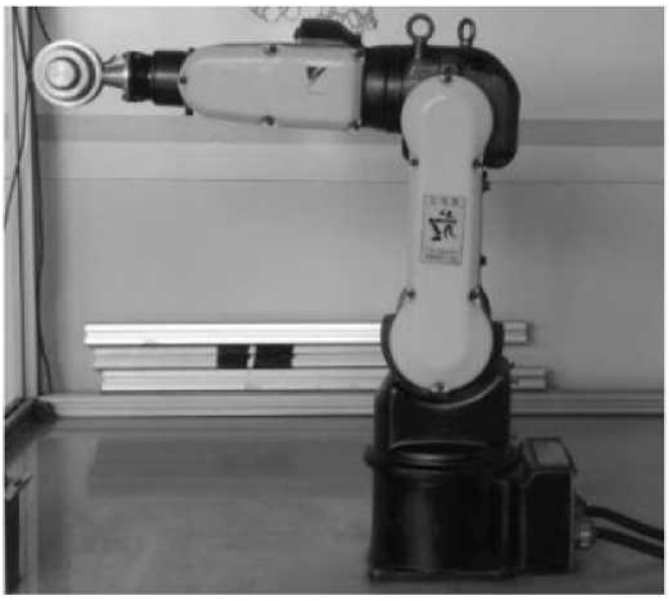

Рисунок 1.14 – Робототехнічний маніпулятор

MIRA

MIRA – це кросплатформова інфраструктура, написана на C++, яка надає

міжплатформне програмне забезпечення, кілька базових функцій та численні інструменти для розробки та тестування розподілених програмних модулів.

Особливість цього програмного забезпечення полягає в тому, що це не лише міжплатформне програмне забезпечення для робототехнічних систем. Основною метою MIRA є розробка роботизованих додатків, але оскільки її основа та структура призначені для забезпечення безпечного обміну даними між програмними модулями, система не обмежується цими типами додатків.

У статті [32] описується досвід із запровадження роботизованого допоміжного персоналу з догляду за людьми похилого віку (рис.1.15) на основі системи MIRA. Авторами з міжуніверситетської наукової групи з Великобританії розглядається 3 різні зміни РТС із різними особливостями.

Обмеженням є відсутність пакетів моделювання, що підтримуються, крім власного. Це накладає низку обмежень використання віртуальних прототипів співтовариством робототехніків.

Рисунок 1.15 – Роботизований персонал

1.3 Огляд форматів опису моделей робототехнічних об'єктів

Для представлення тривимірних об'єктів існує багато форматів. які підтримуються сучасними САПР. Однак для представлення робототехнічних об'єктів важливо не тільки зберігати тривимірну модель, а й динамічні

характеристики, а також зчленування між собою твердих тіл. Більшість із сучасних форматів такого типу ґрунтуються на мові XML. Нині використовується кілька основних форматів. Розглянемо формати. популярні у спільноти розробників.

COLLADA

COLLADA (COLLAborative Design Activity) – це формат файлу обміну для інтерактивних ЗО-додатків. Формат керується некомерційним технологічним консорціумом, Khronos Group, і був прийнятий ISO як загальнодоступна специфікація ISO / PAS 17506.

COLLADA визначає XML-схему відкритого стандарту для обміну цифровими об'єктами між різними програмами, які інакше могли б зберігати свої об'єкти в несумісних форматах файлів.

URDF

Universal Robot Description Format (URDF) [27] – це формат XML-файлів, що використовується в ROS для опису всіх елементів РТС. URDF є стандартизованим форматом у ROS. URDF може визначати кінематичні та динамічні властивості лише одного ізольованого робота.

SDF

SDF[23] – це формат на основі XML, який описує об'єкти та сцени для симуляторів роботів, візуалізації та керування. Спочатку розроблений як частина робота-симулятора Gazebo, SDF був розроблений з урахуванням потреб наукових роботів.

За минулі роки SDF став здатним описувати багато аспектів роботів, статичних та динамічних об'єктів, освітлення та місцевості.

1.4 Огляд засобів віртуального прототипування робототехнічних систем

В даний час існує багато пакетів моделювання, які дозволяють спростити створення прототипів робототехнічних засобів. Сучасні тенденції диктують вимоги до функціональних особливостей таких інструментів, тому більшість

ними схожі[9]. Пакети моделювання мають фізичний та графічний двигун. Фізичний двигун відповідає за розрахунок динамічних показників об'єктів сцени. Графічний двигун візуалізує дані, розраховані пакетом для представлення їх у зручній для людини формі. Типова архітектура пакета моделювання представлена на рис.1.16).

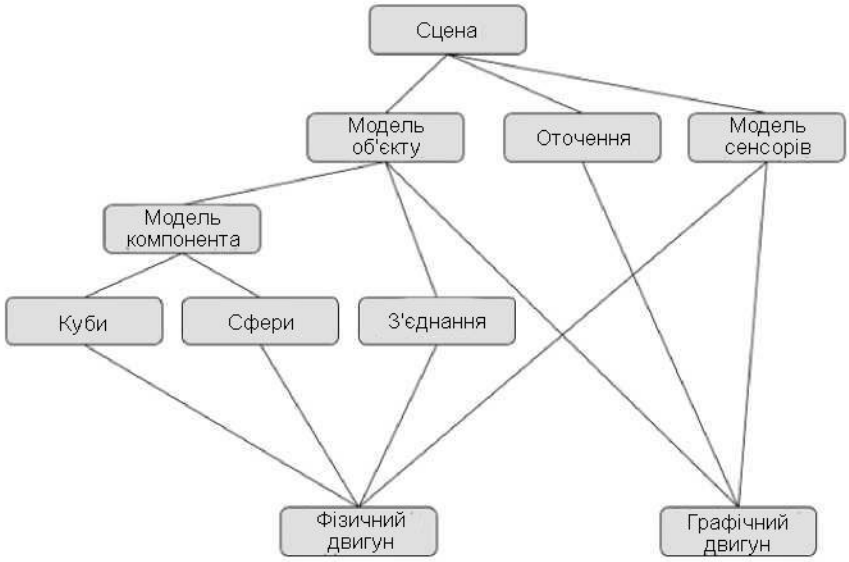

Рисунок 1.16 – Схема взаємодії компонентів пакетів моделювання

У роботі розглядаються пакети моделювання, які користуються популярністю у спільноти розробників.

Gazebo

Gazebof6] призначений для точного відтворення динамічного середовища, з яким може зіткнутися робот. Усі змодельовані об'єкти мають масу, швидкість, тертя та численні інші атрибути. які дозволяють їм вести себе реалістично за зовнішнього впливу.

Особливістю пакета є відкритість вихідного коду, що дозволяє спільноті брати участь у покращенні можливостей проекту. створювати різні плагіни та бібліотеки.

Гідністю Gazebo є можливість симуляції різних динамічних середовищ, наприклад, води. Це дозволяє застосовувати пакет моделювання для прототипування підводних та літальних апаратів. У статті [13] описується досвід

використання середовища ROS та симулятора Gazebo для розробки автономного багатопропелерного безпілотного літального апарату (рис.1.17) науковою групою з університету Дармштадт, який може переміщатися невідомою будівлею та знаходити вихід, використовуючи методи комп'ютерного зору.

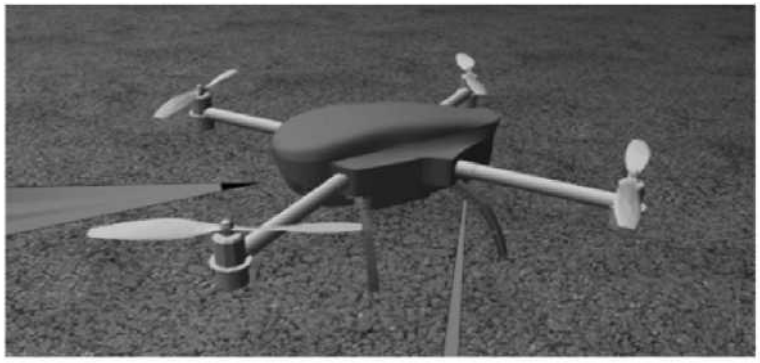

Рисунок 1.17 – Модель квадрокоптера в симуляторі Gazebo

Однак Gazebo не підтримує такі маніпуляції з об'єктами, як різання твердих тіл. Це накладає обмеження на моделювання деяких роботів та механізмів. З іншого боку, складні об'єкти імпортовані з інших засобів моделювання неможливо змінити всередині пакета.

V-REP

V-REP [28] – платформа для експериментів з віртуальними роботами - це програмне забезпечення для тривимірного моделювання роботів з інтегрованим середовищем розробки, яке дозволяє моделювати, редагувати та програмувати будь-якого робота або роботизовану систему (наприклад, датчики, механізми тощо).

Особливістю пакету моделювання є наявність п'яти фізичних двигунів і велика вбудована бібліотека моделей роботів.

Перевагою V-REP є можливість симуляції різання об'єктів, а також створення/редагування складних 3D об'єктів. Ця можливість розширює список маніпуляцій із об'єктами, які можна симулювати. У роботі (5) представлено симулятор дослідницького комплексу da Vinci (dVRK) (рис.1.18). Симулятор містить повну кінематична модель робота та вбудовані датчики. І симулятор, і

приклади сцен доступні спільноти як програмне забезпечення з відкритим вихідним кодом.

Також є можливість розробляти ПЗ для моделей як усередині так і поза пакетом і однією п'ятьма мовами.

Але V-REP має обмеження в моделюванні динамічних середовищ. Це означає, що немає можливості якісно моделювати підводні об'єкти.

Рисунок 1.18 – Модель дослідницького комплексу в V-REP

ARGoS

Симулятор призначений для підтримки великих команд роботів. Його дизайн дуже відрізняється від дизайну інших симуляторів. Його найбільш характерною рисою є те. що сцену можна розділити на регіони, і кожен регіон може бути призначений на окремий фізичний двигун. Крім того, дизайн ARGoS використовує концепцію точності, що настроюється.

Такий підхід дає перевагу пакету у моделюванні великої кількості об'єктів. Це можна застосувати для симуляції роєвого управління з великою кількістю РТС. У [15] розробниками з Шефілдського університету представлений плагін для симулятора ARGoS [2]. розроблений для спрощення та прискорення експериментів з "кілоботами" (рис.1.19). Пакет моделювання дозволив

розробникам симулювати поведінку сотень "кілоботів" для експериментів із ними.

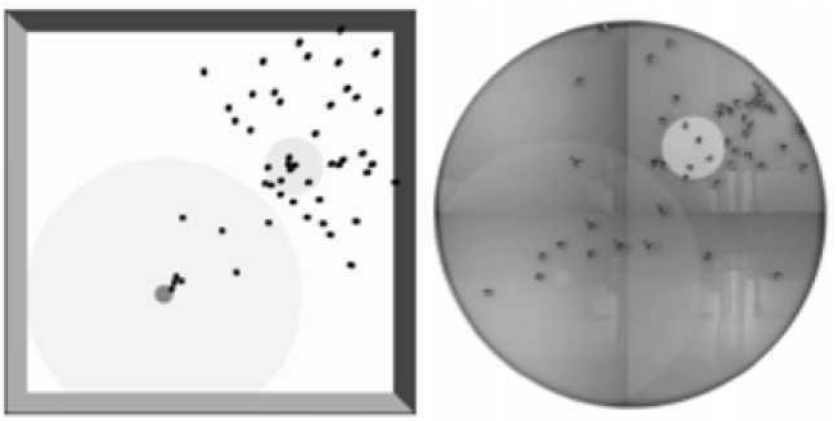

Рисунок 1.19 – Симуляція "кілоботів" (ліворуч) та натурна реалізація (праворуч)

Однак у пакеті відсутня можливість імпорту складних 3D об'єктів та маніпуляції з ними. Також у пакеті відсутня можливість редагування оточення сцени.

Webots

Webots[24] – це симулятор роботів із відкритим вихідним кодом. Він широко використовується в промисловості, освіті та наукових дослідженнях.

У дистрибутив програмного забезпечення входить велика колекція моделей моделей, що вільно змінюються. Крім того, можна створювати нові моделі з нуля. Webots включає набір датчиків та виконавчих механізмів, що часто використовуються в роботизованих експериментах, наприклад, датчики наближення, датчики світла, сенсорні датчики. GPS. акселерометри, камери, випромінювачі та приймачі, серводвигуни (ротаційні та лінійні), датчик положення та тиску, світлодіоди, захоплення, гіроскопи, компас і т. д. Це дозволяє розробникам у короткі терміни створювати системи автономного пілотування або навігації. Оцінка надійного виявлення перешкод із використанням даних, зібраних датчиками LiDAR, стала ключовою проблемою, яку активно вивчає наукова спільнота. Для досягнення мети [24] центром автоматизації та робототехніки з Іспанії було обрано інструмент ЗО-

моделювання Webots Automobile для емуляції взаємодії датчиків у складних умовах керування транспортним засобом (рис.1.20).

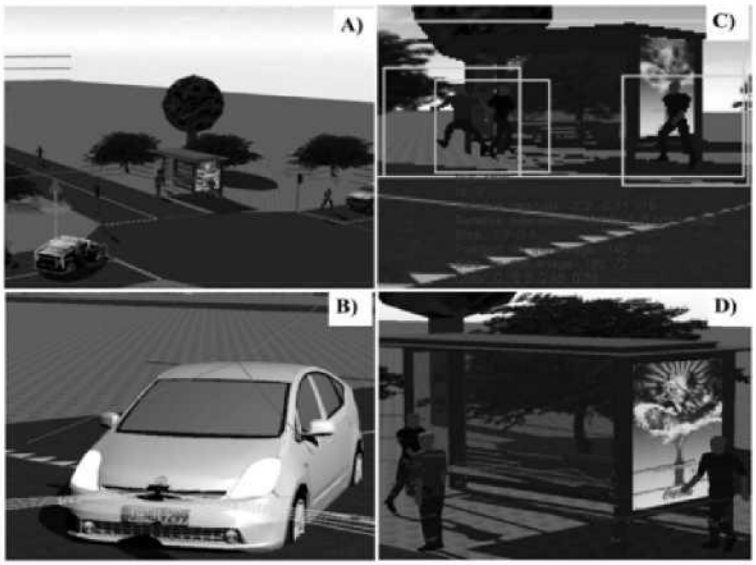

Рисунок 1.20 – Приклад моделювання LiDAR у середовищі моделювання Webots

Webots стала відкритим продуктом лише у грудні 2018 року, тому спільноті знадобиться деякий час, щоб освоїти цей пакет.

MORSE

Основне завдання пакета [12] 3D-моделювання сцен від малих до великих розмірів. Основний акцент зроблено на відтворення різних сенсорів, а не на динамічні характеристики об'єктів. Такий пакет моделювання може бути використаний для симуляції, коли ціна помилки в системі керування роботом надто велика. Наприклад, у роботі [10] група з Брістольської робототехнічної лабораторії розробляє систему інтеграції людей із роботами HI (рис.1.21). Розробники використовували пакет моделювання для симуляції взаємодії роботів з людьми, щоб не наражати на небезпеку людей.

Особливістю MORSE є інтерфейс користувача. Сцени є скриптами, що базуються на Python.

Рисунок 1.21 – Симуляція Human-Robot Integration системи

1.5 Постановка задачі

У роботі потрібно розробити програмно-апаратний комплекс. забезпечує підтримку проектування модульних робототехнічних систем. Узагальнена структура рішення представлена на рис.1.22).

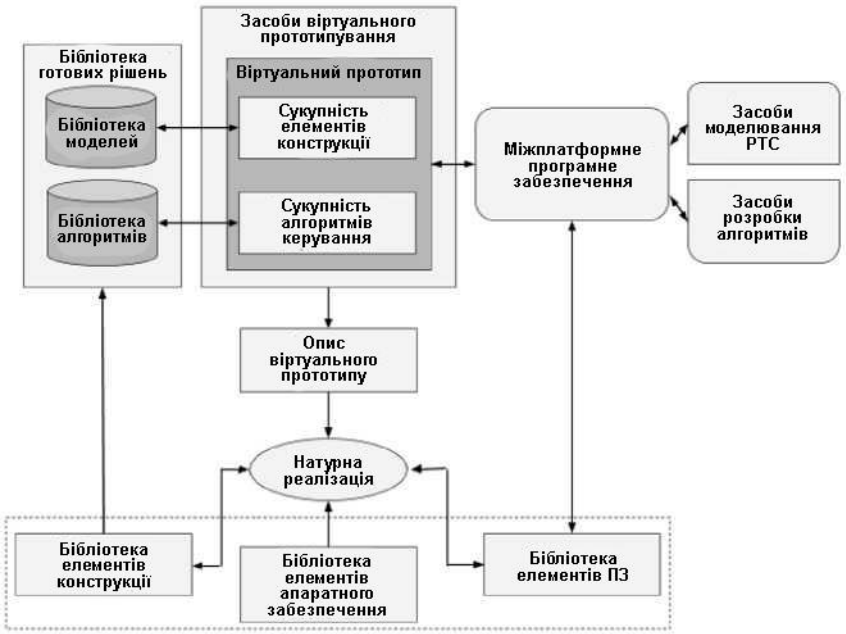

Рисунок 1.22 – Симуляція Human-Robot Integration системи

Для спрощення проектування та моделювання робототехнічних систем

необхідно створити бібліотеку модульних робототехнічних компонентів, що поповнюється. Кожен компонент складається з моделі в універсальному форматі та інтелектуального алгоритму управління. Засоби повинні дозволяти використовувати бібліотеку готових рішень з іншими пакетами моделювання, а алгоритми управління компонентів повинні мати абстрактні інтерфейси взаємодії з іншими компонентами.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– створення бібліотеки готових рішень модульних компонент РТС;

– розробка моделі модульного крокуючого робота;

– розробка алгоритму керування модульними компонентами;

– розробка алгоритму управління модульною роботизованою системою;

– натурна реалізація прототипу модульного крокуючого робота на основі моделі з бібліотеки готових рішень;

– виконання експериментальних досліджень одержаного прототипу.

РОЗДІЛ 2
РОЗРОБКА СТРУКТУРИ КОМПЛЕКСУ ЗАСОБІВ ПРОЕКТУВАННЯ РОБОТОТЕХНІЧНИХ СИСТЕМ

2.1 Вибір міжплатформного програмного забезпечення та формату опису віртуальних прототипів

Для розробки бібліотеки віртуальних прототипів міжплатформне програмне забезпечення для робототехнічних систем повинно мати такі властивості:

– можливість імплементації на мікроконтролері;

– підтримка виробниками робототехнічних компонентів та наявність сумісних пристроїв;

– підтримка пакетів моделювання.

Безліч фреймворків, які відповідають нашим вимогам. Однак з них виділяється ROS за такими параметрами:

– підтримка найбільшої кількості пакетів моделювання;

– широке співтовариство робототехніків, що бере участь у розвитку проекту;

– детальна документація з прикладами;

– підтримка інтеграції OROCOS, ROS2, Rock, Mira, Urbi.

Таким чином, був обраний ROS як міжплатформне програмне забезпечення для розробки екосистеми з проектування модульних робототехнічних засобів. Формат для опису віртуальних прототипів PTC повинен зберігати сполуки, тверді тіла та динамічні характеристики об'єктів. Ці параметри мають усі формати. Однак формат URDF є стандартом Robot Operating System, яка була обрана як міжплатформне програмне забезпечення. А також у ROS є вбудовані бібліотеки для конвертування URDF/COLLADA та URDF/SDF, що дозволяє розширити крут розробників, які зможуть використовувати бібліотеку готових рішень. Таким чином, для зберігання

віртуальних прототипів вибирається формат URDF.

2.2 Вибір пакету моделювання

Fla перший погляд сучасні пакети моделювання мають однакові можливості, однак, при детальному розгляді це не так. Для розробки бібліотеки віртуальних компонентів пакет моделювання повинен володіти такими функціями:

– редактор сцени

– редактор скриптів

– підтримка формату URDF

– редагована бібліотека компонентів

Лише V-REP задовольняє наданим вимогам створення бібліотеки віртуальних компонентів. Однак, оскільки в роботі використовується формат URDF для представлення віртуальних компонентів, можна експортувати моделі з бібліотек готових рішень і використовувати в інших пакетах моделювання, який підтримують Universal Robot Description Format або COLLADA.

2.3 Розробка комплексу засобів віртуального прототипування робототехнічних систем

Першим етапом у проектуванні РТС є віртуальне прототипування натурної реалізації інтелектуального компонента. Віртуальний прототип - це сукупність візуального та керуючого елементів:

– модель інтелектуального робототехнічного компонента;

– керуюче та міжплатформне програмне забезпечення.

Розробник створює модель пристрою у пакеті V-REP. Оскільки модель являє собою не тільки тверді тіла, але й зчленування, а також різні динамічні характеристики звичайні формати 3D моделей не підходять. Для цього було

обрано мову опису робототехнічних компонентів URDF. При використанні пакета моделювання розробник оперує примітивами і налаштовує параметри віртуальних компонентів через інтерфейс користувача. Потім отриманий віртуальний компонент зберігається у вбудовану бібліотеку та може бути повторно використаний. При експорті віртуального прототипу з пакета моделювання файл зберігається в уніфікованому форматі URDF, який надалі може бути використаний іншими розробниками та іншому пакеті моделювання (рис.2.1).

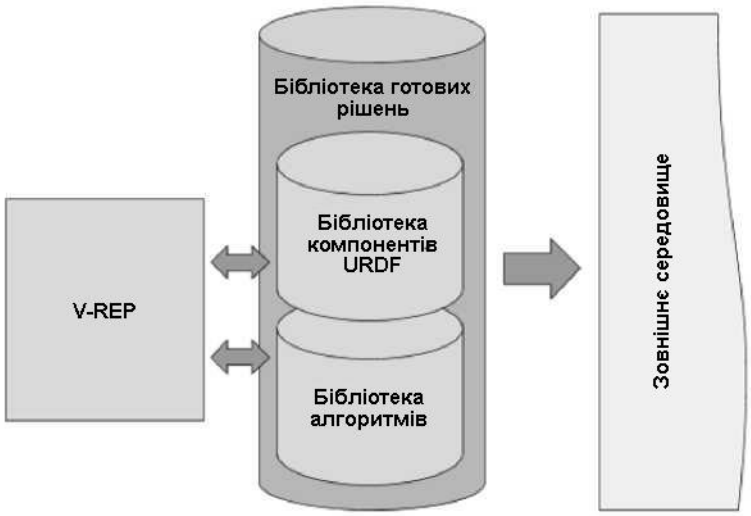

Рисунок 2.1 – Схема роботи бібліотеки готових рішень

Кожен модульний компонент є інтелектуальним пристроєм, тому розробнику необхідно розробити програмне забезпечення для такого компонента. Програмне забезпечення має бути уніфіковано таким чином, щоб алгоритми управління, розроблені для віртуального прототипу, були застосовні для реалізації натури. Для таких цілей застосовується Remote API. Віртуальний прототип зв'язується із програмою керування через інтерфейс пакету моделювання Remote API.

Remote API дає можливість вибрати мову програмування, придатну для використання у системах, що вбудовуються, наприклад. C/C а не тільки мова Lua. Крім того, модульна архітектура передбачає швидку інтеграцію

компонентів. Для цього необхідно використовувати міжплатформне програмне забезпечення. Автором було обрано ROS. принцип роботи якого представлений на рис.2.2).

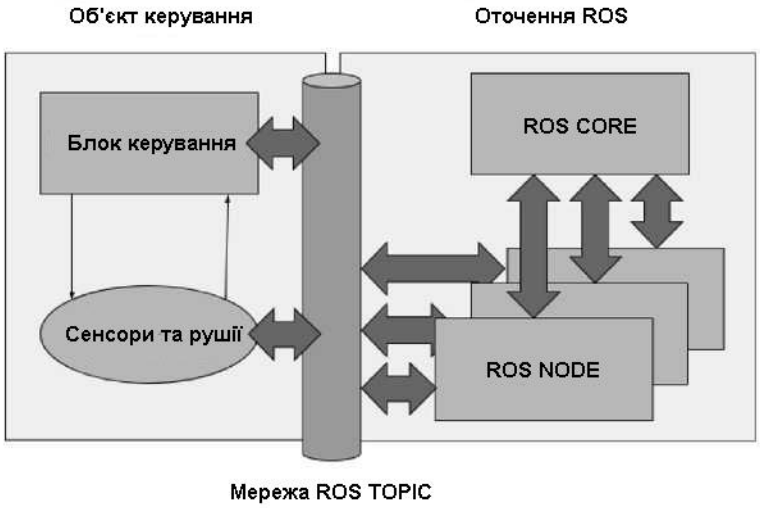

Рисунок 2.2 – Принцип роботи ROS

Кожен інтелектуальний компонент є вузлом ROS NODE, який повідомляється коїться з іншими вузлами через ROS TOPIC. Таким чином, всі компоненти робототехнічної системи знаходяться в оточенні ROS.

Другий етап - це створення віртуального прототипу РТС з бібліотеки компонентів, створених на етапі 1. Для створення робототехнічної системи розробник може використовувати компоненти з бібліотеки готових рішень або створити компонент, що бракує, і додати його в бібліотеку. Модульні компоненти інтегруються між собою через систему ROS. яка дає змогу уніфікувати інтерфейси взаємодії таких компонентів. Кожен компонент є вузлом системи та взаємодіє з іншими компонентами стандартизованими повідомленнями.

Розробник, який використовує бібліотеку готових рішень, не повинен заглиблюватися в деталі реалізації кожного компонента. Компоненти з бібліотеки готових рішень є інтелектуальними та керуються високорівневими командами. Таким чином, при створенні системи управління робототехнічною системою можна сконцентруватися на стратегічному рівні управління. Для

створення такого управління автором пропонується використовувати мову Python. Розробник створює інтелектуальну систему управління мовою Python, яка зв'язується із системою ROS через бібліотеку ROSpy. Система управління транслює команди інтелектуальними компонентами через систему ROS TOPIC, описану вище. Схема взаємодії комплексу віртуального прототипування представлена (рис.2.3).

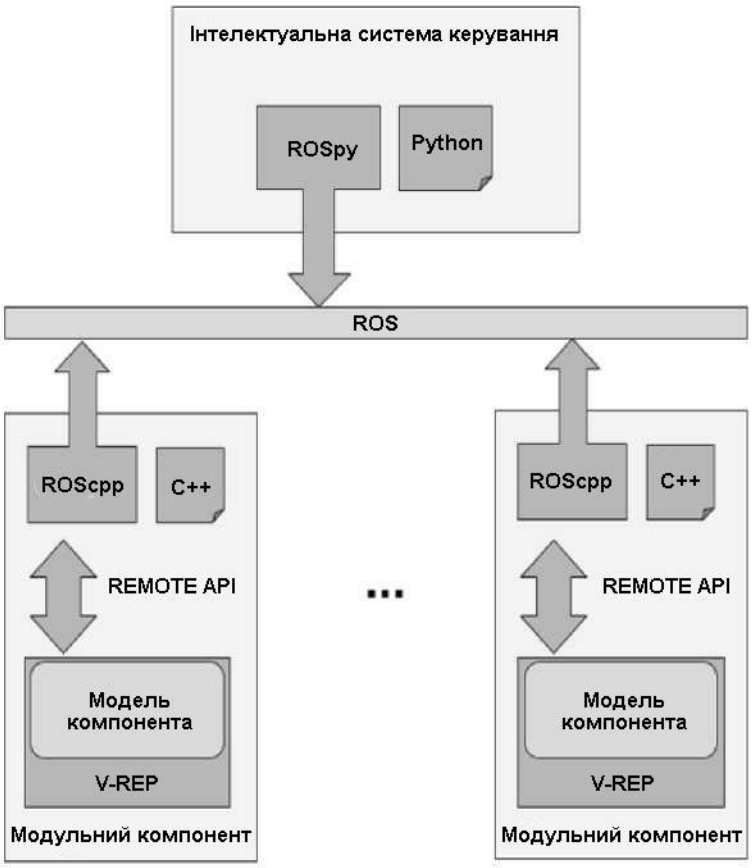

Рисунок 2.3 – Схема взаємодії віртуальних компонентів системи

Наступним етапом є натурна реалізація модульних компонентів. Для натурної реалізації частина програми з Remote API замінюється на периферійну бібліотеку. Таким чином, протестоване на моделі програмне забезпечення при перенесенні на реальний об'єкт залишається незмінним за винятком заміни блоку Remote API на периферійну бібліотеку.

Для віртуального прототипу достатньо використовувати бібліотеки ROS для C або за необхідності розробити власні, оскільки ROS є безкоштовним

продуктом із відкритим вихідним кодом. Для натурної реалізації необхідно використовувати бібліотеку mbedROS оскільки кожен модуль системи має бути вузлом ROS. а з метою здешевлення системи передбачається використовувати у модульному компоненті РТС мікроконтролер.

Після створення віртуальних інтелектуальних компонентів можна поєднувати їх у єдину систему. Сервіс ROS повинен бути запущений на одній із машин, що беруть участь в управлінні. Для натурної реалізації пропонується використовувати одноплатний комп'ютер, який є головним вузлом із запущеним ядром ROS CORE. У природній реалізації всі вузли пов'язані фізично через UART, а програмно через систему ROS TOPIC. Такий спосіб взаємодії компонентів має на увазі апаратну абстракцію та взаємодію моделей та реальних об'єктів через єдину уніфіковану систему. Такий спосіб дозволяє не тільки переносити програмне забезпечення з пакета моделювання на реальний об'єкт, але інтегрувати готові робототехнічні компоненти до робототехнічної системи під управлінням ROS. Схема взаємодії модулів системи представлена (рис.2.4).

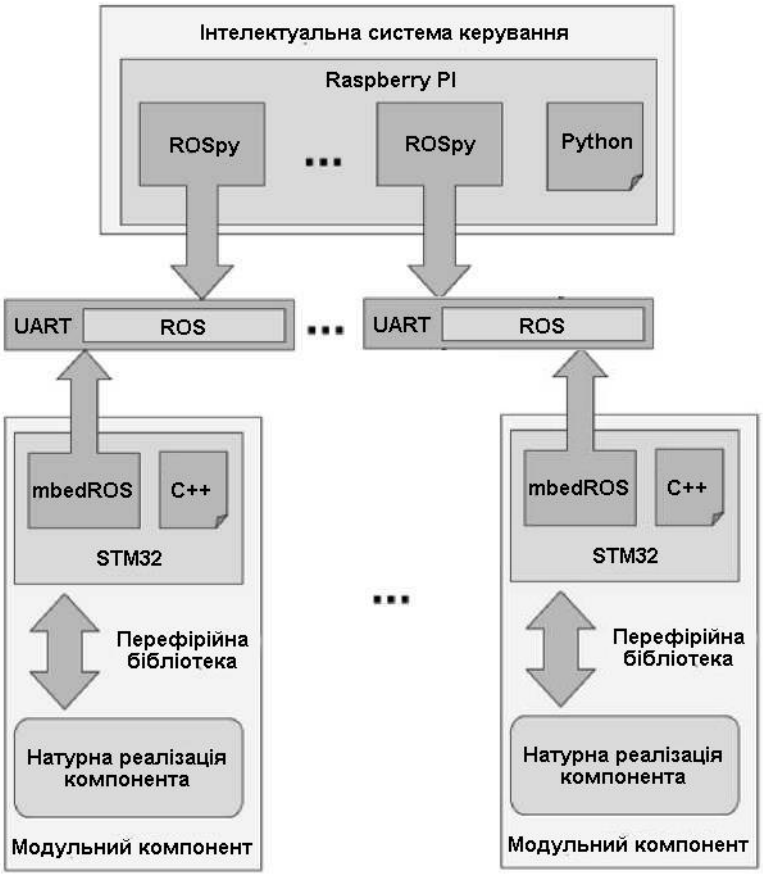

Рисунок 2.4 – Схема взаємодії реальних компонентів системи

РОЗДІЛ 3
КОНФІГУРУВАННЯ І ПАРАМЕТРИЗАЦІЯ ЗАСОБІВ ВІРТУАЛЬНОГО ПРОТОТИПІРУВАННЯРОБОТОТЕХНІЧНИХ СИСТЕМ

3.1 Конфігурація та параметризація Robot Operating System

У цій роботі автором було обрано версію ROS під назвою Kinetic для операційної системи Ubuntu 16.04.

Для використання цього інструменту необхідно його встановити та створити робоче оточення CATKIN_WS. Установка включає наступні етапи:

– налаштування sources.list для отримання пакетів із ros.org;

– втановлення ключів для підключення до сервера;

– отримання списку доступних пакетів;

– встановлення повного набору інструментів;

– перед початком використання необхідно ініціалізувати rosdep, щоб встановити усі залежності;

– щоб змінні ROS автоматично додавалися до bash сесії під час запуску терміналу, необхідно виконати команду;

– для створення та керування своїми власними пакетами, змінними та робочими оточеннями необхідно додати.

```
1  $ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(
       lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.
       list'
2  $ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80
       --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
3  $ sudo apt-get update
4  $ sudo apt-get install ros-kinetic-desktop-full
5  $ sudo rosdep init
6  $ rosdep update
7  $ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
8  $ source ~/.bashrc
9  $ sudo apt install python-rosinstall python-rosinstall-generator
       python-wstool build-essential
```

Рисунок 3.1 – Встановлення ROS

Система встановлена, але ще готова до роботи. Для роботи з операційною

системою необхідно створити робоче оточення:

– створити та зібрати робоче оточення;

– додати змінні оточення.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
$ source devel/setup.bash
```

Рисунок 3.2 – Налаштування робочого оточення

Тепер Robot Operating System готова до роботи.

3.2 Конфігурація та параметризація інструментів для MBED ROS

MBED – це платформа та операційна система для пристроїв на основі 32-бітних мікроконтролерів ARM Cortex-M. Це інструмент для швидкого та простого програмування обладнання. Використовуючи пакет rosserial_mbed, можна використовувати ROS безпосередньо із платою з підтримкою MBED. rosserial надає протокол зв'язку ROS, який працює через UART-пристрої MBED. Це дозволяє MBED бути повноцінним вузлом ROS, який може безпосередньо публікувати та підписуватись на повідомлення ROS, публікувати перетворення TF та отримувати системний час ROS. Є два методи використання бібліотеки:

Використання онлайн компілятора:

1. Необхідно імпортувати бібліотеку rosserial_mbed у свій обліковий запис компілятора, використовуючи наступне посилання:

```
$ https://developer.mbed.org/users/garyservin/code/ros_lib_kinetic/
```

Рисунок 3.3 – Посилання для імпорту бібліотеки

Використання офлайн-компілятора:

– встановити компілятор gcc4mbed;

– клонувати репозиторій rosserial у налаштований робочий простір CATKIN_WS обраного дистрибутива ROS в папку ~catkin_ws/src/;

– створити бібліотеку ros_lib. Для цього необхідно виконати команду $catkin_make у корені робочого простору. Потім $source devel/setup.bash;

– згенерувати бібліотеку ros_lib, необхідну для компіляції проекту з використанням rosserial_mbed;

– додати шляхи у змінні оточення. Необхідно перейти в папку з проектом та додати змінні gcc4mbed та ros_lib.

```
1 $ catkin_make
2 $ source devel/setup.bash
3 $ rosrun rosserial_mbed make_libraries.py ~/ros/lib
4 $ export GCC4MBED_DIR=~/gcc4mbed
5 $ export ROS_LIB_DIR=~/ros/lib/ros_lib
```

Рисунок 3.4 – Приклад конфігурування офлайн-компілятора

3.3 Конфігурація та параметризація пакета моделювання V-REP

Roslnterface є частиною інфраструктури V-REP API. Roslnterface дублює API-інтерфейс C/C++ ROS. V-REP може виступати як вузл ROS, з яким інші вузли можуть зв'язуватися через служби видавців та передплатників ROS. Функціональність Roslnterface у V-REP включена через плагін: libv_repExtRosInterface.so або libv_repExtRosInterface.dylib. Плагін може бути легко адаптований до ваших потреб. p align="justify"> Плагін завантажується при запуску V-REP, але операція завантаження буде успішною тільки в тому випадку, якщо в цей час працював roscore. Необхідно перевірити вікно консолі V-REP або термінал для отримання детальної інформації про операції завантаження плагіна.

Щоб налаштувати ROS інтерфейс у пакета V-REP, необхідно додати vrep_ros_interface до папки catkin_ws/src і виконати збірку. Щоб перевірити, чи готовий пакет до використання, необхідно запустити roscore, а потім V-REP. У лозі запуску крім інших пакетів повинен відобразитися Roslnterface.

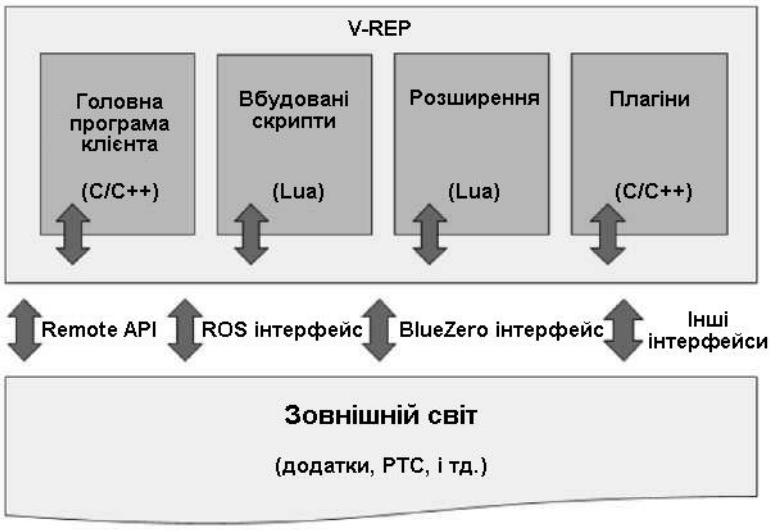

Рисунок 3.5 – Інтерфейси V-REP

```
$ ./vrep.sh
License file 'v_rep':
---> ok
Simulator launched.
Plugin 'BubbleRob': loading...
Plugin 'BubbleRob': load succeeded.
Plugin 'K3': loading...
Plugin 'K3': load succeeded.
Plugin 'RemoteApi': loading...
Plugin 'RemoteApi': load succeeded.
Plugin 'RosInterface': loading...
Plugin 'RosInterface': load succeeded.
```

Рисунок 3.6 – Приклад коректного запуску V-REP із інтерфейсом ROS

3.4 Етапи застосування засобів віртуального прототипування

Автором пропонується проект з наступною структурою:

Рисунок 3.7 – Схема взаємодії демонстраційного проекту

Вузол шле в Topic під назвою /chatter повідомлення “Hello World!” та сигналізує при успішному виконанні операції.

Для віртуального прототипу використовується сфера, яка під час надсилання повідомлення змінює колір.

Для натурної реалізації налагоджувальна плата при надсиланні повідомлення змінює стан світлодіода вкл/викл.

У середовищі віртуального прототипування V-REP створюється сфера і створюється скрипт, який приймає сигнал по REMOTE API від керуючої програми.

Скрипт, який на стороні V-REP відповідає за прийом повідомлень та зміну кольору представлений нижче:

```
1  function sysCall_init()
2
3    simRemoteApi.start(19997)
4
5  end
```

Рисунок 3.8 – Ініціалізація REMOTE API

```
1  if v==1 then
2
3    simSetShapeColor(sphereHandle, nil, 0, green)
4
5  end
6
7  if v==0 then
8
9    simSetShapeColor(sphereHandle, nil, 0, red)
10
11 end
```

Рисунок 3.9 – Блок, який змінює колір сфери, залежить від отриманого сигналу.

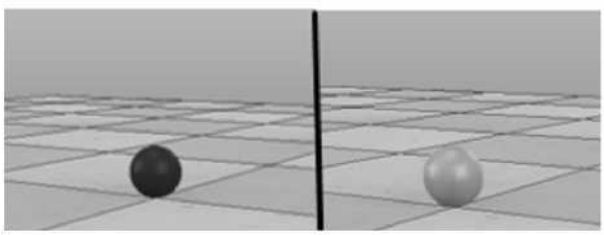

Рисунок 3.10 – Зміна кольору сфери при надсиланні повідомлень

Наступним етапом є написання програми для надсилання повідомлень та

сигналізації про відправку шляхом зміни кольору сфери. Програма складається з наступних блоків:

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "std_msgs/UInt8.h"

#include <sstream>
#include <stdio.h>
#include <stdlib.h>

extern "C" {
#include "remoteApi/extApi.h"
#include "remoteApi/extApiPlatform.h"
#include "remoteApi/extApiInternal.h"
}
```

Рисунок 3.11 – Ініціалізація REMOTE API

```
int portNb = 19997;
int clientID = simxStart ("127.0.0.1", portNb, true, true, 5000, 5);
int led=0;
int x=0;
```

Рисунок 3.12 – Ініціалізація REMOTE API

```
int main(int argc, char ** argv) {

   ros::init(argc,argv,"chatter");
   ros::NodeHandle  nh;
   std_msgs::String str_msg;
   ros::Publisher chatter = nh.advertise<std_msgs::String>("chatter"
        , 1000);
```

Рисунок 3.13 – Ініціалізація REMOTE API

```
   while (1) {

    ...

       x=1-x;
       simxSetIntegerSignal(clientID,"sig",x,simx_opmode_streaming);
       chatter.publish( str_msg );
       ros::spinOnce();
       usleep(1000000);
   }
}
```

Рисунок 3.14 – Ініціалізація REMOTE API

Після запуску програми у списку доступних Topic'oB буде доступним /chatter. За допомогою команди rostopic echo/chatter можна перевірити повідомлення. Таким чином у терміналі буде виводитися раз на секунду повідомлення “Hello World!”, а в пакеті моделювання щоразу при надсиланні

повідомлення сфера змінюватиме колір. Наступним етапом буде перевірка натурної реалізації. Для натурної реалізації Автором було обрано налагоджувальну плату STM32 DISCOVERY та одноплатний комп'ютер RASPBERRY PI 3. Схематичне зображення підключення представлено на (рис.3.15).

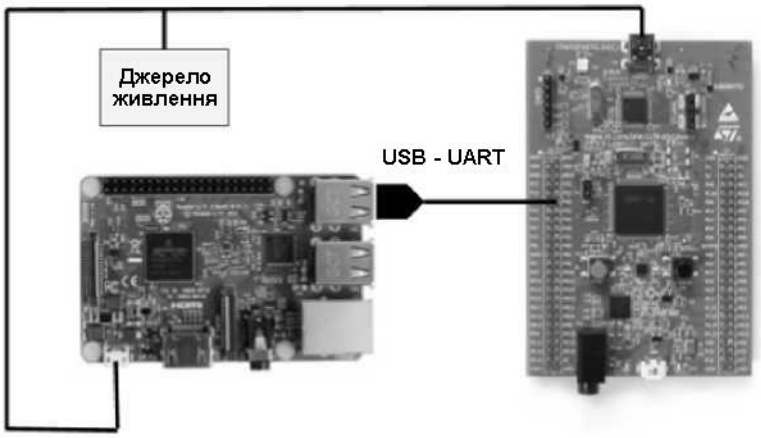

Рисунок 3.15 – Схема підключення натурної реалізації

Для натурної реалізації необхідно програму, що відповідає за надсилання повідомлень та індикацію адаптувати мікроконтролера. Основні блоки програми представлені нижче:

```
1  #include"mbed.h"
2  #include <ros.h>
3  #include <std_msgs/String.h>
```

Рисунок 3.16 – Підключення бібліотек

```
1  ros::NodeHandle  nh;
2  std_msgs::String str_msg;
3  ros::Publisher chatter("chatter", &str_msg);
```

Рисунок 3.17 – Ініціалізація ROS

```
1  DigitalOut led = LED2;
2
3
4      nh.initNode();
5      nh.advertise(chatter);
```

Рисунок 3.18 – Ініціалізація світлодіода налагоджувальної плати, ініціалізація вузла ROS

```
    while (1) {
        led = !led;
        str_msg.data = hello;
        chatter.publish( &str_msg );
        nh.spinOnce();
        wait_ms(1000);
    }
```

Рисунок 3.19 – Основна частина програми, відправлення повідомлення в ROS Topic, перемикання світлодіода-індикатора

Після написання програми для мікроконтролера необхідно налаштувати файл make. Бібліотека mbed підтримує безліч мікроконтролерів, список яких можна переглянути на сайті бібліотеки. Вибрана для тестового прикладу налагоджувальна плата присутня в бібліотеці, тому достатньо вибрати її для компіляції, make файл виглядатиме так:

```
PROJECT          := rosserial_mbed_HelloWorld
DEVICES          := DISCO_F401VC
GCC4MBED_DIR     := /media/evgenylin/LINUX/adamgreen-gcc4mbed-7d79ef3
USER_LIBS        := !$(ROS_LIB_DIR) $(ROS_LIB_DIR)/BufferedSerial
NO_FLOAT_SCANF   := 1
NO_FLOAT_PRINTF  := 1

include $(GCC4MBED_DIR)/build/gcc4mbed.mk
```

Рисунок 3.20 – Makefile для складання проекту Далі необхідно скомпілювати програму для завантаження.

Це досягається виконаному команди make, після чого можна завантажити .elf файл у мікроконтролер.

Після завантаження програми на мікроконтролер та складання тестового макета необхідно ввести команду в терміналі одноплатного комп'ютера для ініціалізації Topic'а в оточенні ROS, підключеного за UART.

```
evg@evg-desktop:~/Desktop$ rostopic list
/rosout
/rosout_agg
```

Рисунок 3.21 – ROS Topic list до ініціалізації підключення UART

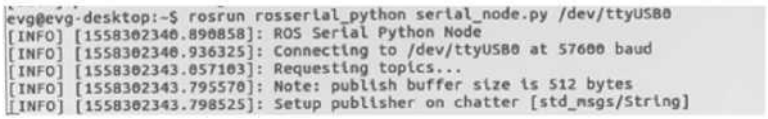

Рисунок 3.22 – Установка зв'язку з вузлом ROS UART

Рисунок 3.23 – ROS Topic list після ініціалізації підключення за UART

За допомогою команди rostopic echo/chatter можна перевірити повідомлення. У терміналі буде із заданою періодичністю виводитись повідомлення “Hello world!”, а світлодіод на налагоджувальній платі змінюватиме свій стан вкл/викл щоразу при надсиланні повідомлення.

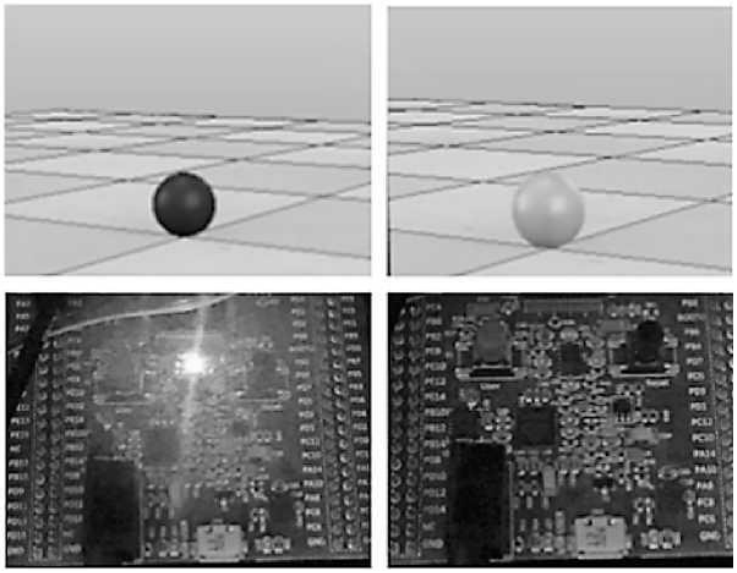

Рисунок 3.24 – Зміна індикації для віртуального прототипу (згори) та натурної реалізації (знизу)

Таким чином, обидва вузли, реалізовані на різних платформах, поводяться взаємоподібним чином. Програмний код написаний для віртуального прототипу, відрізняється від програмного коду, написаного для натурної реалізації, тільки платформозалежною частиною, що включає апаратні бібліотеки і бібліотеки для

комунікації з пакетом моделювання.

3.5 Розробка симетричної крочої роботизованої платформи з використанням засобів віртуального прототипування та її дослідження

У бакалаврській випускній роботі Автором була представлена модель симетричної крокуючої роботизованої платформи (СШРП) та прототип натурної реалізації. У наступних розділах описано процес створення модульного прототипу СШРП та його віртуального прототипу.

У ході попередньої роботи було створено креслення СТД та роздруковано на 3D принтері у кількості 4 штук. Двигун було вирішено продавати на основі сервоприводів компанії Dynamixel моделі A-12. оскільки такі сервоприводи працюють у командному режимі.

Рисунок 3.25 – Сервопривід Dynamixel AX-12

Сервоприводи об'єднуються в єдину мережу, все управління відбувається шляхом відправки на цільовий сервопривід пакета даних, що містить команди.

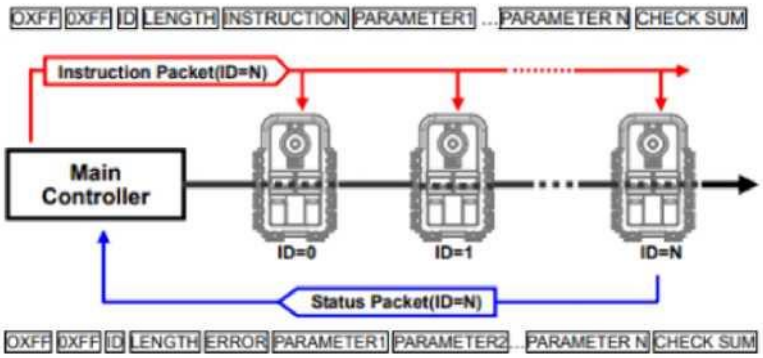

Рисунок 3.26 – Схематичне зображення обміну даними у Dyriamixel AX-12

В результаті складання було створено 4 прототипи СТД. забезпечених сервоприводами.

Рисунок 3.27 – Натурна реалізація СТД

Оскільки у роботі всі елементи конструкції РТС є модульними компонентами, прототип СТД доповнюється микроконтроллером STM32. який дозволить реалізувати вузол ROS для повідомлення компонент через єдину мережу. Схематичне зображення апаратної частини модульного компонента представлено (рис.3.28).

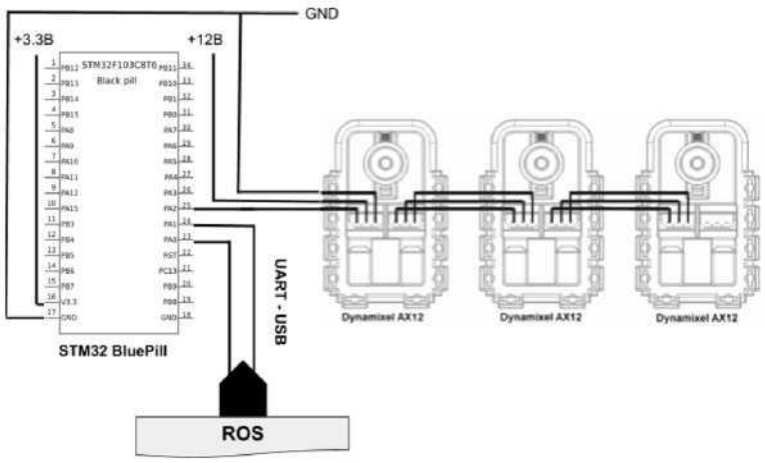

Рисунок 3.28 – Схематичне зображення апаратної частини модульного СТД

Для спрощення підключення кожен компонент використовує перетворювач UART-USB, що дозволяє позбавитися великої кількості проводів. Модульний СТД при підключенні до керуючого пристрою використовує USB-роз'єм. провід заземлення та провід живлення – 12В.

3.6 Розробка віртуального прототипу симетричного трикоординатного рушія

Для створення модульного інтелектуального компонента для бібліотеки готових рішень необхідно створити модель з примітивів. представлених у пакеті моделювання. Кожен примітив має свої параметри. Для створення якісної моделі необхідно враховувати ці параметри. Усі елементи конструкції поєднуються в ієрархію. У елемента, що знаходиться на вершині ієрархії, необхідно вказати, що він є базовим елементом конструкції, щоб модель стала цілісною.

Після створення нового компонента його можна додати до бібліотеки до пакета моделювання. Для цього потрібно вибрати об'єкт на сцені та натиснути “save as model”. Модель стане доступною для використання і відображатиметься у спеціальному вікні. Її можна додати до сцени, перемістивши в потрібне місце з меню вибору компонентів.

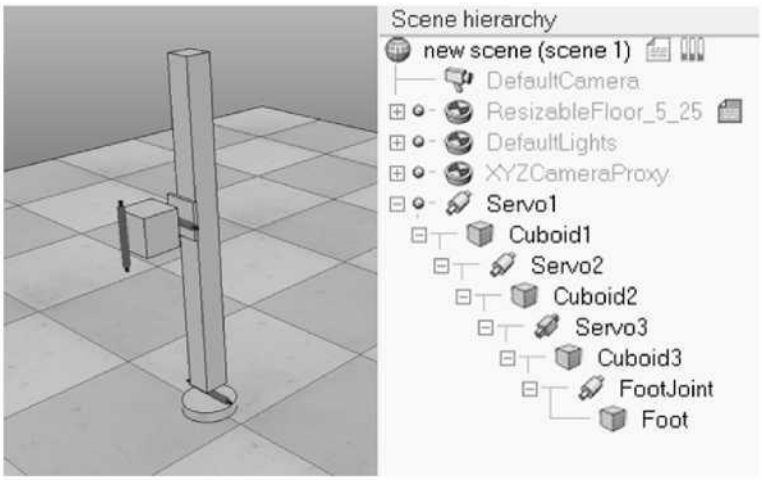

Рисунок 3.29 – Модель СТД у пакеті V-REP

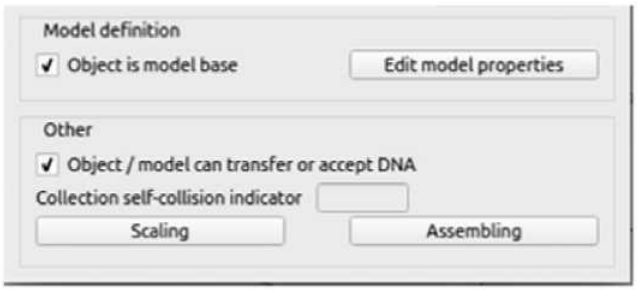

Рисунок 3.30 – Параметри базового елемента конструкції

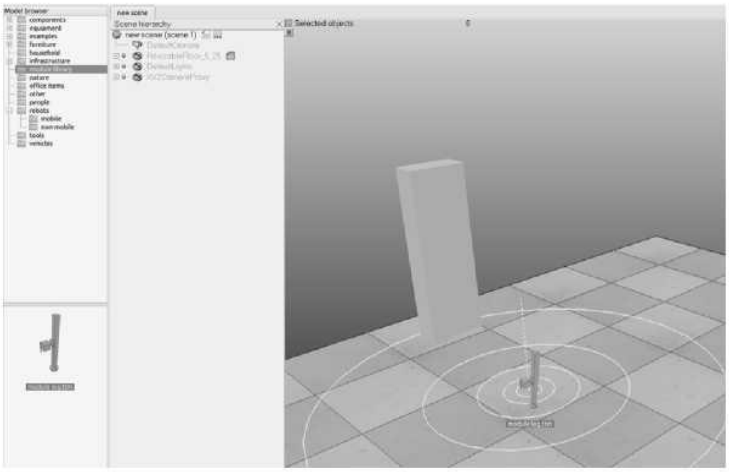

Рисунок 3.31 – Додавання компонента з бібліотеки

Після створення моделі компонента необхідно описати низькорівневу частину керування. Для можливості використання програмного коду надалі автором було обрано мову програмування С.

Для створення універсального модульного компонента необхідно налаштувати взаємодію через ROS. У ROS компоненти взаємодіють за допомогою системи Publisher – Subscriber. Publisher транслює команди Topic, а Subscriber зчитує інформацію з потрібного Topic'а. Приклад подано на (рис.3.32).

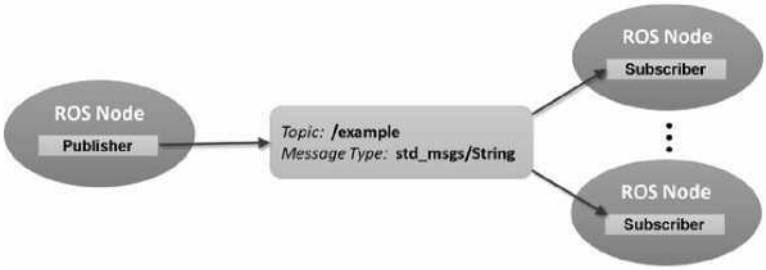

Рисунок 3.32 – Приклад взаємодії елементів ROS

Для використання ROS та Remote API необхідно підключити необхідні бібліотеки

```
#include "ros/ros.h"
#include <ros/service.h>

extern "C" {
#include "remoteApi/extApi.h"
#include "remoteApi/extApiPlatform.h"
#include "remoteApi/extApiInternal.h"
}
```

Рисунок 3.33 – Підключення бібліотек

Система ROS побудована в такий спосіб, що обробляє повідомлення через callback'll. Тому потрібно ініціалізувати callback під потрібний тип повідомлень

```
void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ....
}
```

Рисунок 3.34 – Ініціалізація callback'a

У більшості програм описуються назви Topic'ов. на які підписуватиметься модуль та назва вузла.

```
int main (int argc, char ** argv)
{
  ros::init(....);
  ros::NodeHandle n;
  ros::Subscriber sub = n.subscribe("head_to_leg#", 1000, Callback);

  ....
```

Рисунок 3.35 – Ініціалізація ROS

Після ініціалізації та налаштування ROS вказуються назви рушіїв із моделі компонента для керування через REMOTE API.

```
1  simxGetObjectHandle(clientID, "Servo1#", &Servo,
       simx_opmode_blocking);
2  simxGetObjectHandle(clientID, "Servo2#", &Servo,
       simx_opmode_blocking);
3  simxGetObjectHandle(clientID, "Servo3#", &Servo,
       simx_opmode_blocking);
4
5    ....
6  }
```

Рисунок 3.36 – Ініціалізація віртуальних сервоприводів

Після налаштування всіх рівнів взаємодії між моделлю та вузлами ROS необхідно розробити алгоритм низькорівневого керування окремого модуля. Програмний код для СТД представлений у додатку.

Таким чином, отримано віртуальний двійник фізичного елемента конструкції, який є модульним компонентом і спілкується з майстром через ROS високорівневими командами, що дозволяє знизити трудомісткість при проектуванні РТС з таких компонентів.

3.7 Створення віртуального прототипу симетричної крокуючої роботизованої платформи із застосуванням бібліотеки готових рішень

Наступним етапом є створення моделі робототехнічної системи із модульних компонентів, що входять до складу бібліотеки готових рішень.

При додаванні однакових компонентів вони автоматично нумеруються у порядку зростання. Це дуже зручно при використанні компонентів робототехнічної системи з кількома однаковими модулями. Результат об'єднання чотирьох СТД у симетричну крокуючу роботизовану платформу представлений на рис.3.37).

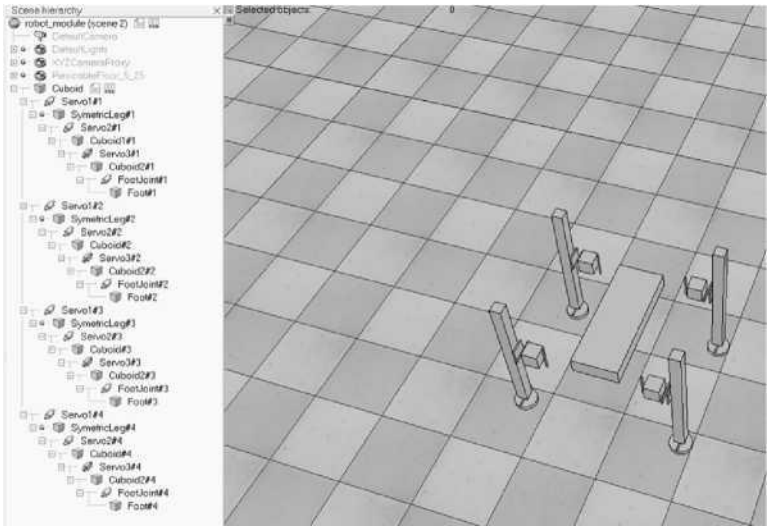

Рисунок 3.37 – Віртуальний прототип СШРП

Наступним етапом проектування є розробка алгоритму керування високого рівня. Для керування системою підходить будь-яка мова програмування за допомогою ROS. Автором було обрано мову Python. Спочатку необхідно додати бібліотеки для підтримки пакета моделювання та Robot Operating System.

```
1 import rospy
2 import vrep
```

Рисунок 3.38 – Підключення бібліотек

Потім відбувається ініціалізація Topic'ів для керування модулями.

```
1 pub_leg1 = rospy.Publisher('head_to_leg1', String, queue_size=10)
2 pub_leg2 = rospy.Publisher('head_to_leg2', String, queue_size=10)
3 pub_leg3 = rospy.Publisher('head_to_leg3', String, queue_size=10)
4 pub_leg4 = rospy.Publisher('head_to_leg4', String, queue_size=10)
```

Рисунок 3.39 – Ініціалізація Торю'ів

Для управління модулем необхідно відправляти команду Topic, який підписаний модуль.

```
1  simxGetObjectHandle(clientID, "Servo1#", &Servo,
       simx_opmode_blocking);
2  simxGetObjectHandle(clientID, "Servo2#", &Servo,
       simx_opmode_blocking);
3  simxGetObjectHandle(clientID, "Servo3#", &Servo,
       simx_opmode_blocking);
4
5    ....
6  }
```

Рисунок 3.40 – Відправлення команди

Повний програмний код представлений у додатку. Після конструювання моделі робототехнічної системи із готових модульних компонентів та розробки високорівневих алгоритмів управління можна протестувати отриману систему.

Коли ROS працює командою rostopic list, можна перевірити активні Topic'll. Бо не модульні компоненти PTC. ні майстер не активні, відображаються лише системні Topic'll.

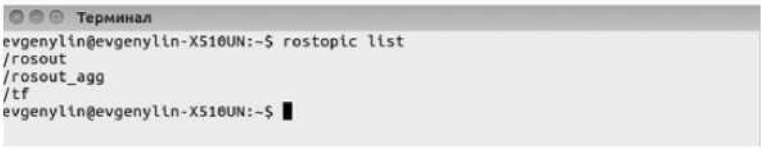

Рисунок 3.41 – Службові Topic'll під час запуску ROS

Після запуску моделі СШРП у ROS створюються Topic'll:

– у Topic'll head_to_leg# транслюються команди кожному СТД від майстра;

– у Topic'll leg#_to_head передаються дані зворотного зв'язку про свій стан майстру;

– у Topic head_bodyFront транслюються спільні команди для коригування положення тулуба;

– у Topic legs_bodyFront транслюються стани СТД для коригування положення тулуба.

Рисунок 3.42 – Topic'll моделі СШРП

Для наочності взаємодії між вузлами системи можна викликати команду rqt_graph. яка автоматично збудує граф з вузлами ROS.

На (рис.3.43) видно, як вузли пов'язані один з одним Topic'ами. Модульні компоненти мають на увазі спілкування високорівневими командами. Це дозволяє знизити трудомісткість розробки робототехнічних систем, оскільки дозволяє розробнику зосередитись на стратегічному рівні керування роботом. Розглянемо систему команд та поведінку моделі докладніше на (рис.3.44).

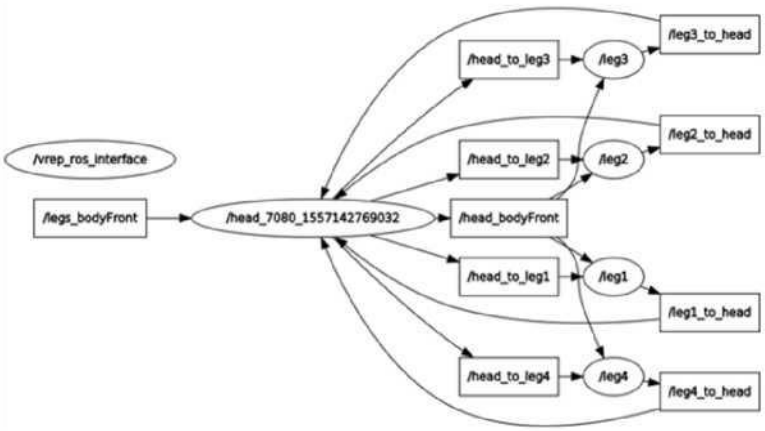

Рисунок 3.43 – Граф взаємодії вузлів ROS для СШРП

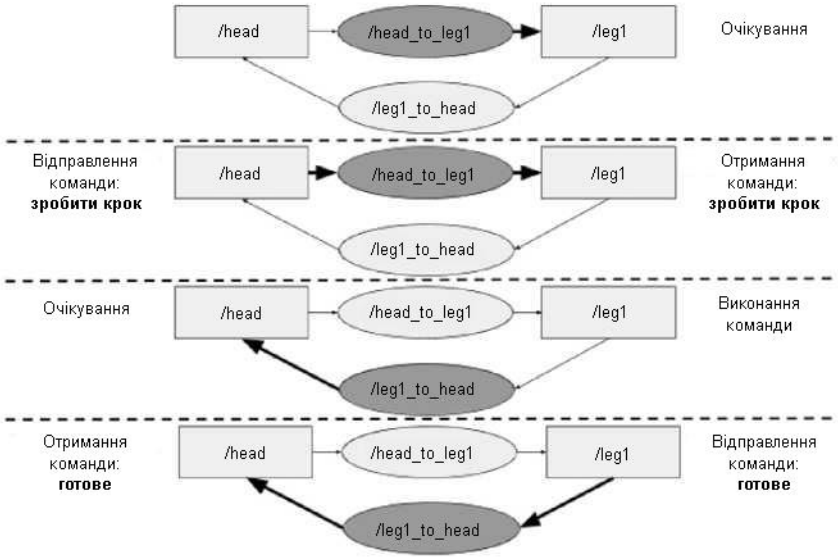

Рисунок 3.44 – Взаємодія вузлів моделі СШРП

Оскільки програма управління модульним СТД уніфікована всім чотирьох елементів конструкції, вона вміє працювати з такими командами:

– clockwise_step для кроку за годинниковою стрілкою;

– counterclockwise_step для кроку проти годинникової стрілки;

– body_forward для переміщення основи конструкції вперед щодо рушіїв;

– body_backward для переміщення основи конструкції назад оносителя рушіїв.

3.8 Натурна реалізація симетричної крокуючої роботизованої платформи

Для керування модульними СТД було обрано одноплатний комп'ютер Raspberry PI. До керуючого пристрою підключаються 4 компоненти USB і до живлення -12В. Сам керуючий пристрій підключається до живлення - 5В. Схематичне зображення апаратної частини СШРП представлено (рис.3.45).

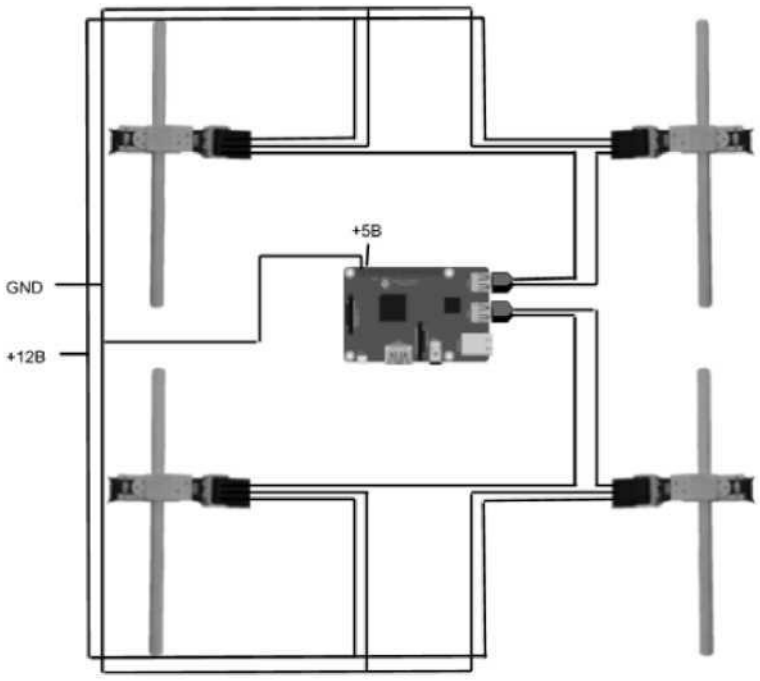

Рисунок 3.45 – Схематичне зображення апаратної частини СШРП

При складанні з'ясувалося, що пластмасові СТД ковзають по більшості поверхонь, тому було прийнято рішення забезпечити кожен рушій нековзною поверхнею. Результат складання модульної симетричної крокуючої роботизованої платформи представлений на рис.3.46).

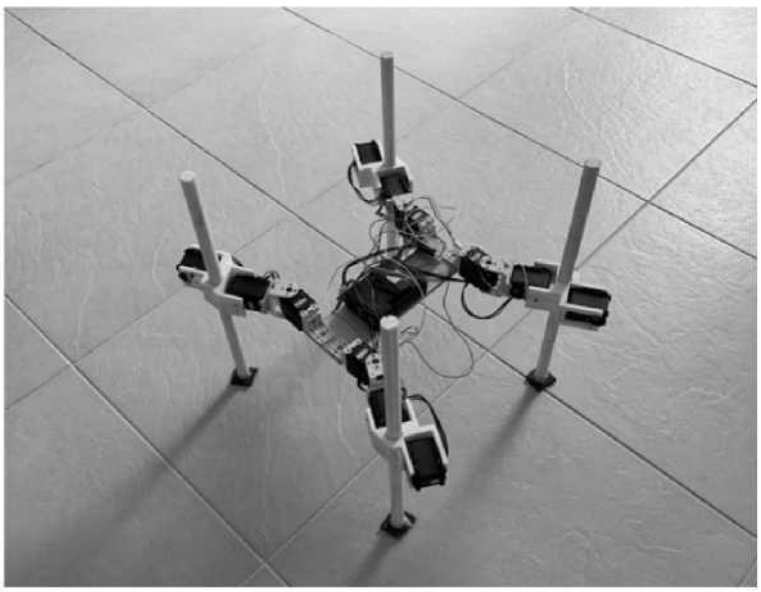

Рисунок 3.46 – Натурна реалізація СШРП

Наступним етапом є розробка програмного забезпечення для керування

крокуючою платформою. Для взаємодії між компонентами Автором пропонується використовувати ROS, оскільки використання міжплатформного програмного забезпечення дозволяє абстрагуватися від апаратної частини розробки керуючого ПЗ. Це дозволить розробити софт, який зможе взаємодіяти із компонентами РТС в уніфікованій формі.

Завдяки засобам віртуального прототипування при розробці РТС можна випробовувати програму, що управляє, на віртуальному прототипі, що дозволить уникнути псування дорогих елементів конструкції, а також прискорити експериментальні дослідження ПЗ. Таким чином, наступним етапом розробки системи буде створення віртуального прототипу та випробування на ньому програм управління.

Після розробки та дослідження алгоритму керування для віртуального прототипу СТД та віртуального прототипу роботизованої платформи можна перейти до заключної фази. Втілення натурної реалізації на основі результатів, отриманих під час роботи з віртуальним прототипом.

Оскільки ROS забезпечує апаратну абстракцію, необхідно замінити лише програмний код вузла ROS, який підключається до пакету віртуального моделювання, бібліотекою для взаємодії з сервоприводами Dynamixel. MB ROS має драйвери для даних сервоприводів. За основу береться програмний код, розроблений для віртуального прототипу СТД. Основні зміни у програмному коді представлені нижче.

```
1  #include "mbed.h"
2  #include "AX12.h"
```

Рисунок 3.47 – Підключення бібліотек

```
1  AX12 servo1 (PA_9, PA_10, 1);
2  AX12 servo2 (PA_9, PA_10, 2);
3  AX12 servo3 (PA_9, PA_10, 3);
```

Рисунок 3.48 – Ініціалізація сервоприводів

```
servo1.SetGoal(0);     // go to 0 degrees
wait (1.0);
servo1.SetGoal(300);   // go to 300 degrees
wait (1.0);
```

Рисунок 3.49 – Зміна кута повороту сервоприводу

Далі необхідно скомпілювати програму управління та завантажити на всі 4 СТД отримані hex-файли. Після підключення модульних рушіїв USB до керуючого пристрою необхідно ініціалізувати Topic'n ROS UART за допомогою команди, представленої в демо-прикладі в розділі 3.4.

Програма управління симетричною платформою, що крокує роботизованою залишається без змін, оскільки виконується на одноплатному комп'ютері Raspberry PI 3. Програмний код представлений в додатку. Після запуску системи можна перевірити працездатність натурної реалізації СШРП після віртуального прототипування та розробки керування модульною платформою з інтелектуальними компонентами.

Як показано на рис.3.50-3.52)Віртуальний прототип і натурна реалізація поводяться подібно.

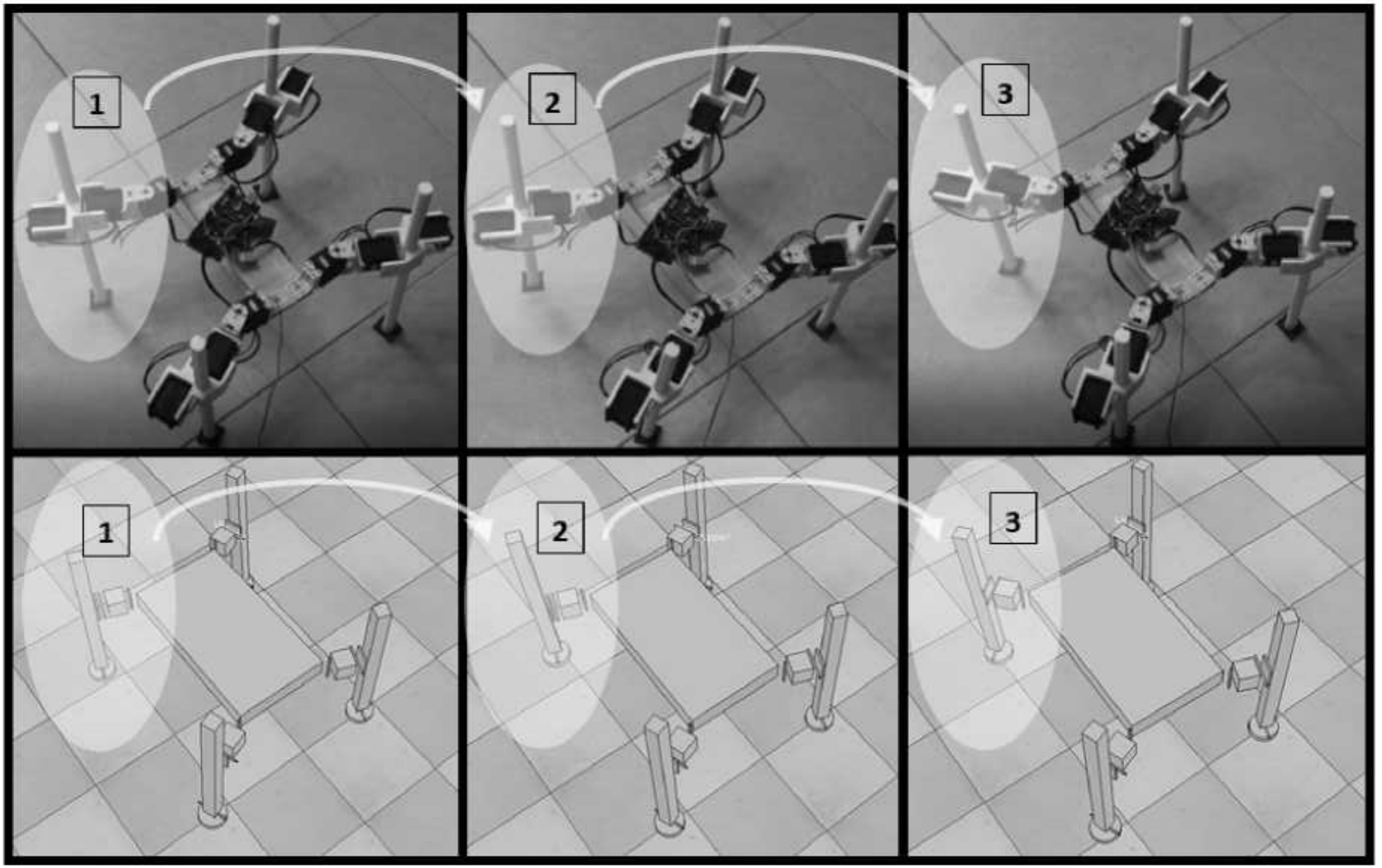

Рисунок 3.50 – Крокувальний цикл СШРП, перший крок

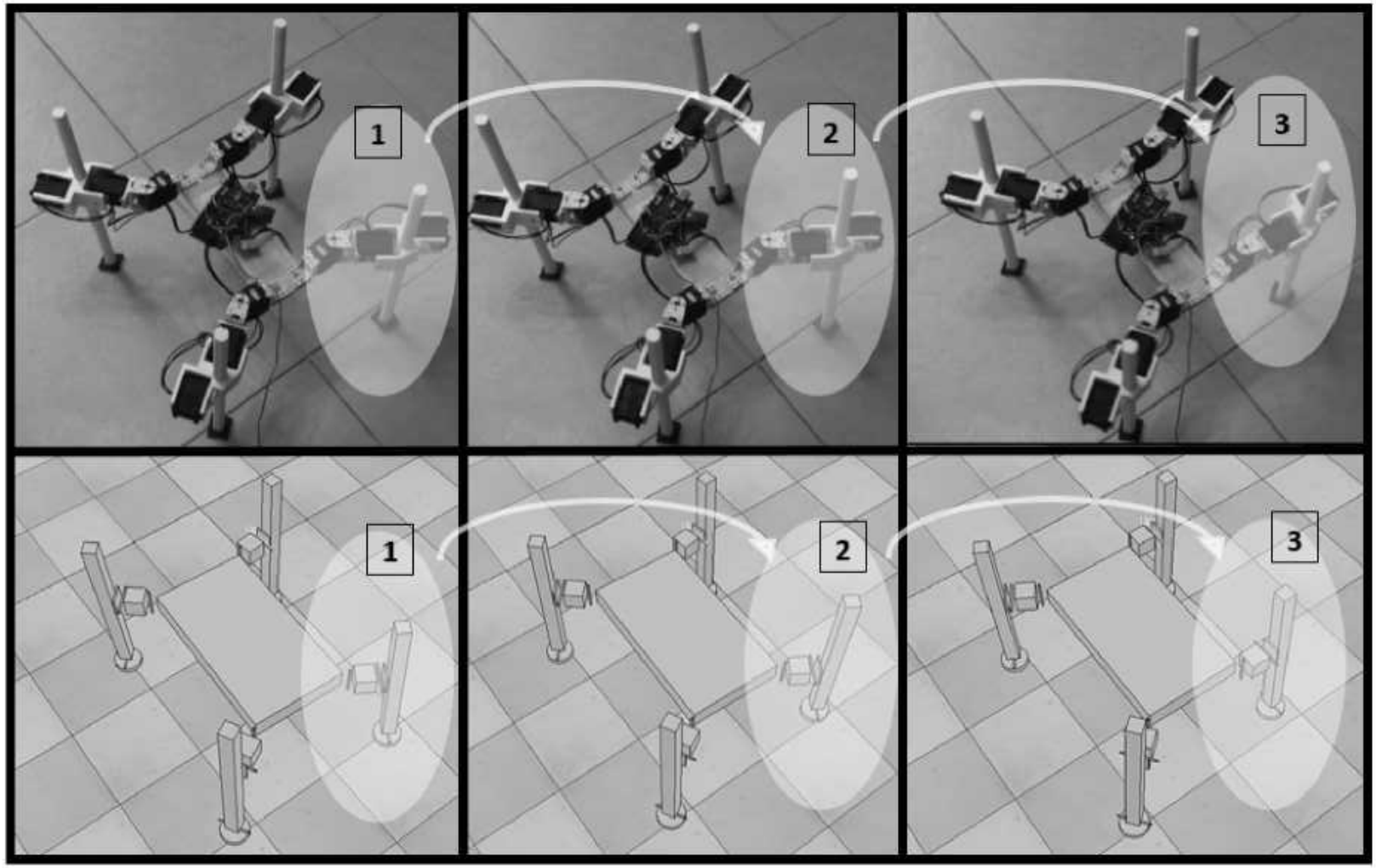

Рисунок 3.51 – Крокувальний цикл СШРП, другий крок

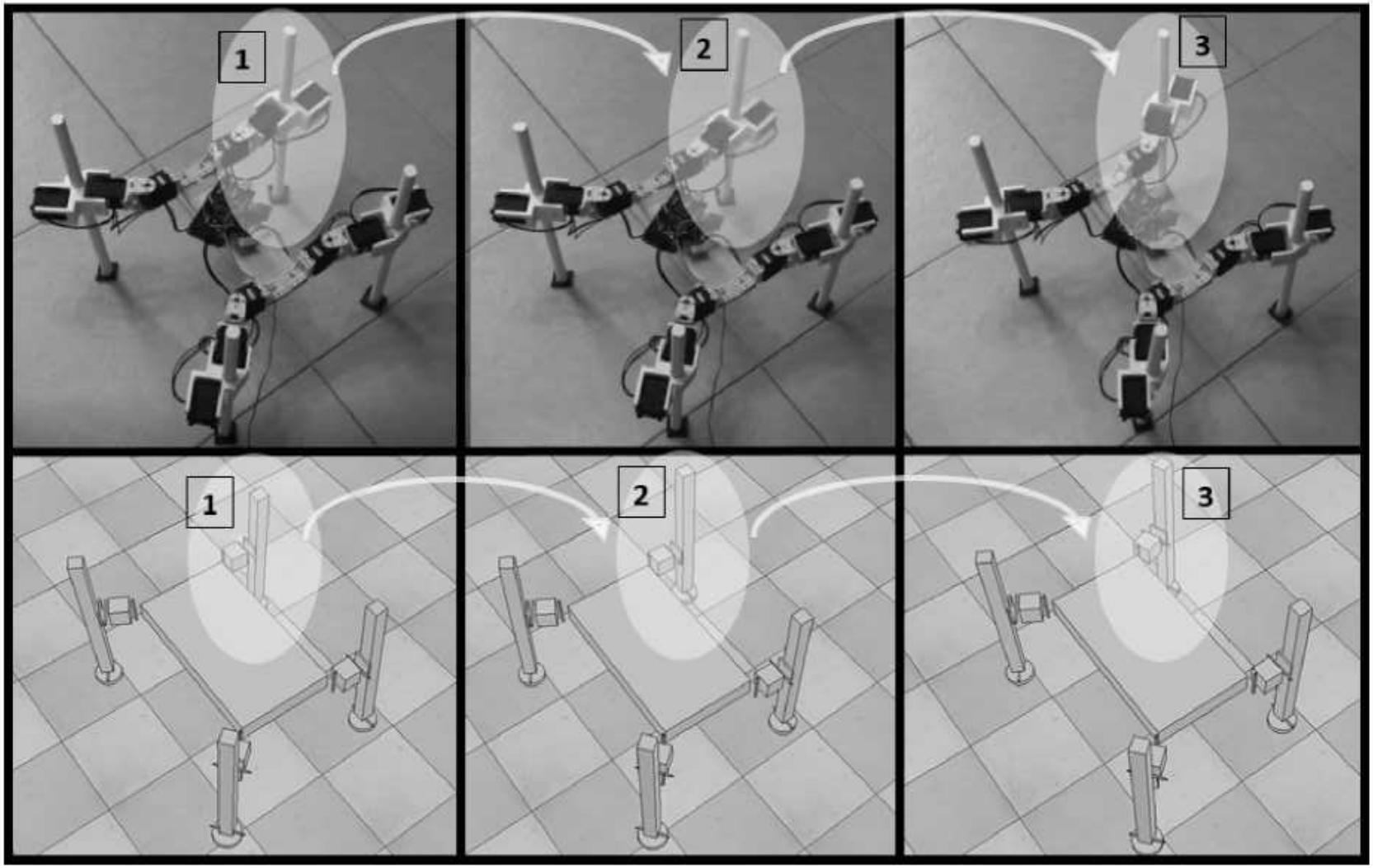

Рисунок 3.52 – Крокувальний цикл СШРП, третій крок

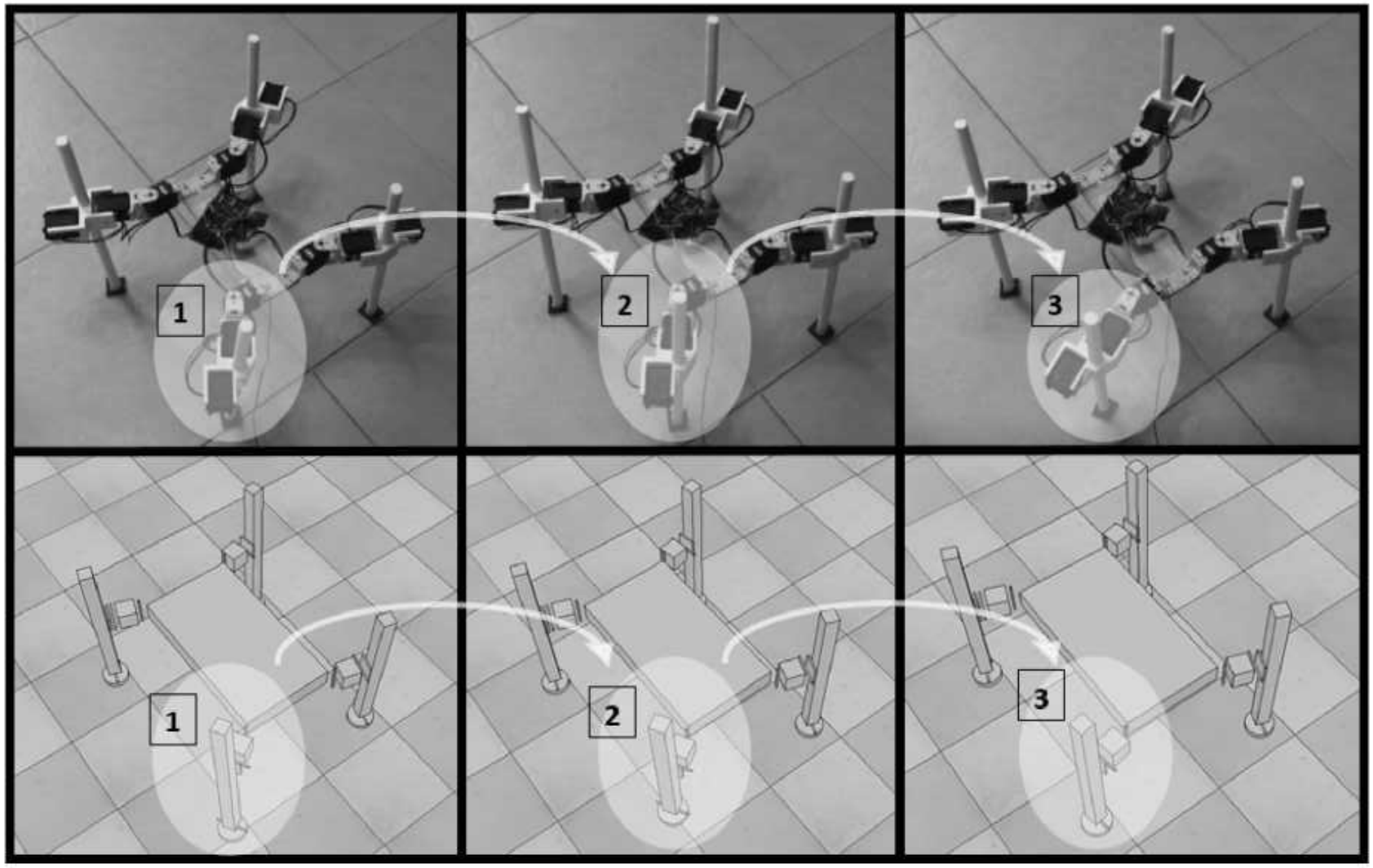

Рисунок 3.53 – Крокувальний цикл СШРП, четвертий крок

РОЗДІЛ 4
ОХОРОНА ПРАЦІ

Охорона праці – це система правових соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці.

Правовою основою законодавчих актів про охорону праці є Конституція України і Кодекс законів паро працю України, а також такі закони:

- Закон пр. охорону праці;
- Закон про використання ядерної енергії і радіаційного захисту;
- Закон про охорону здоров'я;
- Закон про пожежну безпеку;
- Закон про забезпечення санітарного та епідеміологічного благополуччя населення;

Закон про загальноосвітнє державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які призвели до втрати працездатності.

У ст. 43 Конституції записано: «Кожний має право на працю, що включає можливість заробляти собі на життя працею, що він вільно обирає або на який вільно погоджується», «Кожний має право на належні, безпечні і здорові умови праці, на заробітну плату не нижче тієї, котра визначена законом», «Використанні праці жінок і неповнолітніх на небезпечних для їхнього здоров'я роботах забороняється».

Кожний хто працює, має право на відпочинок (ст. 45 Конституції). Це право забезпечується наданням днів щотижневого відпочинку, а також оплачуваної щорічної відпустки, установленням скороченого робочого дня для окремих професій і виробництв, скороченою тривалістю роботи в нічний час.

Основним документом у галузі охорони праці є Закон України «Про охорону праці», дія якого поширюється на всі галузі підприємства, установи і

організації незалежно від форм власності і видів їхньої діяльності, на всіх громадян, що працюють, а також притягнути до роботи на цих підприємствах.

4.1 Аналіз небезпечних та шкідливих факторів, що впливають на людину при роботі в лабораторії і дії людини на випадок пожежі

Особливістю негативного впливу комп'ютерних технологій на працездатність і здоров'я людини є комплексна одночасна дія декількох шкідливих факторів, при значній інтенсивності яких відбувається накопичення акумулювання їх впливу, викликає суттєві зміни в організмі людини, розлад функцій окремих органі і систем.

До основних негативних факторів належать: вимірювання різних діапазонів електромагнітного спектру (рентгенівське та оптичне випромінювання, високочастотні та низькочастотні ЕМП, ЕМП з надто низькими факторами виробничого середовища, нервово емоційна напруга і т.д.

На працездатність та самопочуття негативно впливає шум від роботи електронно-обчислювальних машин. При цьому тривала дія шуму призводить до зниження розумової працездатності на 10-15 %, швидкої зорової втоми, послаблення уваги, порушення психофізіологічних процесів. Вплив шуму ВДТ є однією із причин розвитку стресу, погіршення настрою, сенсорного перевантаження, змін кровопостачання тканин і органів у зв'язку зі спазмами капілярів.

Професійна діяльність працівника на ВДТ є причино. Функціональних змін нервово-м'язового апарата і кровопостачання ока, які призводять до розвитку астеноптичних скарг. Встановлено, що жінки частіше, ніж чоловіки, скаржаться на зоровий дискомфорт. При цьому відмічено, що в більшості випадків частота астенопії зростає зі збільшенням тривалості роботи за ВДТ.

Астеноптичні скарги пов'язані також з освітленням робочого місця, відблиском екрану, тремтінням та мерехтінням зображення, сухістю повітря тощо. Встановлено, що у 72 % користувачів ВДТ мають місце скарги на біль в

очах. Результатом напруженої тривалої зорової роботи на ЕОМ може бути не лише специфічний зоровий дискомфорт, але й виникнення головного болю.

До психологічних і поведінкових розладів відносяться: агресивність, фрустрація, нервозність, дратівливість, порушення сну, швидкий розвиток втоми тощо.

При виникненні пожежі самими небезпечними факторами є токсичні продукти горіння (оксиди вуглецю, ціанід водню та інші).

Тому впершу чергу необхідно терміново залишити активну зону горіння. Перед тим, як вийти з приміщення, потрібно перекрити газ та вимкнути усі електроприлади, а краще здійснити повне знеструмлення об’єкту. При виході щільно закрити двері.

Якщо пожеже виникла на вашому поверсі і безпосередньої загрози для працівників немає, то потрібно здійснити запобіжні заходи від можливого негативного впливу води, яку використовують для гасіння. При цьому знеструмлюють приміщення, відсувають від стін меблі, накривають обладнання і предмети захисною плівкою або іншими підручними засобами. На випадок виникнення пожежі на нижньому поверсі, виникає загроза негативного впливу диму на людей та перешкода для їх евакуації. У такому разі приміщення потрібно негайно залишити, але перед тим, як виходити через двері, їх треба трохи при відчинити (ні в якому разі не можна різко відкривати або вибивати двері, бо миттєвий доступ кисню може викликати викид полум’я).тому підчас пожежі двері треба відчиняти обережно з урахуванням перепаду температури і впливу полум’я.

При рятуванні потерпілих з будівель, що горять, та при гасінні пожежі виконувати наступні правила:

– перед тим як увійти у палаюче приміщення накритися з головою мокрим покривало, плащем тощо;

– двері в задимлене приміщення відкривати обережно, поволі, прикриваючи корпус тіла дверним полотном для того, щоб уникнути спалаху полум’я від швидкого приливу свіжого повітря;

– у дуже задимленому приміщенні пересуватись поповзом або схилившись, бо більшість нагрітих газоподібних речовин та дим скупчуються у верхній частині приміщення;

– якщо виникло займання одягу, лягти на землю (підлогу) та перекочуватися для збиття полум'я (ні в якому разі не бігти, тому що полум'я розгориться ще сильніше);

– при гасінні пожежі використовувати вогнегасники та інші засоби гасіння за призначенням, спрямуванням їх на поверхню, що горить.

4.2 Розрахунок освітлення лабораторії

Маємо приміщення з таким розмірами:

а) довжина – А = 6 м;

б) ширина В = 4 м;

в) висота Н = 3 м;

г) висота робочої поверхні h_P = 1 м.

Для освітлення приймаємо світильник типу УМП-15.мінімальна освітленість лампи розжарювання за нормами E_{min} = 300 лк.

Коефіцієнт відображення стелі $\rho_П$=70%, стін $\rho_С$=50%, робочої поверхні $\rho_Р$=10%.

Напруга живлення 220 В.

Відстань від стелі до робочої поверхні:

$$H_0 = H - h_P = 3 - 1 = 2 \text{ м.}$$

Відстань від стелі до світильника:

$$h_C = 0,2 \cdot H_0 = 0,2 \cdot 2 = 0,4 \text{ м.}$$

Висота підвішування світильник на освітлювальною:

$$h = H_0 - h_C = 2 - 0,4 = 1,6 \text{ м.}$$

Висота підвішування світильник над:

$$H_П = h = h_Р = 1,6 + 1 = 2,6 \text{ м.}$$

Для досягнення найбільшої рівномірності освітленні приймаємо відношення L/h = 1,5, тоді відстань між центрами світильників:

$$L = 1,5 \cdot h = 1,5 \cdot 1,6 = 2,4 \text{ м.}$$

Необхідна кількість світильників:

$$N = \frac{S}{L^2} = \frac{6 \cdot 4}{2,4^2} = 4 \text{ шт.}$$

Приймаємо 4 світильника (2 ряди по 2 світильника). Індекс приміщення:

$$i = \frac{A \cdot B}{h \cdot (A + B)} = \frac{6 \cdot 4}{1,6 \cdot (6 + 4)} = 1,5 .$$

При i = 1,5, $\rho_П$=70%, $\rho_С$=50%, $\rho_Р$=10% для світильників типу УПМ-15 коефіцієнт використання світлового потоку η = 0,55. тоді основний потік однієї лампи:

$$Ф = \frac{E_{\min} \cdot S \cdot K_З \cdot Z}{N \cdot \eta} = \frac{300 \cdot 24 \cdot 1,3 \cdot 1,15}{4 \cdot 0,55} = 4983 \text{ лм.}$$

За найденим світловим потоком обираємо лампу Г 215-225-300 потужністю 300 Вт, що має світловий потік 4610 лм, найбільш близький до розрахункового. При цьому фактична освітленість:

$$E_{Ф} = E_{Н} \frac{Ф_{Л}}{Ф_{Р}} = 300 \frac{4610}{4893} = 283 \text{ лк.}$$

Загальна потужність освітлюваної установки:

$$P_{З} = P_{л} \cdot N = 300 \cdot 4 = 1,2 \text{ кВт.}$$

Таким чином розраховано, що для загального освітлення приміщення необхідно 4 світильники сумарною потужністю 1,2 кВт.

4.3 Запобіжні заходи проти ураження струмом, пожежі та інших нещасних випадків

Відкриті металеві частини електричних машин або устаткування, які не повинні бути під напругою, але можуть виявитись під напругою внаслідок їхньої несправності, повинні заземлюватися, крім випадків, коли машини або устаткування:

– живляться постійним струмом, напруга якого не перевищує 55 В, або змінним струмом, середньоквадратичне значення напруги якого між провідниками не перевищує 55 В, причому для одержання цієї напруги не повинні застосовуватися автотрансформатори;

– живляться струмом напруга якого не перевищує 250 В, від роздільних трансформаторів, що живлять тільки одного споживача;

– виготовлені відповідно до принципу подвійної ізоляції.

Всі електричні апарати повинні бути виготовлені та встановлені таким чином, щоб при їхньому нормальному обслуговуванні або дотику до них вони не викликали травм.

Головні та розподільні щити повинні бути влаштовані так, щоб забезпечувати зручний доступ персоналу до апаратів і устаткування. Бічні та тильні, а якщо необхідно, і лицьові сторони щитів повинні бути оснащенні належним огородженням.

У випадку, коли застосовують незаземлену первинну або вторинну систему розподіл струму для силових, опалювальних або освітлювальних ланцюгів, повинне бути передбачено пристрій, що забезпечує безперервний контроль за рівнем ізоляції щодо землі і подачу звукового та світлового сигналу, що вказує на ненормально низьку величину ізоляції.

Всі електричні кабелі та електропроводка, що перебувають поза устаткування, повинні бути принаймні такого типу, що не поширює полум’я, і бути прокладені так, щоб не погіршувалися їхні первісні властивості відносно нерозповсюдження полум’я.

Кабелі та електропроводка, що живлять відповідальні або аварійні силові ланцюги, а також висвітлення і внутрішньо судновий зв'язок або сигналізацію, повинні прокладатися в обхід камбузів, пралень, машинних приміщень категорії А та їхніх шахт і інших зон високої пожежонебезпеки.

Кабелі повинні бути встановлені і закріплені таким чином, щоб уникнути перетинання або іншого ушкодження.

Підключення та з’єднання всіх провідників повинні бути виконані таким чином, щоб зберігалися первісні електричні та механічні властивості кабелю, а також його властивості щодо нерозповсюдження полум’я і якщо буде потреба вогнестійкі властивості.

РОЗДІЛ 5
ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Україні притаманні такі екологічні проблеми, як кислотні дощі, транскордонне забруднення, руйнування озонового шару, потепління клімату, накопичення відходів, особливо токсичних та радіаційних, зниження біологічного різноманіття. Аварія на ЧАЕС з її величезними медико-біологічними наслідками спричинила в Україні ситуацію, що наближається до рівня глобальної екологічної катастрофи. В Україні відсутній природній приріст населення, а тривалість життя на шість років нижча, ніж у розвинутих країнах. Негативні зміни у здоров'ї сталися, головним чином, за рахунок підвищення рівня злоякісних новоутворень, серцево-судинних хвороб, бронхіальної астми і т.д. Великої гостроти набула проблема радіоактивних відходів. На атомних електростанціях накопичено тисячі тон відпрацьованого ядерного палива, десятки тисяч кубометрів твердих і десятки мільйонів літрів рідких радіоактивних відходів.

Стан навколишнього природного середовища значною мірою визначається рівнем лісистості та якісним станом лісів. Україна малолісна країна. Ліси переважно виконують захисні водоохоронні та санітарно-гігієнічні функції. Однак вони інтенсивно експлуатуються, гинуть від промислових викидів та пожеж, внаслідок недбалого відведення земель під вирубку для різноманітного будівництва. Протягом останніх років в Україні загинуло близько 2,5 тис. га лісових насаджень. Вирубка лісового фонду перевищує його відновлення. Викликає стурбованість інтенсивна експлуатація лісів, особливо в Карпатському та Поліському регіонах, де зосереджено відповідно 29 та 33 % запасів деревини. Значних збитків завдають пожежі. Необгрунтованість при визначенні шляхів розвитку агропромислового та лісохімічного комплексів, не регульований стік забруднень у річки, осушення боліт та стихійний розвиток колективного садівництва призвели до зниження природного потенціалу майже на 70% цінних природних комплексів і ландшафтів України.

5.1 Забруднення морського середовища внаслідок викиду суднових відходів діяльності

Відходи суднові – не потрібні і не придатні для обслуговування судна відпрацьовані види нафти і матеріалів, невживані їх залишки або речовини, що утворюються в процесі експлуатації судна (тверді і рідкі), які не підпадають під утилізацію на самому судні і підлягають постійному або періодичному вилученню із судна.

Всі відходи, що утворюються на судні, можна розбити на дві групи:

- залишки перевезених вантажів, що утворюються наслідок неповного їхнього вивантаження, обмивання палуби, трюмів, танків і т.д.;
- забруднення, що утворюються в результаті життєдіяльності екіпажа й пасажирів, а також у результаті експлуатації суднових механізмів.

Крім того, дотепер, нажаль, досить часті випадки аварійних розливів нафтопродуктів при бункеруванні судів і в результаті різних аварійних ситуацій.

У процесі експлуатації суднових механізмів утворюється особливий вид відходів – нафто місткі води (НВ), які скопичуються в машинному відділенні. Основні причини утворення НВ – це протікання води із трубопроводів, арматури, насосів, дейдвудних пристроїв, донні арматури, а також протікання нафтопродуктів із трубопроводів і арматур при ремонті механізмів.

Такий стан речей став можливим за рахунок розвитку підводної техніки, яка дозволяє вести інтенсивний пошук та дослідження флори і фауни не тільки на малих глибинах але й на середніх і великих глибинах, закріплюватися на місцях видобутку корисних копалин, проводити різного виду комунікації.

Розглянемо деякі негативні впливи від використання підводних апаратів на екологічний стан Світового океану та істот, що його населяють.

Для проведення комунікацій зв’язку зі стаціонарними нафтовими буровими платформами та між островами і материками, а також для прокладання трубопроводів для транспортування видобутих нафти і газу у товщі грунту використовують самохідні робочі апарати, котрі своїми робочими інструментами

при виконана своїх функцій знищують живі організми на дні океану, порушують звичні місця їхнього перебування, змінюють рельєф дна.

При використанні прив'язних підводних систем намагаються досягти максимального економічного ефекту від їх використання так як по закінченню терміну їхньої експлуатації вони залишаються на дні Світового океану, якщо не передбачена утилізація іншим шляхом. Знаходячись на дні прив'язна підводна система піддається агресивному впливу водного середовища, отруюючи простір навколо себе різними хімічними речовинами.

У складі рушіїв є наявність рідкого діелектрика, в основному це гас, який призначений для компенсації гідростатичного тиску на конструктивні елементи.

Для запобігання наведеним негативним факторам світовою спільнотою приймаються заходи, що до зниження забруднення і зміни водного простору, де інтенсивно використовують підводні апарати.

5.2 Заходи боротьби з забрудненням

У 1954 році в Лондоні пройшла міжнародна конференція, що ставила метою виробити погодженні дії по охороні морського середовища від забруднень нафтою. На конференції була прийнята конвенція, що визначає обов'язки держав у цій області. Пізніше у 1958 році в Женеві були прийняті ще чотири документи: про відкрите море, про територіальне море та прилеглу зону, про континентальний шельф, про рибальство й охорону живих ресурсів моря. Ці конвенції юридично закріпили принципи й норми морського права. Вони ставили за обов'язок кожній країні розробити і ввести в дію закони, що забороняють забруднювати морське середовище нафтою, радіо відходами та іншими шкідливими речовинами. В 1973 році в Лондоні конференція прийняла документи по запобіганню забруднення з суден.

Заборонено злив нафтомістких вод з танкерів, всі скидання повинні з них викачувати тільки на берегові приймальні пункти.

З метою запобігання витоків нафти вдосконалюються конструкції нафтоналивних суден. Багато сучасних танкерів мають подвійне дно. При ушкоджені одного з них нафта не виливається, її затримає друга оболонка.

Для систематичного очищення акваторій від випадкових розливів застосовуються плавучі нафтозбирачі й бічні загородження.

У Японії створена й апробована унікальна технологія, за допомогою якої можна в короткий термін ліквідувати гігантську пляму. Було випущено реактив, основним компонентом якого є оброблене рисове лушпиння.

З 1993 року було заборонено скидання рідких радіоактивних відходів, але їх число неухильно росте. Тому з метою захисту навколишнього середовища в 90-ті роки стали розроблятися проекти очищення рідких радіоактивних відходів.

Однак незважаючи на деякі успіхи в пошуку ефективних засобів, які ліквідують забруднення, про рішення проблеми говорити зарано. Тільки впровадженням нових методик очищення акваторій неможливо забезпечити чистоту морів і океанів. Центральне завдання, яке необхідно вирішувати всім країнам спільно – запобігання забрудненню.

ВИСНОВКИ

У цій роботі розглянуто можливі підходи до вирішення проблеми зниження трудомісткості розробки робототехнічних систем за одночасного підвищення її якості за допомогою створення системи віртуального прототипування робототехнічних об'єктів. Такі засоби дозволяють проводити експерименти з різними конфігураціями робототехнічних систем та алгоритмами їх управління у віртуальному середовищі з метою їх налагодження та оптимізації. Автором були розроблені нові віртуальні прототипи компонентів робототехнічних систем, що становлять інтерес для спільноти розробників, та їх натурні реалізації.

Запропонований підхід дозволяє скоротити час, потрібний для розробки робототехнічної системи, знизити кваліфікаційні вимоги до розробників РТС, знизити трудомісткість розробки та дослідження нових робототехнічних компонентів, тим самим дозволяючи підвищити показники, що впливають на ефективність проектування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Юревич Є.І. Основи проектування техніки. – Навчальний посібник, 2012.

2. ARGoS [Електронний ресурс], Веб-сайт проекту ARGoS. – URL:https://www.argos-sim.info/.

3. Cheruiyot GK, Zhu X., Cao Q. OROCOS-базується загальна система управління для 6 DOF індустріальний manipulator // IEEE ARSO. – 2016. – С. 174–179.

4. Collada [Електронний ресурс], Веб-сайт проекту Collada. – URL:https://collada.org/.

5. Fontanelli GA, Selvaggio M., Ferro M. A V-REP Simulator для da Vinci Research Kit Роботичні Platform // IEEE International Conference on Biomedical Robotics and Biomechatronics. - 2018. – № 8. – С. 1056–1061.

6. GAZEBO [Електронний ресурс], Веб-сайт проекту GAZEBO. URL:http://gazebosim.org/.

7. Gniewek A., Bards M., Bettstetter C. Robots що Sync і Swarm: A Proof of Concept in ROS2 // CoRR. – 2019.

8. Gomez C., Hernandez AC, Crespo J. ROS-BASED MIDDLE- COST ROBOTIC PLATFORM WITH HIGHPERFORMANCE // ICERI. – 2015.

9. Ivaldi S., Padois V., Nori F. Інструменти для динамічних simulation of robots: survey based on user feedback // IEEE-RAS International Conference on Humanoid Robots. – 2014. – С. 842-849.

10. Lemaignan, Severin, Hanheide. Simulation and HRI - Recent Perspectives with MORSE Simulator // A Proposed Software Framework Aimed at Energy-Efficient Autonomous Driving of Electric Vehicles. – 2014. – С. 13–24.

11. MIRA [Електронний ресурс], Веб-сайт проекту MIRA. URL:http://www.mira-project.org/joomla-mira/.

12. MORSE [Електронний ресурс], Веб-сайт проекту MORSE.– URL:http://morse-simulator.github.io/.

13. Meyer J., Sendobry A., Kohlbrecher S. Comprehensive Simulation of

Quadrotor UAVs використовуючи ROS і Gazebo // SIMPAR. - 2012.

14. Orocos [Електронний ресурс], Веб-сайт проекту Orocos. – URL:http://www.orocos.org/.

15. Pinciroli C., Talamali MS Simulating Kilobots Within ARGoS: Models and Experimental Validation // ANTS Conference. - 2018.

16. Protopapadakis E., Stentoumis C., Doulamis N. AUTONOMOUS ROBOTIC INSPECTION IN TUNNELS // Remote Sensing and Spatial Information Sciences. – 2016. – No. 3.

17. ROS [Електронний ресурс], Веб-сайт проекту ROS. – URL:https://www.ros.org/.

18. ROS2 [Електронний ресурс], Веб-сайт проекту ROS2. – URL: https://index.ros.org/doc/ros2/.

19. RT-middleware [Електронний ресурс], Веб-сайт проекту RT-middleware. – URL:https://openrtm.org/openrtm/.

20. Reinforcement Learning with Gazebo and ROS 2 in an industrial robot [Електронний ресурс], Веб-сайт проекту Acutronic Robotics.– URL:https://acutronicrobotics.com/news/reinforcement-learning-with-gazebo-and-ros-2/.

21. Robotics, the traditional path and new approaches [Електронний ресурс], Веб-сайт проекту Robohub. – URL:https://robohub.org/robotics-the-traditional-path-and-new-approaches/.

22. Rock [Електронний ресурс], Веб-сайт проекту Rock. – URL:https://www.rock-robotics.org/.

23. SDF [Електронний ресурс], Веб-сайт проекту SDF. – URL:http://sdformat.org/.

24. Self-Tuning Method for Increased Obstacle Detection Reliability Based on Internet Things LiDAR Sensor Models [Електронний ресурс], Веб-сайт проекту MDPL URL:https://www.mdpi.eom/1424-8220/18/5/1508/htm.

25. Shamshiri RR, Hameed IA, Karkee M. Роботський постачання з Fruiting Vegetables: Simulation Approach в V-REP, ROS і MATLAB // IntechOpen. – 2017. –

С. 81-105.

26. Tsagarakis NG, Negrello F., Garabini M. WALK-MAN Humanoid Platform // DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue. – 2018. – № 121. - С. 495-549.

27. URDF [Електронний ресурс], Веб-сайт проекту URDF. – URL:http://wiki.ros.org/urdf.

28. V-REP [Електронний ресурс], Веб-сайт проекту V-REP. – URL:http://www.coppeliarobotics.com/.

29. VirCA [Електронний ресурс], Веб-сайт проекту VirCA. – URL:http://www.virca.hu/.

30. Watanabe, Thomio, Neves. The Rock-Gazebo Integration і Real-Time AUV Simulation // LARS-SBR. – 2019. – С. 132-138.

31. YARP [Електронний ресурс], Веб-сайт проекту YARP. URL:https://www.yarp.it/.

32. Multi-modal interface of Robot-Era multi-robot services tailored for the elderly / A. Di Nuovo, F. Broz, N. Wang, T. Belpaeme // Intel Serv Robotics. – 2018. – № 11. – С. 109-126.