Вінницький національний технічний університет
факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Магістерська кваліфікаційна робота

на тему:

«Моделювання руху безпілотного автомобіля в симуляторі WeBots»

Виконав: студент групи 1АКІТ-22м
Спеціальності
151 – «Автоматизації та комп'ютерно інтегровані технології»
_____ Богдан КОЛЕСНИК
Керівник: к.т.н. доц.
_____ Ярослав КУЛИК
«4» грудня 2023 р.

Опонент: к.т.н. доц.
_____ Володимир ОЗЕРАНСЬКИЙ
«7» грудня 2023 р.

Допущено до захисту
Зав. кафедри АІІТ
д.т.н. проф.
_____ Олег БІСІКАЛО
«11» грудня 2023р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет

Факультет комп'ютерних систем та автоматики
Кафедра автоматики та інформаційно-вимірювальної техніки
Освітньо-кваліфікаційний рівень – магістр
Спеціалізація «Комп'ютеризовані системи управління та автоматика»
Спеціальність151 – «Автоматизація та комп'ютерно-інтегровані технології»
Освітня програма – Інтелектуальні компютерні системи.

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ д.т.н.
проф. Олег БІСІКАЛО
"20" 09 2023 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу

Колесніку Богдану Олеговичу

1. **Тема роботи** Моделювання руху безпілотного автомобіля в симуляторі WeBots

керівник роботи к.т.н., доц. каф. АІІТ Кулик, Я.А,

затверджені наказом ВНТУ №247 від "18" 09 2023 р.

2. **Строк подання студентом роботи** 05.12.2023р.

3. **Вихідні дані до роботи**

- нове положення автомобіля(куди слід рухатись)
- можливість змінювати швидкість до 60 км/год, прискорення автомобіля 40 км в год,
- можливість робити карту на основі даних JOSM.
- Об'їжджати перешкоди.

4. **Зміст розрахунково-пояснювальної записки**(перелік питань які необхідно зробити).1)Аналіз сучасного стану питання, 2)аналіз та вибір середовища розробки, 3)моделювання руху середовищі Webots, економічний розділ.

5. **Перелік графічного матеріалу** (з точним зазначенням обовязових креслень):
1) Діаграма класів, 2)Діаграма розгортки, 3)Діаграма життєвого циклу, 4) діаграма використання.

6.Консультанти розділів МКР

Розділ	Прізвище,ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання отримав
1, 2, 3	к.т.н. доц. Ярослав КУЛИК	20.09.23	
4	Професор кафедри ЕПВМ к.е.н. доц. Володимир КОЗЛОВСЬКИЙ	20.09	06.12.

7. Дата видачі завдання 20 09 2023

КЛАДЕНДАРНИЙ ПЛАН

№ з/п	Навза етапів МКР	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	21.09.23 – 28.09.23	вик.
2	Аналіз та вибір з існуючих симуляторів	16.10.23 – 28.10.23	вик.
3	Визначеня вимог до обраного симулятору	23.10.23 – 31.10.23	вик.
4	Налашування необхідних додатків	06.11.23 – 13.11.23	вик.
5	Моделювання руху в середовищі Webots	20.11.23 – 27.11.23	вик.
6	Економічна частина	20.11.23 – 27.11.23	вик.
7	Оформлення матеріалів та до захисту МКР	28.11.23 – 19.12.23	вик.

Студент _____ Богдан КОЛЕСНІК
(підпис) *(прізвище ініціали)*

Керівник роботи _____ Ярослав КУЛИК
(підпис) *(прізвище ініціали)*

АНОТАЦІЯ

УДК 004.94

Колеснік Богдан Олегович. та моделювання руху безпілотного автомобіля в симуляторі WeBots

Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація комп'ютерно інтегрованих систем управління. Інтелектуальні компютерні системи.

Вінниця ВНТУ, 2023 80 с.

На укр мові Бібліогр. 44 назв, рис 27, Табл 5.

У даній магістерській роботі здійснено всебічний аналіз та вибір оптимальних алгоритмів для застосування в галузі безпілотного транспорту. Зроблено огляд існуючих симуляторів, внаслідок чого було обрано середовище WeBots для емуляції руху безпілотних транспортних засобів.

У роботі представлений докладний теоретичний огляд, що охоплює ключові аспекти алгоритмів та методів, які застосовуються в безпілотних системах керування транспортом.

Для ефективного тестування розробленої системи було проведено експерименти у середовищі WeBots, що дозволяє точно емулювати рух безпілотних транспортних засобів. Отримані результати тестування підтвердили ефективність обраного алгоритму та правильність його реалізації.

Додатково, у роботі розглянуті технічні деталі, включаючи використання зовнішніх бібліотек та інструментів, таких як JOSM, pyproj та shaply, для забезпечення більш точного та надійного функціонування розробленої системи.

Ключові слова: Безпілотний транспорт, Симулятори моделювання руху, WeBots, Python, JOSM (Java OpenStreetMap Editor), pyproj, shaply, Емуляція руху безпілотних транспортних засобів

ANNOTATION

Bohdan Kolesnik and modeling the movement of an unmanned vehicle in the WeBots simulator

Master's thesis of a student on specialty 151 - Automation of computer-integrated control systems.

Vinnytsia VNTU, 2023 80 p.

In the Ukrainian language Bibliogr. 44 titles, figure 27, Table 5.

In this master's thesis, analysis and selection of optimal algorithms for application in the field of unmanned transport was carried out. A review of existing simulators was conducted, resulting in the selection of the WeBots environment for simulating the movement of driverless cars.

The paper presents a detailed theoretical overview covering key aspects of algorithms and methods used in unmanned vehicle control systems. The principles of operation of the selected algorithm, its features and advantages in the context of unmanned transport are considered.

In order to effectively test the developed system, experiments were conducted among WeBots, which allows you to accurately emulate the movement of unmanned vehicles.

Additionally, the paper discusses technical details, including the use of external libraries and tools such as JOSM, pyproj, and shaply, to ensure more accurate and reliable functioning of the developed system.

Keywords: Unmanned transport, Movement simulation simulators, WeBots, Python, JOSM (Java OpenStreetMap Editor), pyproj, shaply, Emulation of movement of unmanned vehicles

ЗМІСТ

ВСТУП

Актуальність дослідження полягає в тому, що розробка підсистеми уникнення зіткнень для безпілотних автомобілів допоможе запобігти можливим зіткненням безпілотних автомобілів з іншими транспортними засобами і перешкодами, на результатах емуляції можна розробляти нові рішення для дорожньо-транспортного руху.

Сучасний світ свідчить про швидкий розвиток безпілотних автомобілів як перспективної галузі автомобільної промисловості. Використання таких транспортних засобів може значно покращити безпеку дорожнього руху, зменшити кількість аварій і полегшити життя людей. Однак однією з основних проблем у реалізації безпілотного руху є забезпечення високого рівня безпеки через уникнення зіткнень з іншими об'єктами на дорозі.

Метою даного дослідження є моделювання підсистеми уникнення зіткнень для безпілотного автомобіля в симуляторі WeBots. Ця підсистема буде призначена для ефективного виявлення та уникнення зіткнень під час руху безпілотного автомобіля в симулювальному середовищі, Що повинно призвести до зменьшення кількості зіткнень, та людського фактору у порівнянні зі станом до запровадження цієї системи

Об'єктом дослідження є процес керування безпілотного автомобіля.

Предметом дослідження є підсистема уникнення зіткнень в симуляторі WeBots.

Для досягнення поставленої мети необхідно розв'язати наступні задачі:

Аналіз і літературний огляд - Провести огляд існуючих методів та технологій уникнення зіткнень для безпілотних автомобілів, а також оцінити їх ефективність і обмеження.

Аналіз та вибір середовища розробки – провести огляд існуючих емуляторів, обрати той який задовольняє наступні вимоги, доступність, можливість програмування поведінки,

Імплементація в обраному симуляторі - Реалізування алгоритму у симуляторі WeBots для подальших тестувань та експериментів.

Експерименти та оцінка ефективності - Провести серію експериментів для оцінки ефективності розробленої підсистеми уникнення зіткнень у різних сценаріях та умовах.

Аналіз результатів і висновки - Проаналізувати результати експериментів і зробити висновки щодо роботи підсистеми уникнення зіткнень в симуляторі WeBots, а також її можливого застосування в реальних умовах.

Методи дослідження:

Аналіз літератури і існуючих рішень – Цей метод дозволить оцінити існуючі підходи та технології у сфері уникнення зіткнень для безпілотних автомобілів, враховуючи їх переваги та обмеження.

Розробка алгоритму – Розробка алгоритму уникнення зіткнень буде базуватися на математичних методах та аналізі даних для виявлення потенційних загроз та прийняття рішень.

Симуляція в середовищі WeBots – Для перевірки алгоритму буде використано симулятор WeBots, що дозволить створити різні дорожні сценарії та виконати тестування безпілотного автомобіля у віртуальному середовищі.

Експерименти і аналіз даних – Дані, отримані під час експериментів, будуть аналізуватися статистично для визначення ефективності розробленого алгоритму та виявлення можливих вдосконалень.

Новизна одержаних результатів

В результаті роботи проведені моделювання дозволять виявити можливості оптимізації алгоритму уникнення зіткнень, що є важливим для подальшого розвитку цієї технології.

Практична цінність

В результаті виконання роботи буде отримано вдосконалений алгоритм, програмне забезпечення та налаштоване середовище

Апробація результатів. Основні результати магістерської дипломної роботи прийшли апробацію на LII Науково-технічній конференції підрозділів Вінницького національного технічного університету (м. Вінниця, 2023).

Публікація. За результатами магістерської дипломної роботи опубліковано доповідь у матеріалах LII Науково-технічної конференції підрозділів Вінницького національного технічного університету [45]

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ

1.1 Основні види автоматизованого транспорту

Розрізняють різні види автоматизації автомобіля в залежності від необхідної участі людини, серед яких можна виділити три основних, а саме інформування(попередження при небезпечних ситуаціях), допомога(зміна швидкості, гальмування), та контроль,(див рис 1.1)[1]

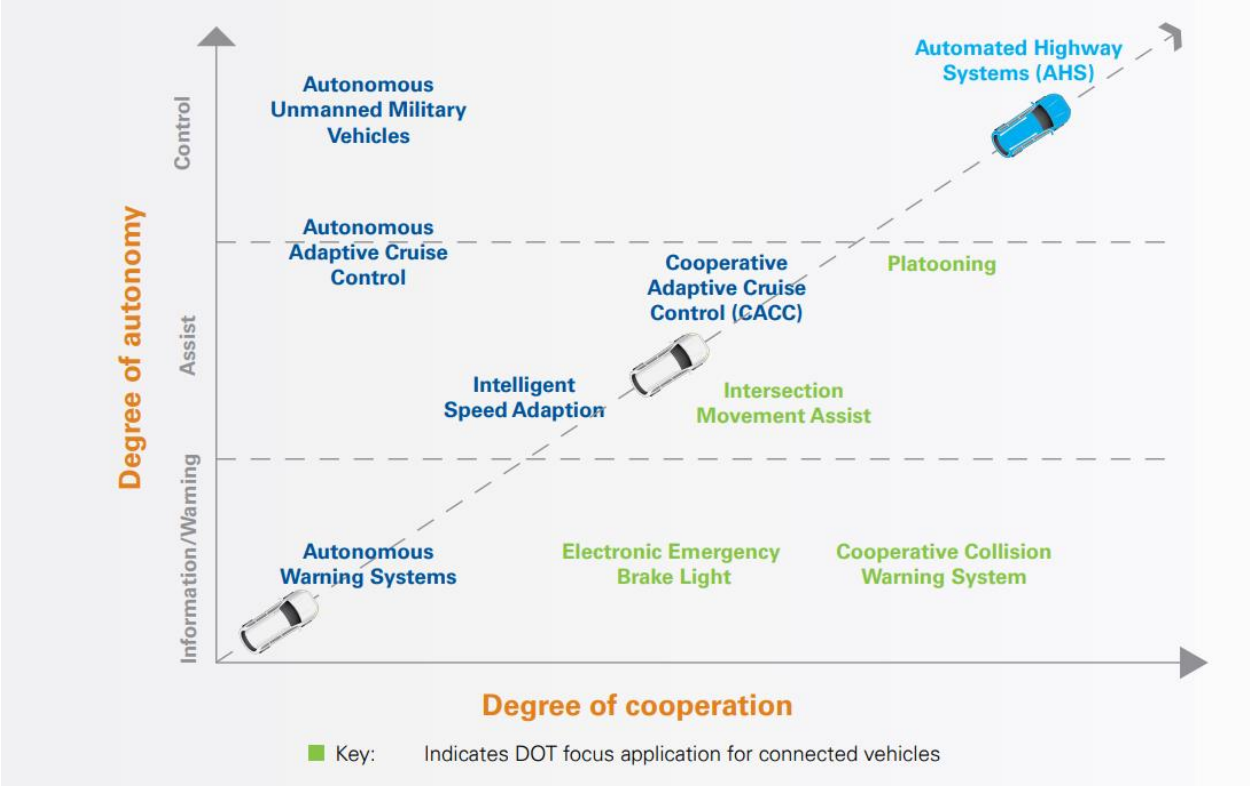


Рисунок 1.1 – Види автомобілів у залежності від рівня автоматизації керуванням

Для виконання поставленої задачі необхідно створити, або скористатись уже готовими рішеннями, повністю автоматизований транспорт. Сучасні

інновації у області автоматичного керування транспортом являються невідємною частиною нашої швидкої та динамічної епохи, вносять значні та перспективні рішення у наший повсякденний образ життя.

Дана технологія має багато застосувань, та може якщо не вирішити то зменшити кількість проблем при дорожньо транспортнму русі. Приклади застосування даної технології;

Автоматизовані автомобілі:

Безпілотні автомобілі[2] Це транспортні засоби, які можуть переміщатися без участі водія, або з мінімальним його втручанням Вони можуть використовувати різні сенсори, камери, радари та інші прилади для навігації та орієнтування себе у просторі.

Безпілотні вантажівки[3]:

Вантажні транспортні засоби без водія, або з мінімальним його втрчанням: Подібні технології також можуть використовуватись для автоматизації вантажних перевезень, що може підвищити ефективність та безпеку у сфері вантажоперевезень.

Безпілотні дрони:

Автономні безпілотні літальні апарати (Дрон): Ці пристрої можуть використовуватися для доставки вантажів, аерозйомки, агрокультур та інших завдань, на момент написання магістерської, застосовуються воєнними. Вони керуються віддалено чи програмно.

1.2 Аналіз та вибір алгоритмів для безпілотного транспорту

Для початку необхідно сформувати діаграмму класів, щоб зрозуміти яка має бути архітектура для программи(див рисунок 1.2.1)

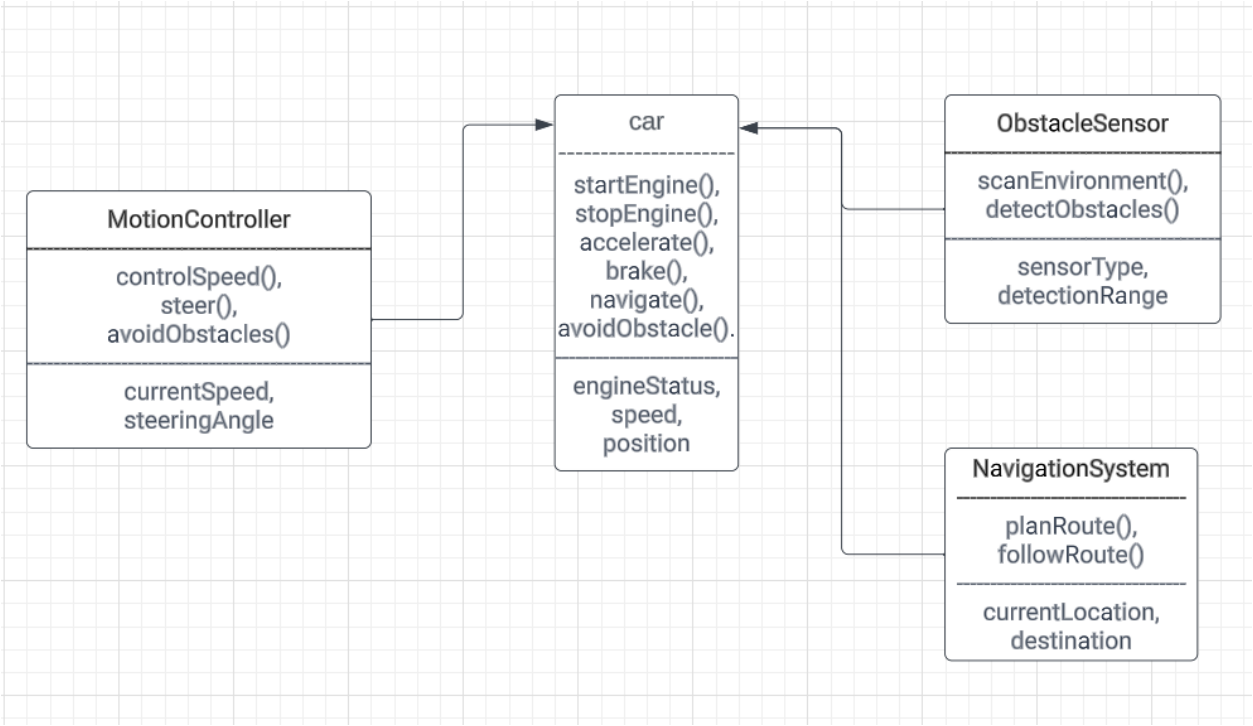


Рисунок 1.1 – Діаграма класів

Серед основних алгоритмів можна виділити чотири:

SLAM, A*, алгоритм косяку риб, нейромережі, розглянемо кожен із них, виділимо переваги та недоліки кожного із них.

1.2.1 SLAM

SLAM (Simultaneous Localization and Mapping) - це алгоритмічний підхід, що використовується в робототехніці та комп'ютерному зору для вирішення проблеми одночасної локалізації і картографування в незнайомому середовищі. дати змогу роботам або іншим мобільним системам створювати карту навколишнього світу і визначати своє положення в цій карті одночасно, користуючись даними з датчиків, таких як камери, лідари, акселерометри і гіроскопи [4] (див Рисунок 1.2.1)

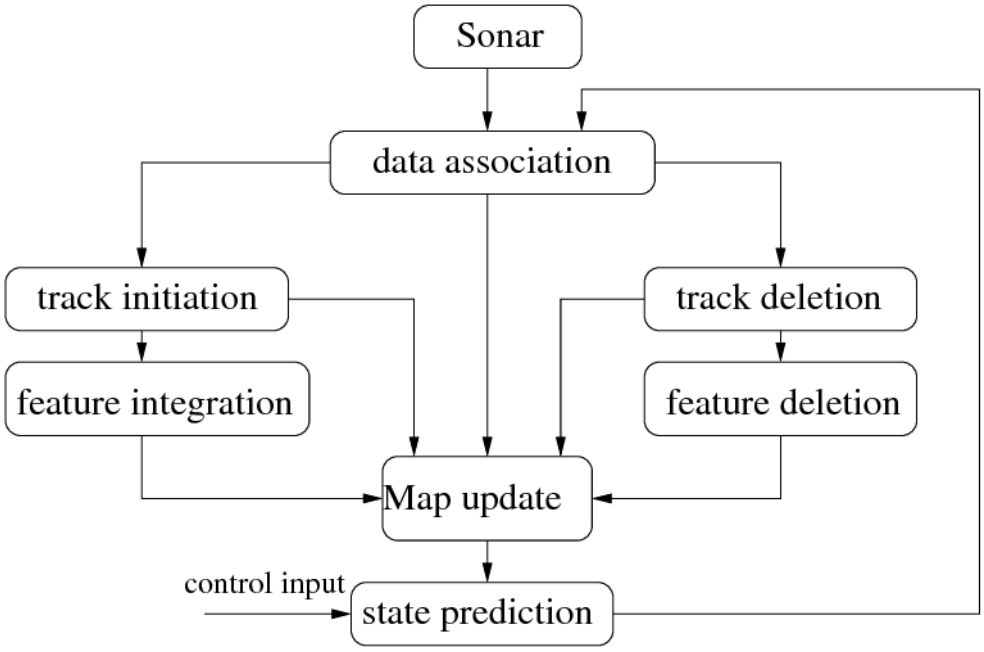


Рисунок 1.2 – Алгоритм SLAM

Основні етапи алгоритму SLAM:

- Збір даних– Система отримує дані з різних датчиків, такі як візуальні дані від камери, відстані від лідара, орієнтаційні дані від гіроскопів і акселерометрів тощо.

- Екстракція ознак– З оброблених сенсорних даних вилучаються ключові ознаки, такі як розпізнавання особливих точок на Рисунок або вимірювання відстаней і кутів на плані лідару.
- Локалізація– На цьому етапі робот або система обчислює своє поточне положення відносно створеної карти, використовуючи вихідні дані та інформацію про розташування ознак на карті.
- Картографування– За допомогою отриманих даних про локалізацію і визначення ознак, система оновлює свою карту середовища, додавши нові об'єкти та ознаки до неї.
- Покращення локалізації і картографування – Зазвичай SLAM використовує ітеративний підхід, в якому локалізація та картографування оновлюються разом, покращуючи точність з кожною ітерацією.
- Оцінка невизначеності – Важливо враховувати невизначеність в оцінках локалізації і картографування і розглядати можливість помилок і неточностей в даних датчиків.
- Підтримка актуальності картографії – Карта середовища постійно оновлюється з урахуванням нових даних і виправлень.
- Використання картографії і локалізації для навігації – Після завершення процесу SLAM система може використовувати створену карту для навігації в середовищі, визначення оптимального маршруту і виконання завдань.

Переваги SLAM:

- Автономність – SLAM дозволяє роботам і системам працювати в незнайомих середовищах без попередньої інформації про карту. Він дозволяє системам створювати і поновлювати карту "на льоту".
- Придатність для різних датчиків – SLAM може використовувати різні типи датчиків, включаючи камери, лідари, радіолокацію тощо, що робить його витратним та гнучким у застосуванні.

- Широкий спектр застосувань – Він може бути використаний в багатьох областях, включаючи робототехніку, беспілотні автомобілі, маппінг середовищ, внутрішню навігацію тощо.
- Реалізація в реальному часі – Деякі варіації SLAM можуть працювати в реальному часі, що важливо для роботів і систем, що потребують швидкої реакції на зміни в навколишньому середовищі.

Недоліки SLAM:

- Обчислювальна складність – SLAM може вимагати значних обчислювальних ресурсів, особливо при використанні багатьох датчиків або в складних середовищах.
- Невизначеність і помилки – Вимірювання датчиків завжди мають певну невизначеність, і це може призвести до накопичення помилок у карті і локалізації з часом.
- Потреба в налагодженні – SLAM вимагає налагодження та калібрування датчиків для досягнення найкращих результатів. Це може бути трудомістким завданням.
- Залежність від освітлення і середовища – Використання камер для SLAM може бути обмеженим у поганих умовах освітлення або в складних середовищах зі значними змінами.
- Витратність енергії – В деяких випадках SLAM може вимагати великої кількості енергії, що може бути проблемою для портативних систем, якими машини і являються.
- Етичні питання і конфіденційність – Використання SLAM у великих містах або в інших областях може породжувати питання щодо приватності і використання даних, чим можна знехтувати в умовах емульованого простору.

1.2.2 A*

A* - це алгоритм пошуку шляху, який використовується для пошуку найкоротшого шляху між двома точками на графі або сітці. Він поєднує в собі ідеї алгоритму Дейкстри та евристичного підходу для оптимізації пошуку. A* широко застосовується в різних областях, таких як штучний інтелект, комп'ютерні ігри, робототехніка та інші.[5]

Основна ідея A* полягає в тому, щоб обчислити собівартість (або "вартість") кожного вузла у графі на основі двох компонентів:

- G-значення (cost-so-far, вартість від початкового вузла до поточного вузла) – Це фактична вартість шляху від початкового вузла до поточного вузла.
- H-значення (heuristic, евристична оцінка вартості від поточного вузла до кінцевого вузла) – Це оцінка вартості шляху від поточного вузла до кінцевого вузла на основі якогось правила чи евристичного підходу.
- A* обчислює F-значення (total-cost, загальна вартість) кожного вузла, яке є сумою G- та H-значень – F = G + H. Потім він вибирає вузел з найменшим F-значенням та розглядає його сусідів для подальшого розгляду.

Процес вибору та розгляду вузлів триває до тих пір, поки не буде знайдено шлях до кінцевого вузла або виявлено, що такого шляху не існує. A* забезпечує збереження коротших шляхів на початку пошуку завдяки використанню F-значень і евристичних оцінок.

Реалізація алгоритму A* у псевдокоді

```
function AStar(start, goal)
    openSet := {start}
    closedSet := empty set
    cameFrom := empty map
```

```
gScore := map with default value of infinity
gScore[start] := 0
fScore := map with default value of infinity
fScore[start] := heuristic_cost_estimate(start, goal)
while openSet is not empty
    current := node in openSet with the lowest fScore value
    if current == goal
        return reconstruct_path(cameFrom, goal)
    openSet := openSet - {current}
    closedSet := closedSet + {current}
    for each neighbor of current
        if neighbor in closedSet
            continue
        tentative_gScore := gScore[current] + distance_between(current, neighbor)
        if neighbor not in openSet or tentative_gScore < gScore[neighbor]
            cameFrom[neighbor] := current
            gScore[neighbor] := tentative_gScore
            fScore[neighbor] := gScore[neighbor] + heuristic_cost_estimate(neighbor, goal)
            if neighbor not in openSet
                openSet := openSet + {neighbor}
return failure
```

```
function reconstruct_path(cameFrom, current)
    total_path := [current]
    while current in cameFrom.keys()
        current := cameFrom[current]
        total_path := [current] + total_path
    return total_path
```

Переваги A*:

- При правильному виборі евристичної функції H A* гарантує знаходження найкоротшого шляху між початковим і кінцевим вузлами.
- A* зазвичай працює ефективно і швидко, особливо якщо використовується пріоритетна черга для вибору вузлів для розгляду.
- Алгоритм може бути використаний в різних застосуваннях і налаштований під конкретні умови завдання шляху, вибираючи різні евристичні функції.
- A* може легко враховувати вагу ребер графа, що дозволяє враховувати обмеження шляху відносно ваги ребер.

Недоліки A*:

- Евристична оцінка H – Вибір неправильної евристичної оцінки H може призвести до неправильних результатів або навіть до того, що A* буде не оптимальним.
- Пам'ять – Алгоритм може вимагати значної кількості пам'яті для збереження великих графів або сіток, особливо в надзвичайно великих задачах.

- Завдання з великою кількістю вузлів – В задачах з великою кількістю вузлів A* може бути вельми обчислювально витратним, особливо якщо евристична оцінка надто погана.
- Неможливість роботи в обмеженому часі – На практиці A* не завжди може знаходити рішення в обмеженому часі, особливо в задачах з великою кількістю можливих шляхів.
- Залежність від структури графа – У деяких випадках, якщо структура графа не вдало підходить для A*, або якщо граф неадекватно представлений, алгоритм може давати неправильні результати.

1.2.3 рої риб

Алгоритм рої риб - це один із способів оптимізації обчислень, який використовує прикладну паралельність для розв'язання обчислювальних завдань. Він отримав свою назву через аналогію зі способом, яким риби спільно вирушають в косяк для максимальної вигоди. Основна ідея полягає в тому, що багато невеликих обчислювальних задач обробляються паралельно декількома обчислювальними вузлами, які працюють над однією та ж задачею.[6] (див Рисунок 1.2.2)

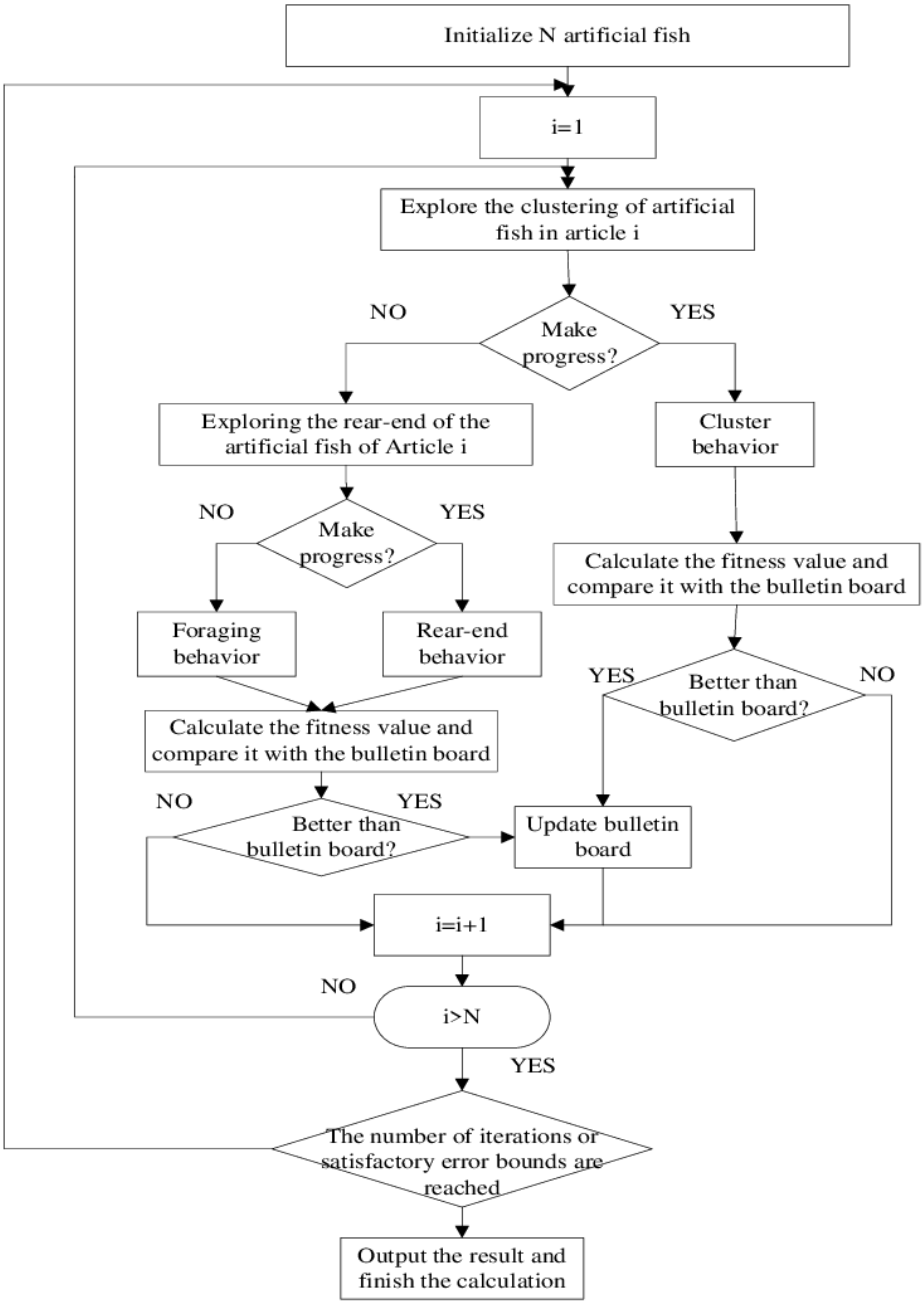


Рисунок 1.2 – Алгоритм рою риб

Переваги алгоритму " рої риб " включають

– Алгоритм може бути легко паралелізований, що дозволяє використовувати багато обчислювальних вузлів або процесорів для рішення задачі. Це значно збільшує продуктивність і швидкість виконання.

– За необхідності можна додавати нові вузли або ресурси для розширення потужності обчислень. Це дозволяє ефективно розв'язувати великі задачі.

– Завдяки розділенню задачі на багато менших підзадач, система залишається стійкою до відмов окремих вузлів. Якщо один з вузлів відмовляє, інші можуть продовжувати обробку.

Недоліки алгоритму "косяк риб" включають.

– Для ефективного використання алгоритму необхідна досить складна система управління та координації між обчислювальними вузлами. Це може призвести до значних накладних витрат на обмін даними та синхронізацію.

– Розробка програмного забезпечення для роботи з алгоритмом "косяк риб" може бути складною через необхідність детального проектування та управління паралельністю.

– Не всі види завдань підходять для оптимізації за допомогою алгоритму "косяк риб". Він найбільш ефективний для задач, які можна розділити на незалежні підзадачі.

– Алгоритм нейромережі для системи уникнення зіткнень в автономних транспортних засобах (наприклад, в автономних автомобілях) може включати в себе кілька етапів. Ось загальний опис такого алгоритму.

– Сенсори, такі як радари, лідари, камери і ультразвукові датчики, збирають інформацію про оточуюче середовище. Ця інформація може включати в себе дані про інші автомобілі, пішоходів, дорожні знаки, сигнальні світлофори тощо.

– Отримана інформація обробляється, фільтрується і аналізується для виділення важливих об'єктів та передачі цих даних нейромережі.

1.2.4 Нейромережі

– Нейромережа, зазвичай, глибокий навчальний персептрон (Deep Neural Network), приймає на вхід дані з сенсорів та іншу інформацію про стан автомобіля і оточуючого середовища. Мережа навчається прогнозувати траєкторію руху і рішення для уникнення зіткнень. Це може включати в себе розрахунок шляху, швидкості, гальмування і керування кермом.[7]

– На основі виходів нейромережі приймаються рішення про те, як реагувати на потенційні загрози зіткнень. Це може включати в себе зміну швидкості, напрямку руху, зупинку або об'їзд об'єктів.

– Автомобіль виконує розраховані дії, щоб уникнути зіткнень. Це може включати в себе автоматичне керування кермом, гальмування, рух по іншій смузі або інші маневри.

Переваги алгоритму нейромережі для уникнення зіткнень:

– Нейромережі можуть навчатися і підлаштовуватися до різних умов дорожнього руху та оточуючого середовища.

– Завдяки апаратній прискореній обробці, нейромережі можуть досить швидко реагувати на зміни в оточуючому середовищі.

Недоліки нейромереж:

– Нейромережі потребують великої кількості даних для ефективного навчання, і навіть тоді вони можуть бути схильні до навчального перенавчання або недонавчання.

Виходячи із особливостей кожного із алгоритмів, можна сказати що ні 1 із них сам по собі не виконує обидві задачі а саме уникнення перешкод для

безпілотного автомобіля, але надає можливість зробити його безпілотним в умовах емуляції. Виходячи з переваг та недоліків, для оптимальної емуляції було обрано метод A*.

1.3 Аналіз та вибір методів для уникнення зіткнень

Варто виділити наступні методи Системи детекції і візуального спостереження, Глобальні системи позиціонування.

1.3.1 Системи детекції і візуального спостереження

Системи детекції і візуального спостереження представляють із себе лідари, камери, інфрачервоні датчики і тд.[8]

Переваги:

- Системи детекції дозволяють безпілотним автомобілям виявляти і реагувати на небезпечні ситуації швидше, ніж людина. Це може сприяти зменшенню аварій і травматичних подій на дорогах.

- Безпілотні автомобілі можуть функціонувати без перерви та втоми, що робить їх ефективними для довгих поїздок і комерційних перевезень.

- Збільшення автоматизації дорожнього руху може допомогти уникнути заторів, оптимізувати рух та зменшити викиди CO_2 завдяки ефективнішій їзді.

- Безпілотні автомобілі можуть вести транспортні послуги для людей, які не можуть водити з різних причин, таких як вік, інвалідність або алкогольне сп'яніння.

– безпілотний автомобіль, система зіткнень якого реалізована завдяки даному методові, незалежить від отримання іншої інформації, так як всю інфформацію збирає сам, що робить його універсальним для різних умов експлуатації.

Недоліки:

- Системи детекції і візуального спостереження є дорогими. Вони вимагають встановлення і обслуговування спеціалізованого обладнання, яке може бути дорожчим за сам автомобіль.

- Погодні умови, особливо туман, дощ або сніг, можуть впливати на здатність систем до нормальної роботи.

- Збір великих обсягів даних про дорожні умови та відомостей про користувачів може створити проблеми з приватністю та безпекою цих даних.

1.3.2 Глобальні системи позиціонування

- це сучасні технології, які дозволяють точно визначити місцезнаходження об'єкта на поверхні Землі за допомогою сигналів, що надходять від спеціальних супутникових систем. Найвідомішою та широко використовуваною ГСП є система GPS (Global Positioning System), проте існують інші аналогічні системи[9]

Переваги:

- ГСП може надавати дуже високу точність при визначенні місцезнаходження, часто на рівні кількох метрів або менше.

- ГСП працює в будь-якій точці земної поверхні, де є видимість до достатньої кількості супутників.

- Велика кількість супутників і поширена інфраструктура дозволяють використовувати ГСП практично всюди.

- ГСП використовується у багатьох галузях, включаючи автомобільну навігацію, авіацію, мореплавство, агрокультуру, геодезію, рятувальні операції та багато інших.

Недоліки:

- ГСП потребує доступу до сигналів від супутників, що може бути ускладненим або неможливим в глибоких долинах, ущелинах або в густому лісі.

- Фізичні перешкоди, такі як високі будівлі або гірські масиви, можуть вплинути на якість сигналів та точність ГСП.

- Використання ГСП може викликати питання стосовно приватності, оскільки можливо відслідковувати місцезнаходження користувачів без їхньої згоди.

Виходячи із переваг та недоліків було обрано метод системи детекції.

1.4 Висновки до першого розділу

У цьому розділі проведений ретельний аналіз класифікації автомобілів залежно від ступеня автоматизації. Увага була приділена ключовим методам, які застосовуються при автоматизації процесів прокладання маршрутів та локалізації. В результаті цього аналізу виділено чотири перспективні методи, кожен з яких є потенційним рішенням для певних сценаріїв застосування. Після уважного розгляду цих методів було зроблено вибір на користь одного з них з огляду на вимоги проекту.

Крім цього, висвітлено методи відстеження перешкод та запобігання зіткненням. Проаналізовано різні технології, включаючи сенсори, системи комп'ютерного зору та радіолокацію, що застосовуються для забезпечення безпеки руху. На основі даного аналізу було зроблено вибір найкращого методу, який забезпечує ефективне уникнення зіткнень у різних сценаріях руху автомобілів.

Цей всебічний аналіз як надає огляд сучасного стану технологій автоматизації у транспортній сфері, а й визначає конкретні рішення, які можуть бути успішно інтегровані у проект автоматизованого транспорту. Отримані результати забезпечують основу для розробки та впровадження ефективних та інноваційних систем керування, сфокусованих на підвищенні безпеки та ефективності автомобільного руху.

2 АНАЛІЗ ТА ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

2.1 Аналіз та вибір з існуючих Симуляторів

Симулятор – імітатор, механічний або комп'ютерний, який відтворює управління будь-яким процесом, технічним чи апаратним засобом. Також можна трактувати це наступним чином: симулятори – набір програмних та апаратних засобів, що створюють враження дійсності, відображаючи частину реальних явищ і властивостей у віртуальному середовищі.

Симуляція – використання комп'ютерів для імітації реально ситуацій. Одні симулятори використовуються для навчання та тренування в керуванні машинами, інші ж є програмами, які передбачають (або роблять спробу передбачити) події реального світу[10].

2.1.1 SolidWorks Simulation

Пакет SolidWorks Simulation – це програмне забезпечення для проведення комп'ютерного моделювання та аналізу механічних інженерних систем. Він дозволяє створювати віртуальні прототипи, визначати статичні, динамічні та інші параметри системи.[11]

Переваги:

- SolidWorks Simulation інтегрується безпосередньо з CAD-системою SolidWorks, що дозволяє легко створювати та аналізувати моделі без необхідності переходити між різними програмами.

- Він дозволяє виконувати широкий спектр аналізів, таких як статичний, динамічний, термальний, міцнісний, оптичний та інші. Це дозволяє оцінювати різні аспекти продукту.

- Програма використовує точні алгоритми для розв'язання складних завдань аналізу, що дозволяє отримувати достовірні результати.

- Він надає інструменти для відображення результатів аналізу у вигляді візуальної інформації, такої як анімації, графікіи та діаграми.

Недоліки:

- SolidWorks Simulation є комерційним програмним забезпеченням, його вартість може бути високою, що може бути обмежуючим фактором для невеликих компаній чи індивідуальних користувачів.

- Деякі складні типи аналізу можуть вимагати значної обчислювальної потужності та години для завершення.

- Simulation дозволяє моделювати багато видів систем, він може бути обмежений у моделюванні дуже складних фізичних процесів.

2.1.2 NXT-G

NXT-G (NXT Graphical Programming Environment) - це графічне середовище програмування, створене для програмування роботів з лінійки LEGO Mindstorms NXT. Воно розроблено спеціально для навчання дітей та початківців основам програмування та робототехніки[14]

Переваги:

- NXT-G використовує графічний інтерфейс, що дозволяє користувачам створювати програми, перетягуючи та з'єднуючи блоки з візуального інтерфейсу, що робить програмування більш доступним і зрозумілим для початківців.

- Це програмне забезпечення дозволяє створювати різноманітні програми для керування роботами, включаючи вимірювання відстані, рух, звуковий сигнал і багато інших функцій.

Недоліки:

- NXT-G призначений для початківців, і він може бути обмежений в розв'язанні складних програмних завдань. Він не надає той же рівень гнучкості та розширюваності, що інші мови програмування.

- NXT-G розроблено спеціально для роботів LEGO Mindstorms NXT, і він не підходить для інших апаратних платформ.

- LEGO більше не розвиває NXT-G, оскільки його припинено в 2009 році. Це означає, що NXT-G не має останніх оновлень та підтримки.

2.1.3 Robotics System Toolbox

- це набір інструментів для програмування та моделювання роботів в середовищі MATLAB і Simulink. Він призначений для розробки, аналізу і симуляції робототехнічних систем.[14]

Переваги:

- Цей набір інструментів надає широкий спектр функцій і функціональностей для робототехнічних досліджень і розробки, включаючи кінематику, динаміку, планування траєкторії, взаємодію з оточенням і багато іншого.

– має можливість нтеграції з MATLAB і Simulink, що спрощує інтеграцію роботів з іншими системами і алгоритмами.

- Robotics System Toolbox дозволяє створювати динамічні моделі роботів та виконувати симуляції для аналізу їхньої поведінки перед фізичною реалізацією.

- Він підтримує роботів різних типів і конфігурацій, включаючи маніпулятори, мобільні роботи, дрони і багато інших.

- Існують навчальні ресурси, приклади і документація, що спрощують процес навчання і розробки.

Недоліки:

- MATLAB і відповідний Robotics System Toolbox є комерційними продуктами що обмежує їх використання та розповсюдження.

- Великі і складні симуляції робототехнічних систем можуть вимагати потужних обчислювальних ресурсів.

- Щоб ефективно використовувати цей інструмент, користувачам може знадобитися досвід у програмуванні та математичному моделюванні.

- Деякі конкретні роботи можуть вимагати додаткової розробки або інтеграції для повної підтримки.

2.1.4 AnyKode Marilou Robotics Studio

- це програмне забезпечення для робототехнічного моделювання і симуляції, яке призначено для розробки та тестування роботів та автономних систем. Воно дозволяє користувачам створювати та тестувати віртуальні роботи та програми для них[14]

Переваги:

- Marilou Robotics Studio надає користувачам можливості для створення та налаштування роботів, та середовища для симуляції. Є можливість створення власних моделей роботів та налаштовування їх параметрів.

- Він дозволяє моделювати різні види роботів, включаючи маніпулятори, мобільні роботи, дрони та інші, розробляти для них різні програми та сценарії.

- Marilou Robotics Studio має потужні засоби візуалізації, які допомагають користувачам спостерігати за роботами та аналізувати їхню поведінку в реальному часі.

- Це дозволяє зменшити витрати та час, пов'язані з фізичними тестами роботів, оскільки ви можете спробувати різні сценарії та алгоритми віртуально.

Недоліки:

- Програмне забезпечення Marilou Robotics Studio є комерційними продуктами що обмежує їх використання та розповсюдження.

- Для новачків у робототехніці може бути складним зрозуміти та використовувати всі функції цього програмного забезпечення.

- Великі та складні симуляції можуть вимагати потужних комп'ютерів для ефективної роботи.

- Marilou Robotics Studio може бути обмеженим у підтримці деяких робототехнічних платформ або обладнання.

2.1.5 Webots

Webots - це віртуальне середовище для симуляції робототехнічних систем та інших автономних агентів. Воно використовується для розробки, тестування та валідації робототехнічних програм та алгоритмів перед їхньою реалізацією на фізичних роботах.[17]

Переваги:

- Webots надає широкий спектр можливостей для створення власних симуляцій та моделей роботів. Ви можете визначати фізичні характеристики робота та його оточення.

- Веботс підтримує різні робототехнічні платформи, включаючи роботів від таких виробників, як LEGO, NAO, Pepper та багатьох інших.

- Ви можете моделювати різні типи сенсорів та актуаторів для роботів, що дозволяє тестувати різні алгоритми для взаємодії з оточенням.

- Webots має потужні інструменти візуалізації, які допомагають користувачам спостерігати за роботами та аналізувати їхню поведінку в реальному часі.

– є можливість програмувати роботів в Webots за допомогою різних мов програмування, таких як C++, Python, Java.

Недоліки:

Великі обчислювальні вимоги - Великі та складні симуляції в Webots можуть вимагати значної обчислювальної потужності, що може бути обмежуючим фактором для деяких користувачів.

- Для комерційного використання Webots може бути високим витратами, хоча є безкоштовна версія для академічного використання.

- Веботс може бути складним для вивчення, особливо для початківців у робототехніці та програмуванні.

2.1.6 OROCOS

(Open Robot Control Software project) [17] – текстовий симулятор. Є безкоштовним продуктом і використовується як додатковий інструмент для різних симуляторів. Підтримує мову програмуванняC/C++

Переваги:

- Відкрите програмне забезпечення: OROCOS є проектом з відкритим вихідним кодом, що дозволяє користувачам вільно поширювати, модифікувати та покращувати його відповідно до їхніх потреб.
- Модульна архітектура: OROCOS побудований з урахуванням модульної архітектури, що полегшує інтеграцію та перевикористання різних компонентів системи управління роботами.
- Реальний час: OROCOS надає засоби для роботи в режимі реального часу, що робить його придатним для додатків, де потрібна висока чутливість системи, таких як робототехніка.
- Підтримка різних мов програмування: OROCOS підтримує кілька мов програмування, включаючи C++, що може бути зручним для розробників з різним досвідом та уподобаннями.

Недоліки:

- Крута крива навчання: Використання OROCOS може вимагати від розробників час на освоєння та розуміння його концепцій та методів програмування.
- Обмежена спільнота: Порівняно з деякими іншими фреймворками, спільнота OROCOS може бути меншою, що може утруднити отримання підтримки та вирішення проблем.
- Не настільки поширений: На відміну від деяких популярних фреймворків для робототехніки, таких як ROS (Robot Operating System), OROCOS може бути менш поширеним і менш популярним серед спільноти розробників.
- Необхідність ресурсів: Використання реального часу потребує більш високих вимог до ресурсів, що може зробити системи, побудовані з використанням OROCOS, менш придатними для більш обмежених ресурсів систем.

2.1.7 V-REP

є одним з найкращих рішень серед симуляторів, володіє великою кількістю функцій, а також складним API. Універсальний симулятор робототехніки V-REP, з інтегрованим середовищем розробки, заснований на розподіленій архітектурі управління: кожен об'єкт/модель може керуватися індивідуально за допомогою вбудованого сценарію, плагіна, вузла ROS, клієнта віддаленого API або рішення користувача. Підходить для застосування в кількох роботах. Програмний код може бути написаний на C / C ++, Python, Java, Lua, Matlab, Octave, але генерацію коду потрібно реалізувати через API, що є трудомістким процесом. Використовується для швидкої розробки алгоритмів, моделювання автоматизації виробництва, швидкого прототипування та верифікації. робототехніки, віддалений моніторинг і т.д.

Підтримує інтегрування сторонніх симуляторів, а також різні GUI-інтерфейси та користувальницькі драйвера[17].

Переваги:

- Багатозадачність: V-REP підтримує одночасне виконання кількох сценаріїв та сцен, що забезпечує гнучкість при моделюванні різних робототехнічних завдань.
- Широкий вибір моделей роботів та датчиків: Платформа надає велику бібліотеку готових моделей роботів та датчиків, що спрощує процес створення різноманітних сценаріїв симуляції.
- Інтеграція з ROS: V-REP має інтеграцію з ROS, що полегшує спільне використання даних між симуляцією та реальними роботами, що працюють на ROS.

Недоліки:

- Обмежена продуктивність: При роботі з великими та складними моделями роботів та сцен V-REP користувач може зіткнутися з обмеженнями продуктивності.
- Закрите програмне забезпечення: Незважаючи на те, що V-REP надає безкоштовну версію, він не є повністю відкритим кодом. Це може обмежувати можливості користувачів у модифікації та покращенні платформи.
- Обмежена документація та спільнота: користувачі можуть зіткнутися з обмеженою документацією та відсутністю розвиненого співтовариства, що ускладнює отримання підтримки та вирішення проблем.

2.1.8 RobotC

текстовий симулятор, є лідером серед мов програмування для вивчення роботів та підготовки до змагань. Він заснований на мові програмування C, і має простий у використанні середовищем розробки. Є платним програмним забезпечення[18].

Переваги:

- RobotC базується на мові програмування C, що робить його відносно простою для вивчення, особливо для тих, хто вже знайомий з мовою C.
- RobotC часто використовується в освітніх програмах, таких як FIRST Tech Challenge та FIRST Lego League, що полегшує впровадження у навчальних навчальних програмах.
- RobotC підтримує широкий спектр робототехнічних платформ, таких як LEGO Mindstorms та VEX Robotics, що робить його універсальним інструментом.
- Дозволяє програмістам використовувати безліч функцій та бібліотек для складнішого програмування роботів, включаючи керування моторами, сенсорами та іншими аспектами робототехніки.

Недоліки

- У порівнянні з потужнішими мовами програмування, RobotC може мати обмежені можливості, особливо для більш складних проектів.
- Як вихідний код на основі C, RobotC вимагає нижчого рівня абстракції, що може зробити його менш доступним для програмістів-початківців.

- Можливості налагодження в RobotC можуть бути менш розвиненими порівняно з більш розвиненими середовищами розробки.

2.1.9 BricxCC

BricxCC – текстовий симулятор, найпоширеніший інструмент, підтримує мову програмування NXC. Це вільно поширювана програма, що має велику кількість різних інструментів для роботи з блоками Lego Mindstorms фактично може повністю замінити стандартне програмне забезпечення Lego (крім драйверів). Вбудовані бібліотеки мови дозволяють працювати з пристроєм на різних рівнях, присутні низькорівневі засоби звернення до входів та виходів пристрою, звернення до фізичних адрес пам'яті, а також високорівневі команди управління моторами та отримання даних із датчиків[19].

Переваги:

- BricxCC підтримує кілька мов програмування, таких як NQC (Not Quite C), NBC (Next Byte Codes) та інші, що дозволяє програмістам вибирати відповідну мову залежно від своїх переваг та досвіду.
- Надає безліч функцій та можливостей для програмування роботів, включаючи керування моторами, сенсорами, обробку подій та інші робототехнічні завдання.
- BricxCC сумісний із різними робототехнічними платформами, що робить його універсальним інструментом для програмування роботів.
- Програма має відкритий вихідний код, що означає, що користувачі можуть змінювати та доповнювати функціональність, якщо це необхідно.

Недоліки:

- Інтерфейс BricxCC може бути менш інтуїтивним для новачків порівняно з іншими середовищами розробки. Це може створювати певні проблеми при першому використанні.
- Обмежена підтримка: Оскільки BricxCC є проектом з відкритим вихідним кодом, він може мати обмежену офіційну підтримку в порівнянні з комерційними середовищами розробки.
- Оскільки BricxCC не так широко використовується, як деякі інші середовища розробки, спільнота користувачів може менш активна, і менше можливостей для отримання допомоги.

2.1.10 Microsoft Robotics

це пакет програм, який можна використовувати для управління різними роботами і включає повноцінний універсальний симулятор. Підтримує мови програмування VPL, C#. Цей симулятор дозволяє працювати з такими віртуальними пристроями: GPS, лазерний далекомір, інфрачервоний далекомір, компас, сенсор кольору, сенсор яскравості, веб-камера. Є умовно безкоштовним продуктом. Компоненти у Robotics Studio представлені як незалежно виконуваних сервісів як сервісів з GUIинтерфейсом. Протокол SOAP, що використовується для взаємодії розподілених сервісів, не призначений для додатків, що працюють у режимі реального часу. Підтримує імпорт таких типів CAD-файлів: *.dae, *.obj і *.x. Крім цього підтримує інтеграцію в різні симулятори через GUI- та COM-інтерфейси та користувальницькі драйвера[20].

Переваги:

- RDS інтегрований з популярним середовищем розробки Microsoft Visual Studio, що забезпечує комфортне середовище для розробників, знайомих із інструментами Microsoft.
- Microsoft RDS підтримує кілька мов програмування, включаючи C# та Visual Programming Language (VPL), що забезпечує гнучкість у виборі мови залежно від переваг розробника.
- Можливість симулювати поведінку роботів у віртуальному середовищі до їхньої фізичної реалізації може прискорити процес розробки та налагодження.
- RDS пропонує абстракції для роботи з різними робототехнічними апаратами, що може спростити створення програмного забезпечення для різноманітних роботів.

Недоліки:

- У зв'язку з тим, що RDS може не оновлюватися так часто, як деякі інші платформи, може виникнути питання підтримки та актуальності.
- У порівнянні з деякими іншими робототехнічними платформами RDS може мати меншу спільноту розробників, що може позначитися на доступності ресурсів та підтримки.
- Використання RDS може вимагати знання інструментів Microsoft, що може бути обмежуючим фактором для тих, хто віддає перевагу іншим середовищам розробки та мови програмування.

Провівши детальний аналіз серед існуючих симуляторів, можна прийти до висновку що серед наведених способів досягнення поставленої задачі, а саме вибору необхідного симулятору можна прийти до того що SolidWorks не підходить через велику складність аналізу та вартість, а його переваги неактуальні для даної роботи, NXT-G не надає остатнього функціоналу для досягнення мети роботи. Robotics System Toolbox не дивлячись на широкий спектр можливостей, більш направлений на комерційні проекти, через

відсутність можливості користуватись ним безкоштовно. У AnyKode аналогічна ситуація, тому серед наведених симуляторів було вирішено користуватися саме Webots, оскільки він відповідає вимогам, та надає необхідний функціонал для досягнення мети роботи.

2.2 Вибір додаткових інструментів для роботи із Webots

Webots надає додатковий фукнкціонал із метою полегшення роботи, було вирішено створити мапу на прикладі вулиці, на якій знаходиться ВНТУ, для досягнення поставленої мети було проведено аналіз досупних рішень існують наступні можливості інтеграції реальних географічних даних у саме середовище для подальшої емуляції.

А для цього потрібно виконати наступний список дій, або знайти готові рішення які допоможуть досягнути поставленої задачі:

- Отримання географічних данних,
- Перетворення цих даних у підтримуваний формат самого середовища, як наприклад VRML, OBJ.
- Імпорт даних у webots.
- Налаштування сцени.
- Інтеграція з роботами.

Серед оглянутих рішень не було знайдено того, що дає можливість досягнути усіх поставлених цілей, але було знайдено рішення, JOSM , яке дає можливість отримати данні.

JOSM[21] - відкритий програмний інструмент для редагування геоданих. Які знеаходяться у OpenStreetMap, був розроблений на Java, спочатку був створений Іммануелем Шольцем але на даний момент підтримуваний Дірком

Штекером. Має безліч можливостей у порівнянні із онлайн-редактором iD, але і більш складний інтерфейс користувача, ніж стандартний онлайн-редактор iD

Основними особливостями JOSM, полягає у моживості імпорту GPX файлів (GPS треків) та робота з аерофотознімками (включаючи такі протоколи як WMS, TMS та WMTS), підтримка кількох картографічних проекцій, шарів, редагування відносин, інструменти перевірки даних, фільтрація даних, автономна робота, пресети та стилі рендерингу.

Багато додаткових можливостей які надаються програмою окремими бібліотеками, які доволі легко встановити, наприклад інструментів, які дають можливість рисування будівель, додавання посилань на Вікіпедію або перегляду даних у 3D, та низка сугубо косметичних рішень,

Рисунок 2.1 – Логотип JOSM

Для перетворення даних потрібно буде скористатись влаштованим у саму программу, імпортером, основна проблема його використання полягає у відсутності нормального інтерфейсу використання, та відстуність необхідних

для роботи бібліотек. У результаті чого для користання імпортером потрібні будуть знання роботи із консоллю, та уміння встановлювати бібліотеки,

Імпортер.

Для роботи з імпортером потрібно встановити декілька бібліотек, а саме

Lxml[22] - це багатофункціональна та проста у використанні бібліотека для обробки мовою Python, яка необхідна для форматування.

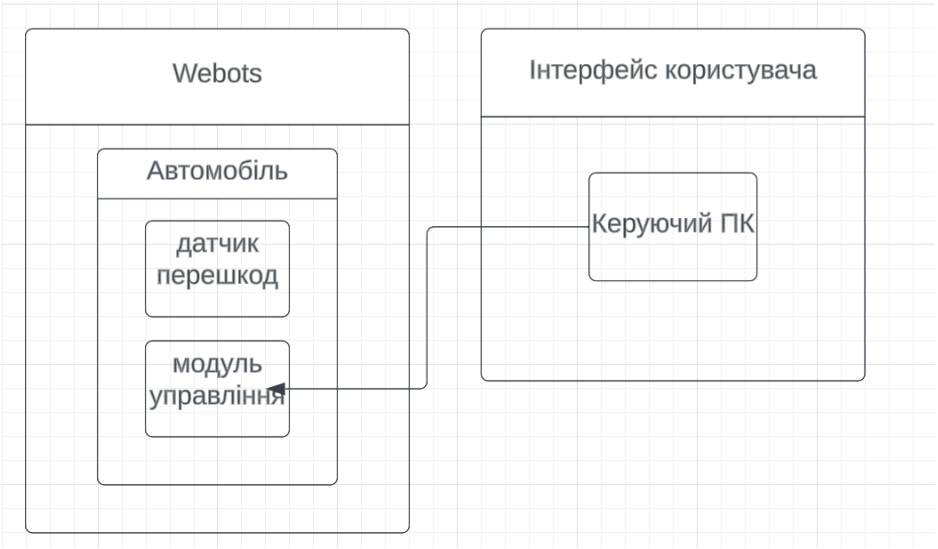
Pyproj[23] - Інтерфейс Python для PROJ (картографічні проекції та бібліотека перетворень координат)

Webcolors[24] - це модуль для роботи з визначеннями кольорів HTML/CSS

Configparser[25] - Цей модуль реалізовує клас ConfigParser, який реалізує базову конфігурацію, яка забезпечує структуру,схожу до тієї, що міститься у файлах Microsoft Windows INI.

Shapely[26] — це пакет Python з ліцензією BSD для маніпулювання та аналізу плоских геометричних об’єктів. Він використовує широко розгорнуту бібліотеку геометрії з відкритим кодом GEOS[27] (двигун PostGIS і порт JTS). Shapely обгортає геометрії та операції GEOS, щоб забезпечити як багатофункціональний інтерфейс Geometry для сингулярних (скалярних) геометрій, так і високопродуктивний NumPy ufuncs для операцій із використанням масивів геометрій. Shapely не зосереджений на форматах серіалізації даних або системах координат, але його можна легко інтегрувати з такими пакетами.

Також слід прийняти до уваги фізичну організацію програмного забезпечення на апаратному обладнанні, яка має виглядати наступним чином



Зображення 2.2.2 – Діаграма розгортки

2.3 Висновок до другого розділу

У даному розділові було проаналізовано різні емулятори, детально розглянуо переваги та недоліки кожного із них, та виходячи з цього було обрано емулятор Webots який відповідає вимогам, та надає весь необхідний функціонал для осягнення поставленої даною роботою задачі

3 МОДЕЛЮВАННЯ РУХУ СЕРЕДОВИЩІ WEBOTS

3.1 Встановлення необхідного програмного забезпечення

Для початку нам необхідне саме середовище для цього необхідно перейти на офіційний сайт, та встановити його, [28](див, Рисунок 3.1.1)

Рисунок 3.1 – Інтерфейс встановлення додатку з офіційного сайту відповідальний за встановлення самого додатку

При встановленні додатку необхідно вказати шлях для того щоб розуміти де буде знаходитись програма.(див. Рисунок 3.2)

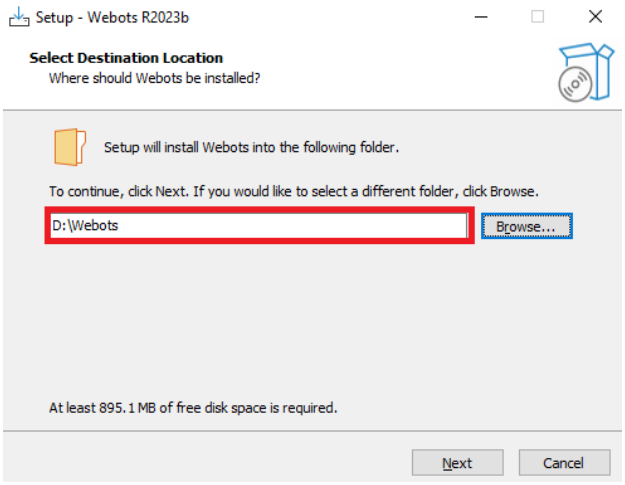


Рисунок 3.2 – Меню вказання шляху при встановленні додатку

Також для генерації середовища у самій webots, було прийнято рішення скористатись JOSM, оскільки webots має можливість конвертації карт з використанням стороннього програмного забезпечення, який також необхідно встановити та налаштувати,

JOSM –Java OpenStreetMap Editor, це програмне забезпечення вільного розповсюдження для редагування географічних даних OpenStreetMap розроблене за допомогою Java. Не має вимог що до під'єднання до Інтернету та під час редагування, для встановлення необхідно перейти на офіційний сайт [29] та почати завантаження необхідного файлу(див. Рисунок 3.3).

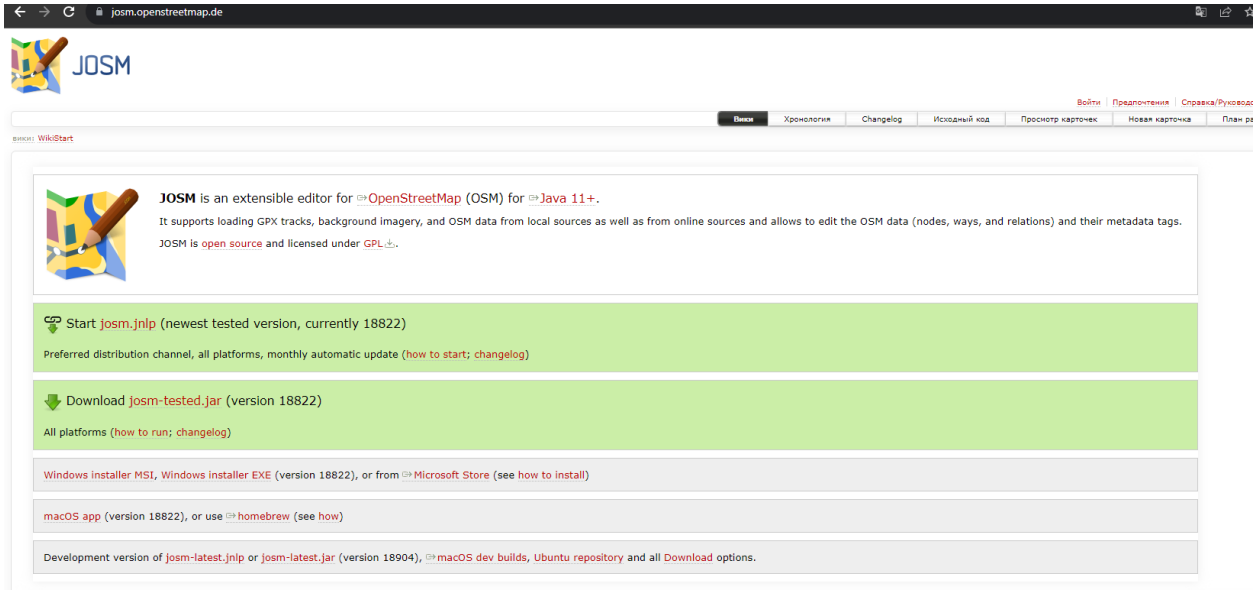


Рисунок 3.3 – Веб сторінка на якій можна встановити JOSM

Після встановлення програми її необхідно налаштувати для більш зручного користування, а саме перейти в вкладку налаштування(див. Рисунок 3.4), та увімкнути експертний режим для того щоб мати можливість більш широкого спектру налаштування самої програми,

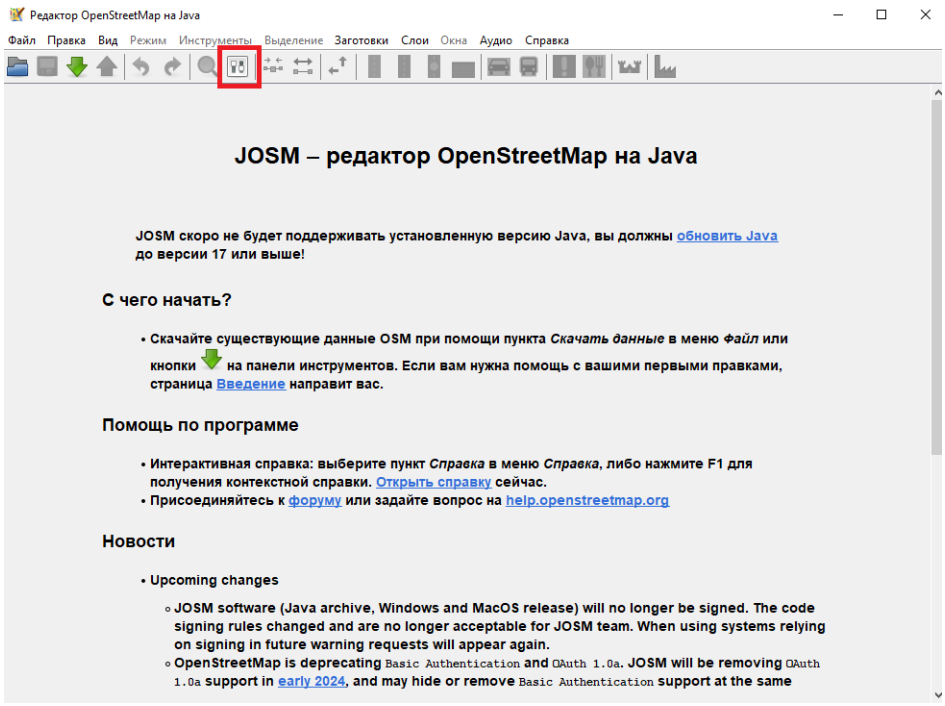


Рисунок 3.4 – Вкладка налатштування

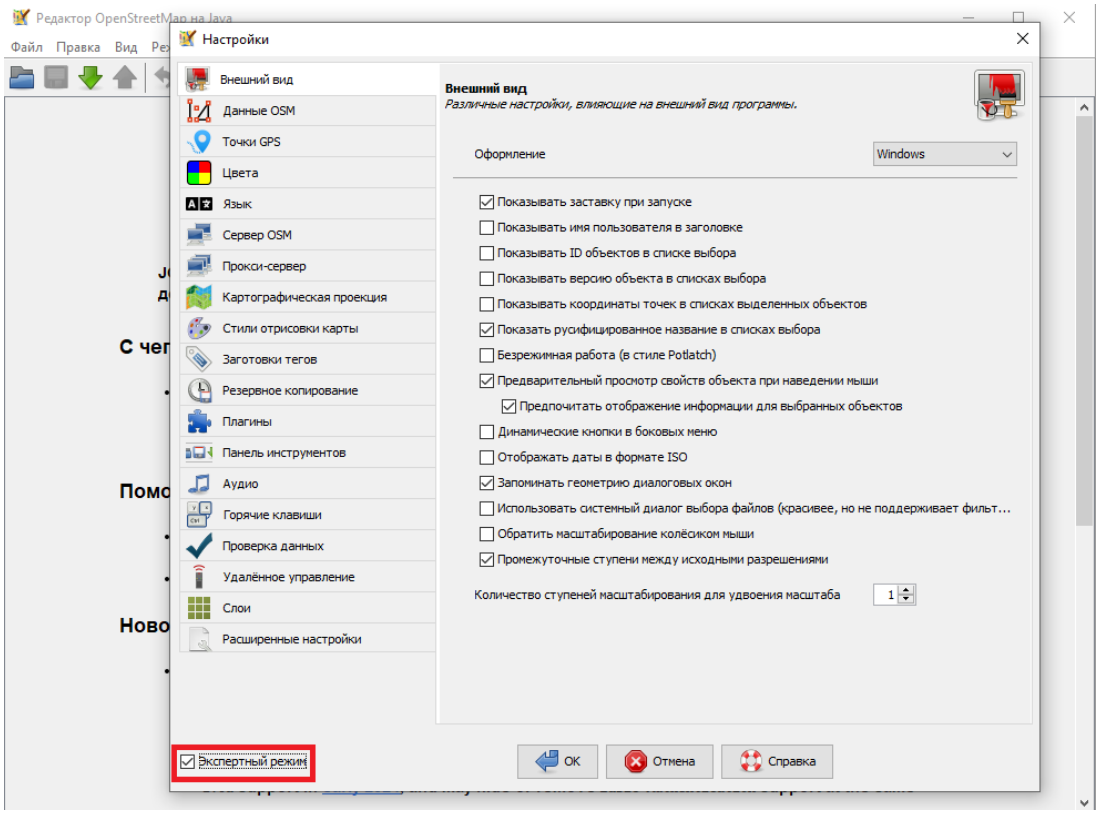


Рисунок 3.5 – Експертний режим, необхідний для більш тонкого налаштування

При увімнкненому експертному режимові режимі, відкривається вкладка даних OSM, у ній необхідно увімкнути режим, рисувати лише контур області(див. Рисунок 3.5), також необхідно перейти в меню, плагіни(див. рисунок 3.6), та встановити наступні плагіни, building_tools, imagery_offset_db, reltoolbox, turnretstricsions, після чого необхідно налаштувати регіональні шари знімків,

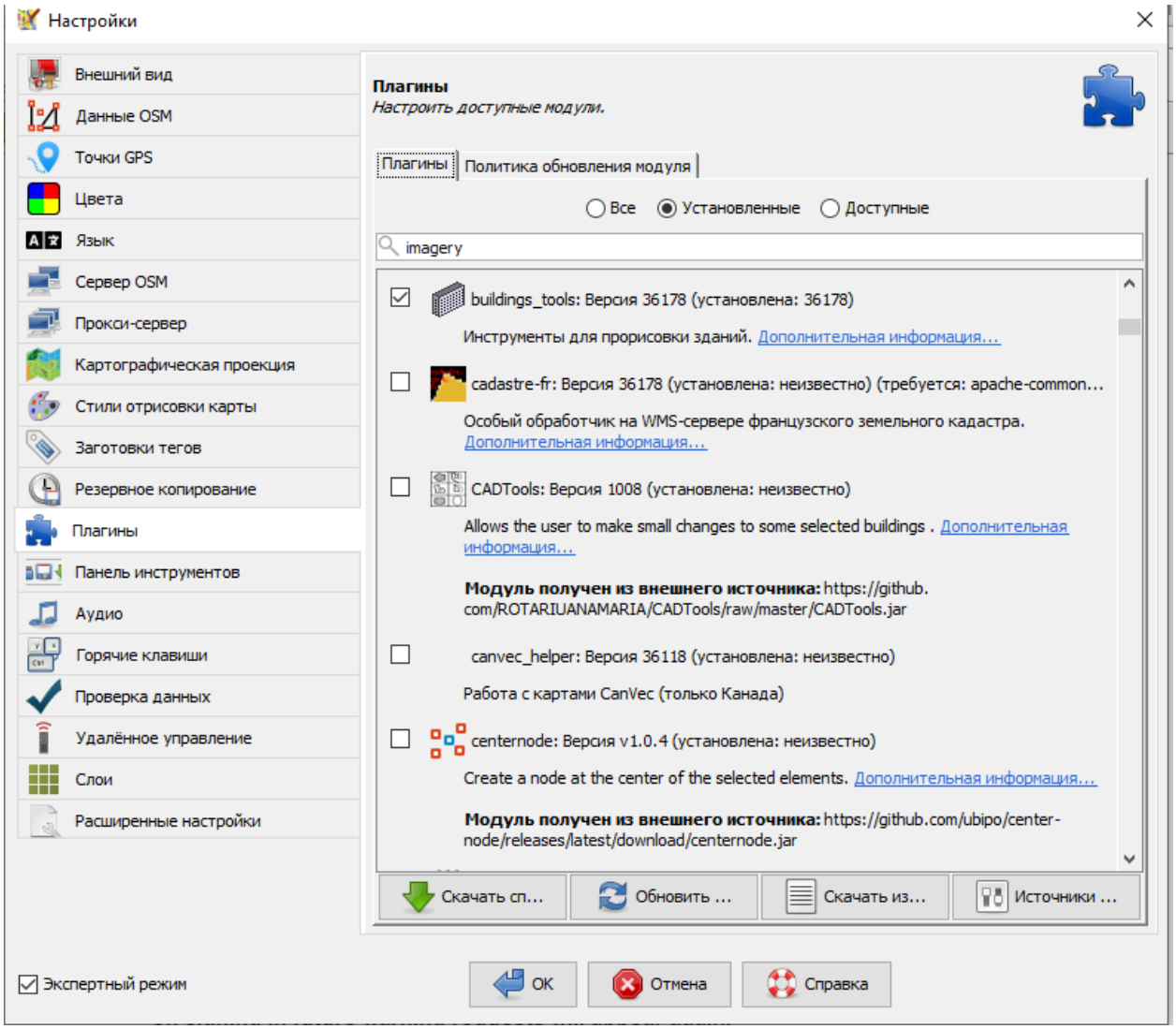


Рисунок 3.6 – Вкладка плагінів

Після чого необхідно перейти у вкладку шарів, та обрати знімки Вінниці,(див. Рисунок 3.7)

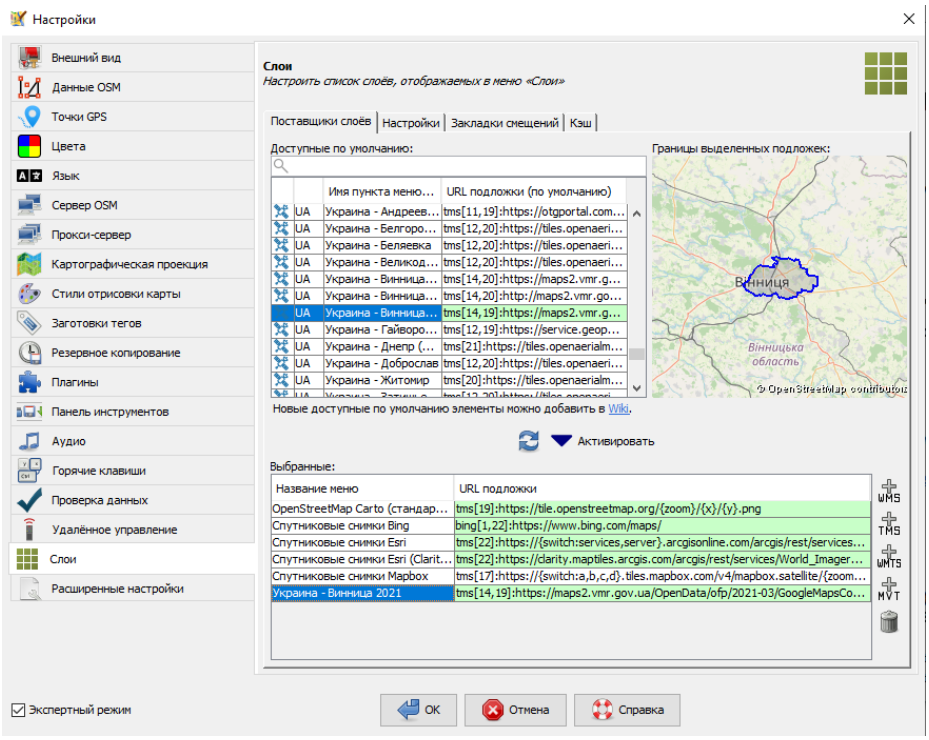


Рисунок 3.7 – Вкладка шарів, де встановлені знімки Вінниці

Після того натискаємо ок, і переходимо у меню встановити картографічні дані, та знаходимо Україну, вінницю, ВНТУ, та встановлюємо дані.(див Рисунок 3.1.8)

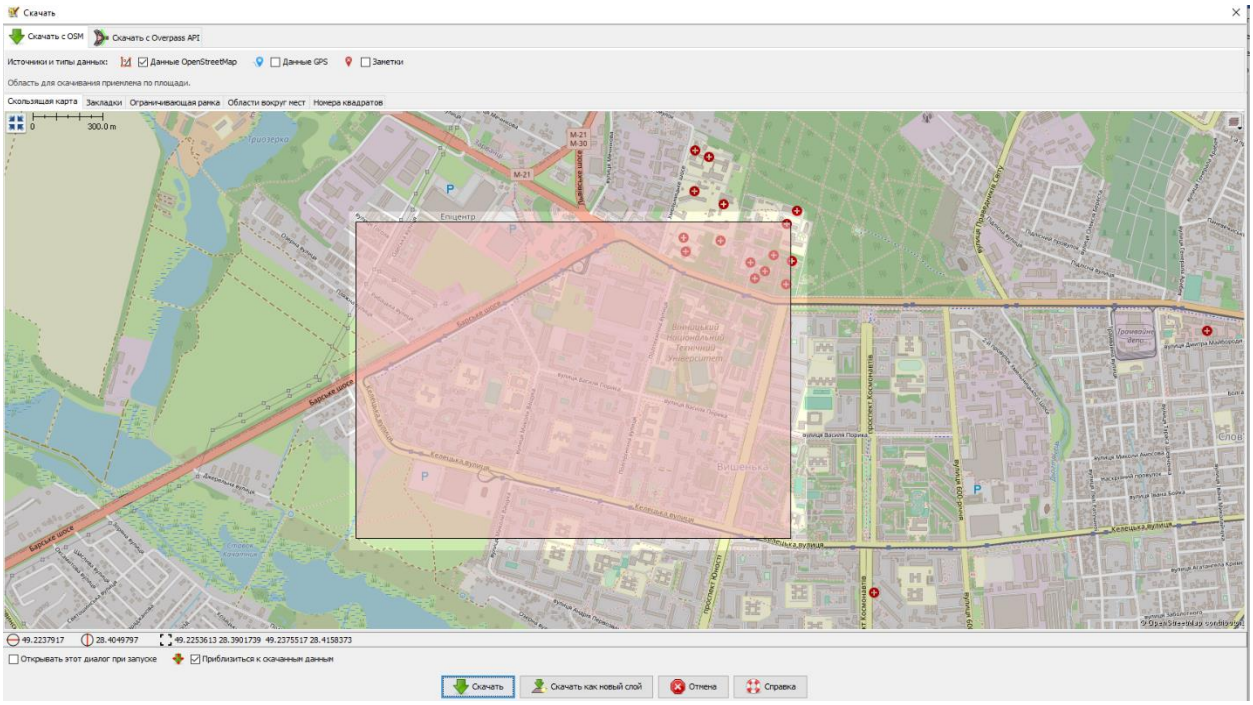


Рисунок 3.8 – Встановлення карти

Також знадобиться Python, перейдемо на офіційну сторінку Python[30]

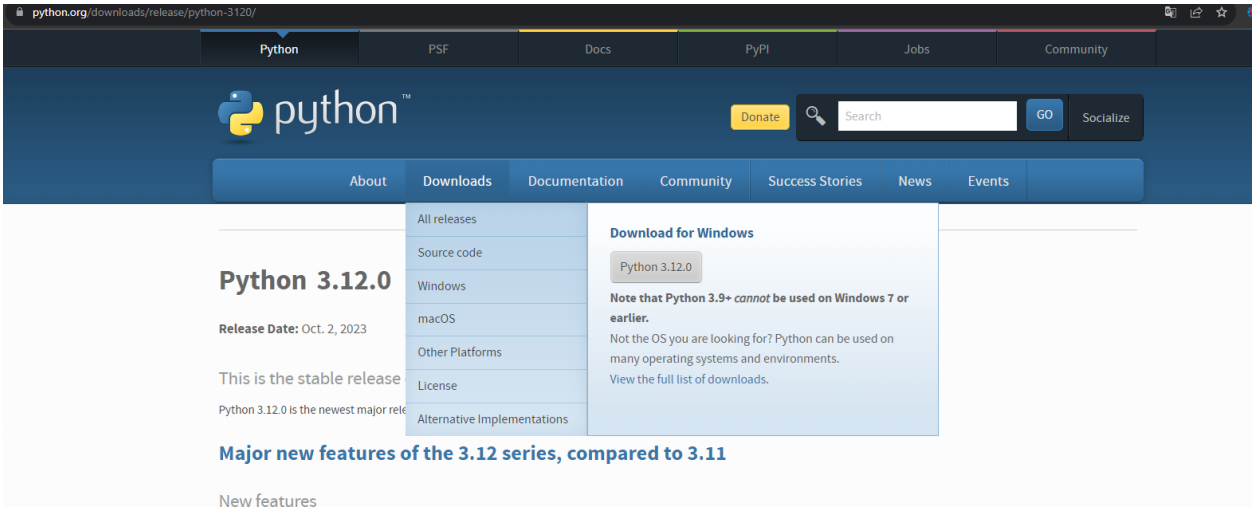


Рисунок 3.9 – Офіційний сайт python

Після встановлення python необхідно встановити декілька додаткових бібліотек, а саме

%PYTHON_PATH%\Scripts\pip.exe install lxml

%PYTHON_PATH%\Scripts\pip.exe install %HOME%\Downloads\pyproj-<<version>>-cp<<python_version>>-cp<<python_version>>m-win_amd64.whl

%PYTHON_PATH%\Scripts\pip.exe install webcolors

%PYTHON_PATH%\Scripts\pip.exe install configparser

%PYTHON_PATH%\Scripts\pip.exe install %HOME%\Downloads\Shapely-<<version>>-cp<<python_version>>-cp<<python_version>>m-win_amd64.whl

При тому що pyproj і Shapely, я просто встановив через pip, з downloads, тобто скачав їх до цього, з сайту[31]

3.2 Робота з самим середовищем

Розглянемо інтерфейс программи.

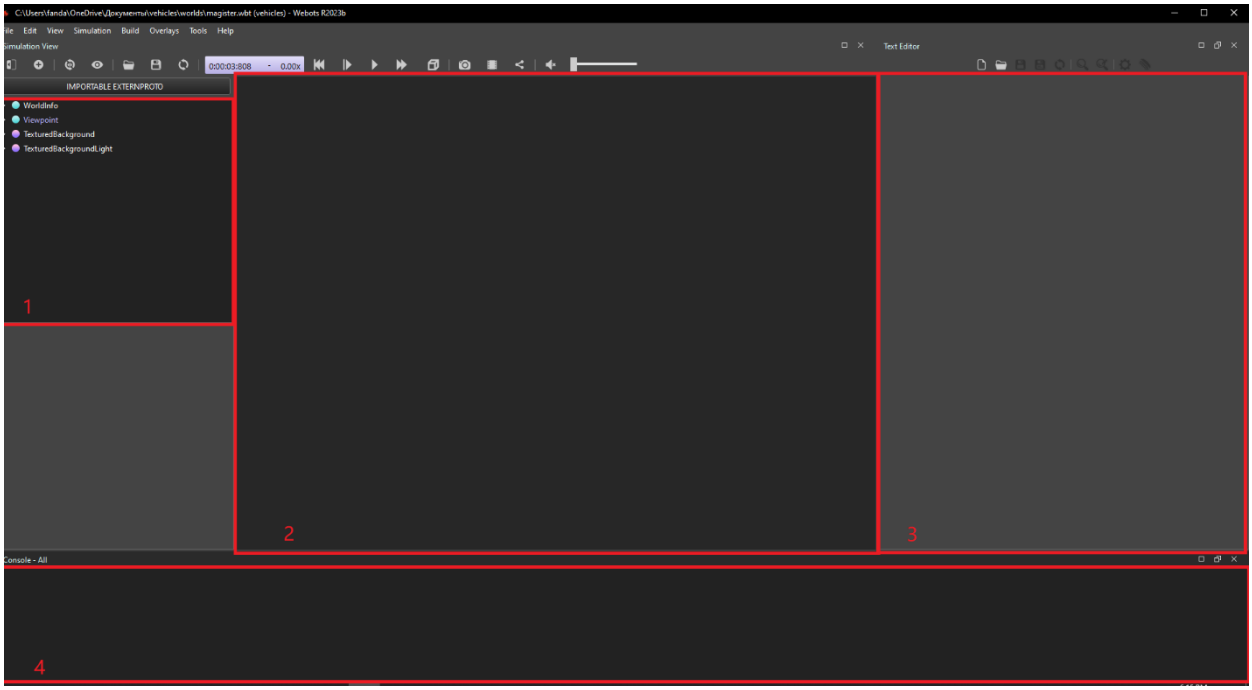


Рисунок 3.1 – Інтерфейс середовища webots

Під номером 1 ми можемо спостерігати дерево звязків, у Toolsменю або натиснути Show Scene Treeкнопку на головній панелі інструментів. Дерево сцени містить інформацію, що описує світ, що моделюється, включаючи роботів і навколишнє середовище, а також його графічне представлення. Дерево сцен Webots структуровано як файл VRML97. Він складається зі списку вузлів, кожен із яких містить поля. Поля можуть містити значення (текстові рядки, числові значення) чи інші вузли.(див. Рисунок 3.2)

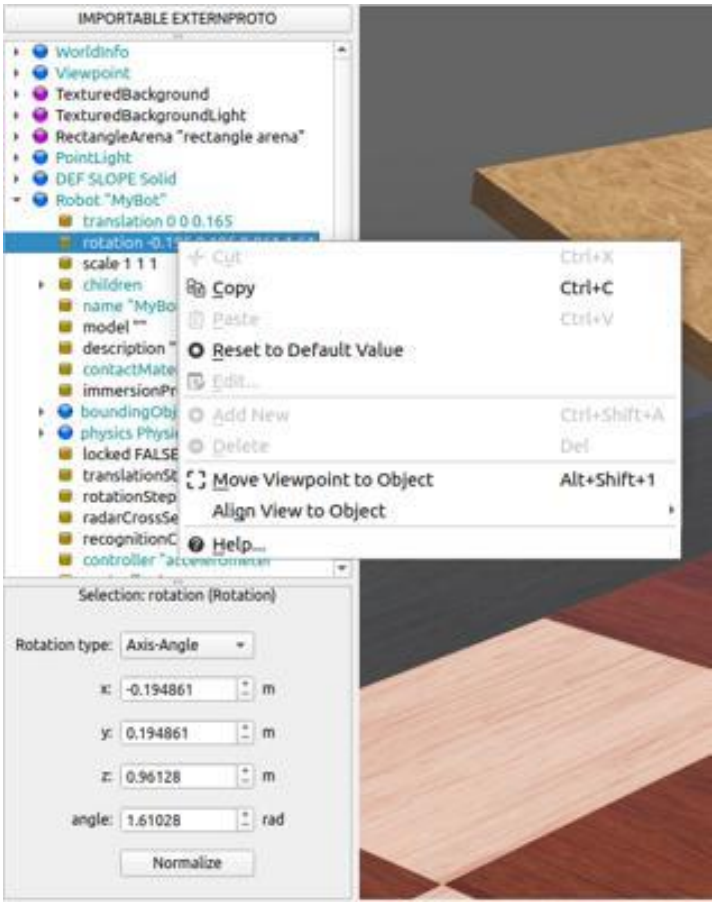


Рисунок 3.2 – Інше Рисунок дерева звязків

Вузли можна розгорнути подвійним клацанням миші. Коли вибрано поле, його значення можна редагувати внизу дерева сцен. Подвійне клацання або натискання enter клавіші на полі вибирає перший редагований елемент панелі редактора полів. Фокус клавіатури можна повернути в дерево сцени, переглядаючи всі елементи на панелі редактора полів. Для текстових полів зміни використовуються натисканням клавіші enter. Те саме стосується і числових полів, але клавіші зі стрілками вгору і вниз також можна використовувати для коригування значень вгору і вниз з негайним застосуванням змін. Для прапорців значення змінюються за допомогою Space смуги. Застосовані зміни негайно відображаються у вікні 3D. У розділі редактора полів доступні такі кнопки: [32]

Під номером 2 ми можемо спостерігати графічне Рисунок сцени. Для перетягування миші під час натискання кнопки миші переміщує камеру

тривимірного вікна. Для обертання камери: у 3D-вікні клацніть на об’єкті лівою кнопкою миші та перетягніть мишу, щоб повернути точку огляду навколо нього. Якщо натиснути на фон, камера обертатиметься навколо свого положення. Горизонтальне перетягування миші обертатиме камеру навколо осі світу вгору. Вертикальне перетягування миші обертатиме камеру навколо горизонтальної осі.(див Рисунок 3.3)[33]

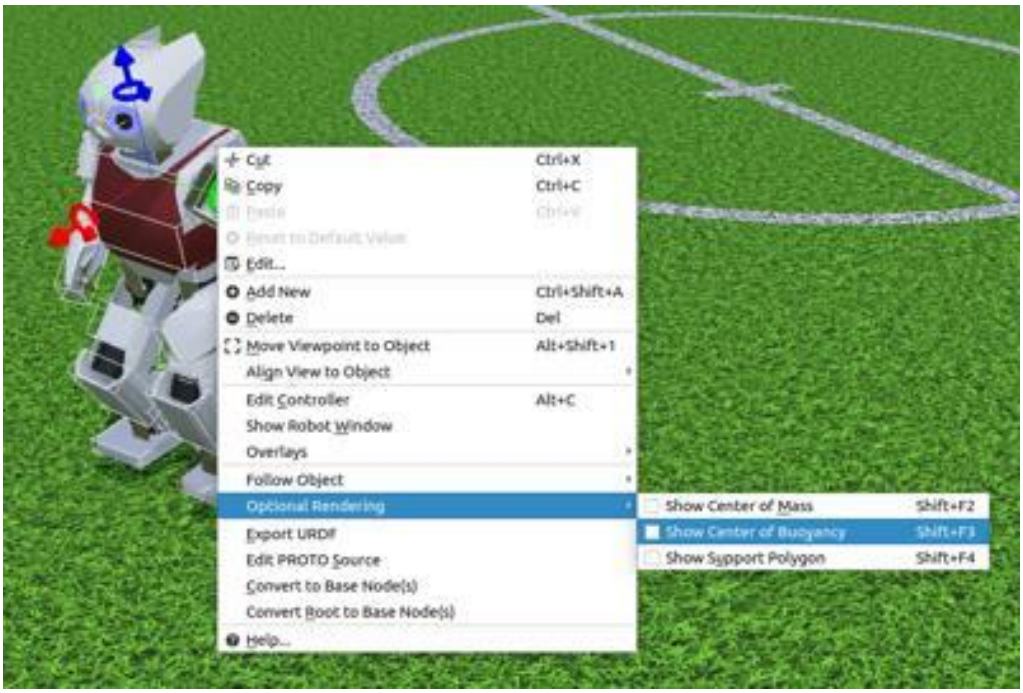


Рисунок 3.3 – Інше Рисунок сцени

Переклад камери: у 3D-вікні натисніть праву кнопку та перетягніть мишу, щоб перекласти камеру.

Масштабування/обертання камери: у 3D-вікні одночасно натисніть ліву та праву кнопки миші (або лише середню кнопку) і перетягніть мишу вертикально, щоб збільшити та зменшити масштаб. Горизонтальне перетягування миші обертатиме камеру навколо осі огляду. Крім того, для масштабування можна використовувати лише колесо миші.

Під номером 3 ми можемо відкривати файли, та редагувати їх, це необхідно для встановлення поведінки та налаштування взаємозвязку між модулями щоб отримати доступ до вікна дерева сцен, ви можете вибрати

«Дерево сцен» у меню «Інструменти» або натиснути кнопку «Показати дерево сцен» на головній панелі інструментів. Дерево сцени містить інформацію, яка описує змодельований світ, включаючи роботів і середовище, а також його графічне представлення. Дерево сцен Webots структуроване як файл VRML97. Який складається зі списку вузлів, кожен з яких містить поля. Поля можуть містити значення (текстові рядки, числові значення) або інші вузли.(див Рисунок 3.2.4)[34]

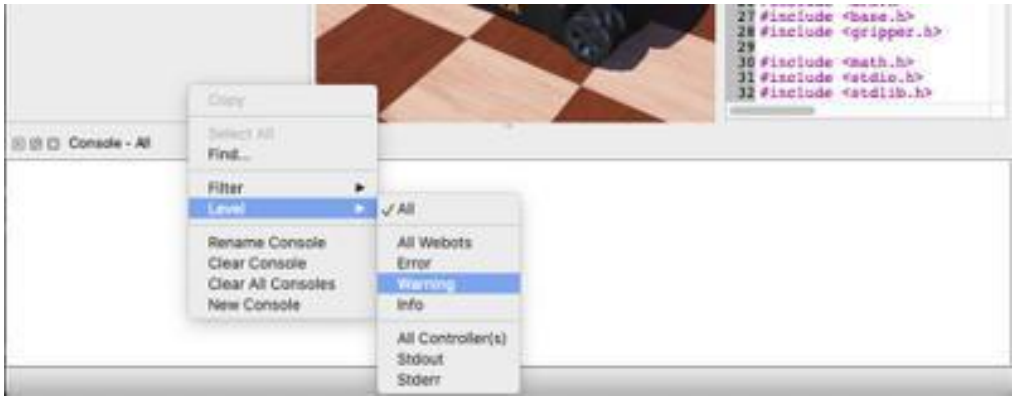


Рисунок 3.4 – Інше Рисунок консолі.

Під номером 4 зображено debug console Консоль Webots яка відображає журнали, що надходять від Webots (такі як попередження аналізу або помилки під час відкриття світу, результати компіляції, попередження ODE тощо) і виходи контролерів. Щоб гарантувати, що журнал, надрукований у консолі Webots, є детермінованим, вихідні дані, що надходять від контролерів, не друкуються відразу після отримання, а групуються за контролерами та друкуються в кінці етапу моделювання. Це означає, що якщо ваша симуляція містить робота A і робота B, вихідні дані робота A будуть надруковані на консолі перед виводом робота B незалежно, якщо деякі вихідні повідомлення робота B були отримані раніше, ніж повідомлення робота A.

Меню «Файл» дозволяє виконувати звичайні операції з файлами: завантаження, збереження тощо.

Підменю «new» дозволяє створювати нові файли моделювання:

Пункт меню «New Project Directory…» спочатку пропонує вам вибрати розташування файлової системи, а потім створює каталог проекту. Каталог проекту містить кілька підкаталогів, які використовуються для зберігання файлів, пов'язаних із певним проектом Webots, тобто файлів світу, файлів контролера, файлів даних, плагінів тощо. Webots запам'ятовує поточний каталог проекту та автоматично відкриває та зберігає будь-який тип файлу з відповідного підкаталогу поточного каталогу проекту.

Пункт меню New World File… дозволяє створити простий файл світу в поточному проекті симуляції, який може містити деякі додаткові компоненти, зокрема прямокутну арену.

Пункт меню New Robot Controller… дозволяє створити нову програму контролера робота. Спочатку вам буде запропоновано вибрати контролер C, C++, Java, Python або MATLABTM. Якщо ви виберете C або C++ у Windows, Webots запропонує вам створити проект Makefile / gcc або проект Visual Studio. Потім Webots попросить вас ввести ім'я вашого контролера, і, нарешті, він створить усі необхідні файли (включно з файлом вихідного коду шаблону) у вашому поточному каталозі проекту.

Пункт меню New Physics Plugin… дозволить вам створити новий фізичний модуль для вашого проекту. Webots просить вас вибрати мову програмування (C або C++) і назву для нового фізичного плагіна. Потім він створює каталог, файл вихідного коду шаблону та Makefile у вашому поточному проекті.[35]

Пункт меню New PROTO… дозволить вам створити новий PROTO для вашого проекту. Webots просить вас визначити ім'я для вузла PROTO, теги, які мають бути включені (якщо такі є), і базовий вузол, від якого успадковуватиме сам PROTO. На основі вибору базового вузла Webots запитає вас, яке з його полів має бути відкритим (тобто видимим у дереві сцени), і відповідно створить необхідні параметри. Потім Webots запропонує вам відкрити файл PROTO в текстовому редакторі, щоб ви могли продовжити його

редагування. Нарешті, ви зможете вставити екземпляр вашого нового PROTO в дерево сцени, як ви зробили б для будь-якого іншого PROTO.

Для початку відформатуємо данні з JOSM, для цього скористаємось вбудованим імпортером (див. Рисунок) 3.2.5

```
:\magus>Python D:\Webots\resources\osm_importer\importer.py  --input=VNTU_MAP.osm --output=vntu
ap.wbt
arning: Failed to determine the country.
* OSM filed parsed
```

Рисунок 3.5 -У консолі віндовс конвертуємо карту з формату osm до формату wbt,

Запустивши згенерований файл ми маємо сцену на якій зображен ВНТУ, (див зобр 3.6)

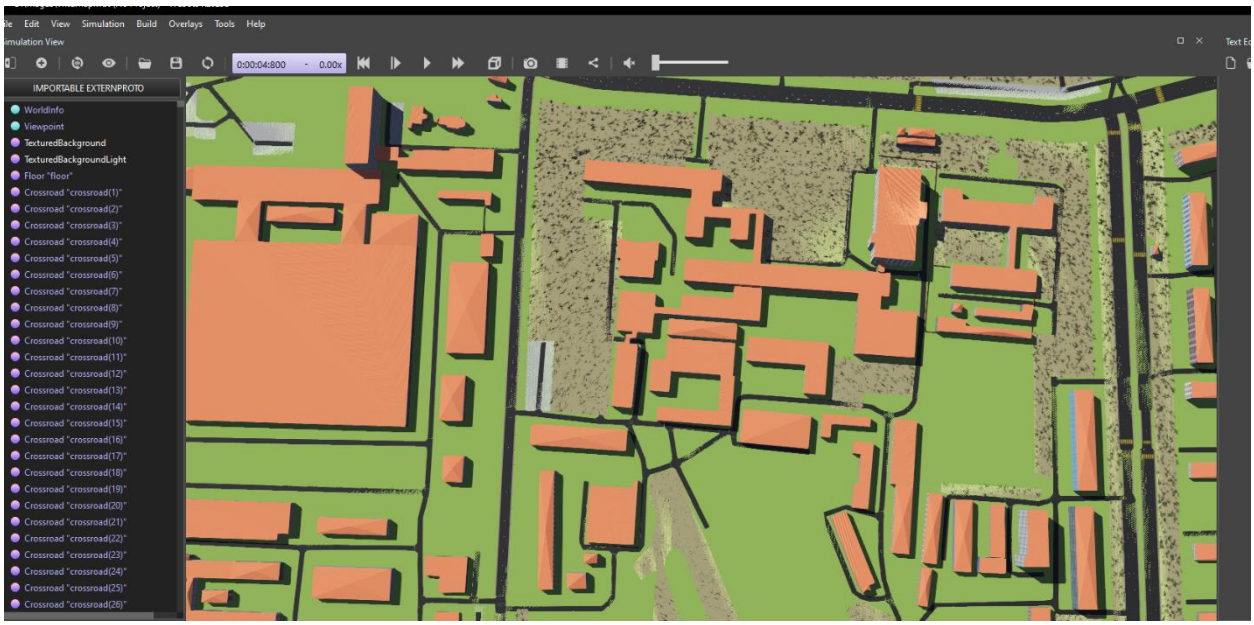


Рисунок 3.6 – Сцена на якій зображений ВНТУ

Після чого нам потрібно додати автомобіль, для цього потрібно додати новий вузол, та серед готових вузлів обрати довільну машину(див зораження 3.7) та прикріпити до нього 2 датчика руху, на даний момент по замувчуванню встановлюються декілька розділів вузлів, в який входять конкретні приклади машин, але якщо ваша симуляція потребує інших приладів, яких у цьому пакетові немає, ви можете встановити його з офіційного сайту [36],

Рисунок 3.7 – Приклад створеного вузла машини

До цього вузла автоматично прикріплений файл відповідальний за функціонування в рамках емуляції, тобто автомобіль буде слідувати по дорозі, орієнуючись на данні самої сцени, але ми можемо додати 2 датчика відстані та модифікувати його, щоб він автоматично зупинявся або повертав вправо, або у ліво, у випадках якщо перед ним на певній відстані з’являється перешкода, для цього необхідно перейти на сайт та встановити один із датчиків який задовольняє зазначені потреби.(див Рисунок 3.8)

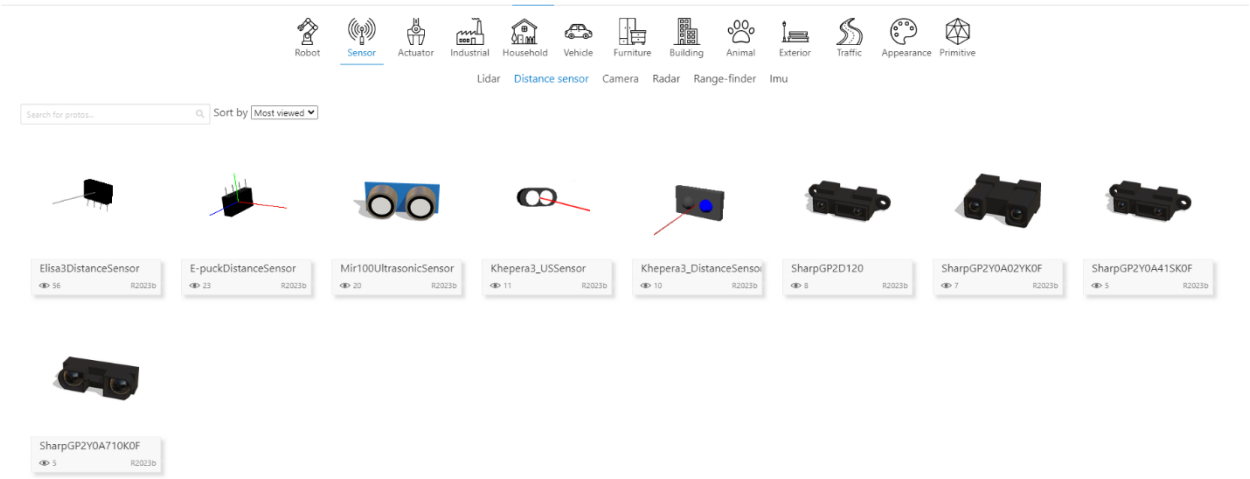


Рисунок 3.8 – Датчики відстані, які можна встановити для подальшого використання у weebots

Для того щоб додати встановлений датчик, необхідно при додаванні нового вузла, у існуючому обрати вказати його фізичне розташування на вашому пк. Та помістити його у sensors front, sensors rear, хоча деякі сенсори, уже присутні у програмі.

Та модифікувати контроллер керування автомобіля, а саме обрати датчики

front_left_sensor = robot.getDistanceSensor(“front_ sensor”)

front_right_sensor = robot.getDistanceSensor(“rear _sensor”)

та на основі данних сенсорів змінити швидкість автомобіля.

While robot.step(64) != -1:

Зчитуємо значения сенсоров расстояния спереди

front_left_distance = front_distance_sensors[0].getValue()

front_right_distance = front_distance_sensors[1].getValue()

```
    Зчитаем значения сенсоров відстані позаду
    rear_left_distance = rear_distance_sensors[0].getValue()
    rear_right_distance = rear_distance_sensors[1].getValue()

Задаємо кут повороту в залежності від відстані до перешкоди
    if front_left_distance < 1.0 or front_right_distance < 1.0:
        # Если близко к препятствию, поворачиваем налево
        wheels[0].setVelocity(-max_speed)
        wheels[1].setVelocity(max_speed)
        wheels[2].setVelocity(max_speed)
        wheels[3].setVelocity(-max_speed)
    else:
        # В противном случае, двигаемся вперед
        for wheel in wheels:
            wheel.setVelocity(max_speed)
```

у готової моделі автомобіля, уже готовий контроллер, але якщо є необхідність додати новий контроллер то це можна зробити якщо перейти у файл, новий, новий контроллер робота, у якому можна обрати активну мову програмування.

API контролера – це програмний інтерфейс, який дає вам доступ до змодельованих датчиків та виконавчих механізмів робота. Наприклад, включення webots/distance_sensor.h файлу дозволяє використовувати

wb_distance_sensor_*функції, і за допомогою цих функцій можна зчитувати значення вузлів DistanceSensor

Маємо программу з наступним життєвим циклом(див зображення 3.2.6)

Рисунок 3.9 – Діаграма життєвого циклу программи

для тестування додам обьєкт, який буде виконувати роль перешкоди, та запустимо емуляцію.(див. Рисунок 3.10. 3.11). З резутатів емуляції, ми можемо дійти до висновку про те, що автомобіль, обїжджає парешкоди,

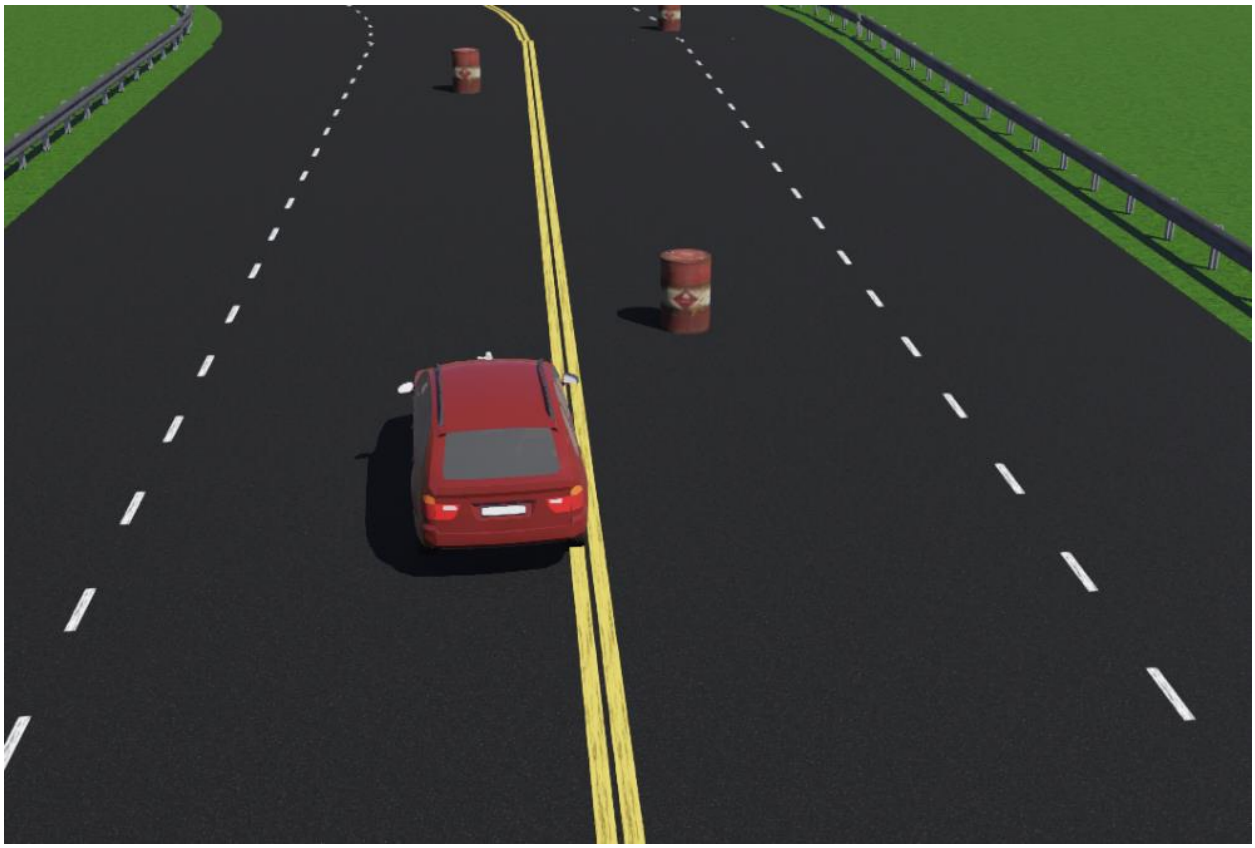


Рисунок 3.10 – Автомобіль об’їжджає першу перешкоду

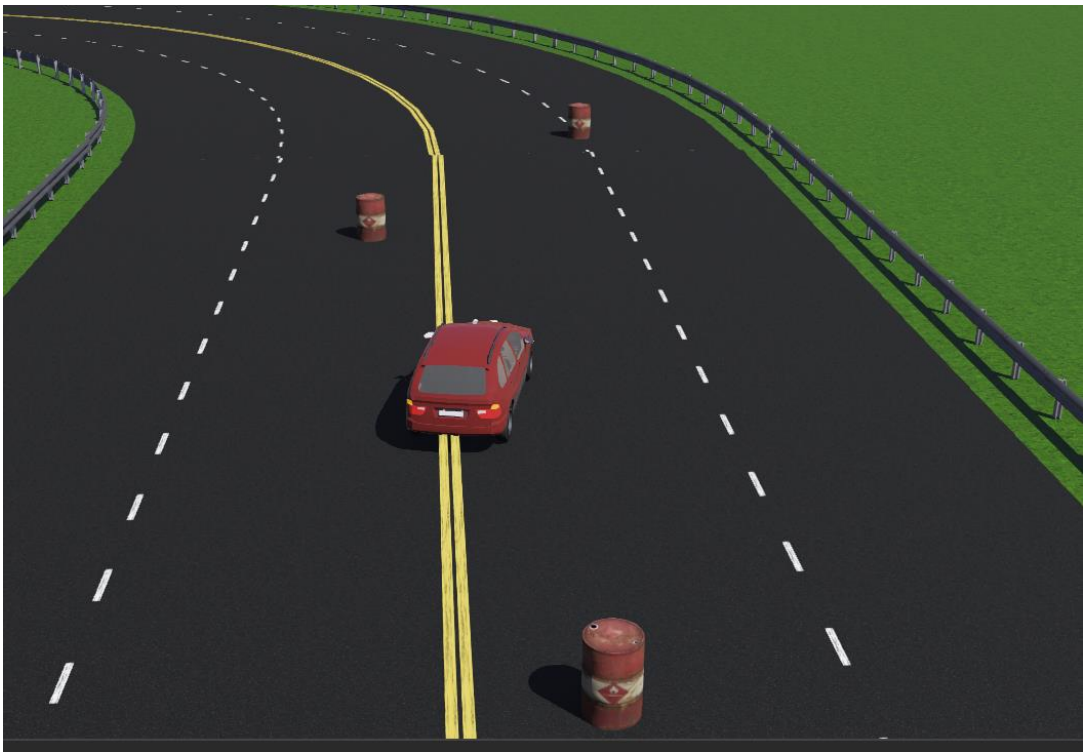


Рисунок 3.11 – Автомобіль об’їжджає другу перешкоду

Схема використання виглядає наступним чином

Рисунок 3.12 – Діаграма використання.

3. 3 Висновок до третього розділу

У даному розділові було встановлено JOSM для можливості зручного редагування, та створення емуляцій на прикладі реальної інфраструктури, яка береться з вільних супутникових даних, та налаштовано імпортер, який і дає можливість перетворити формат osm, в wbt, також було розглянуто інтерфейс середовища емуляції webots, за допомогою якого було виконано додавання інших об’єктів, та редагування їх характеристик, також було модифіковано алгоритм, а саме було додано можливість орієнтуванні за допомогою лідарів. для того щоб надати машині властивості об’їжджати перешкоди на основі даних які отримують 2 датчика, позаду та спереду машини.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Оцінювання комерційного потенціалу розробки

У сфері безпілотних автомобілів, де технологічні революції нерозривно пов'язані із суспільними потребами та вимогами ринку, розуміння комерційного потенціалу є невід'ємною частиною успішного впровадження.[36]

Проведемо оцінювання комерційного потенціалу автопілотів для автомобілів. Це передбачає оцінку технічного рівня даної розробки та визначення потенційних можливостей її комерційного використання. Для проведення технологічного аудиту скористаємося експертним методом. Для цього запросимо 3-х експертів к.т.н. Маслій Р. В., старший викладач Кулик Я. А, к.т.н. доцент Папінов В. М.

Експерти є фахівцями у даній галузі науки і техніки, мають значний досвід роботи, тривалий час займаються вивченням даної проблеми, мають вагомі наукові здобутки у даній галузі тощо.

Оцінювання комерційного потенціалу розробки будемо здійснювати за дванадцятьма критеріями, рекомендованими Державним комітетом України з питань науки, інновацій та інформатики (2010 р.), які наведені в таблиці 4.1.

Для визначення загального рівню комерційного потенціалу розробки, скористаємося даними таблиці 4.3, в якій наведено рекомендації щодо можливих рівнів комерційного потенціалу будь-якої розробки.

Таблиця 4.1 – Критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно найматифахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та звільнення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні.	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування.
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термінокупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продуктувимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Продовження таблиці 4.1

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Декілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогівна великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче заціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Результати оцінювання комерційного потенціалу розробки зведені в таблицю 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Маслій Р. В. к.т.н.	Кулик Я. А. ст. викладач	Папінов В. М. к.т.н., доц.
	Бали, виставлені експертами:		
1	3	4	3
2	3	4	3
3	2	3	3
4	4	3	5
5	1	3	3
6	2	3	2
7	1	3	1
8	2	2	0
9	1	2	3
10	3	4	4
11	4	3	4
12	3	4	4
Сума балів	$СБ_1 = 29$	$СБ_2 = 38$	$СБ_3 = 34$
Середньоарифметична сума балів СБ	$\sum^3 СБ_1 \quad 29 + 38 + 34$ $(29+38+34)/3 = 33.6(6)$		

Для визначення комерційного потенціалу розробки за даними таблиці 4.2 потрібно скористатися таблицею 4.3, у якій представлені рівні комерційного потенціалу.

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівні комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

За результатами оцінювання комерційного потенціалу рівень розробки є «вище середнього». Оскільки за балами експертів середньоквадратична сума балівдорівнює 33. Це означає що технологічний рівень нашої розробки можна пояснити наступним чином, розроблена система має досить великі функціональні можливості, забезпечує високі показники ефективності.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Прогнозування витрат на виконання науково – дослідної чи дослідно конструкторської роботи НДДМР роботи може складатися з таких етапів:

1) розрахунок витрат, які безпосередньо стосуються виконавців даного розділу (частини) НДДМР;

2) розрахунок загальних витрат на виконання даної НДДМР;

3) прогнозування загальних витрат на виконання та впровадження результатів НДДКР.

Перший етап:

Основна заробітна плата кожного із розробників[37] (дослідників) $З_0$, якщо вони працюють в наукових установах бюджетної сфери:

$$З_0 = \frac{M}{T} * xt \text{ грн} \quad (4.1)$$

де *M* – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн. Величини окладів (разом з встановленими доплатами і надбавками) знаходяться в межах (8000 - 15 000) грн. за місяць;

T_p – число робочих днів в місяці; приймемо *Tp* = 21 день;

t – число робочих днів роботи розробника (дослідника).

Оскільки на виконання даної роботи керівнику дається 46 годин, а робоча зміна триває 8 годин, то відповідно керівнику розробки потрібно використати 5.75 днів. Експерту по webots дається 34 годин, а це 4.25 дні роботи, експертові по автопілотові дається 47 годин, а це 5.8 дні роботи. Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 5.4 – Розрахунок основної заробітної плати

Найменування посади виконавця	Місячний посадовий оклад грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
1.Керівник роботи	13000	619	5.75	3,559
2.Експерт по webots	10000	476	4.25	2,023
3. експерт по автопілотові	9000	428	5.6	2,396.8

Всього $З_о = 7978$

Додаткова заробітна плата[38] для всіх розробників розраховується як (10…12)% від величини основної заробітної плати, тобто:

$$З_{д1} = (0.1) * З_0 = 797{,}8 \text{ грн} \quad (4.2)$$

$$З_{д2} = (0.12) * З_0 = 957 \text{ грн} \quad (4.3)$$

Нарахування на заробітну плату Нзп фахівців, які беруть участь у виконанні наукової роботи, розраховуються за формулою 4.4

$$Н_{зп} = (З_0 + З_д) * \frac{\beta}{100} \quad (4.4)$$

Ставка єдиного внеску на загальнообов'язкове державне соціальне страхування[39] для бюджетних установ визначена в 36.3%. Тоді у нашому випадку

$$Н_{зп1} = (7978 + 797) * \frac{36.3}{100} = 3\ 185$$

$$Н_{зп2} = (7978 + 957) * \frac{36.3}{100} = 3\ 243$$

Амортизація[40] обладнання, комп'ютерів та приміщень *А*, які використовувались під час виконання даної роботи.

У спрощеному вигляді амортизаційні відрахування *А* в цілому можуть бути розраховані за формулою 4.5.

$$А = \frac{Ц * Н_а}{100} * \frac{Т}{12} \text{ грн} \quad (4.5)$$

Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даної роботи, грн.;

$Н_а$– річна норма амортизаційних відрахувань. У нашому випадку можна прийняти, що На = (10...25)%;

Т – термін, використання обладнання, приміщень тощо, місяці.

Зроблені розрахунки зведені до таблиці 4.6

Таблиця 4.6 – Розрахунок амортизації обладнання, приміщень.

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
1.Персональні комп'ютери, принтери,інші пристрої	34 000	20	6	3400
2.Приміщення	15 000	10	6	750
Всього				А = 4 150

Витрати на матеріали М, що були використані під час виконання роботи, розраховуються по кожному виду матеріалів за формулою 4.6.

$$М = \sum_{1}^{n} Н_i * Ц_i * К_i - \sum_{1}^{n} В_i * Ц_в \text{ грн} \quad (4.6)$$

H_i – витрати матеріалу *i*-го найменування, кг;

$Ц_i$ – вартість матеріалу *i*-го найменування, грн./кг.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

$В_i$ – маса відходів матеріалу *i*-го найменування, кг;

$Ц_в$ – ціна відходів матеріалу *i*-го найменування, грн/кг;n – кількість видів матеріалів.

Витрати на комплектуючі *К*, що були використані під час виконання роботи, розраховуються за формулою 4.7.

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн}, \quad (4.7)$$

При виконані роботи були використані основні матеріали на суму 640 грн. Витрати на силову електроенергію[41] Ве розраховуються за формулою 4.8

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ грн}, \quad (4.8)$$

В – вартість 1 кВт/год. електроенергії до 100 використаних кВт/год, в 2023р. В ≈ 2.7 грн./кВт-год.

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи обладнання. нехай, Ф = 180 годин;

$К_п$ – коефіцієнт використання потужності; $К_п < 1 = 0{,}85$.

У виконанні роботи були задіяні силові блоки потужністю 1,0 кВт.

У такому випадку витрати на елетроенергію складатимуть

$$В_е = 2.6 * 1 * 0{,}85 * 180 \approx 397 (грн).$$

Інші витрати $В_{ін}$ можна прийняти як (100…300)% від суми основної заробітної плати розробників, які виконували дану роботу. Інші витрати розраховуються за формулою 4.9.

$$В_{ін} = (1 \ \ldots 3) * \ З_0 = 1{,}5 * 7978 = 11\,967 (грн)$$

Сума всіх попередніх статей витрат дає витрати на виконання даної роботи безпосередньо самим розробником – В (формула 5.9).

Для нашого випадку:

$$В = 7978 + 957 + 3243 + 4150 + 397 + 11967 + 640 = 29332$$

Другий етап передбачає розрахунок загальних витрат виконання даної роботи всіма виконавцями. Загальна вартість даної роботи визначається за Взаг формулою 4.11

$$В_{заг} = \frac{В}{\alpha}, \quad (4.9)$$

a – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях

У нашому випадку а =0.9, тому загальна кількість роботи буде

$$В_{заг} = \frac{29332}{0,9} = 32591 \text{ (грн)}$$

Третій етап передбачає прогнозування загальних витрат на розробку та впровадження результатів виконаної роботи. Прогнозування загальних витрат здійснюється за формулою 4.12

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (4.10)$$

β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Оскільки розробка знаходиться на стадії впровадження, то $\beta \approx 0,8\text{-}0,9$.

Тоді, у нашому випадку.

$$ЗВ = \frac{32591}{0,9} = 36\ 212 \text{ (грн)}.$$

Отже прогнозовані витрати на розробку системи емуляції у webots та розробки автопілоту, дослідження та впровадження результатів даної роботи становлять приблизно 36 212 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Аналіз місткості ринку даної продукції показує, що в даний час в Україні кількість потенційних користувачів подібних систем, складає щорічну аудиторію приблизно 35 тис, тобто, Середня ціна подібних розробок вартує від 15 000 до 32 000 грн. приймемо середню ціну тобто 21 тис грн.

Оскільки надається не лише еможливість емуляції, а й впровадження автопілоту, та можливість працювати з різним набором інструментів і передбачає то це дозволяє реалізовувати розробку приблизно на (35%) дорожче, тобто на 7.35 тис грн, тобто в суммі 28.35 тис грн, але потрібно

Забезпечити збільшення попиту на розроблену систему. Припустимо, що розробка буде користуватися підвищеним попитом на ринку протягом 5-ти років після впровадження. Після цього високою є ймовірність, що іншими фахівцями будуть розроблені ще більш ефективні, в плані якості або ціни рішення.

За нашими розрахунками, результати нашої розробки можуть бути впроваджені з 1 січня 2023 року, а її результати будуть виявлятися протягом 2023- го , 2024-го, 2025-го, 2026-го та 2027-го років

Прогноз зростання попиту на нашу розробку складає по роках:

1-й рік після впровадження (2023 р.) – приблизно 35 шт.;

2-й рік після впровадження (2024 р.) – приблизно 31 шт.;

3-й рік після впровадження (2025 р.) – приблизно 27 шт.;

4-й рік після впровадження (2026 р.) – приблизно 23 шт.;

5-й рік після впровадження (2027 р.) – приблизно 20 шт.

У 2028 р. не планується отримання прибутків для потенційних інвесторів, оскільки високою є ймовірність, що з'являться нові, більш якісні та ефективні розробки

Розрахуємо очікуване збільшення прибутку

ΔПі,[42] що його може отримати потенційний інвестор від впровадження результатів нашої розробки, для кожного із років, починаючи з першого року впровадження:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right), \quad (4.11)$$

$\Delta Цо$ - покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником є збільшення ціни нової розробки, грн.; у нашому випадку ΔЦо = 6.3 тис. грн.; [43]

N – основний кількісний показник, який визначає обсяг діяльності у даному році до впровадження результатів наукової розробки; було встановлено, що

N = 35 шт;

ΔN – покращення основного кількісного показника від впровадження

результатів розробки;

$Цо$ – основний оціночний показник, який визначає обсяг діяльності у даному році після впровадження результатів розробки, грн.; Цо = 25 тис. грн.;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ – коефіцієнт, який враховує сплату податку на додану вартість. У 2023 році ставка податку на додану вартість встановлена на рівні 20%, а коефіцієнт λ = 0.83

p – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати

p = 0,2...0,3; візьмемо p = 0,25

v – ставка податку на прибуток у 2023 році, v = 18%

Тоді, збільшення чистого прибутку для потенційного інвестора ΔП1 протягом першого року від реалізації нашої розробки (2023 р.) складатиме:

$$\Delta П1 = [6 * 35 + 25 * (35 - 7)] * 0{,}83 * 0{,}25 * \left(1 - \frac{18}{100}\right) = (210 + 700) * 0{,}2075 * 0{,}82 = 154.8 \text{ тис грн}$$

Протягом другого року

$$\Delta П1 = [6 * 35 + 25 * (31 - 7)] * 0{,}83 * 0{,}25 * \left(1 - \frac{18}{100}\right) = (210 + 600) * 0{,}2075 * 0{,}82 = 114.1 \text{ тис грн}$$

Протягом третього року

$$\Delta П1 = [6 * 35 + 25 * (27 - 7)] * 0{,}83 * 0{,}25 * \left(1 - \frac{18}{100}\right) = (210 + 500) * 0{,}2075 * 0{,}82 = 137{,}82 \text{ тис грн}$$

Протягом четвертого року

$$\Delta П1 = [6 * 35 + 25 * (23 - 7)] * 0{,}83 * 0{,}25 * \left(1 - \frac{18}{100}\right) = (210 + 400) * 0{,}2075 * 0{,}82 = 103.79 \text{ тис грн}$$

Протягом пятого року

$$\Delta П1 = [6 * 35 + 25 * (20 - 7)] * 0{,}83 * 0{,}25 * \left(1 - \frac{18}{100}\right) = (210 + 325) * 0{,}2075 * 0{,}82 = 91{,}03 \text{ тис грн}$$

Основними показниками, які визначають доцільність фінансування розробки потенційним інвестором, є абсолютний ефект і відносна ефективність [44] вкладених в розробку інвестицій та термін їх окупності.Розрахуємо теперішню вартість інвестицій PV, що вкладаються в розробку. Такою вартістю можна вважати прогнозовану величину загальних витрат *ЗВ* на виконання та впровадження результатів роботи, розраховану раніше за формулою (4.12), з врахуванням додаткових витрат *Кдод* = 1,7, пов'язаних з врахуванням непередбачених обставин. Тобто будемо вважати, що

$$PV = ЗВ * К_{дод} = 36\,212 * 1{,}7 = 61{,}560$$

Розраховуємо абсолютну ефективність вкладених інвестицій Еабс. Для цього користуються формулою 4.14.

$$Е_{абс} = ПП - PV, \quad (4.12)$$

де *ПП* – приведена вартість всіх чистих прибутків від реалізації результатів розробки, грн. Розраховується за формулою 5.16.

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.13)$$

ΔПі – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн.;

m – період часу, протягом якого виявляються результати впровадженої наукової роботи, роки;

r – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, що r = 0,10 (або 10%);

t – період часу (в роках) від моменту отримання прибутків до точки „0”.

Якщо *Еабс* ≤ 0, то результат від впровадження нашої розробки буде збитковим і вкладати кошти в розробку ніхто не буде. Якщо Еабс > 0, то результат від впровадження нашої розробки принесе прибуток і вкладати кошти в дану розробку в можна.

Тоді приведена вартість всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від реалізації результатів нашої розробки, складе

$$ПП = \frac{154800}{(0.1+1)^3} + \frac{114100}{(0.1+1)^4} + \frac{137820}{(0.1+1)^5} + \frac{103079}{(0.1+1)^6} + \frac{91030}{(0.1+1)^7} = 533\,414 \text{ грн}$$

Абсолютний ефект нашої розробки (при прогнозованому ринку збуту) складе:

$$Еабс = 533\ 414 - 36\ 212 = 497\ 202 \text{ грн}$$

Оскільки *Еабс* > 0, то вкладання коштів на виконання та впровадження результатів розробки може бути доцільним. Але це не свідчить про те, що інвестор буде зацікавлений у фінансуванні цього проекту. Він буде зацікавлений у тому випадку, коли ефективність вкладених інвестицій буде перевищувати певний критичний рівень. Для цього розрахуємо відносну ефективність Ев вкладених у розробку коштів. Для цього скористаємося формулою 4.17.

$$E_В = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.14)$$

Тж – життєвий цикл наукової розробки, роки

$$Ев = \sqrt[6]{1 + \frac{497202}{36\ 212}} - 1 = 0{,}18 = 18\%$$

Визначаємо мінімальну дохідність, нижче за яку кошти в розробку проекту вкладатися не будуть. У загальному вигляді мінімальна дохідність або мінімальна ставка дисконтування τ мін визначається за формулою 4.18.

$$\tau = d + f, \tag{4.15}$$

d – середньозважена ставка за депозитними операціями в комерційних банках d = (0,10...0,15);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,2).

$$\tau = 0{,}13 + 0{,}1 = 0{,}23 = 23\%.$$

Оскільки величини $EB < \tau$, то можна припустити що у інвестора не виникне зацікавленість у фінансуванні емуляції моделювання руху автопілота. Він буде не зацікавлений вкладати гроші, оскільки у нього не буде можливості отримати більший прибуток, аніж якщо просто покладе свої гроші на депозит у комерційному банку.

ВИСНОВКИ

В результаті виконання розділу 1 магістерської кваліфікаційної роботи був проведений ретельний аналіз класифікації автомобілів залежно від ступеня автоматизації. Увага була приділена ключовим методам, які застосовуються при автоматизації процесів прокладання маршрутів та локалізації. В результаті цього аналізу виділено чотири перспективні методи, кожен з яких є потенційним рішенням для певних сценаріїв застосування. Після уважного розгляду цих методів було зроблено вибір на користь одного з них з огляду на вимоги роботи.

Крім цього, висвітлено методи відстеження перешкод та запобігання зіткненням. Проаналізовано різні технології, включаючи сенсори, системи комп'ютерного зору та радіолокацію, що застосовуються для забезпечення безпеки руху. На основі даного аналізу було зроблено вибір найоптимальнішого методу, який забезпечує ефективне уникнення зіткнень у різних сценаріях руху автомобілів. Цей всебічний аналіз як надає огляд сучасного стану технологій автоматизації у транспортній сфері, а й визначає конкретні рішення, які можуть бути успішно інтегровані у проект автоматизованого транспорту.

Отримані результати забезпечують основу для розробки та впровадження ефективних та інноваційних систем керування, сфокусованих на підвищенні безпеки та ефективності автомобільного руху.

У другому розділові було проаналізовано різні емулятори, детально розглянуо переваги та недоліки кожного із них, та виходячи з цього було обрано емулятор Webots який відповідає вимогам, та надає весь необхідний функціонал для досягнення поставленої даною роботою задачі.

У третьому розділові було встановлено JOSM для можливості зручного редагування, та створення емуляцій на прикладі реальної інфраструктури, яка береться з вільних супутникових даних, та налаштовано імпортер, який і дає можливість перетворити формат osm, в wbt, також було розглянуто інтерфейс середовища емуляції webots, за допомогою якого було виконано додавання інших об’єктів, та редагування їх характеристик, також було модифіковано алгоритм, а саме було додано можливість орієнтуванні за допомогою лідарів. для того щоб надати машині властивості об’їжджати перешкоди на основі даних які отримують 2 датчика, позаду та спереду машини.

Також було проведено економічний аналіз, та комерційна перспективність використання досліджуваної технології, та не дивлячись на те що перспективність проекту на даний момент не приносить достатнього прибутку, та не купається на даний момент, хоч і дає можливість запобігати аварійних ситуацій на дорозі, та спрощує життя людям які мають відношення до досліджуваної теми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Forms of Transport Automation [Електронний ресурс]: URL: https:// transportgeography.org/ contents/conclusion/future-transportation-systems/forms-of-transport-automation/

2. What is an Autonomous Car? [Електронний ресурс]: URL: https:/ /www. Synopsys .com/automotive/what-is-autonomous-car.html

3. Autonomous Trucking [Електронний ресурс]. URL: https://guides.loc.gov/trucking-industry/autonomous-trucking

4. Why SLAM Matters [Електронний ресурс]. URL: https://www.mathworks.com/discovery/slam.html

5. A* Algorithm Concepts and Implementation [Електронний ресурс]. URL – Режим доступу: https:// www.simplilearn.com/ tutorials/artificial-intelligence-tutorial/a-star-algorithm

6. Fish swarm optimization algorithm applied to engineering system design [Електронний ресурс]. URL: Режим доступу: https: //www.scielo.br/j/ lajss/a/ ZsdRkGWRVtDdHJP8WTDFFpB/?lang=en

7. What Is a Neural Network [Електронний ресурс]. URL: Режим доступу: https:/ /www.investopedia. com/terms/n/neuralnetwork.asp

8. A Complete Guide to Proximity Warning Systems [Електронний ресурс]. URL: https://zonesafe.com/a-complete-guide-to-proximity-warning-alert-systems/

9. What Is GPS and how do global positioning systems work? [Електронний ресурс]. URL: Режим доступу: https://www. geotab.com/blog/what-is-gps/

10. Эмуляторы и симуляторы vs реальные устройства для автоматизации тестирования [Електронний ресурс]. URL: Режим доступу: https://habr.com/ru/companies/otus/articles/596371/

11. SOLIDWORKS Simulation [Електронний ресурс] URL: Режим доступу: https://www.javelin-tech.com /3d/technology/solidworks-simulation/

12. NXT-G: the development environment supplied with Lego Mindstorms [Електронний ресурс]. URL: Режим доступу: https://www.generationrobots.com/blog/en/nxt-g-the-development-environment-supplied-with-lego-mindstorms-nxt-g/

13. Robotics System Toolbox [Електронний ресурс].URL: https: //www.mathworks.com/ products/robotics.html

14. Simulate your Robotics Applications [Електронний ресурс]. URL: http://www.anykode.com/index.php

15. Webots User Guide [Електронний ресурс]. URL https://cyberbotics.com/doc/guide/index

16. What is Orocos [Електронний ресурс]. URL: https://docs.orocos.org/

17. V-REP — гибкая и масштабируемая платформа для робомоделирования [Електронний ресурс]. URL: https://habr. com/ru/articles/383009/

18. ROBOTC is a cross-robotics-platform programming language for popular educational robotics systems [Електронний ресурс]. URL: https://www.robotc.net/

19. Bricx Command Center 3.3 [Електронний ресурс]. URL: https://bricxcc.sourceforge.net/

20. Microsoft Robotics Developer Studio [Електронний ресурс]. URL: https:// cxem.net/ software/MRDS.php

21. JOSM [Електронний ресурс]. URL: https://josm.openstreetmap.de/

22. What is Orocos [Електронний ресурс] URL: https://docs.orocos.org/

23. lxml - XML and HTML with Python [Електронний ресурс]. URL: https://lxml.de/

24. pyproj [Електронний ресурс]. URL: Режим доступу: https://pypi.org/project/pyproj/

25. webcolors [Електронний ресурс]. URL: https://pypi.org/ project/webcolors/

26. Configparser [Електронний ресурс]. URL: https://docs.python.org/uk/3/library/configparser.html

27. shapely [Електронний ресурс]. URL: https://pypi.org/project/shapely/

28. GEOS [Електронний ресурс]. URL: Режим доступу: https://libgeos.org/usage/c_api/

29. webots [Електронний ресурс]. URL: – Режим доступу: https://www.cyberbotics.com/

30. JOSM download [Електронний ресурс]. URL: https://josm.openstreetmap/– Режим доступу: https://josm.openstreetmap/

31. Python official site [Електронний ресурс]. URL: Режим доступу: https://www.python.org/

32. Christoph Gohlke [Електронний ресурс]. URL: Режим доступу: https://www.cgohlke.com

33. Webots Scene tree [Електронний ресурс].URL: https:// www. Cyberbotics .com/ doc/guide/the-scene-tree

34. The 3d window [Електронний ресурс]/ URL: https://www.cyberbotics.com/doc/guide/the-3d-window

35. Webots the console [Електронний ресурс]. URL: Режим доступу: https://www.cyberbotics.com /doc/guide/the-console

36. Webots the user interface [Електронний ресурс]. URL: https://www.cyberbotics.com/doc /guide/the-user-interface

37. Webots cloud [Електронний ресурс]. URL: https://webots.cloud/proto?keyword=sensor%2Fdistance+sensor

38. Начисление зарплаты 2022 [Електронний ресурс]. URL: Режим доступу: https://www.golovbukh.ua/ article/ru/8031-nachislenie-zarabotnoy-platy

39. Поняття заробітної плати. Порядок і строки виплати заробітної плати [Електронний ресурс], URL: Режим доступу: https://wiki .legalaid. gov.ua/index.php/

40. Розмір єдиного соціального внеску [Електронний ресурс], URL: Режим доступу: https://sumy.tax.gov.ua/media-ark/news-ark/623890.html

41. Амортизація [Електронний ресурс]. URL: Режим доступу: https://uk.wikipedia.org/wiki

42. Розрахунок витрат на енергоносії [Електронний ресурс]. URL: Режим доступу: https://studfile.net/ preview/7365087/page:8/

43. Прибуток: порядок і формула для визначення [Електронний ресурс]. URL: https:// osvita.ua/ vnz/reports/econom_pidpr/19856/

44. Прогнозування комерційних ефектів від реалізації результатів розробки [Електронний ресурс], URL: https://web.posibnyky. vntu.edu.ua/fmib/ 9kozlovskij_metodvkaz_ ekonomchastini_magisterskih_kvalifikacrobit/4.htm

45. МОДЕЛЮВАННЯ РУХУ БЕЗПІЛОТНОГО АВТОМОБІЛЯ В СИМУЛЯТОРІ WEBOTS [Електронний ресурс]: URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/author/submission/19695

ДОДАТКИ

86

Додаток А

(обов'язковий)

Технічне Завдання.

ЗАТВЕРДЖЕНО
Зав. кафедри АІІТ
Олег БІСІКАЛО.
«12» жовтня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«Моделювання руху безпілотного автомобіля в симуляторі WeBots»
08-31.МКР.009.02.000 ТЗ

Керівник роботи:
к.т.н. доц. Ярослав КУЛИК.
—"12" жовтня 2023 р.

Виконавець:
ст. гр. 1АКІТ-22м
Богдан КОЛЕСНІК
—"12" жовтня 2023 р.

Вінниця 2023

1. Назва та галузь застосування: Моделювання руху безпілотного автомобіля в симуляторі WeBots.

2. Підстава для проведення робіт: Підставою для виконання роботи є наказ № 298 по ВНТУ від «18» 03 2023р., та індивідуальне завдання на МКР, затверджене протоколом № 1 засідання кафедри АІІТ від «30» 08 2023р.

3. Мета та призначення роботи: Метою цієї магістерської роботи є моделювання підсистеми уникнення зіткнень для безпілотного автомобіля в симуляторі WeBots

4. Джерела розробки:

1)webots [Електронний ресурс].URL : https://www.cyberbotics.com/

2)JOSM download [Електронний ресурс]. URL: Режим доступу: https://josm.openstreetmap/Вихідні дані для проведення робіт: глобальна комп'ютерна мережа; операційна система Windows; підтримка усіх доступних браузерів; протокол передачі даних HTTPS; фреймворк клієнта vue.js та Laravel.

5.Методи дослідження: використання реальних географічних рішень, Визначення ефективності системи автоматичного гальмування та уникнення зіткнень.

6. Економічні показники

До економічних показників входять:

- витрати на розробку 36 212 гривень

- приведена вартість прибутку за 1 рік 28.35 тисяч гривень - мінімальна дохідність 23%

- термін окупності розробка, не окупляться

7. Стадії розробки:

а) Аналіз предметної області 28.09 – 28.09

б) Аналіз та вибір середовища для розробки 16.10 – 29.10

в) Моделювання руху у середовищі Webots 30.10 – 24.11

г) Економічна частина 29.11 – 2911

ж) Оформлення матеріалів до захисту МКР 29.11 – 19.12

8. Порядок контролю та приймання

88

Рубіжний контроль провести до «5» 12 2023 р.

Попередній захист МКР провести «8» 12 2023 р.

Захист МКР провести до «19» 12 2023 р.

Розробив студент групи 1АКІТ-22м _____ Богдан Колеснік

Додаток Б

(обов’язковий)

Ілюстративна частина

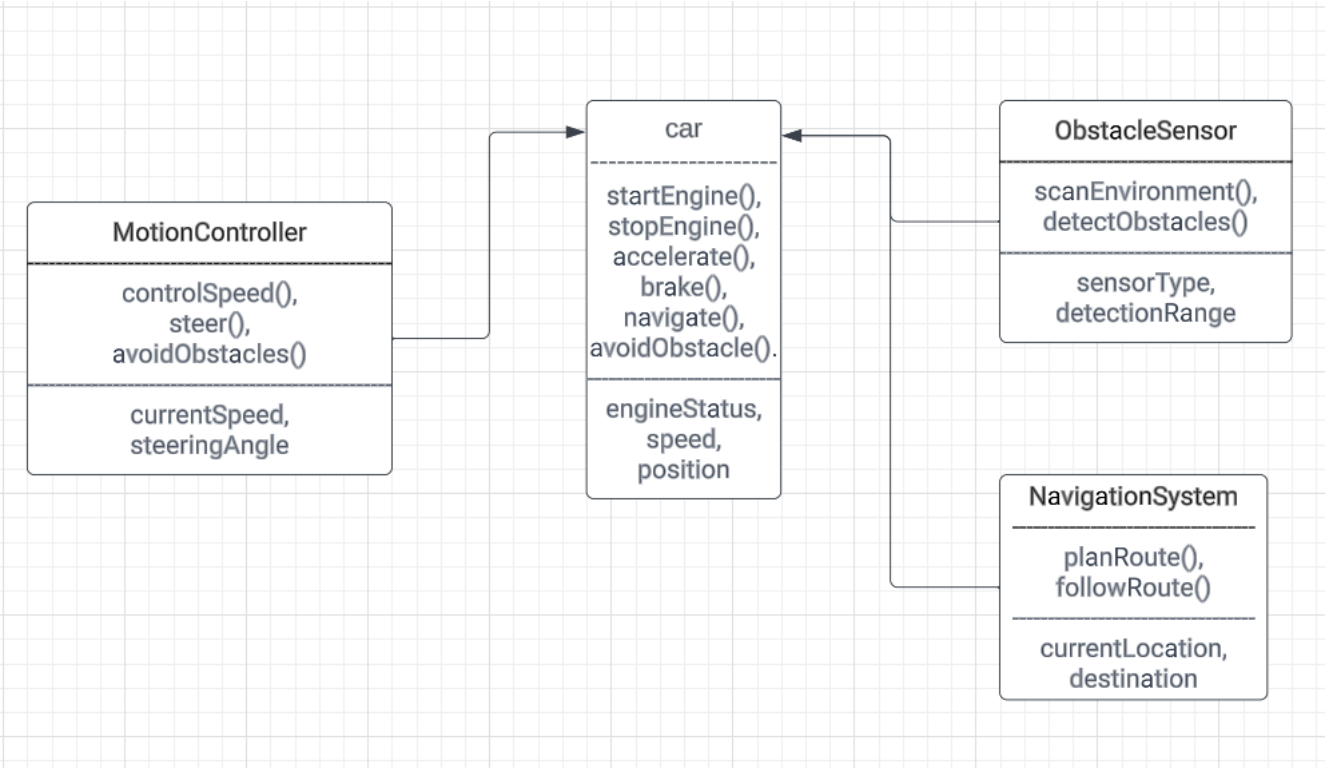


Рисунок Б.1 – Діаграма класів

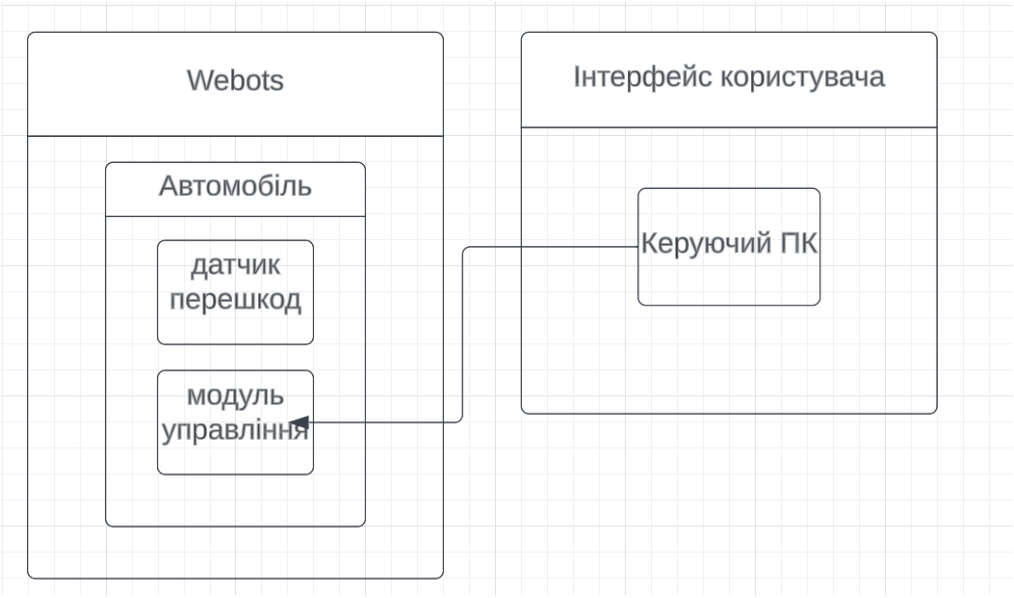


Рисунок Б.2 – Діаграма розгортки

Рисунок Б.3 – Діаграма життєвого циклу

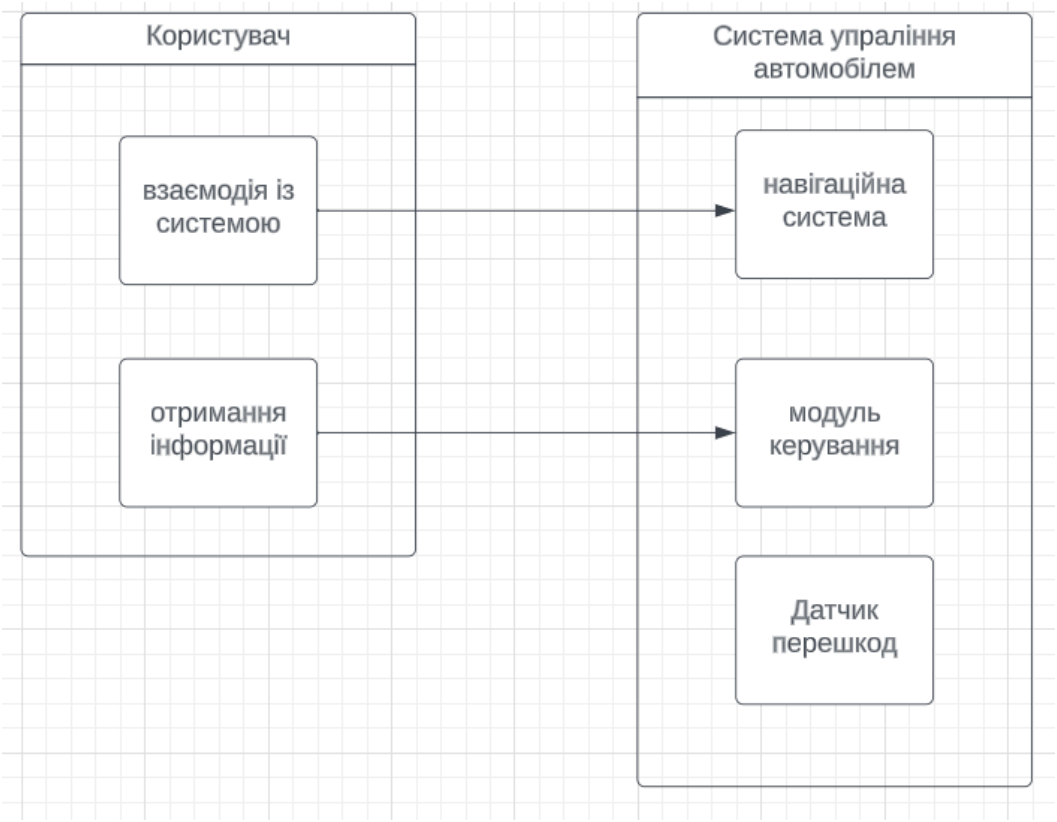


Рисунок Б.4 – Діаграма використання

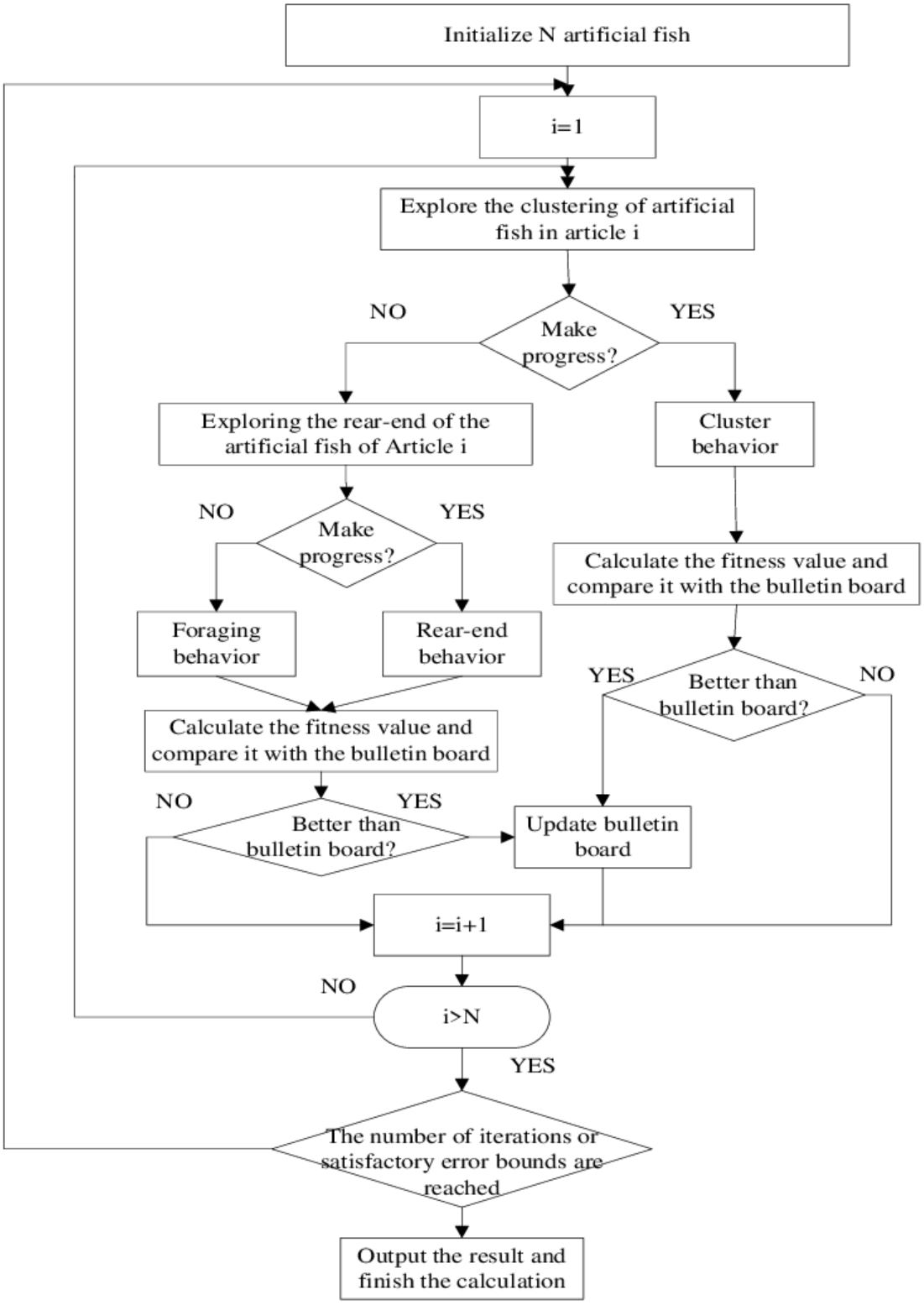


Рисунок Б.6 – Алгоритм рою риб

Додаток В

(обов’язковий)

Лістинг програми

```
#include <webots/camera.h>
#include <webots/device.h>
#include <webots/display.h>
#include <webots/gps.h>
#include <webots/keyboard.h>
#include <webots/lidar.h>
#include <webots/robot.h>
#include <webots/vehicle/driver.h>

#include <math.h>
#include <stdio.h>
#include <string.h>

enum { X, Y, Z };

#define TIME_STEP 50
#define UNKNOWN 99999.99
#define KP 0.25
#define KI 0.006
#define KD 2

bool PID_need_reset = false;

#define FILTER_SIZE 3

bool enable_collision_avoidance = false;
bool enable_display = false;
bool has_gps = false;
bool has_camera = false;

WbDeviceTag camera;
int camera_width = -1;
int camera_height = -1;
double camera_fov = -1.0;

WbDeviceTag sick;
int sick_width = -1;
double sick_range = -1.0;
double sick_fov = -1.0;

WbDeviceTag display;
int display_width = 0;
int display_height = 0;
WbImageRef speedometer_image = NULL;

WbDeviceTag gps;
double gps_coords[3] = {0.0, 0.0, 0.0};
double gps_speed = 0.0;

double speed = 0.0;
```

```
double steering_angle = 0.0;
int manual_steering = 0;
bool autodrive = true;

void print_help() {
  printf("You can drive this car!\n");
  printf("Select the 3D window and then use the cursor keys to:\n");
  printf("[LEFT]/[RIGHT] - steer\n");
  printf("[UP]/[DOWN] - accelerate/slow down\n");
}

void set_autodrive(bool onoff) {
  if (autodrive == onoff)
    return;
  autodrive = onoff;
  switch (autodrive) {
    case false:
      printf("switching to manual drive...\n");
      printf("hit [A] to return to auto-drive.\n");
      break;
    case true:
      if (has_camera)
        printf("switching to auto-drive...\n");
      else
        printf("impossible to switch auto-drive on without camera...\n");
      break;
  }
}

// set target speed
void set_speed(double kmh) {
  // max speed
  if (kmh > 250.0)
    kmh = 250.0;

  speed = kmh;

  printf("setting speed to %g km/h\n", kmh);
  wbu_driver_set_cruising_speed(kmh);
}

// positive: turn right, negative: turn left
void set_steering_angle(double wheel_angle) {
  // limit the difference with previous steering_angle
  if (wheel_angle - steering_angle > 0.1)
    wheel_angle = steering_angle + 0.1;
  if (wheel_angle - steering_angle < -0.1)
    wheel_angle = steering_angle - 0.1;
  steering_angle = wheel_angle;
  // limit range of the steering angle
  if (wheel_angle > 0.5)
    wheel_angle = 0.5;
  else if (wheel_angle < -0.5)
    wheel_angle = -0.5;
  wbu_driver_set_steering_angle(wheel_angle);
}
```

```
void change_manual_steer_angle(int inc) {
  set_autodrive(false);

  double new_manual_steering = manual_steering + inc;
  if (new_manual_steering <= 25.0 && new_manual_steering >= -25.0) {
    manual_steering = new_manual_steering;
    set_steering_angle(manual_steering * 0.02);
  }

  if (manual_steering == 0)
    printf("going straight\n");
  else
    printf("turning %.2f rad (%s)\n", steering_angle, steering_angle < 0 ? "left" : "right");
}

void check_keyboard() {
  int key = wb_keyboard_get_key();
  switch (key) {
    case WB_KEYBOARD_UP:
      set_speed(speed + 5.0);
      break;
    case WB_KEYBOARD_DOWN:
      set_speed(speed - 5.0);
      break;
    case WB_KEYBOARD_RIGHT:
      change_manual_steer_angle(+1);
      break;
    case WB_KEYBOARD_LEFT:
      change_manual_steer_angle(-1);
      break;
    case 'A':
      set_autodrive(true);
      break;
  }
}

// compute rgb difference
int color_diff(const unsigned char a[3], const unsigned char b[3]) {
  int i, diff = 0;
  for (i = 0; i < 3; i++) {
    int d = a[i] - b[i];
    diff += d > 0 ? d : -d;
  }
  return diff;
}

double process_camera_image(const unsigned char *image) {
  int num_pixels = camera_height * camera_width;  // number of pixels in the image
  const unsigned char REF[3] = {95, 187, 203};    // road yellow (BGR format)
  int sumx = 0;                                   // summed x position of pixels
  int pixel_count = 0;                            // yellow pixels count

  const unsigned char *pixel = image;
  int x;
  for (x = 0; x < num_pixels; x++, pixel += 4) {
```

```
    if (color_diff(pixel, REF) < 30) {
      sumx += x % camera_width;
      pixel_count++;  // count yellow pixels
    }
  }
  if (pixel_count == 0)
    return UNKNOWN;

  return ((double)sumx / pixel_count / camera_width - 0.5) * camera_fov;
}
double filter_angle(double new_value) {
  static bool first_call = true;
  static double old_value[FILTER_SIZE];
  int i;

  if (first_call || new_value == UNKNOWN) {  // reset all the old values to 0.0
    first_call = false;
    for (i = 0; i < FILTER_SIZE; ++i)
      old_value[i] = 0.0;
  } else {  // shift old values
    for (i = 0; i < FILTER_SIZE - 1; ++i)
      old_value[i] = old_value[i + 1];
  }

  if (new_value == UNKNOWN)
    return UNKNOWN;
  else {
    old_value[FILTER_SIZE - 1] = new_value;
    double sum = 0.0;
    for (i = 0; i < FILTER_SIZE; ++i)
      sum += old_value[i];
    return (double)sum / FILTER_SIZE;
  }
}
double process_sick_data(const float *sick_data, double *obstacle_dist) {
  const int HALF_AREA = 20;  // check 20 degrees wide middle area
  int sumx = 0;
  int collision_count = 0;
  int x;
  *obstacle_dist = 0.0;
  for (x = sick_width / 2 - HALF_AREA; x < sick_width / 2 + HALF_AREA; x++) {
    float range = sick_data[x];
    if (range < 20.0) {
      sumx += x;
      collision_count++;
      *obstacle_dist += range;
    }
  }
  if (collision_count == 0)
    return UNKNOWN;

  *obstacle_dist = *obstacle_dist / collision_count;
  return ((double)sumx / collision_count / sick_width - 0.5) * sick_fov;
}

void update_display() {
```

```
  const double NEEDLE_LENGTH = 50.0;
  wb_display_image_paste(display, speedometer_image, 0, 0, false);
  double current_speed = wbu_driver_get_current_speed();
  if (isnan(current_speed))
    current_speed = 0.0;
  double alpha = current_speed / 260.0 * 3.72 - 0.27;
  int x = -NEEDLE_LENGTH * cos(alpha);
  int y = -NEEDLE_LENGTH * sin(alpha);
  wb_display_draw_line(display, 100, 95, 100 + x, 95 + y);
  char txt[64];
  sprintf(txt, "GPS coords: %.1f %.1f", gps_coords[X], gps_coords[Z]);
  wb_display_draw_text(display, txt, 10, 130);
  sprintf(txt, "GPS speed:  %.1f", gps_speed);
  wb_display_draw_text(display, txt, 10, 140);
}

void compute_gps_speed() {
  const double *coords = wb_gps_get_values(gps);
  const double speed_ms = wb_gps_get_speed(gps);
  gps_speed = speed_ms * 3.6;  // convert from m/s to km/h
  memcpy(gps_coords, coords, sizeof(gps_coords));
}

double applyPID(double yellow_line_angle) {
  static double oldValue = 0.0;
  static double integral = 0.0;

  if (PID_need_reset) {
    oldValue = yellow_line_angle;
    integral = 0.0;
    PID_need_reset = false;
  }
  if (signbit(yellow_line_angle) != signbit(oldValue))
    integral = 0.0;

  double diff = yellow_line_angle - oldValue;
  if (integral < 30 && integral > -30)
    integral += yellow_line_angle;

  oldValue = yellow_line_angle;
  return KP * yellow_line_angle + KI * integral + KD * diff;
}

int main(int argc, char **argv) {
  wbu_driver_init();

  int j = 0;
  for (j = 0; j < wb_robot_get_number_of_devices(); ++j) {
    WbDeviceTag device = wb_robot_get_device_by_index(j);
    const char *name = wb_device_get_name(device);
    if (strcmp(name, "Sick LMS 291") == 0)
      enable_collision_avoidance = true;
    else if (strcmp(name, "display") == 0)
      enable_display = true;
    else if (strcmp(name, "gps") == 0)
      has_gps = true;
```

```
  else if (strcmp(name, "camera") == 0)
    has_camera = true;
}
if (has_camera) {
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, TIME_STEP);
  camera_width = wb_camera_get_width(camera);
  camera_height = wb_camera_get_height(camera);
  camera_fov = wb_camera_get_fov(camera);
}
if (enable_collision_avoidance) {
  sick = wb_robot_get_device("Sick LMS 291");
  wb_lidar_enable(sick, TIME_STEP);
  sick_width = wb_lidar_get_horizontal_resolution(sick);
  sick_range = wb_lidar_get_max_range(sick);
  sick_fov = wb_lidar_get_fov(sick);
}
if (has_gps) {
  gps = wb_robot_get_device("gps");
  wb_gps_enable(gps, TIME_STEP);
}
if (enable_display) {
  display = wb_robot_get_device("display");
  speedometer_image = wb_display_image_load(display, "speedometer.png");
}
if (has_camera)
  set_speed(50.0);  // km/h
wbu_driver_set_hazard_flashers(true);
wbu_driver_set_dipped_beams(true);
wbu_driver_set_antifog_lights(true);
wbu_driver_set_wiper_mode(SLOW);

print_help();

// allow to switch to manual control
wb_keyboard_enable(TIME_STEP);

// main loop
while (wbu_driver_step() != -1) {
  // get user input
  check_keyboard();
  static int i = 0;

  // updates sensors only every TIME_STEP milliseconds
  if (i % (int)(TIME_STEP / wb_robot_get_basic_time_step()) == 0) {
    // read sensors
    const unsigned char *camera_image = NULL;
    const float *sick_data = NULL;
    if (has_camera)
      camera_image = wb_camera_get_image(camera);
    if (enable_collision_avoidance)
      sick_data = wb_lidar_get_range_image(sick);

    if (autodrive && has_camera) {
      double yellow_line_angle = filter_angle(process_camera_image(camera_image));
      double obstacle_dist;
```

```
        double obstacle_angle;
        if (enable_collision_avoidance)
          obstacle_angle = process_sick_data(sick_data, &obstacle_dist);
        else {
          obstacle_angle = UNKNOWN;
          obstacle_dist = 0;
        }
        if (enable_collision_avoidance && obstacle_angle != UNKNOWN) {
          wbu_driver_set_brake_intensity(0.0);
          double obstacle_steering = steering_angle;
          if (obstacle_angle > 0.0 && obstacle_angle < 0.4)
            obstacle_steering = steering_angle + (obstacle_angle - 0.25) / obstacle_dist;
          else if (obstacle_angle > -0.4)
            obstacle_steering = steering_angle + (obstacle_angle + 0.25) / obstacle_dist;
          double steer = steering_angle;
          if (yellow_line_angle != UNKNOWN) {
            const double line_following_steering = applyPID(yellow_line_angle);
            if (obstacle_steering > 0 && line_following_steering > 0)
              steer = obstacle_steering > line_following_steering ? obstacle_steering :
line_following_steering;
            else if (obstacle_steering < 0 && line_following_steering < 0)
              steer = obstacle_steering < line_following_steering ? obstacle_steering :
line_following_steering;
          } else
            PID_need_reset = true;
          set_steering_angle(steer);
        } else if (yellow_line_angle != UNKNOWN) {
          wbu_driver_set_brake_intensity(0.0);
          set_steering_angle(applyPID(yellow_line_angle));
        } else {

          wbu_driver_set_brake_intensity(0.4);
          PID_need_reset = true;
        }
      }

      // update stuff
      if (has_gps)
        compute_gps_speed();
      if (enable_display)
        update_display();
    }

    ++i;
  }

  wbu_driver_cleanup();

  return 0;  // ignored
}
```

Додаток Д

(обов'язковий)

зразок протоколу перевірки кваліфікаційної роботи на наявність запозичень

ПРОТОКОЛ

ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Моделювання руху безпілотного автомобіля в симуляторі WeBots

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та комп'ютерно інтегрованих технологій, та автоматизації, факультет інформаційних технологі та автоматизації

Показники звіту подібності Plagiat.pl (StrikePlagiarism)

Оригінальність 98.1% Схожість 1.9%

Аналіз звіту подібності (відмітити потрібне)

[ ] Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

[ ] Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

[ ] Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Роман МАСЛІЙ (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Plagiat.pl (StrikePlagiatism) щодо роботи.

Автор роботи _____ (підпис) Богдан КОЛЕСНІК. (прізвище, ініціали)

Керівник роботи _____ (підпис) Ярослав КУЛИК (прізвище, ініціали)