Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

«**СИСТЕМИ УПРАВЛІННЯ РОБОТАМИ**»

для здобувачів освіти другого (магістерського) рівня вищої освіти спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
(освітня програма «Автоматизація, мехатроніка та робототехніка»)
усіх форм навчання

2024

Методичні вказівки до лабораторних робіт з дисципліни «Системи управління роботами» для здобувачів освіти другого (магістерського) рівня вищої освіти спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» (освітня програма «Автоматизація, мехатроніка та робототехніка») усіх форм навчання / Уклад. : Н. О. Миронова, С.В. Шаптала. – Запоріжжя: НУ «Запорізька політехніка», 2024. – 33 с.

Укладачі: Наталя МИРОНОВА, к.т.н., доцент каф. ІТЕЗ;
Станіслав ШАПТАЛА, ст. викладач каф. ІТЕЗ;

Рецензент: Микола ЄФИМЕНКО, д.т.н., доцент каф. ІТЕЗ;

Відповідальний
за випуск: Наталя МИРОНОВА, к.т.н., доцент каф. ІТЕЗ;

Затверджено на засіданні кафедри ІТЕЗ протокол № 1 від 10.09.24 р.

Рекомендовано до видання на засіданні НМК ФІБЕК протокол № 2 від 19.09.24р.

ЗМІСТ

ВСТУП

Метою лабораторних робіт є ознайомлення з основними принципами та методами управління роботизованими системами через практичне використання інструментів та середовищ симуляції, таких як Webots та ROS; розвиток навичок розробки та програмування контролерів руху роботів за допомогою створення алгоритмів керування, розпізнавання об’єктів, оминання перешкод тощо. Це дозволить здобувачам оволодіти практичними навичками у сфері робототехніки, підготуватися до роботи над реальними проєктами та отримати глибше розуміння принципів функціонування роботизованих систем.

Кожному студенту при підготовці до виконання лабораторної роботи потрібно ознайомитись з методичними вказівками та конспектом лекцій по даному розділу. Виконання лабораторної роботи розбивається на два етапи. На першому етапі необхідно ознайомитись з конспектом лекцій. На другому етапі виконується індивідуальне завдання до лабораторної роботи. Наприкінці заняття результати роботи подаються викладачу для перевірки. Звіт з лабораторної роботи оформлюється на стандартних аркушах паперу формату А4 (297х210мм).

Лабораторна робота обов’язково захищається на наступному після виконання роботи навчальному занятті. За несвоєчасний захист роботи оцінка знижується. Захист лабораторних робіт проводиться під час навчальних занять. Студент не допускається до виконання наступної роботи, якщо має дві незахищені роботи.

ЛАБОРАТОРНА РОБОТА №1
ІНСТАЛЯЦІЯ ТА НАЛАШТУВАННЯ WEBOTS

1.1 Мета роботи

Ознайомитися з системою симуляції Webots, здійснити інсталяцію та налаштування середовища для симуляції робототехнічних систем, а також вивчити основні функції й можливості для подальшого створення та тестування роботів.

1.2 Теоретичні відомості

Технологія розробки програм для систем симуляції полягає в створенні програмного забезпечення, що дозволяє моделювати процеси, які відбуваються в реальному світі. Такі програми можуть бути використані для вирішення різноманітних завдань, від тестування нових технологій до тренування персоналу.

Основні етапи розробки програм для систем симуляції:

а) розробка математичної моделі: на цьому етапі визначаються математичні залежності, які описують процеси, що моделюються;

б) розробка програмного забезпечення: на цьому етапі розробляються програмні модулі, які забезпечують взаємодію з користувачем, візуалізацію результатів симуляції та інші функції;

в) тестування та налагодження: на цьому етапі проводяться тестування програмного забезпечення і виявляються його недоліки;

г) впровадження та експлуатація: на останньому етапі програмне забезпечення системи симуляції впроваджується в дію, а далі проводяться регулярні оновлення та технічна підтримка [1].

Технологія розробки програм для систем симуляції приведена на рисунку 1.1.

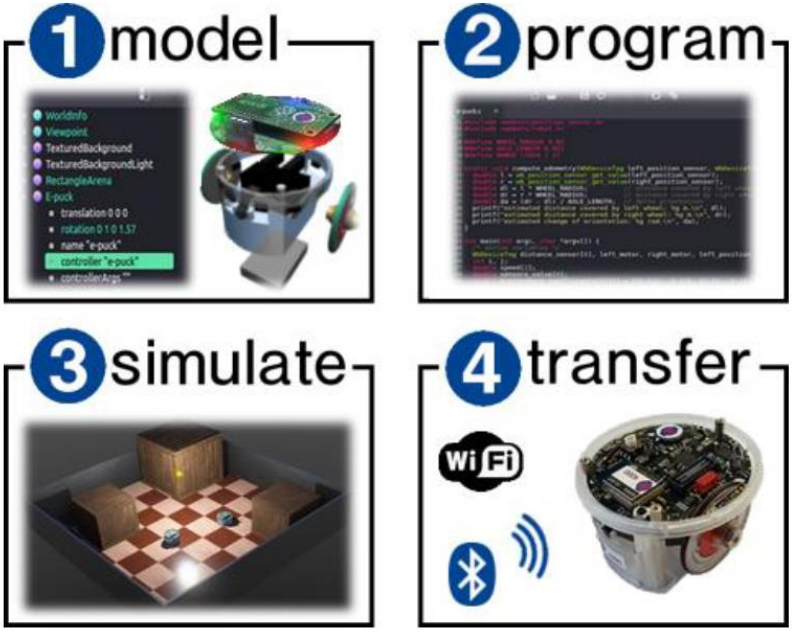

Рисунок 1.1 – Технологія систем симуляцій [1]

1.3 Порядок виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями та матеріалами лекції.
2. Затвердити індивідуальне завдання у викладача.
3. Відповісти на контрольні питання.
4. Оформити звіт та завантажити до «Системи дистанційного навчання» НУ «Запорізька політехніка».

1.4 Індивідуальні завдання

1. Обрати конкретну модель робота в середовищі Webots із доступних або самостійно створених варіантів.
2. Ознайомитися з технічними характеристиками та можливостями обраного робота.
3. Дослідити готові рішення для реалізації рухів робота, включаючи алгоритми керування його моторами та сенсорами.
4. Проаналізувати існуючі контролери руху робота та визначити

їх переваги і недоліки.

5. Розглянути можливості модифікації або вдосконалення наявних рішень для більш ефективного керування рухами робота.

1.5 Контрольні запитання

1. Які параметри важливо враховувати при виборі моделі робота в середовищі Webots?
2. Як обраний робот взаємодіє з навколишнім середовищем у Webots?
3. Які сенсори та двигуни використовує обраний робот?
4. Які типи рухів може виконувати обраний робот, і які механізми відповідають за їх реалізацію?
5. Які готові рішення для контролю рухів робота було знайдено, і чим вони відрізняються один від одного?
6. Як працює типовий контролер руху для робота в Webots?
7. Які ключові елементи містить такий контролер?
8. Які переваги та недоліки було виявлено в існуючих алгоритмах керування рухами робота?
9. Які можливі способи вдосконалення готових рішень щодо руху робота можна запропонувати?
10. Які кроки необхідні для впровадження нового або модифікованого контролера руху для обраного робота в Webots?

ЛАБОРАТОРНА РОБОТА №2
РОЗРОБКА КЛАСУ КОНТРОЛЕРУ РУХІВ РОБОТА В WEBOTS

2.1 Мета роботи

Розробити клас контролеру для керування рухами робота в середовищі Webots, налаштувати алгоритм керування та протестувати його на моделі робота, досліджуючи можливості автоматизації та точності виконання рухів.

2.2 Теоретичні відомості

Опишемо процес розробки контролеру руху наприкладі робототехнічній платформи Robotis OP2.

Робототехнічна платформа Robotis OP2 призначена для розробки та дослідження різноманітних робототехнічних систем. Ця платформа є однією з найбільш популярних серед робототехнічних платформ, що використовуються у наукових дослідженнях, навчальних закладах та промисловості [1].

Загалом, Robotis OP2 є досить потужною робототехнічною платформою, яка має високий ступінь конфігурованості та розширюваності.

Деякі з її основних характеристик включають:

а) два 32-бітних мікроконтролера, які забезпечують високу швидкість обробки даних та керування роботом;

б) шістьнадцять сервоприводів Dynamixel MX-28, які забезпечують точну та динамічну роботу з високим рівнем керованості та точності;

в) вбудований набір датчиків, таких як гіроскоп, акселерометр, енкодери та датчики тиску, які дозволяють роботу збирати та обробляти різноманітну інформацію про своє оточення;

г) відкрита програмна архітектура, що дозволяє розробникам створювати свої власні програми та алгоритми для керування роботом;

д) наявність готових бібліотек та SDK для розробки программного забезпечення, що спрощує процес розробки та програмування.

Robotis OP2 може бути використаний для різноманітних

завдань, включаючи:

а) розробку мобільних та стаціонарних роботів, що можуть працювати в різних умовах;

б) розробку роботів, які можуть взаємодіяти з людьми, наприклад, для розваг, освіти або медичної допомоги;

в) дослідження в області робототехніки та програмування, включаючи навчання студентів та дослідників;

г) використання в наукових дослідженнях у різних галузях, таких як автоматизація виробництва, медицина, військова техніка, аерокосмічна техніка та інші.

Отже, призначення робототехнічної платформи Robotis OP2 полягає у використанні її для розробки, дослідження та навчання в різноманітних галузях робототехніки та автоматизації [1].

Для розробки контролеру робототехнічної платформи Robotis OP2 в безкоштовном 3D симуляторі роботів Webots спочатку потрібно створити файл світу. Для цього потрібно виконати наступні операції: File->New->New World File; Після чого вибирати властивості для світу:

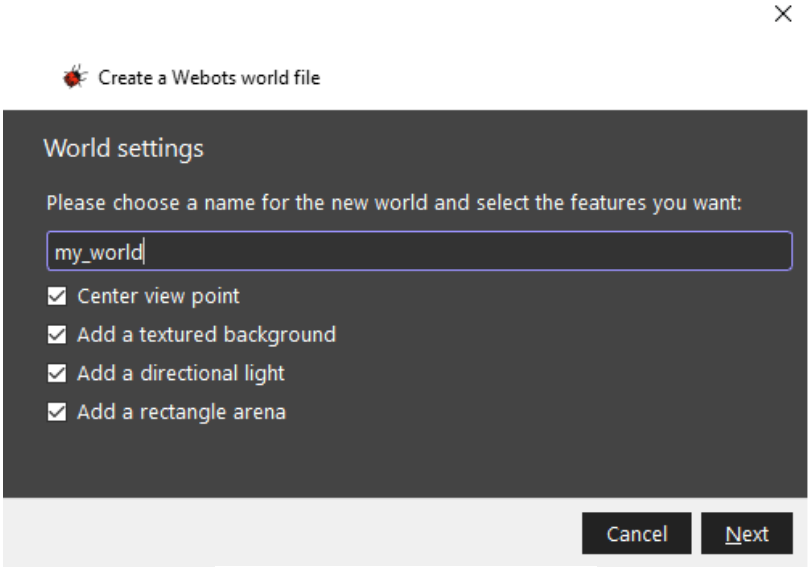

Рисунок 2.1 – Створення світу

Після того як світ було створено та завантажено, він буде виглядати так:

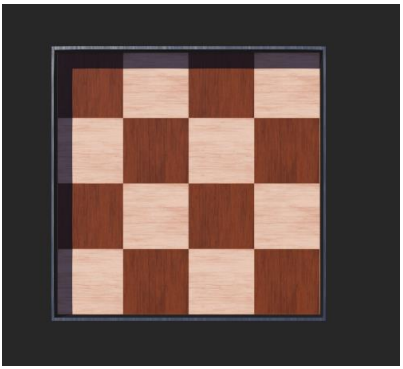

Рисунок 2.2 – Зовнішній вид світу

Так як поле для робота не дуже велике то потрібно його збільшити. Для цього потрібно в вікні зліва обрати RectangleArena, відкрити її властивості, знайти floorsize, та задати розмір для x та y = 3. Виглядає це наступним чином:

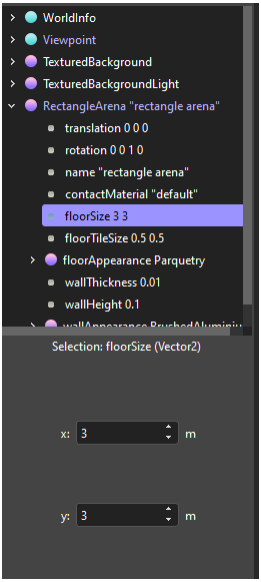

Рисунок 2.3 – Властивості RectangleArena

Після редагування поле буде мати наступний вигляд:

Рисунок 2.4 – RectangleArena після змін розміру

Після цього потрібно добавити робота. Для цього потрібно натиснути правою кнопкою миші на RectangleArena в полі зліва, та обрати Add New. У відкрившомуся вікні, знайти потрібного робота, в даному випадку це Robotis OP2. Для цього відкрити вкладку PROTO nodes (Webots Projects), там обрати robots, потім robotis, потім darwin-op, і тут вже знаходиться сам робот з назвою RobotisOp2 (Robot), треба вибрати його та натиснути Add.

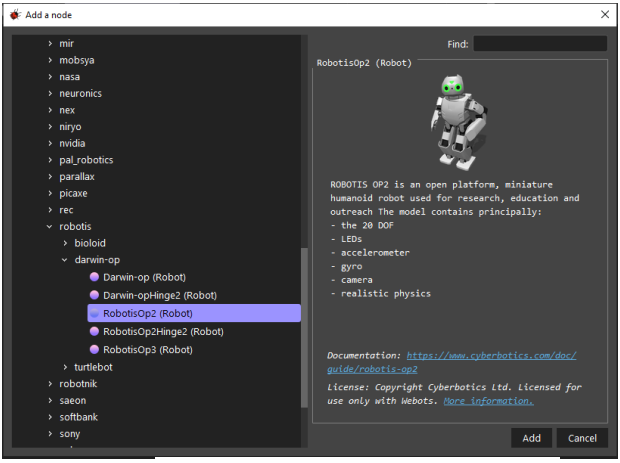

Рисунок 2.5 – Додавання робота

Такий вигляд має робот після того як його додали в світ:

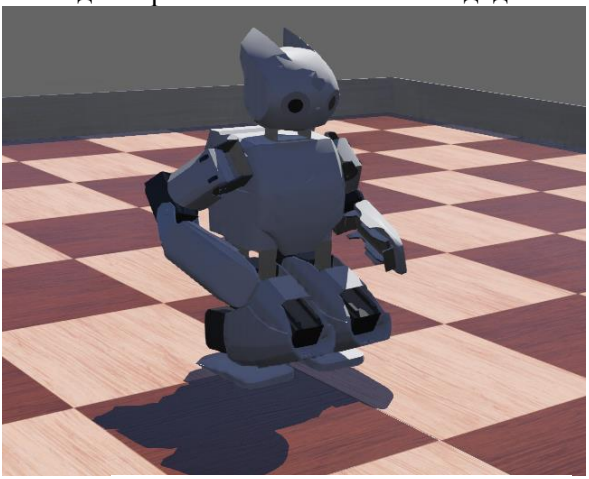

Рисунок 2.6 – Робот після додавання

Для програмування робота потрібно створити та обрати контролер для нього. Для цього потрібно обрати File->New->New Robot Controller; В відкрившомуся вікні спочатку потрібно натиснути Next, далі пропонується мова на якій буде програмуватися контролер, де потрібно обрати C++ та натиснути Next:

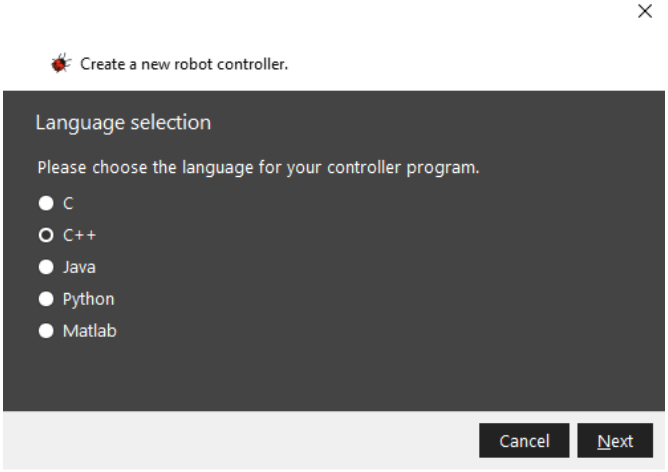

Рисунок 2.7 – Вибір мови для програмування контролера

В наступній вкладці пропонують обрати середовище, в якого буде писатися код контролера, треба обрати Webots:

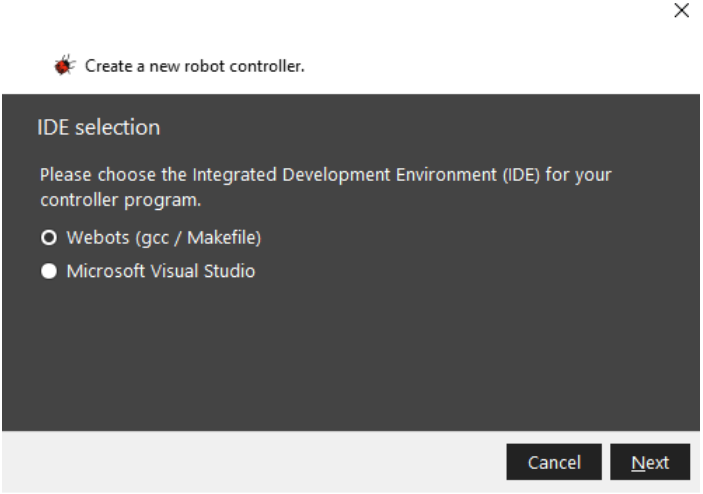

Рисунок 2.8 – Вибір середовища для програмування

І в останній вкладці треба ввести назву контролера, в даному випадку це Walk та натиснути Next:

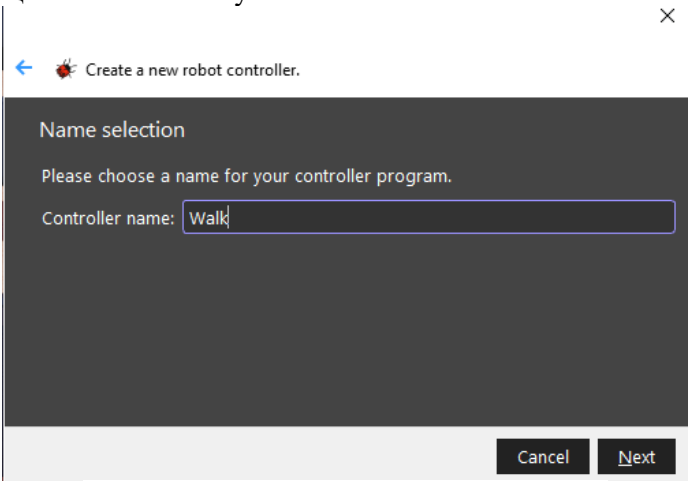

Рисунок 2.9 – Введення назви контролера

Після того як файл для програмування контролера створено, його потрібно обрати та відкрити у середовищі для написання коду. Для цього в вікні зліва треба обрати робота RobotisOp2 “Robotis OP2”, знайти та натиснути на властивість controller, тут натиснути на Select та знайти відповідний контролер, в даному випадку це walk:

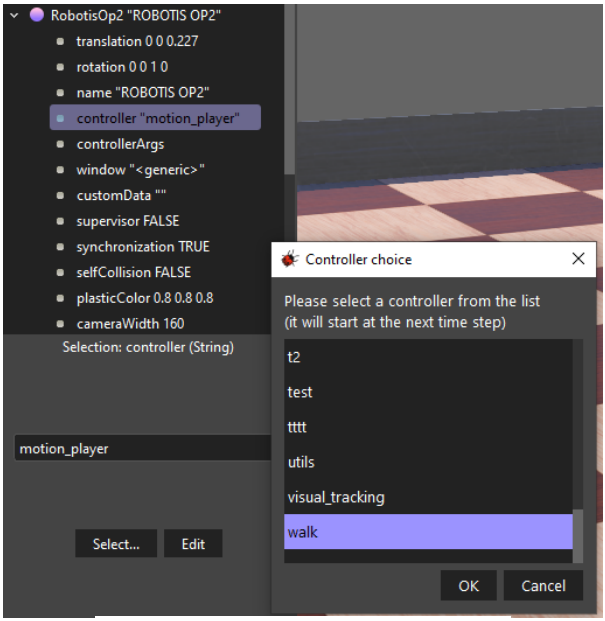

Рисунок 2.10 – Вибір контролера

Після цього треба натиснути на Edit, і файл контролера відкриється у вікні справа, що дозволить писати текст програми. Текст контролеру руху робототехнічної платформи Robotis OP2 наведено в додатку А.

В розробленому контролері walk використовуються наступні сенсори та пристрої: акселерометр: для вимірювання прискорення робота; гіроскоп: для вимірювання швидкості обертання робота; клавіатура: для зчитування введених користувачем команд; двигуни: для керування рухом робота; позиційні датчики: для вимірювання кутової позиції кожного з двигунів; світлодіоди: для відображення стану робота.

В конструкторі Walk() встановлюється час кроку, ініціалізуються сенсори і пристрої, а також генерація гейта менеджера.

У функції myStep() виконується один крок симуляції.

У функції wait(int ms) задається пауза в мілісекундах.

У функції run() описано головний цикл управління роботом. У цьому циклі зчитується введення з клавіатури, щоб керувати роботом, який рухається. При натисканні клавіші "пробіл" запускається або зупиняється генерація руху за допомогою гейта менеджера. Функція checkIfFallen() використовується для визначення, чи впав робот. Якщо робот впав, то генерується відповідна траєкторія руху, щоб підняти його.

В деструкторі ~Walk() відбувається звільнення пам'яті, яка була виділена під гейт менеджер і менеджер руху.

У головній функції run() здійснюється основний цикл обробки даних від сенсорів і управління рухами робота.

Спочатку виводяться на екран деякі діагностичні повідомлення, тоді викликається функція myStep(), яка оновлює дані від сенсорів і виконує крок симуляції. Далі здійснюється відтворення вихідної позиції робота, а після цього відбувається основний цикл обробки введення від клавіатури. Клавіша "Пробіл" використовується для початку або зупинки руху робота.

Також в тексті програми є функція checkIfFallen(), яка визначає, чи лежить робот на землі, на основі даних, які надають акселерометр та гіроскоп. Ця функція викликається на кожному кроці основного циклу run(), тому робот завжди може перевірити своє положення і, якщо необхідно, повернутися в вихідне становище

Після того як текст програми для контролера буде написаний, потрібно його скомпілювати. Для цього треба у вікну справа натиснути на шестерню (Build the current project). Коли контролер скомпілювався, його можна тестувати.

Для запуску контролера треба в верхній панелі натиснути на кнопку Run the simulation in real-time.

Після запуску робот відразу переходить з сидячого положення, в стояче. Також вмикається його підсвітка.

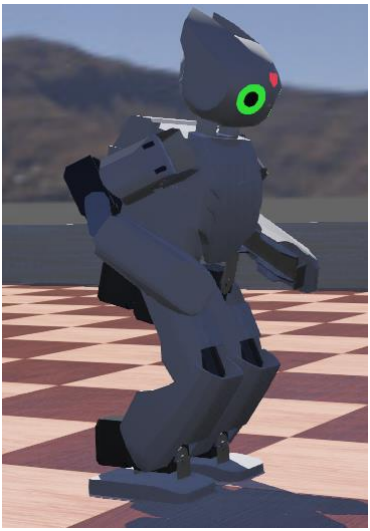

Рисунок 2.11 – Робот в стоячому положенні

Для того щоб робот почав ходити потрібно натиснути на пробіл. Після цього робот починає йти вперед. Якщо робот впаде, то за допомогою функції на перевірку того чи не впав він, буде виконано код на його підйом в залежності того впав він на спину або живіт, і запущено відповідну анімацію.

Для написання контролеру для робототехнічної платформи Robotis OP2 була використана бібліотека Managers, яка використовується для того, щоб реалізувати всі ключові функціональні можливості Robotis Framework в симуляції. Ця бібліотека розділена на три частини, які називаються менеджерами. Кожен менеджер реалізує окремий модуль фреймворку.

2.3 Порядок виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями та матеріалами лекції.
2. Реалізувати контролер руху для обраного робота з лабораторної роботи №1.
3. Відповісти на контрольні питання.
4. Оформити звіт та завантажити до «Системи дистанційного навчання» НУ «Запорізька політехніка».

2.4 Індивідуальні завдання

1. Ознайомитися з середовищем симуляції Webots та вибрати модель робота для розробки контролера руху.

2. Проаналізувати структуру існуючих контролерів руху роботів у Webots, їх архітектуру та алгоритми керування.

3. Створити новий клас контролера руху для обраної моделі робота, визначивши основні параметри руху (швидкість, напрям, прискорення).

4. Реалізувати алгоритм керування рухами робота, що враховує отримання даних із сенсорів для корекції траєкторії та швидкості.

5. Протестувати роботу контролера в системі симуляції Webots, забезпечивши коректність та стабільність рухів робота.

6. Оформити звіт, який включає опис процесу розробки, результати тестування та аналіз роботи контролера, з графічними ілюстраціями результатів симуляції.

2.5 Контрольні запитання

1. Які основні етапи розробки контролера руху для робота в середовищі Webots?

2. Яка структура класу контролера руху, і які основні функції він повинен містити?

3.Як обраний робот отримує та обробляє дані від сенсорів для контролю руху?

4. Які параметри руху необхідно враховувати при розробці контролера (наприклад, швидкість, напрямок, прискорення)?

5. Які типи алгоритмів керування рухом використовуються для роботів у Webots?

6. Які основні помилки можуть виникати під час розробки контролера, і як їх уникнути?

7. Як проводиться тестування контролера в симуляційному середовищі Webots, і які метрики використовуються для оцінки його ефективності?

8. Які методи можна використовувати для оптимізації роботи контролера руху робота?

9. Як інтегрувати сенсори робота в процес контролю його рухів?

10. Які можливі шляхи вдосконалення контролера руху після тестування в симуляції?

ЛАБОРАТОРНА РОБОТА №3
ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПІДХОДІВ КОМП'ЮТЕРНОГО ЗОРУ РОБОТІВ В WEBOTS

3.1 Мета роботи

Вивчити основи комп'ютерного зору, дослідити різні підходи до його реалізації для роботів у середовищі Webots, а також реалізувати алгоритми обробки зображень для навігації роботів та виявлення об'єктів.

3.2 Теоретичні відомості

Комп'ютерний зір – це технологія, яка дозволяє комп’ютеру інтерпретувати та розуміти візуальну інформацію з навколишнього середовища. Він базується на алгоритмах обробки зображень та комп'ютерного навчання. Основні компоненти комп'ютерного зору включають отримання зображень, обробку та аналіз. Роботи, оснащені камерами, можуть використовувати комп'ютерний зір для виконання різних завдань, таких як розпізнавання об'єктів, слідкування за рухом, визначення відстаней до об'єктів і навігація в складних середовищах.

Одним з основних підходів до комп'ютерного зору є обробка зображень за допомогою фільтрації, сегментації та виявлення контурів. Алгоритми, такі як Canny, Sobel та інші, дозволяють виділити ключові особливості в зображеннях. Також важливим аспектом є використання алгоритмів машинного навчання, таких як нейронні мережі, для навчання моделей розпізнавання об'єктів на основі навчальних наборів даних.

Webots надає інструменти для симуляції роботів з вбудованими камерами, що дозволяє тестувати різні алгоритми комп'ютерного зору в контрольованому середовищі. Завдяки інтеграції з бібліотеками Python, такими як OpenCV, можна реалізувати складні алгоритми обробки зображень та тестувати їх на віртуальних моделях роботів.

3.3 Порядок виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями та матеріалами лекції.
2. Реалізувати алгоритм розпізнавання перешкод на основі

інструментів Webots для обраного робота.

3. Відповісти на контрольні питання.

4. Оформити звіт та завантажити до «Системи дистанційного навчання» НУ «Запорізька політехніка».

3.4 Індивідуальні завдання

1. Ознайомитися з можливостями Webots для роботи з камерами та комп'ютерним зором.

2. Додати, за необхідності, камеру для захоплення зображень для обраного роботу з лабораторної роботи №1.

3. Розробити алгоритм обробки зображень для виявлення об'єктів у середовищі Webots.

4. Провести експерименти з різними алгоритмами комп'ютерного зору та оцінити їх точність.

5. Оформити звіт, який включає опис алгориму, результати експериментів, з графічними ілюстраціями результатів симуляції.

3.5 Контрольні запитання

1. Поняття комп'ютерного зору та наведіть його складові.

2. Які алгоритми обробки зображень використовуються в комп'ютерному зорі?

3. Які переваги має використання алгоритмів машинного навчання у комп'ютерному зорі?

4. Які алгоритми виявлення об'єктів є найпоширенішими?

5. Як можна оцінити ефективність алгоритмів комп'ютерного зору в симуляційному середовищі Webots?

ЛАБОРАТОРНА РОБОТА №4
ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ РОЗПІЗНАВАННЯ ПЕРЕШКОД РОБОТІВ В WEBOTS

4.1 Мета роботи

Вивчити та реалізувати алгоритми розпізнавання перешкод для роботів у середовищі Webots.

4.2 Теоретичні відомості

Розпізнавання перешкод є важливим аспектом автономної навігації роботів, що дозволяє їм ефективно пересуватися в складних середовищах. Основною метою даного процесу є виявлення (ідентифікація) об'єктів, які можуть заважати руху робота. Для досягнення цієї мети використовуються різні типи сенсорів, серед яких найбільш поширені ультразвукові, інфрачервоні та камери. Кожен з цих сенсорів має свої переваги та обмеження. Наприклад, ультразвукові сенсори забезпечують простоту в застосуванні та економічність, але мають обмежену точність на великих відстанях. Інфрачервоні сенсори здатні виявляти об'єкти в умовах низького освітлення, проте їхній радіус дії зазвичай менший.

Для обробки даних, отриманих від сенсорів, застосовуються різноманітні алгоритми, які дозволяють фільтрувати шуми та визначати відстань до перешкод. Основні методи включають методи комп'ютерного зору, а також просторові фільтри. Використання таких алгоритмів дозволяє досягати високої точності в розпізнаванні перешкод, що критично важливо для автономних систем.

У Webots можна інтегрувати різноманітні сенсори та алгоритми, що забезпечує реалістичне середовище для дослідження розпізнавання перешкод роботами. Моделі роботів у Webots можуть бути запрограмовані на виконання різних навігаційних алгоритмів, таких як A* або Dijkstra, що дозволяють планувати оптимальні траєкторії. Крім того, машинне навчання також знаходить застосування в розпізнаванні перешкод, дозволяючи роботам адаптуватися до нових умов середовища. Застосування сучасних технологій у поєднанні з класичними методами розпізнавання перешкод створює нові можливості для розвитку автономних робототехнічних систем.

4.3 Порядок виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями та матеріалами лекції.
2. Реалізувати алгоритм розпізнавання перешкод за допомогою інструментів Webots та алгоритм навігації в середовищі з перешкодами для обраного робота.
3. Відповісти на контрольні питання.
4. Оформити звіт та завантажити до «Системи дистанційного навчання» НУ «Запорізька політехніка».

4.4 Індивідуальні завдання

1. Ознайомитися можливостями Webots для моделювання роботів.
2. Додати за необхідності сенсори для виявлення перешкод для обраного роботу з лабораторної роботи №1.
3. Розробити алгоритм, що використовує дані сенсорів для ідентифікації перешкод у Webots.
4. Провести експерименти з різними типами перешкод та оцінити точність алгоритму.
5. Оформити звіт, який включає опис алгориму, результати експериментів, з графічними ілюстраціями результатів симуляції.

4.5 Контрольні запитання

1. Які основні типи сенсорів використовуються для розпізнавання перешкод?
2. Які алгоритми обробки даних сенсорів є найпоширенішими?
3. Які переваги та недоліки має використання різних сенсорів для розпізнавання перешкод?
4. Як можна оптимізувати алгоритм для розпізнавання перешкод?
5. Наведіть фактори, що впливають на точність розпізнавання перешкод у середовищі Webots.

ЛАБОРАТОРНА РОБОТА №5
ВИКОРИСТАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ В СИСТЕМАХ СИМУЛЯЦІЇ РОБОТІВ

5.1 Мета роботи

Дослідити застосування алгоритмів машинного навчання в системах симуляції роботів, розробити та реалізувати просту модель машинного навчання для покращення точності розпізнавання перешкод у симуляційному середовищі.

5.2 Теоретичні відомості

Машинне навчання є підгалуззю штучного інтелекту, що дозволяє комп'ютерним системам навчатися з даних без явного програмування. У контексті робототехніки машинне навчання може бути використано для вирішення різноманітних задач, таких як виявлення об'єктів, планування траєкторій, навігація та адаптивне навчання. Основними типами машинного навчання є:

– контрольоване навчання: модель навчається на мітках, що надаються разом з даними, що дозволяє їй робити прогнози на нових, невідомих даних. Застосовується для задач, таких як класифікація та регресія;

– неконтрольоване навчання: модель виявляє структуру даних без міток, що дозволяє їй знаходити патерни або класи у даних. Застосовується для кластеризації та зниження розмірності;

– навчання з підкріпленням: алгоритми навчання з підкріпленням дозволяють агенту навчатися через взаємодію з середовищем, отримуючи винагороду за виконання правильних дій. Цей підхід часто використовується для навчання роботів автономно виконувати завдання.

У системах симуляції роботів, таких як Webots, машинне навчання дозволяє швидко тестувати та адаптувати алгоритми в контрольованому середовищі. Завдяки цьому можна легко перевіряти різні гіпотези та оптимізувати параметри алгоритмів без ризику для фізичних роботів.

5.3 Порядок виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями та матеріалами лекції.
2. Реалізувати алгоритм розпізнавання та оминання перешкод на основі машинного навчання для обраного робота.
3. Відповісти на контрольні питання.
4. Оформити звіт та завантажити до «Системи дистанційного навчання» НУ «Запорізька політехніка».

5.4 Індивідуальні завдання

1. Ознайомитися з основами машинного навчання та його підходами.
2. Вибрати алгоритм машинного навчання, який буде використаний для розпізнавання перешкод (класифікація або навчання з підкріпленням).
3. Розробити симуляцію в Webots, де робот повинен навчитися розпізнавати та оминати перешкоди на основі зібраних даних.
4. Реалізувати алгоритм машинного навчання, використовуючи наявні бібліотеки (TensorFlow або PyTorch).
5. Провести експерименти, розрахувати показник достовірності розпізнавання перешкод навченої моделі.
6. Оформити звіт, який включає опис алгориму, результати експериментів, з графічними ілюстраціями результатів симуляції.

5.5 Контрольні запитання

1. Машинне навчання та його основні види.
2. Як відрізняється контрольоване навчання від неконтрольованого?
3. Які переваги має навчання з підкріпленням в контексті робототехніки?
4. Як машинне навчання може покращити автономію роботів?
5. Які бібліотеки та інструменти можуть бути використані для реалізації алгоритмів машинного навчання в системах симуляції?
6. Які метрики використовуються для оцінки показника достовірності розпізнавання перешкод?

ЛАБОРАТОРНА РОБОТА №6
РОЗРОБКА РУХІВ РОБОТУ З ВИКОРИСТАННЯМ ПАКЕТУ ROBOTICS TOOLBOX FOR PYTHON

6.1 Мета роботи

Розробити та реалізувати алгоритми для управління рухами робота, використовуючи пакет Robotics Toolbox for Python, з метою дослідження основних методів планування та контролю руху.

6.2 Теоретичні відомості

Пакет Robotics Toolbox for Python є потужним інструментом для моделювання та аналізу робототехнічних систем. Він надає функції для роботи з кінематики, динаміки, планування руху та контролю. Основні концепції, що стосуються розробки рухів робота, включають:

– кінематика: вивчає зв'язок між рухом роботів і їх геометричними характеристиками. Основні елементи кінематики включають прямий та зворотний розрахунок кінематики, які дозволяють визначити положення та орієнтацію робота на основі його суглобових кутів;

– планування руху: процес визначення оптимальної траєкторії, яку має пройти робот для досягнення заданої мети. Для цього використовуються різні алгоритми, такі як A*, RRT (Rapidly-exploring Random Tree), та інші методи, які допомагають уникнути перешкоди;

– контроль руху: включає реалізацію алгоритмів, які дозволяють роботу виконувати заплановану траєкторію. Це може включати PID-регулятори та інші методи, що забезпечують точність і стабільність руху;

– симуляція: Robotics Toolbox дозволяє симулювати рухи роботів у віртуальному середовищі, що є важливим для тестування алгоритмів перед їх реалізацією на фізичних роботах.

6.3 Порядок виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями та матеріалами лекції.
2. Реалізувати просту модель робота з визначеними кінематичними характеристиками.

3. Реалізувати алгоритм руху з використанням одного з алгоритмів, таких як A* або RRT.

4. Відповісти на контрольні питання.

5. Оформити звіт та завантажити до «Системи дистанційного навчання» НУ «Запорізька політехніка».

6.4 Індивідуальні завдання

1. Ознайомитися з основами пакету Robotics Toolbox for Python та його функціональними можливостями.

2. Реалізувати просту модель робота з визначеними кінематичними характеристиками.

3. Розробити алгоритм для прямого та зворотного розрахунку кінематики робота.

4. Реалізувати планування руху з використанням одного з алгоритмів, таких як A* або RRT.

5. Провести симуляцію руху робота та оцінити точність виконання траєкторії алгоритму руху.

6. Оформити звіт, який включає опис алгориму, результати експериментів, з графічними ілюстраціями результатів симуляції.

6.5 Контрольні запитання

1. Які основні елементи кінематики використовуються в робототехніці?

2. Які методи розрахунку кінематики існують, і в чому полягає різниця між прямим та зворотним розрахунком кінематики?

3. Як працює алгоритм планування руху A* і які його переваги?

4. Які методи контролю руху можуть бути використані для забезпечення точності виконання траєкторії?

5. Які можливості надає Robotics Toolbox for Python для симуляції роботів?

6. Як можна оцінити точність виконання траєкторії алгоритму руху?

ЛАБОРАТОРНА РОБОТА №7
РОЗРОБКА РУХІВ ПРОМИСЛОВИХ РОБОТІВ З ВИКОРИСТАННЯМ ROBODK

7.1 Мета роботи

Дослідити основи розробки рухів промислових роботів, використовуючи програмне забезпечення RoboDK. Реалізувати та протестувати алгоритми керування рухами для виконання промислових завдань, таких як маніпулювання об'єктами або збирання деталей.

7.2 Теоретичні відомості

RoboDK – це платформа для моделювання, програмування та симуляції промислових роботів, що дозволяє створювати моделі, розробляти траєкторії руху і навіть генерувати тексти програм для реальних роботів. У контексті розробки рухів промислових роботів важливо враховувати такі поняття:

– кінематика робота: кінематика вивчає рух механічної системи без урахування її маси та сил. Прямий розрахунок кінематики визначає положення і орієнтацію кінцевого ефектора на основі заданих суглобових кутів, тоді як зворотний розрахунок знаходить кути суглобів для досягнення певного положення кінцевого ефектора;

– планування траєкторії: робот має виконувати завдання, слідуючи заданій траєкторії, уникати перешкод та рухатися оптимальним шляхом для досягнення мети. RoboDK дозволяє легко налаштувати та оптимізувати траєкторії для конкретних завдань, наприклад, зварювання, пакування чи збирання;

– контроль руху: включає налаштування швидкості, точності й повторюваності рухів робота. У промислових системах важлива стабільність руху, щоб уникнути помилок під час виконання точних завдань;

– інтерфейс RoboDK: програма надає простий інтерфейс для симуляції, де користувач може вибрати різні моделі роботів, налаштувати їх траєкторії та виконувати симуляцію. Додатково RoboDK може генерувати тексти програм для роботів на різних мовах програмування C++, Python та ін., що робить його універсальним інструментом;

– синхронізація з фізичними роботами: RoboDK дозволяє експортувати розроблені алгоритми рухів у вигляді текстів програм, які можна завантажити на реальні роботи різних виробників, таких як ABB, KUKA, Fanuc та ін., що дозволить прискорити процес впровадження та тестування програм у реальних умовах.

7.3 Порядок виконання лабораторної роботи

1. Ознайомитися з теоретичними відомостями та матеріалами лекції.
2. Обрати модель промислового робота в RoboDK та налаштувати його кінематичні параметри.
3. Розробити алгоритм планування траєкторії робота для виконання виробничої операції (наприклад, збирання об'єктів, маніпуляція деталями або зварювання).
4. Відповісти на контрольні питання.
5. Оформити звіт та завантажити до «Системи дистанційного навчання» НУ «Запорізька політехніка».

7.4 Індивідуальні завдання

1 Ознайомитися з функціональними можливостями RoboDK для симуляції рухів промислових роботів.
2. Обрати модель промислового робота в RoboDK та налаштувати його кінематичні параметри.
3. Розробити алгоритм для прямого та зворотного розрахунку кінематики для керування рухом промислового робота.
4. Розробити алгоритм планування траєкторії робота для виконання однієї виробничої операції (наприклад, збирання об'єктів, маніпуляція деталями або зварювання).
5. Виконати симуляцію роботи робота та проаналізувати точність та плавність рухів у виконанні поставлених завдань.
6. Згенерувати текст програми для реального робота та дослідити можливості його використання у фізичних системах.
7. Оформити звіт, який включає опис алгориму, результати експериментів, з графічними ілюстраціями результатів симуляції.

7.5 Контрольні запитання

1. Що таке кінематика робота і в чому полягає різниця між прямим та зворотним розрахунком кінематики?
2. Які методи планування траєкторій використовуються для оптимізації рухів промислових роботів?
3. Як RoboDK допомагає у плануванні та моделюванні рухів промислових роботів?
4. Які фактори впливають на точність і стабільність рухів робота під час виконання завдань?
5. Які особливості синхронізації алгоритмів, розроблених у RoboDK, з реальними роботами?
6. Які показники важливі для оцінки якості виконання промислового завдання роботом?

ЛІТЕРАТУРА

1. Robotis simulation services [Електроний ресурс]: режим доступу - https://cyberbotics.com/

2. Robot Modeling and Control [Режим доступу]: http://www.coep.ufrj.br/~ramon/COE-841/robotics/book%202005%20-%20Robot%20Modeling%20and%20Control%20-%20Spong,%20Hutchinson%20&%20Vidyasagar.pdf

3. Проць Я. І. Захоплювальні пристрої промислових роботів: Навчальний посібник. /Я. І. Проць — Т: Тернопільський державний технічний університет ім. І. Пулюя, 2008. — 232 с.

4. David J. C. MacKay[en]. Information Theory, Inference, and Learning Algorithms Cambridge: Cambridge University Press, 2003.

5. Довідник ROS, [Режим доступу]: https://wiki.ros.org/ 6. Readme Robotics Toolbox for Python, [Режим доступу]: https://github.com/petercorke/robotics-toolbox-python

7. Peter Corke Robotics, Vision and Control: Fundamental Algorithms in Python (Springer Tracts in Advanced Robotics, 146) 3rd ed. 2023 Edition

8. Довідник «Robotics Toolbox for Python package», [Режим доступу]: https://petercorke.github.io/robotics-toolboxpython/arm_erobot.html#erobot

9. Basic Guide https://robodk.com/doc/en/Basic-Guide.html

10. Poole C. P., Owens F. J. Introduction to Nanotechnology. — New Jersey: Wiley–Interscience, 2003. — 388 p.

Додаток А

Текст програми контролеру руху walk для робототехнічної платформи Robotis OP2

```
#include "Walk.hpp"
#include <RobotisOp2GaitManager.hpp>
#include <RobotisOp2MotionManager.hpp>
#include <webots/Accelerometer.hpp>
#include <webots/Gyro.hpp>
#include <webots/Keyboard.hpp>
#include <webots/LED.hpp>
#include <webots/Motor.hpp>
#include <webots/PositionSensor.hpp>
#include <iostream>

using namespace webots;
using namespace managers;
using namespace std;

static const char *motorNames[NMOTORS] = {
  "ShoulderR", "ShoulderL", "ArmUpperR", "ArmUpperL", "ArmLowerR",
  "ArmLowerL", "PelvYR", "PelvYL", "PelvR", "PelvL",
  "LegUpperR", "LegUpperL", "LegLowerR", "LegLowerL", "AnkleR",
  "AnkleL", "FootR", "FootL", "Neck", "Head"
};

Walk::Walk() : Robot() {
  mTimeStep = getBasicTimeStep();

  getLED("HeadLed")->set(0xFF0000);
  getLED("EyeLed")->set(0x00FF00);
  mAccelerometer = getAccelerometer("Accelerometer");
  mAccelerometer->enable(mTimeStep);

  getGyro("Gyro")->enable(mTimeStep);

  for (int i = 0; i < NMOTORS; i++) {
    mMotors[i] = getMotor(motorNames[i]);
    string sensorName = motorNames[i];
    sensorName.push_back('S');
    mPositionSensors[i] = getPositionSensor(sensorName);
```

```
    mPositionSensors[i]->enable(mTimeStep);
  }

  mKeyboard = getKeyboard();
  mKeyboard->enable(mTimeStep);

  mMotionManager = new RobotisOp2MotionManager(this);
  mGaitManager = new RobotisOp2GaitManager(this, "config.ini");
}

Walk::~Walk() {
}

void Walk::myStep() {
  int ret = step(mTimeStep);
  if (ret == -1)
    exit(EXIT_SUCCESS);
}

void Walk::wait(int ms) {
  double startTime = getTime();
  double s = (double)ms / 1000.0;
  while (s + startTime >= getTime())
    myStep();
}

void Walk::run() {
  cout << "Press the space bar to start/stop walking" << endl;

  myStep();
  mMotionManager->playPage(9);
  wait(200);

  bool isWalking = false;

  while (true) {
    checkIfFallen();

    mGaitManager->setXAmplitude(0.0);
    mGaitManager->setAAmplitude(0.0);
```

```
    int key = 0;

    if (isWalking == true){
      mGaitManager->setXAmplitude(1.0);

        }
    while ((key = mKeyboard->getKey()) >= 0) {

      switch (key) {
        case ' ':
          if (isWalking) {
            mGaitManager->stop();
            isWalking = false;
            wait(200);
          } else {
            mGaitManager->start();
            isWalking = true;
            wait(200);
          }
      }
    }
    mGaitManager->step(mTimeStep);
    myStep();
  }
}

void Walk::checkIfFallen() {
  static int fup = 0;
  static int fdown = 0;

  static const double acc_step = 10;

  const double *acc = mAccelerometer->getValues();

  if (acc[1] < 450.0)
    fup++;
  else
    fup = 0;

  if (acc[1] > 550.0)
    fdown++;
```

```
  else
    fdown = 0;

  if (fup > acc_step) {
    mMotionManager->playPage(10);
    mMotionManager->playPage(9);
    fup = 0;
  }

  else if (fdown > acc_step) {
    mMotionManager->playPage(11);
    mMotionManager->playPage(9);

    fdown = 0;
  }
}
```