Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Автоматизація та комп'ютерно-інтегровані технології та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

перший (бакалаврський)
(рівень вищої освіти)

Розробка віртуального макету для дослідження навчання робота на основі досвіду
(тема)

Виконав:

студент 4 курсу, групи АКТСІ-20-3

Пара І. І.

Спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології,

освітньої програми «Системна інженерія»

Тип програми Освітньо-професійна

Освітня програма Системна інженерія

Керівник Гурін Д.В.

Допускається до захисту
Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

Харків 2024

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології, освітньої програми «Системна інженерія»
Тип програми Освітньо-професійна
Освітня програма Системна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри КІТАР

(підпис)

«____» _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Пари Іллі Ігоровича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): Розробка віртуального макету для дослідження навчання роботів на основі досвіду

Затверджена наказом № 545 Ст по університету від 03.06.2024

2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024 р.

3. Вихідні дані до роботи: Arduino Uno, фізичний макет мобільного роботу, генетичний алгоритм, розроблене віртуальне середовище, Python.

4. Перелік питань, що потрібно опрацювати в роботі: 4.1 Вступ; 4.2 Аналіз предметної області; 4.3 Розробка віртуального макету; 4.4 Розробка фізичного макету і дослідження впливу генетичного алгоритму на роботу мобільної платформи; 4.5 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt). – с. Формату А4.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз літератури за темою кваліфікаційної роботи	от 02.05.2024 до 05.05.2024	Виконано
2	Аналіз методів навчання роботів	от 13.05.2024 до 15.05.2024	Виконано
3	Розробка віртуального макету для дослідження роботи генетичного алгоритму	от 15.05.2024 до 20.05.2024	Виконано
4	Розробка фізичного макету мобільної платформи та інтеграція генетичного алгоритму до робота.	от 20.05.2024 до 28.05.2024	Виконано
5	Оформлення пояснювальної записки	от 28.05.2024 до 14.06.2024	Виконано
6	Подання кваліфікаційної роботи в ЕК	19.06.2024	Виконано

Дата видачі завдання 12.04.2024 р.

Студент _____ Пара І. І.
(підпис) (прізвище, ім'я та по батькові)

Керівник роботи _____ ст. викл. каф. КІТАР Гурін Д.В.
(підпис) (прізвище, ім'я та по батькові)

РЕФЕРАТ

Пояснювальна записка містить: 91 с., 29 рис., 2 дод., джерел.

РОЗРОБКА ВІРТУАЛЬНОГО МАКЕТУ ДЛЯ РОБОТИ С ГЕНЕТИЧНИМ АЛГОРИТМОМ, МОБІЛЬНА ПЛАТФОРМА, СИСТЕМА КЕРУВАННЯ НА ОСНОВІ ГА.

Мета роботи – розробка віртуального макету, спрямованого на дослідження теоретичних аспектів навчання агентів на основі досвіду за допомогою генетичного алгоритму. Використовуючи цей алгоритм, мобільна платформа буде відтворювати та аналізувати різноманітні стратегії навчання агентів, включаючи їх взаємодію з навколишнім середовищем.

Об’єкт розробки – дослідження навчання робота на основі досвіду.

Предмет розробки – макет для дослідження навчання робота на основі досвіду.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– аналіз сучасних методів навчання роботів;

– розробка віртуального макету для дослідження роботи генетичного алгоритму;

– аналіз ефективності генетичного алоритму;

– реалізація керування мобільною платформою за допомогою генетичного алгоритму;

– порівняння ефективності керування мобільною платформою за допомогою генетичного алгоритму та за допомогою звичайних методів.

ABSTRACT

The explanatory note contains: 91 p., 29 fig., 2 app., sources.

DEVELOPMENT OF A VIRTUAL MODEL FOR WORKING WITH A GENETIC ALGORITHM, MOBILE PLATFORM, GA-BASED CONTROL SYSTEM.

Objective of the work – development of a virtual model aimed at researching the theoretical aspects of training agent based on experience using a genetic algorithm. Using this algorithm, the mobile platform will reproduce and analyze various strategies for training agents including their interaction with the environment.

Object of development – research on experience-based training of robot.

Subject of development – model for researching experience-based training of robot.

To achieve the set objective, the following tasks need to be solved:

– analysis of modern methods of robot training;

– development of a virtual model for researching the genetic algorithm;

– analysis of the effectiveness of the genetic algorithm;

– implementation of mobile platform control using the genetic algorithm;

– comparison of the effectiveness of mobile platform control using the genetic algorithm and conventional methods.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ГА – генетичний алгоритм;

ПЗ – програмне забезпечення.

ВСТУП

В сучасному світі робототехніка стрімко розвивається, впроваджуючи інноваційні рішення у різні галузі науки та промисловості. Однією з ключових проблем є навчання роботів ефективним стратегіям взаємодії з навколишнім середовищем, що дозволяє їм адаптуватися та виконувати складні завдання. Традиційні методи навчання мають свої обмеження, тому зростає інтерес до використання генетичних алгоритмів, які імітують процес природного відбору і здатні знаходити оптимальні рішення у складних системах.

Генетичні алгоритми, завдяки своїй здатності еволюціонувати і покращуватися з кожним циклом, відкривають нові горизонти у навчанні роботів. Використовуючи цей підхід, можливо не лише покращити ефективність роботів, але й розробити нові методи взаємодії з навколишнім середовищем. Проте, для дослідження і практичного впровадження цих методів необхідна відповідна інфраструктура, зокрема, віртуальні моделі, що дозволяють проводити експерименти без ризику і великих витрат.

Ця робота спрямована на розробку віртуального макету для дослідження навчання роботів на основі досвіду за допомогою генетичних алгоритмів. Такий підхід дозволить дослідити ефективність різних стратегій навчання, оптимізувати процеси адаптації та взаємодії з середовищем.

Об’єкт розробки – дослідження навчання робота на основі досвіду.

Предмет розробки – макет для дослідження навчання робота на основі досвіду.

Для досягнення цієї мети буде вирішено такі завдання:

– аналіз сучасних методів навчання роботів. Це включає в себе дослідження різних методів навчання роботів, розгляд існуючих переваг та недоліків різних систем;

– розробка віртуального макету для дослідження роботи генетичного алгоритму. Це включає в себе розробку програми для аналізу роботи ГА та його впливу на керування агентом;

– аналіз ефективності генетичного алоритму. Це включає в себе роботу із вхідними даними для аналізу роботи ГА та порівняння його ефективності в залежності від наявних даних;

– реалізація керування мобільною платформою за допомогою генетичного алгоритму. Це включає в себе збірку фізичного макету мобільної платформи та реалізація роботи програм для керування платформою;

– порівняння ефективності керування мобільною платформою за допомогою генетичного алгоритму та за допомогою звичайних методів. Це включає в себе аналіз ефективності роботи програми на основі ГА та без використання ГА.

Кваліфікаційна робота виконана згідно ДСТУ 3008 – 15 [1], та керуючись навчальним посібником з дипломного проекту [2] та методичними вказівками [3].

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ НАЧАННЯ РОБОТІВ

1.1 Машинне навчання

Машинне навчання є одним з найбільш перспективних і складних напрямків в сучасній науці. Воно є важливою складовою науки про дані, яка швидко розвивається. За допомогою статистичних методів алгоритми навчаються класифікувати дані, прогнозувати результати та виділяти важливі моменти в процесі збирання та аналізу даних.

Машинне навчання є великою підгалуззю штучного інтелекту, яка зосереджується на розробці алгоритмів, здатних до самонавчання. У цьому процесі система обробляє значну кількість прикладів, виявляє закономірності і використовує їх для прогнозування результатів на нових вхідних даних. Мета машинного навчання полягає в частковій або повній автоматизації вирішення складних аналітичних завдань. Основними інструментами машинного навчання є алгоритми, які повинні бути реалізовані за допомогою певних мов програмування та працювати в специфічних середовищах на певному апаратному забезпеченні. До мов програмування для машинного навчання належать: C++, Octave, Python, Java, Ruby, R, Matlab, Scala.

Методи машинного навчання класифікуються за типом навчання:

– алгоритми навчання з учителем: ці алгоритми використовують штучні нейронні мережі для вирішення ряду прикладів, кожен з яких складається з двох частин: вхідні дані та бажане вихідне значення. У процесі "навчання" вагові коефіцієнти коригуються, щоб вхідні дані давали вихідні значення, максимально наближені до бажаних;

– алгоритми навчання з підкріпленням: у цьому випадку алгоритми не отримують бажаного вихідного значення, але після кожного прикладу їм надається оцінка виконання завдання, яка може бути позитивною або негативною;

– алгоритми навчання без учителя: ці алгоритми працюють із наборами прикладів без зазначеного бажаного значення на виході. Під час обробки даних у штучних нейронних мережах відбуваються процеси самоорганізації, що призводять до модифікації вагових коефіцієнтів, і, як результат, мережі здатні вирішувати певні завдання [4].

1.2 Огляд існуючих методів навчання роботів

1.2.1 Навчання з підкріпленням

Навчання з підкріпленням – це метод машинного навчання, який використовується для навчання агентів (роботів) діяти в середовищі таким чином, щоб максимізувати очікуване значення винагороди.

Основні принципи навчання з підкрипленням:

У навчанні з підкріпленням є агент – це автономна сутність, яка може сприймати середовище та діяти в ньому. Середовище – це все, що оточує агента, включаючи інші об'єкти, фізичні закони та правила. Агент виконє дію. Дія – це те, що агент може/повинен робити відносно стану середовища. Стан – це опис середовища в певний момент часу. Винагорода за виконану дію агента – це скалярне значення, що відображає бажаність стану або дії. Функція цінності- це функція, яка оцінює очікуване значення винагороди за певний стан. Політика – це функція, яка визначає, яку дію агент буде виконувати в певному стані.

Алгоритми, які використовуються:

– метод Монте-Карло: цей метод оцінює цінність стану, використовуючи середнє значення винагород, отриманих з епізодів, які починаються з цього стану;

– TD-навчання: цей метод оцінює цінність стану, використовуючи різницю між поточною оцінкою та очікуваною винагородою за наступний стан;

– Q-навчання: цей метод використовує таблицю Q-значень для зберігання очікуваної винагороди за виконання певної дії в певному стані.

Переваги:

– підкріплення навчання може використовуватися для навчання агентів в складних середовищах, де складно або неможливо заздалегідь визначити оптимальну політику;

– підкріплення навчання може використовуватися для навчання агентів адаптувати свою поведінку до мінливих умов середовища;

– підкріплення навчання може використовуватися для навчання агентів виконувати складні завдання, які неможливо запрограмувати вручну.

Недоліки:

– підкріплення навчання може потребувати багато даних для навчання;

– підкріплення навчання може призвести до небажаної поведінки агента, якщо винагорода не правильно визначена.

1.2.2 Навчання з демонстрацією

Навчання з демонстрацією – це метод машинного навчання, який використовується для навчання агентів (роботів) на основі прикладів поведінки, продемонстрованої людиною або іншим роботом.

Основні принципи, які використовуються у навчанні з демонстрацією:

Демонстрація – це послідовність дій, які виконує людина або інший робот. Імітація – це процес, за допомогою якого агент намагається відтворити дії, продемонстровані в демонстрації. Узагальнення – це процес, за допомогою якого агент використовує знання, отримані з демонстрації, для виконання нових завдань.

Методи, які використовуються в навчанні з демонстрацією:

– навчання за траєкторією: цей метод вчить агента слідувати за траєкторією, продемонстрованою в демонстрації;

– навчання за інваріантними характеристиками: цей метод вчить агента використовувати інваріантні характеристики для узагальнення з демонстрації;

– навчання за допомогою зворотного зв'язку: цей метод використовує зворотний зв'язок від людини або іншого робота для покращення поведінки агента.

Переваги:

– навчання з демонстрацією може використовуватися для навчання агентів завданням, які важко або неможливо запрограмувати вручну;

– навчання з демонстрацією може використовуватися для навчання агентів виконувати завдання з високою точністю.

Недоліки:

– навчання з демонстрацією може призвести до перенавчання, коли агент вчиться відтворювати поведінку демонстратора, а не узагальнювати її на нові завдання;

– залежно від якості демонстрацій, агент може навчитися неправильній або неповній стратегії виконання завдання.

1.2.3 Імітаційне навчання

Імітаційне навчання – це метод машинного навчання, який використовується для навчання агентів (роботів) на основі симуляцій.

Основні принципи, які використовуються в імітаційному навчанні:

Симулятор – це комп'ютерна програма, яка моделює поведінку реального світу. Віртуальне середовище – це середовище, створене симулятором.

Методи, які використовуються в імітаційному навчанні:

– навчання за траєкторією: цей метод вчить агента слідувати за траєкторією, згенерованою в симуляторі;

– навчання за допомогою зворотного зв'язку: Цей метод використовує зворотний зв'язок з симулятора для покращення поведінки агента;

– навчання з підкріпленням: цей метод використовує метод підкріплення навчання для навчання агента в симуляторі.

Переваги:

– імітаційне навчання може використовуватися для навчання агентів в небезпечних або складних середовищах;

– імітаційне навчання може використовуватися для навчання агентів завданням, які важко або неможливо демонструвати в реальному світі.

Недоліки:

– імітаційне навчання може призвести до перенавчання, коли агент вчиться відтворювати поведінку в симуляторі, а не узагальнювати її на реальний світ.

1.2.4 Навчання на основі досвіду

Навчання на основі досвіду – це потужний підхід до машинного навчання, який використовується для навчання роботів виконувати завдання та приймати рішення на основі їхнього власного досвіду. Цей метод відрізняється від традиційного програмування, де роботам чітко описують, як виконувати кожен крок завдання. Натомість навчання на основі досвіду дозволяє роботам досліджувати своє середовище, взаємодіяти з об'єктами та вчитися на власних помилках.

Основні принципи, які відносяться до навчання на основі досвіду представленні нижче.

Роботи мають свободу досліджувати та взаємодіяти з середовищем без постійного втручання людини. Також є взаємодія між роботами, де вони навчаються на основі взаємодії з об'єктами та іншими агентами в своєму середовищі. Роботи не мають чітко визначених цілей чи завдань, а натомість вони вільні досліджувати та вивчати те, що їм цікаво. Роботи навчаються на власних помилках, роблячи висновки та покращуючи свою поведінку з часом. Роботи здатні адаптуватися до нових ситуацій та завдань на основі свого попереднього досвіду. Навчання адаптується до потреб та стилю навчання кожного учня.

Методи, які використовуються у навчанні на основі досвіду:

– підкріплення навчання: роботи навчаються виконувати дії, отримуючи винагороду або покарання за свою поведінку;

– навчання з демонстрацією: роботи навчаються, спостерігаючи за тим, як люди або інші роботи виконують завдання;

– еволюційні алгоритми: роботи еволюціонують через процес природного відбору, де більш успішні особини з більшою ймовірністю передають свої гени наступному поколінню.

Переваги:

– агенти, які навчаються на основі досвіду, можуть адаптуватися до змін у середовищі та ситуаціях, оскільки їхнє навчання базується на отриманому досвіді;

– цей метод дозволяє агентам навчатися різноманітним стратегіям та поведінці, що робить їх гнучкими в різних ситуаціях;

– замість того, щоб програмувати жорсткі правила поведінки, можна навчити агента, як діяти на основі досвіду, що спрощує розробку систем штучного інтелекту;

– агенти можуть постійно вдосконалювати свою поведінку, навчаючись на основі нового досвіду, що дозволяє їм стати більш ефективними з часом;

– навчання на основі досвіду дозволяє розв'язувати складні завдання, для яких немає чітких правил або рішень.

Недоліки:

– для ефективного навчання на основі досвіду може бути необхідно значна кількість обчислювальних ресурсів, особливо для складних задач та великих навчальних наборів;

– навчання на основі досвіду вимагає наявності великої кількості якісних даних, що може бути складним для забезпечення в деяких випадках;

– якщо агент навчиться лише на основі обмеженого досвіду або неправильної інтерпретації даних, це може призвести до перенавчання та неправильної поведінки в недосліджених ситуаціях;

– іноді модель, навчена на основі досвіду, може бути недостатньо узагальнювальною і не в змозі адекватно реагувати на нові, раніше не бачені ситуації;

– якщо середовище або умови змінюються, може знадобитися постійне оновлення даних, на яких базується навчання, щоб агент залишався ефективним.

1.3 Нейроні мережі

Штучна нейронна мережа – це математична модель та пристрій для паралельних обчислень, який представляє собою систему з'єднаних і взаємодіючих простих процесорів (штучних нейронів). Як математична модель, штучна нейронна мережа є специфічним випадком методів розпізнавання образів або дискримінантного аналізу. Ці процесори зазвичай досить прості, особливо в порівнянні з тими, що використовуються в персональних комп'ютерах. Кожен процесор обробляє сигнали, які періодично отримує і надсилає іншим процесорам. Проте, коли такі локально прості процесори об'єднані у велику мережу з керованою взаємодією, вони здатні вирішувати складні завдання [6].

Це поняття виникло під час дослідження процесів у мозку під час мислення і спроб моделювання цих процесів. Отримані моделі називаються штучними нейронними мережами, які використовують велику кількість зв'язків між окремими нейронами. У мозку людини інформація обробляється динамічно, інтерактивно і самоорганізовано. Біологічні нейронні мережі існують у тривимірному просторі з мікроскопічних компонентів і мають безліч можливих з'єднань, тоді як штучні мережі мають фізичні обмеження.

Штучна нейронна мережа – це набір штучних нейронів, з'єднаних між собою. Зазвичай передатні функції всіх нейронів у мережі фіксовані, а ваги є параметрами мережі і можуть змінюватися. Деякі входи нейронів визначені як зовнішні входи мережі, а деякі виходи – як зовнішні виходи. Подавши числа на входи мережі, отримуємо набір чисел на виходах. Таким чином, робота нейронної мережі полягає в перетворенні вхідного вектора у вихідний, де це перетворення визначається вагами мережі.

Практично будь-яке завдання можна перетворити на задачу, яку вирішує нейронна мережа. Наприклад, задачу розпізнавання рукописних літер можна сформулювати в термінах нейронної мережі.

Розглянемо, чому вибирається вихід із максимальним рівнем сигналу. Рівень вихідного сигналу зазвичай може приймати будь-яке значення в певному діапазоні.

Проте у задачі розпізнавання нас цікавить не аналогова відповідь, а номер категорії (номер літери в алфавіті). Тому використовують підхід, де кожній категорії відповідає свій вихід, і відповіддю мережі є та категорія, на виході якої рівень сигналу максимальний. Таким чином, рівень сигналу на виході А вказує на ймовірність того, що на вхід була подана рукописна літера А. Задачі, де потрібно віднести вхідні дані до однієї з відомих категорій, називаються задачами класифікації. Викладений підхід – це стандартний спосіб класифікації за допомогою нейронних мереж.

Існуючі на даний час нейромережі є групуванням штучних нейронів, у виді з'єднаних між собою шарів (рис.1.1).

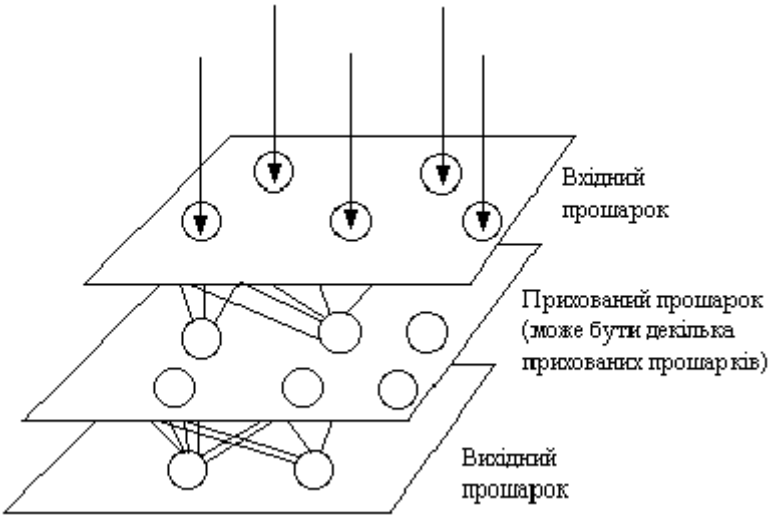

Рисунок 1.1 – Діаграма простої нейроної мережі

На рис. 1.1 показана типова структура штучних нейромереж. Хоча існують мережі, що містять лише один шар, або навіть один елемент, більшість реалізацій використовують мережі, що містять як мінімум три типи шарів - вхідний, схований і вихідний. Шар вхідних нейронів одержує дані або з вхідних файлів, або безпосередньо з електронних датчиків. Вихідний шар пересилає інформацію безпосередньо в зовнішнє середовище, до вторинного комп'ютерного процесу, або до іншого пристрою. Між цими двома шарами може бути кілька схованих шарів, що містять багато різноманітно зв'язаних нейронів. Входи і виходи кожного зі схованих нейронів з'єднані з іншими нейронами. Напрямок зв'язку від одного

нейрона до іншого є важливим аспектом нейромереж. У більшості мереж кожен нейрон схованого шару одержує сигнали від усіх нейронів попереднього шару і звичайно від нейронів вхідного шару. Після виконання операцій над сигналами, нейрон передає свій вихід усім нейронам наступних шарів, забезпечуючи передачу сигналу вперед (feedforward) на вихід. При зворотному зв'язку, вихід нейронів шару направляється до нейронів попереднього шару (рис. 1.2).

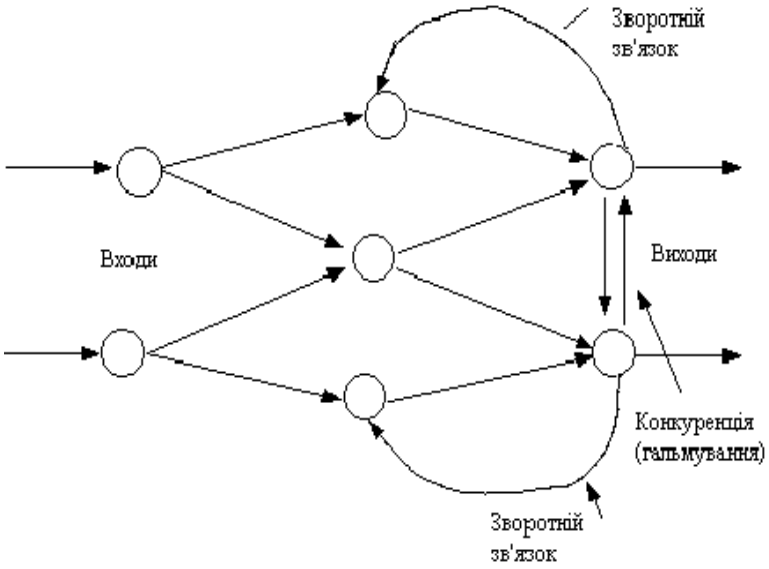

Рисунок 1.2 – Принцип взаємодії нейронів

Вид з'єднання між нейронами має великий вплив на роботу мережі. Більшість пакетів програмних реалізацій нейроних мереж дозволяють користувачеві додавати, віднімати і керувати з'єднаннями як завгодно. Постійно корегуємі параметри зв'язку можна зробити як збудливими так і гальмуючими. [6].

1.4 Генетичний алгоритм

Генетичний алгоритм – це евристичний метод пошуку, натхненний біологічною еволюцією. Він використовується для вирішення задач оптимізації та моделювання шляхом ітеративного вдосконалення популяції кандидатів на рішення. Генетичні алгоритми є потужним інструментом для розв'язання різних комбінаторних задач і задач оптимізації та стали стандартним методом

інтелектуальних обчислень. Назва методу походить від його імітації процесу природного відбору.

Якщо нам потрібно знайти оптимальне рішення задачі за певним критерієм, де кожне рішення можна описати набором чисел або величин, генетичні алгоритми стають дуже корисними. Наприклад, для вибору сукупності параметрів ринку, що впливають на його динаміку, ці параметри можна розглядати як хромосоми, що визначають властивості індивіда – конкретного рішення задачі. Значення параметрів називаються генами. Пошук оптимального рішення нагадує еволюцію популяції індивідів, представлених наборами хромосом. В еволюції діють три основні механізми: відбір найсильніших (найкращих рішень), схрещування (створення нових рішень шляхом комбінації хромосом відібраних індивідів) і мутації (випадкові зміни генів для збереження різноманітності в популяції). У результаті цих процесів з покоління в покоління виробляється оптимальне рішення задачі.

Процес роботи генетичного алгоритму зазвичай включає такі етапи:

– ініціалізація – створення випадкової початкової популяції рішень;

– оцінка – оцінювання кожного рішення в популяції за допомогою функції придатності, яка визначає, наскільки добре рішення вирішує задачу;

– відбір – вибір кращих рішень для наступного покоління, зазвичай з вищою оцінкою придатності;

– схрещування – створення нових рішень шляхом обміну частинами між батьківськими рішеннями (кросовер;

– мутація – випадкові зміни деяких елементів рішення для збереження різноманітності в популяції;

– повторення – повторення процесів відбору, схрещування та мутації протягом кількох поколінь або до досягнення заданої умови зупинки;

– завершення – зупинка алгоритму, коли досягнуто критерій зупинки, наприклад, задану кількість поколінь або достатньо добре рішення.

Генетичні алгоритми ефективні для оптимізації великих, складних проблем, де простий перебір усіх можливих рішень недоцільний. Вони застосовуються в багатьох галузях, включаючи інженерію, економіку, біологію, комп'ютерні науки та інші [6].

2 РОЗРОБКА ВІРТУАЛЬНОГО МАКЕТУ ДЛЯ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ГЕНЕТИЧНОГО АЛГОРИТМУ

2.1 Навчання із підкріпленням

Машинка, яка керується генетичним алгоритмом, матиме назву агент. А середовище, в якому відбувається її переміщення разом із законами фізики, – оточенням. Саме в таких термінах формулюються завдання навчання з підкріпленням – сучасного наукового спрямування, яке використовується для навчання поведінки ігрових персонажів, управління динамічними об'єктами (машинами, літаками, кораблями тощо). Скрізь, де завдання можна подати у вигляді агента та його взаємодії з навколишнім середовищем, можна застосувати ідею навчання з підкріпленням.

Метою навчання агента є можливість формувати такі дії a, залежно від поточного стану середовища, щоб максимізувати сумарні (загальні) винагороди r, рис. 2.1.

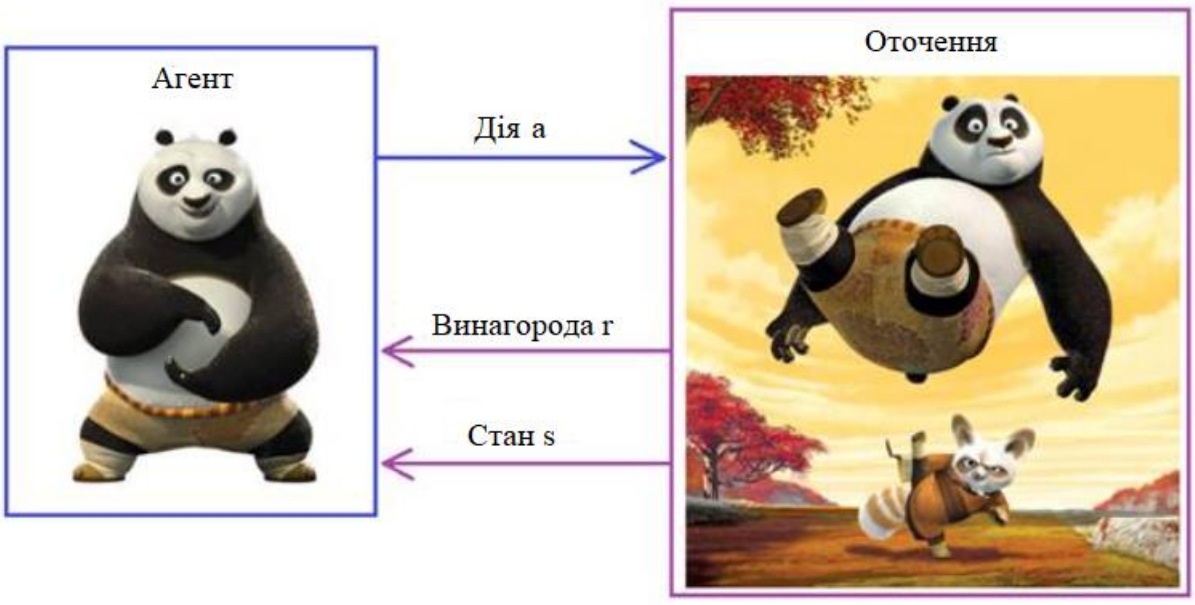

Рисунок 2.1 – Принцип роботи навчання з підкріпленням

2.2 Принцип роботи програми

2.2.1 Принцип дії генетичного алгоритму

Для того, щоб генетичний алгоритм виконав задачу, яка перед ним стоїть, йому треба отримати заповнений набір генів у хромосомах. Так само ми і розглядаємо задачу з машинкою, рис. 2.2.

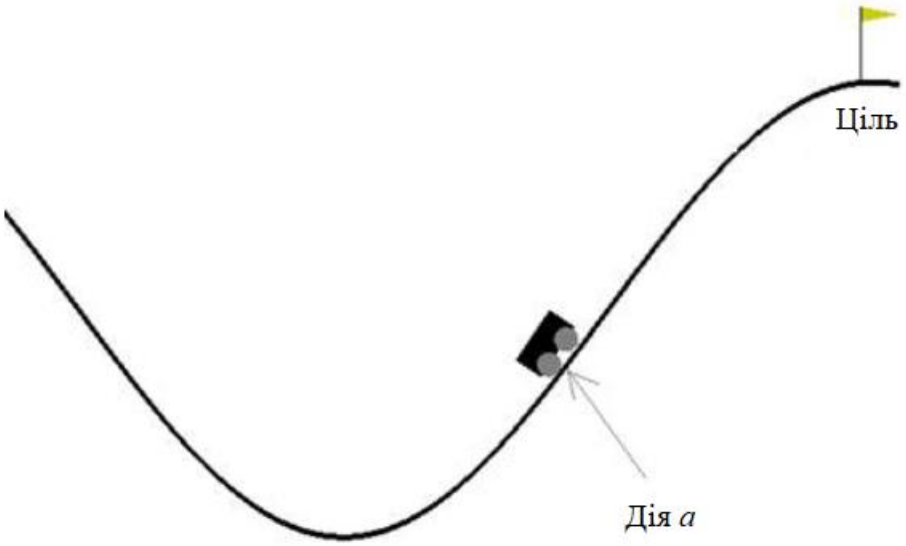

Рисунок 2.2 – Взаємодія агенту із середовищем

В рівні проміжки часу на агента надається один із видів впливу (дія):

– 0 – розгін вліво;
– 1 – рух по інерції;
– 2 – розгін вправо.

Якщо на поточній ітерації машинка не дісталася прапорця, то винагорода дорівнюватиме r = -1. Як тільки машинка дістається мети, алгоритм завершується і підраховується сумарна винагорода за формулою 2.1:

$$R = r_1 + r_2 + \cdots + r_N, \qquad (2.1)$$

де N – кількість виконанних кроків;

R – сумарна винагорода;

r – винагорода за окрему дію.

В рамках поставленного завдання обмежимося максимальним числом кроків Nmax = 200. Цього цілком достатньо, щоб машина забралася на правий пагорб. Відповідно, чим швидше (за меншу кількість ітерацій) агент дістанеться до мети, тим краща буде сумарна винагорода R. Фактично, ця величина і буде використовуватися як пристосованість індивіда в популяції.

2.2.2 Визначення пристосованості індивіда

Для того, щоб почати роботу із агентом і його навчанням взаємодії з оточенням, необхідно визначитися з методом кодування інформації в генах хромосом. Цілком очевидним рішенням тут є хромосома, що складається з 200 ген, і в кожному гені може бути записана одна з команд: 0, 1 або 2 (рис.2.3):

Рисунок 2.3 – Приклад вигляду хромосоми

Далі на основі цієї хромосоми потрібно визначати її пристосованість. Зробити це можна досить легко. Для кожного індивідуума ми будемо запускати віртуальне оточення (без візуалізації) і підраховувати кількість кроків (actionCounter), за яку було досягнуто мети (машинка доїхала до прапорця). Потім обчислювати значення пристосованості за формулою:

score = 0 - (LENGTH_CHROM - actionCounter) / LENGTH_CHROM

В цій задачі береться негативне значення частки кроків і чим вона більша, тим більше пристосований індивід. Тобто генетичний алгоритм прагнутиме мінімізувати функцію пристосованості особин у популяції.

Але на самому початку цілком можливі ситуації, коли для жодної з хромосом машинка не добирається до прапорця за 200 кроків. Тоді значення пристосованості всіх індивідуумів дорівнюватимуть нулю. Щоб цього уникнути, було прописано таку умову:

```
if actionCounter < LENGTH_CHROM:
                    score = 0 - (LENGTH_CHROM - actionCounter) / LENGTH_CHROM else:
                    score = abs(observation[0] - FLAG_LOCATION)
```

Якщо кількість ітерацій actionCounter менше довжини хромосоми, то машинка досягла мети і обчислення відбуваються за раніше визначеною формулою. А інакше обчислюється відстань між машинкою та метою (прапором). Таким чином, якщо агент не досягає мети, то найкращою хромосомою буде вважатися та, для якої машинка виявляється найближче до мети.

Це лише приклад формування значення пристосованості індивіда. Формула для обчислення пристосованості може відрізнятися і це залежить від розробника. Це елемент творчого процесу: визначення пристосувань особин залежно стану агента у навколишньому середовищі. Кожен може реалізувати це по-своєму. Головне, щоб значення функції зростало (або спадало) для кращих рішень.

2.2.3 Віртуальне оточення

Перш ніж реалізовувати сам алгоритм потрібно виконати імітацію віртуального оточення. У мові Python існує бібліотека OpenAI Gym із набором віртуальних оточень для вирішення завдань навчання із підкріпленням.

Саме за допомогою цієї бібліотеки було створено модуляцію руху машинки на холмах. Для підключення цього віртуального оточення було прописано:

```
env = gym.make("MountainCar-v0")
```

Після чого, для ініціалізації створеного оточення слід викликати метод reset():

```
observation = env.reset()
```

Цей метод повертає об'єкт observation (спостереження), що містить поточну інформацію про стан оточення. Яка саме інформація входить у цей об'єкт залежить від типу довкілля. У нашому випадку (для MountainCar-v0) цей об'єкт складатиметься з двох дійсних чисел: положення та поточної швидкості машинки.

Далі, для по крокової симуляції та адаптації агента в оточенні, було прописано метод step ().

```
observation, reward, done, info = env.step(action)
```

В цьому випадку метод step повертає чотири об’єкти:

– reward – винагорода за виконання команди (зазвичай, повертається значення -1 за кожну виконану команду);

– done – флаг (True – якщо ціль була досягнута; False – в іншому випадку);

– info – словник з інформацією для відлагодження;

– observation – поточну інформацію про оточення після виконання дії.

Візуалізацію поточного стану середовища для найкращого індивіду запускаємо командою: env.render().

2.2.4 Реалізація генетичного алгоритму на Python

Тепер реалізуємо генетичний алгоритм Python з використанням пакету DEAP. Спочатку, як завжди, імпортуємо необхідні бібліотеки:

```
from deap import base, algorithms
from deap import creator
from deap import tools
import algelitism
import random
import matplotlib.pyplot as plt
import numpy as np
import gym
```

Створюємо об'єкт віртуального оточення MountainCar-v0:

```
env = gym.make("MountainCar-v0")
```

І визначаємо необхідні параметри разом із об'єктом Залу слави та елемент генератора випадкових чисел:

```
LENGTH_CHROM = 200    # довжина хромосоми, підлеглої оптимізації
 # константи генетичного алгоритму
POPULATION_SIZE = 50   # кількість індивидів в популяції P_CROSSOVER
= 0.9       # вірогідність схрещення
P_MUTATION    =    0.2                    # вірогідність мутації індивіда
MAX_GENERATIONS    =    150          # максимальна кількість поколінь
HALL_OF_FAME_SIZE = 3
hof = tools.HallOfFame(HALL_OF_FAME_SIZE)
RANDOM_SEED = 42
```

random.seed(RANDOM_SEED)

Далі, два стандартних класи для опису індивідуума у популяції зі значенням weights -1 – для відшукання мінімуму функції пристосованості:

creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

Потім, реєструємо функції для генерації випадкових команд, створення індивіда і всієї популяції.

toolbox = base.Toolbox() toolbox.register("randomAction", random.randint, 0, 2)
toolbox.register("individualCreator", tools.initRepeat, creator.Individual, toolbox.randomAction, LENGTH_CHROM)
toolbox.register("populationCreator", tools.initRepeat, list,toolbox.individualCreator)
population = toolbox. populationCreator(n=POPULATION_SIZE)

Після цього оголосимо функцію для визначення пристосованості конкретного індивіду:

```
def getCarScore(individual):
FLAG_LOCATION = 0.5
observation = env.reset()
actionCounter = 0
for action in individual:
        actionCounter += 1
        observation, reward, done, info = env.step(action)
        if done:
                break
```

```
if actionCounter < LENGTH_CHROM:
        score = 0 - (LENGTH_CHROM - actionCounter) / LENGTH_CHROM else:
         score = abs(observation[0] - FLAG_LOCATION)
return score,
```

Все працює наступним чином: робиться цикл по командах у хромосомі, на кожній ітерації циклу збільшується лічильник actionCounter, поки не пройдемо по всій хромосомі або не буде досягнуто мети. Далі, обчислюємо значення пристосованості та повертаємо його у вигляді кортежу.

Нижче зареєструємо функції для роботи генетичного алгоритму у пакеті DEAP, а також об'єкт Statistics для збору статистики:

```
toolbox.register("evaluate", getCarScore)
toolbox.register("select", tools.selTournament, tournsize=2)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutUniformInt, low=0, up=2, indpb=1.0/LENGTH_CHROM)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("min", np.min)
stats.register("avg", np.mean)
```

Запускаємо генетичний алгоритм за допомогою модифікованої функції eaSimpleElitism():

```
population, logbook = algelitism.eaSimpleElitism(population, toolbox,
                                                 cxpb=P_CROSSOVER,
                                                      mutpb=P_MUTATION,
                                                      ngen=MAX_GENERATIONS,
                                                      halloffame=hof,
                                                      stats=stats,
```

```
                                                    verbose=True)
```

Відображаємо статистику разом із найкращою хромосомою:

```
maxFitnessValues, meanFitnessValues = logbook.select("min", "avg")
best = hof.items[0]
print(best)
plt.plot(maxFitnessValues, color='red')
plt.plot(meanFitnessValues, color='green')
plt.xlabel('Покоління')
plt.ylabel('Макс/середня пристосованість')
plt.title('Залежність максимальної и середньої пристосовоності від покоління')
plt.show()
```

Наприкінці запустимо візуалізацію руху машинки для кращого індивіду:

```
observation = env.reset()
for action in best:
    env.step(action)
    env.render()
env.close()
```

2.3 Дослідження ефективності роботи алгоритму

Проведемо перший дослід із такими вхідними даними:

```
LENGTH_CHROM = 120  # довжина хромосоми
P_CROSSOVER = 0.9       # вірогідність схрещення
P_MUTATION = 0.4        # вірогідність мутації індивіда
MAX_GENERATIONS = 120    # максимальна кількість поколінь
```

Результати досліду представлені в таблиці 2.1:

Таблиця 2.1 – Дані з консолі

Gen	Nevals	Min	Avg
90	44	0.15281	0.225638
91	44	0.146859	0.226897
92	43	0.146859	0.209586
93	43	0.14289	0.222847
94	44	0.134179	0.213361
95	46	0.134179	0.223072
96	44	0.120164	0.195466
97	45	0.113813	0.204039
98	47	0.101752	0.208533
99	44	0.101752	0.174922
100	46	0.101752	0.173604
101	46	0.094078	0.170149
102	44	0.0890326	0.16482
103	44	0.0647749	0.157794
104	46	0.0518049	0.162881
105	43	0.0518049	0.140467
106	42	0.0518049	0.144865
107	41	0.0453941	0.122038
108	42	0.0381117	0.105857
109	43	0.0381117	0.109449
110	44	0.0381117	0.101893
111	45	0.0381117	0.107069
112	44	0.0207756	0.102209
113	42	0.0207756	0.0932253
115	40	0.0205188	0.0682096
116	44	0.0047437	0.0748295
117	45	0.0047437	0.0724302
118	46	0.0047437	0.0630273
119	44	0.0041863	0.060694
120	46	0.00379875	0.0537434

Висновок: Індивід не знайшов рішення задачі. Параметри не оптимальні.

Проведемо другий дослід із такими вхідними даними:

LENGTH_CHROM = 150 # довжина хромосоми

POPULATION_SIZE = 50 # кількість індивидів в популяції

P_CROSSOVER = 0.7 # вірогідність схрещення

P_MUTATION = 0.6 # вірогідність мутації індивіда

MAX_GENERATIONS = 150 # максимальна кількість поколінь

Результати досліду представлені в таблиці 2.2:

Таблиця 2.2 – Дані з консолі

Gen	Nevals	Min	Avg
90	40	0.126563	0.221743
91	45	0.126563	0.207317
92	41	0.125196	0.197931
93	45	0.125196	0.202211
94	39	0.125196	0.195019
95	42	0.121582	0.194685
96	35	0.121582	0.19583
97	39	0.111461	0.186373
98	39	0.110315	0.188328
99	45	0.109379	0.186408
100	37	0.109379	0.164905
101	39	0.0665054	0.167646
102	42	0.0665054	0.158816
103	41	0.0665054	0.149297
104	41	0.0665054	0.142713
105	41	-0.00666667	0.12871
106	45	-0.00666667	0.123708
107	42	-0.00666667	0.104916
108	43	-0.00666667	0.0960496
109	44	-0.0133333	0.085802
110	44	-0.0133333	0.0801561
124	35	-0.0266667	0.00500797
125	40	-0.0266667	-0.00497338
126	37	-0.0266667	-0.0111733
127	39	-0.0266667	-0.00594288
128	36	-0.0266667	-0.0112301
150	41	-0.0266667	-0.00589278

Висновок: Індивід знайшов рішення задачі. Індивід адаптувався на 105 ітерації. Кінцева адаптація індивіду далека від ідеалу.

Проведемо третій дослід із такими вхідними даними:

LENGTH_CHROM = 170 # довжина хромосоми

POPULATION_SIZE = 50 # кількість індивидів в популяції

P_CROSSOVER = 0.4 # вірогідність схрещення

P_MUTATION = 0.8 # вірогідність мутації індивіда

MAX_GENERATIONS = 150 # максимальна кількість поколінь

Результати досліду представлені в таблиці 2.3:

Таблиця 2.3 – Дані з консолі

Gen	Nevals	Min	Avg
80	42	0.134871	0.209257
81	41	0.119301	0.201926
82	42	0.0995644	0.200007
83	39	0.0995644	0.188123
84	40	0.0543983	0.168524
85	41	0.0543983	0.161577
86	43	0.0543983	0.146396
87	40	0.0543983	0.159084
88	42	0.0521911	0.136601
89	43	0.0543983	0.140094
90	43	0.0396588	0.131375
91	42	0.0232977	0.132904
92	43	-0.00588235	0.113885
93	42	-0.00588235	0.0962995
94	38	-0.00588235	0.0880136
95	45	-0.0117647	0.0818633
96	41	-0.0117647	0.0719504
97	42	-0.0117647	0.0676885
98	38	-0.0176471	0.0602996
99	40	-0.0176471	0.0576169
100	42	-0.0176471	0.050541
145	43	-0.0411765	-0.00518804
146	42	-0.0411765	-0.0074543
147	36	-0.0411765	-0.00709925
148	40	-0.0411765	-0.00759931

Продовження таблиці 2.3

149	37	-0.0411765	-0.0133693
150	42	-0.0411765	-0.0102894

Висновок: Індивід знайшов рішення задачі. Індивід адаптувався на 92 ітерації. Кінцева адаптація індивіда є однією з найкращих можливих.

Проведемо четвертий дослід із такими вхідними даними:

LENGTH_CHROM = 200 # довжина хромосоми

POPULATION_SIZE = 50 # кількість індивидів в популяції

P_CROSSOVER = 0.9 # вірогідність схрещення

P_MUTATION = 0.9 # вірогідність мутації індивіда

MAX_GENERATIONS = 150 # максимальна кількість поколінь

Результати досліду представлені в таблиці 2.4:

Таблиця 2.4– Дані з консолі

Gen	Nevals	Min	Avg
60	46	0.206569	0.285262
61	47	0.195702	0.276689
62	46	0.17907	0.265139
63	47	0.17907	0.259581
64	47	0.143026	0.2436
65	47	0.143026	0.251632
66	47	0.139429	0.246276
67	47	0.139429	0.262531
68	46	0.139429	0.225814
69	46	0.0838899	0.222948
70	47	0.0838899	0.225704
71	47	0.0838899	0.228315
72	47	0.0838899	0.203584
73	47	0.0838899	0.209474
74	46	0.0838899	0.208081
75	47	0.0671878	0.181353
76	46	0.0671878	0.17687
77	47	0.0671878	0.172127
78	47	-0.005	0.177533
79	47	-0.005	0.157746

Продовження таблиці 2.4

80	47	-0.005	0.142005
145	47	-0.035	-0.0191503
146	47	-0.035	-0.0153386
147	47	-0.035	-0.00665154
148	45	-0.035	-0.0165889
149	46	-0.035	-0.0130038
150	46	-0.035	-0.0129072

Висновок: Індивід знайшов рішення задачі. Індивід адаптувався на 78 ітерації, що є найшвидшим результатом. Кінцева адаптація індивіда є гарною, але не ідеальною. Такі вхідні дані є найоптимальнішими, але є небезпека втрати корисних хромосом.

2.4 Результати роботи програми

Результатом роботи програми є статистика у вигляді графіку залежності максимальної та середньої пристосованості від покоління рис.2.4:

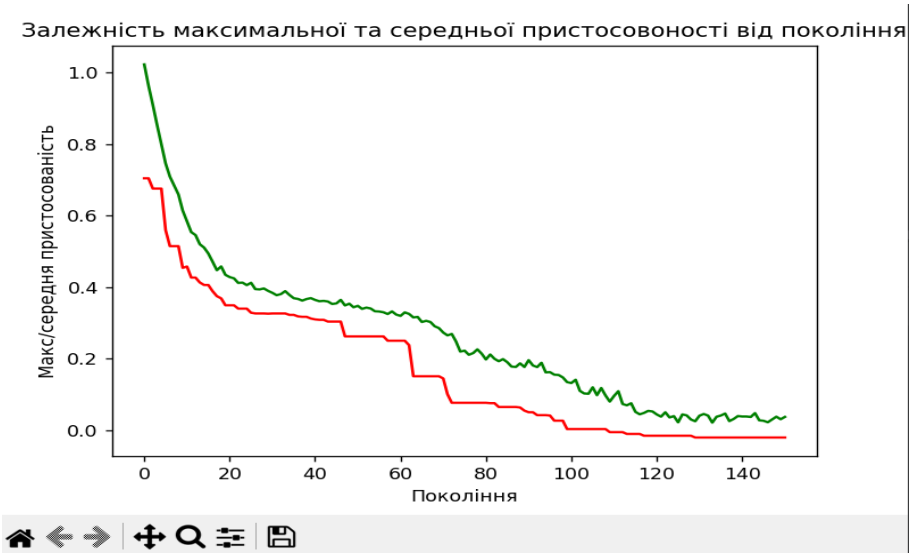

Рисунок 2.4 – Графік залежності максимальної та середньої пристосованості від покоління

На основі отриманого графіку можна зробити висновок, що чим більше покоління, тим краще пристосованість агента до оточення. А вже після 109

покоління агент повністю пристосувався до задачі і далі лише оптимізує свої дії, для отримання найшвидшого і найоптимальнішого варіанту виконання своєї місії.

Також результатом роботи програми є візуалізація найкращого індивіду з найоптимальнішим рішенням задачі, яке він знайшов рис. 2.5- рис.2.8:

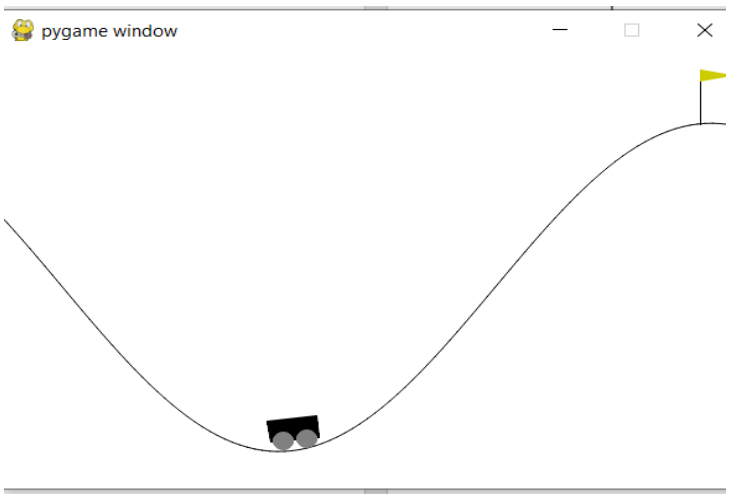

Рисунок 2.5 – Початок виконання задачі

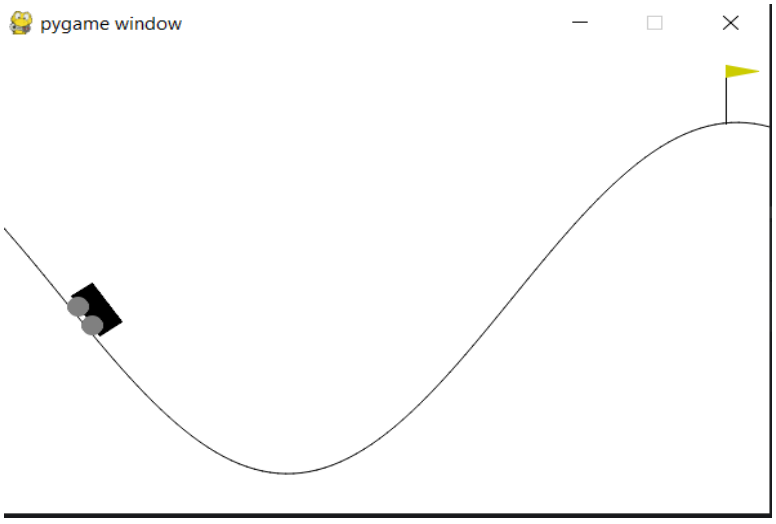

Рисунок 2.6 – Проміжний результат виконання задачі

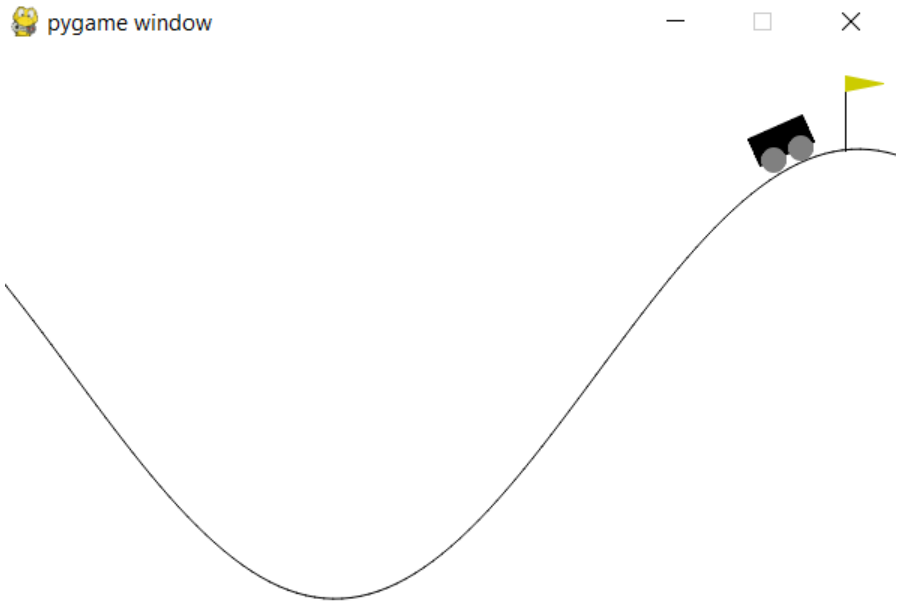

Рисунок 2.7 – Проміжний результат виконання програми

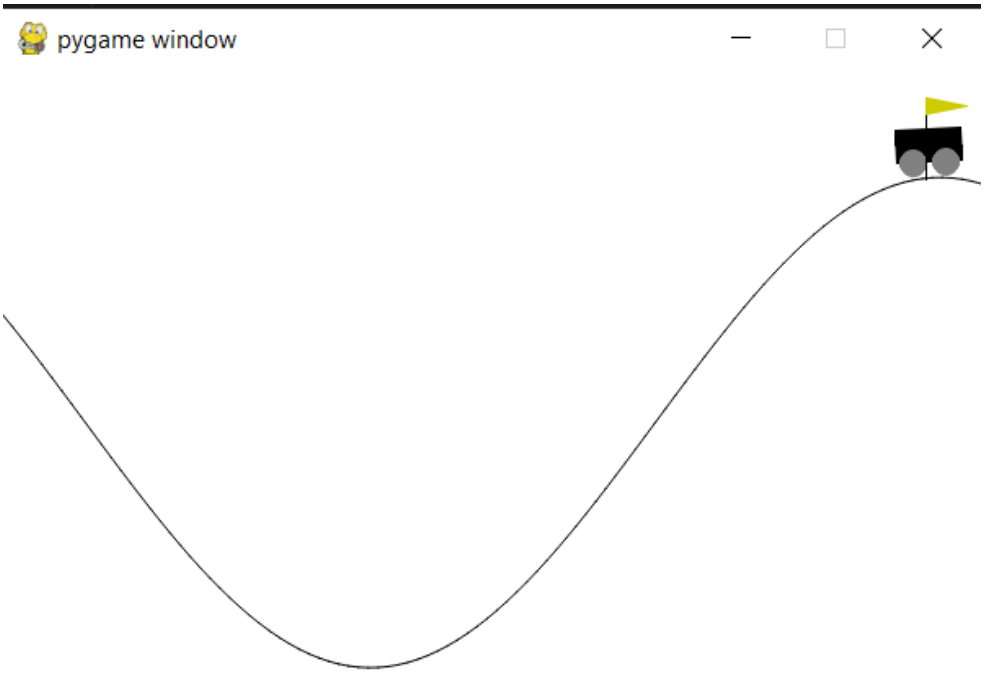

Рисунок 2.8 – Фінальний результат виконання програми

3 РОЗРОБКА ФІЗИЧНОГО МАКЕТУ РОБОТА ТА СИСТЕМИ УПРАВЛІННЯ МОБІЛЬНОЮ ПЛАТФОРМОЮ ЗА ДОПОМОГОЮ ГА

3.1 Вибір середовища розробки

Arduino IDE (Integrated Development Environment) є офіційним середовищем розробки для програмування мікроконтролерів Arduino. Воно забезпечує простий і інтуїтивно зрозумілий інтерфейс, що дозволяє швидко почати працювати з апаратним забезпеченням Arduino. Основні особливості Arduino IDE включають:

– інтерфейс і процес розробки розраховані на початківців, з інтуїтивно зрозумілими елементами керування та мінімальними налаштуваннями;

– підтримка програмування, завантаження коду та моніторингу серійного порту для різних плат Arduino;

– велика кількість вбудованих бібліотек і прикладів коду, які полегшують розробку проектів;

– Arduino IDE підтримує Windows, MacOS та Linux, що робить його доступним для широкого кола користувачів.

PyCharm є потужним середовищем розробки для мови програмування Python, розробленим компанією JetBrains. Воно забезпечує широкий спектр інструментів для професійної розробки програмного забезпечення. Основні особливості PyCharm включають:

– інтелектуальний автозавершення, аналіз коду в реальному часі, підсвічування синтаксису та багато іншого;

– підтримка Git, SVN, Mercurial та інших систем контролю версій;

– можливість додавання плагінів для розширення функціональності середовища;

– вбудовані інструменти для написання та запуску тестів, зокрема підтримка unittest, pytest, nose;

– потужні інструменти для налагодження та аналізу продуктивності коду.

Обидва середовища мають свої переваги в залежності від конкретних потреб проекту і в випадку виконання поставленного завдання до проекту є ідеальними варіантами для обох макетів. Arduino IDE є ідеальним вибором для проектів, пов’язаних з мікроконтролерами та апаратним забезпеченням, тоді як PyCharm підходить для комплексної розробки програмного забезпечення на Python.

3.2 Вибір апаратного забезпечення

Для збірки фізичного макету прешочергово було створено схему, за якою збиралася мобільна платформа, див. рис. 3.1.

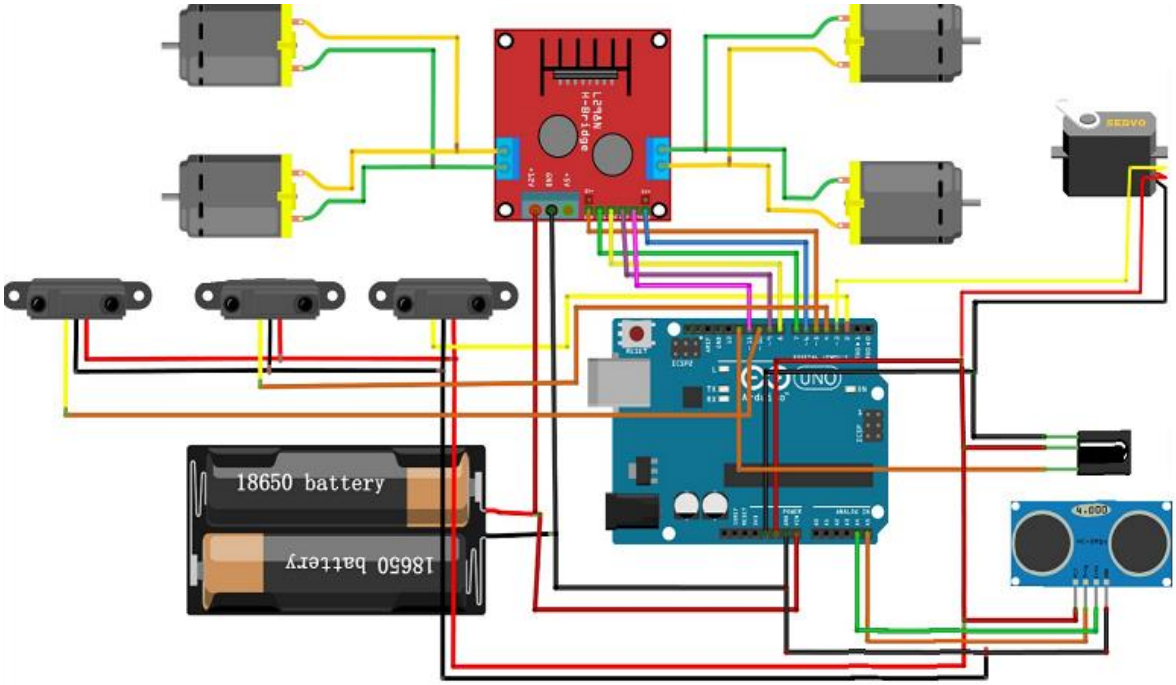

Рисунок 3.1 – Схема мобільної платформи

Перед розробкою схеми і збіркою фізичнох моделі було детально переглянуто існуючи датчики та контролери, необхідні для вирішення

поставленної задачі. Після чого було обрано наступні комплектуючі: Arduino UNO R3 board, DC motor, SG90 servo, HCSR04, L298N, Line Tracking Module.

Детальніше про Arduino UNO R3 board: Arduino UNO R3 є однією з найпопулярніших і широко використовуваних плат мікроконтролерів у світі. Вона призначена для початківців, а також для досвідчених розробників, які створюють прототипи електронних пристроїв. Плата базується на мікроконтролері ATmega328P від Atmel і пропонує широкі можливості для розробки завдяки своїй простоті та функціональності (рис 3.2).

Основні характеристики:

а) мікроконтролер: ATmega328P;

б) тактова частота: 16 МГц;

в) пам'ять: Flash 32 KB (з них 0.5 KB використовується для завантажувача) ;

г) SRAM: 2 KB;

ґ) EEPROM: 1 KB;

д) вхідні/вихідні порти (I/O): ;

- цифрові: 14 (з яких 6 можуть бути використані як PWM-виходи) ;
- аналогові: 6;

е) напруга живлення:

- робоча напруга: 5 В;
- вхідна напруга (рекомендована): 7-12 В;
- вхідна напруга (межі): 6-20 В;

є) максимальний струм:

- для I/O пінів: 40 мА на пін;
- для 3.3V піну: 50 мА;

ж) інтерфейси: UART, I2C, SPI;

з) роз'єми: USB, живлення (барел-джек і піновий роз'єм), ICSP заголовок, роз'єм для розширення.

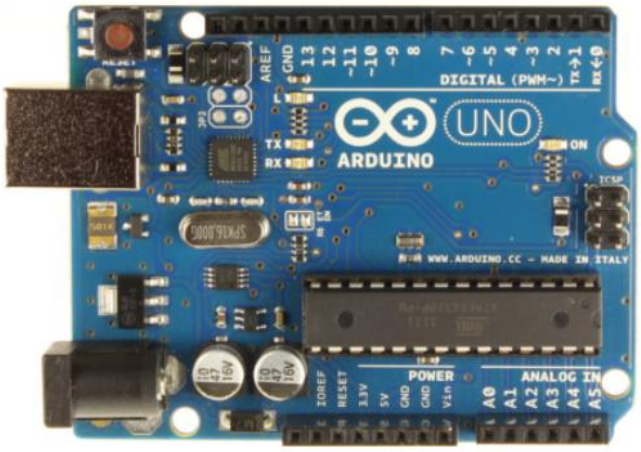

Рисунок 3.2 – Arduino UNO R3 board.[10].

Після того, як було обрано платформу для програмування та контролю за роботом переходимо до рушійниих можливостей робота. Тут в силу вступають мотори та драйвер до них, а саме L298N та DC motors.

Детальніше про рухову чатину. L298N – це популярний двоканальний драйвер для керування двигунами, який дозволяє керувати напрямком і швидкістю обертання двох постійних двигунів або одного двофазного крокового двигуна. Він часто використовується в проектах з робототехніки та автоматизації завдяки своїй надійності та простоті використання (рис 3.3).

Основні характеристики:

– мікросхема: L298N;

– кількість каналів: 2 (може керувати двома двигунами одночасно) ;

– робоча напруга: 5В - 35В;

– максимальний струм: 2А на канал (піковий струм до 3А) ;

– логічна напруга: 5В;

– вбудований стабілізатор: Може забезпечити 5В вихід для живлення логічних схем;

– розміри модуля: Зазвичай близько 43 мм x 43 мм x 27 мм (можуть бути варіації в залежності від виробника).

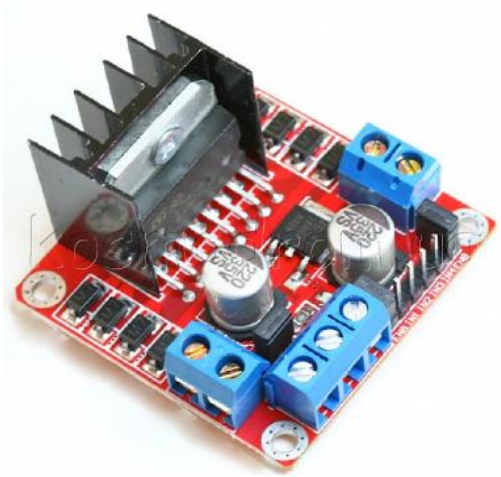

Рисунок 3.3 – драйвер двигунів L298N. [17].

Розібравшись з руховою частиною переходимо до апаратної частини, яка дозволяє керувати рухами робота, а також за допомогою якої отримуємо дані для створення популяції хромосомів і подальшої їх мутації. Такий датчик як HCSR04 найліпше доможе у виконанні цієї задачі. Ультразвуковий датчик HCSR04 використовує сонар для визначення відстані до об'єкта, подібно до того, як це роблять кажани або дельфіни. Він забезпечує відмінне безконтактне виявлення з високою точністю та стабільними показниками у простому для використання корпусі. Датчик здатен вимірювати відстані від 2 см до 400 см (або від 1" до 13 футів). Його робота не залежить від сонячного світла або чорних матеріалів, як у випадку з дальномірами Sharp (хоча акустично м'які матеріали, такі як тканина, можуть бути важкими для виявлення). Датчик комплектується ультразвуковим передавачем та приймачем (рис 3.4).

Характеристики:

– живлення: +5V DC;
– струм у стані спокою: <2mA;
– робочий струм: 15mA;
– ефективний кут: <15°;
– діапазон вимірювань: 2 см – 400 см (1" – 13 футів) ;
– роздільна здатність: 0.3 см;
– кут вимірювання: 30°;

– ширина вхідного імпульсу тригера: 10 мкс;
– розміри: 45 мм x 20 мм x 15 мм.

Рисунок 3.4 – Ультразвуковий датчик HCSR04

Також для більшого радіусу охвату середовища було додано SG90 servo додатково до HCSR04. Таким чином простор, який охоплюється цим датчиком стає більшим, бо він починає обертатися у різні сторони. Сервопривод SG90 є одним з найпопулярніших і широко використовуваних міні-сервоприводів завдяки своїй компактності, легкості та надійності. Цей сервопривод ідеально підходить для різноманітних проектів у робототехніці, моделюванні, автоматизації та DIY-проектах (рис. 3.5).

Основні характеристики:

– робоча напруга: 4.8V - 6V;
– максимальний крутний момент: 1.8 кг·см при 4.8V;
– швидкість: 0.1 с/60° при 4.8V;
– розміри: 22.2 мм x 11.8 мм x 31 мм;
– вага: 9 г;
– кут повороту: 180° (90° в кожну сторону від середнього положення) ;
– тип двигуна: 3-полюсний;
– шестерні: Пластикові;
– інтерфейс керування: PWM (імпульсно-широтна модуляція) ;
– довжина кабелю: Приблизно 25 см;

– конектор: 3-контактний (VCC, GND, сигнал).

Рисунок 3.5 – Сервопривод SG90

Також варто згадати додаткову функцію мобільної платформи, це можливість слідкувати і їхати по лінії. Line Tracking Module використовується для виявлення чорної лінії на світлій поверхні або світлої лінії на темній поверхні. Це робить його ідеальним для використання в робототехніці, особливо в проектах, де робот повинен слідувати певному маршруту (рис 3.6).

Основні характеристики:

– джерело живлення: 3.3V - 5V;

– тип сенсора: Інфрачервоний (IR) ;

– вихідний сигнал: Цифровий (HIGH або LOW) ;

– діапазон виявлення: Зазвичай 1 см - 2 см;

– налаштування чутливості: За допомогою потенціометра;

– індикація: Світлодіод (LED) для візуального підтвердження виявлення лінії;

– розміри: Зазвичай близько 3 см x 1 см x 1 см (можуть бути варіації в залежності від виробника).

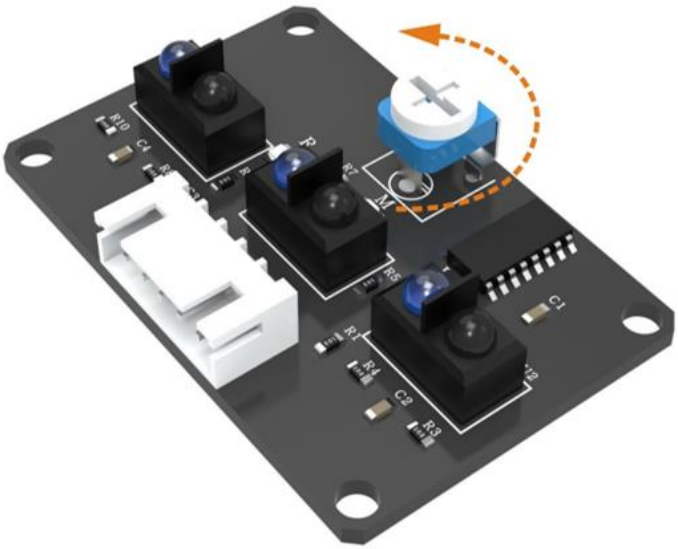

Рисунок 3.6 – Модуль стеження за лінією

Компонент, на який вказано, – це потенціометр. Він може налаштовувати чутливість модуля стеження за лінією шляхом зміни його опору.

Необхідно поставити автомобіль на трасу. Якщо модуль стеження за лінією може відрізнити землю (світлодіод не світиться) і чорну лінію (світлодіод світиться), його можна використовувати нормально. В іншому випадку потрібно налаштувати потенціометр, поки не буде досягнуто цей ефект. [7].

Після розгляду всіх комплектуючих та створення схеми мобільної платформи і маєму наступну структуру, за якою збирається робот, див. рис 3.7 та рис.3.8.

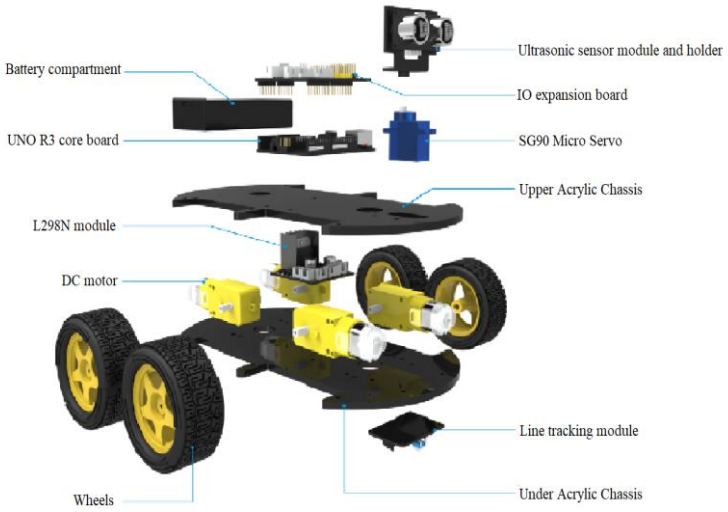

Рисунок 3.7 – Структура мобільної платформи

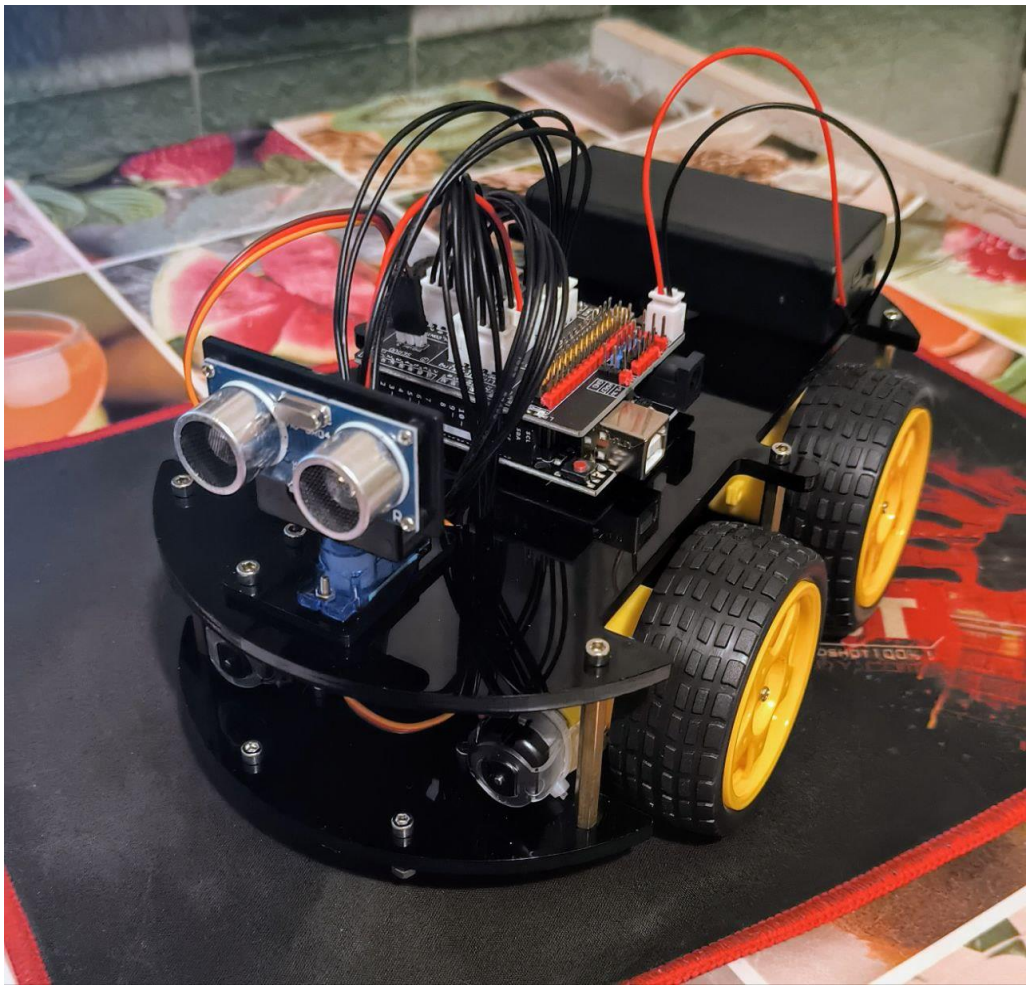

Рисунок 3.8 – Результат збірки макету

3.3 Теорія автоматичного управління для мобільної платформи

Структурна схема, рис.3.9, включає блоки $W_0(s)$, W_1(s), W_2(s), W_3(s), і W_4(s). Зазначимо, що:

– u(t) – вхідний сигнал;

– h(t) – вихідний сигнал;

– y_0, y_1, y_2, y_3, y_4 – проміжні сигнали;

– W0(s): Передавальна функція PID-контролера;

– W1(s): Передавальна функція ультразвукового датчику;

– W2(s): Передавальна функція драйверу моторів (на базі L298);

– W3(s): Передавальна функція моторів;

– W4(s): Передавальна функція елементу зворотного зв'язку.

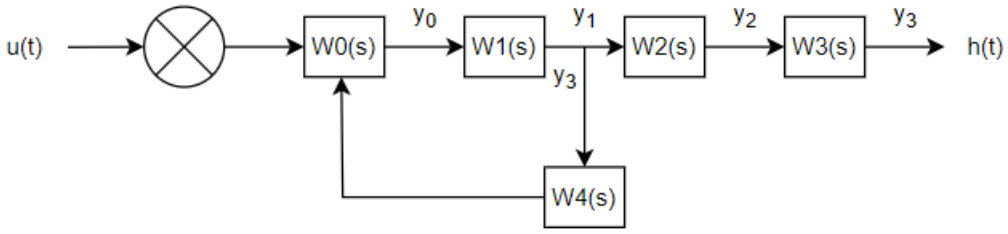

Рисунок 3.9 – Структурна схема мобільної платформи

На структурній схемі видно принцип взаємодії модулів між собою, а також як вхідний сигнал, проходячи через систему стає вихідним. Ця схема дозволяє візуалізувати та легко зрозуміти принципе роботи всієї мобільної платформи.

Враховуючи зворотній зв’язок і послідовних з'єднанб блоків, загальну передавальну функцію H(s) можна знайти наступним шляхом:

Передавальна функція відкритого контуру H(s):

$$H(s) = W_0(s) \cdot W_1(s) \cdot W_2(s) \cdot W_3(s), \quad (3.1)$$

Передавальна функція зворотного зв’язку Hfb(s):

$$H_{fb}(s) = W_4(s), \quad (3.2)$$

Загальна передавальна функція системи зворотного зв’язку:

$$T(s) = \frac{H(s)}{1+H(s)\cdot H_{fb}(s)}, \quad (3.3)$$

Розгляд передавальних функції:

– блок W_0(s):

$$W_0(s) = K_p + \frac{K_i}{s} + K_d s, \quad (3.4)$$

– блок W_1(s):

$$W_1(s) = \frac{1}{1+\tau_1 s}, \tag{3.5}$$

– блок W_2(s):

$$W_2(s) = \frac{1}{1+\tau_2 s}, \tag{3.6}$$

– блок W_3(s):

$$W_3(s) = \frac{1}{1+\tau_3 s}, \tag{3.7}$$

– блок W_4(s):

$$W_4(s) = K_f, \tag{3.8}$$

Тепер підставимо ці значення в загальні формули і отримаємо:
Передавальна функція відкритого контуру H(s):

$$H(s) = (K_p + \frac{K_i}{s} + K_d s) \cdot \frac{1}{1+\tau_1 s} \cdot \frac{1}{1+\tau_2 s} \cdot \frac{1}{1+\tau_3 s}, \tag{3.9}$$

Передавальна функція зворотного зв’язку Hfb(s):

$$H_{fb}(s) = K_f, \tag{3.10}$$

Загальна передавальна функція системи зворотного зв’язку:

$$T(s) = \frac{H(s)}{1+H(s)\cdot H_{fb}(s)}, \tag{3.11}$$

На рисунку 3.10 представлений графік АЧХ, який показує як змінюється амплітуда вихідного сигналу залежно від частоти.

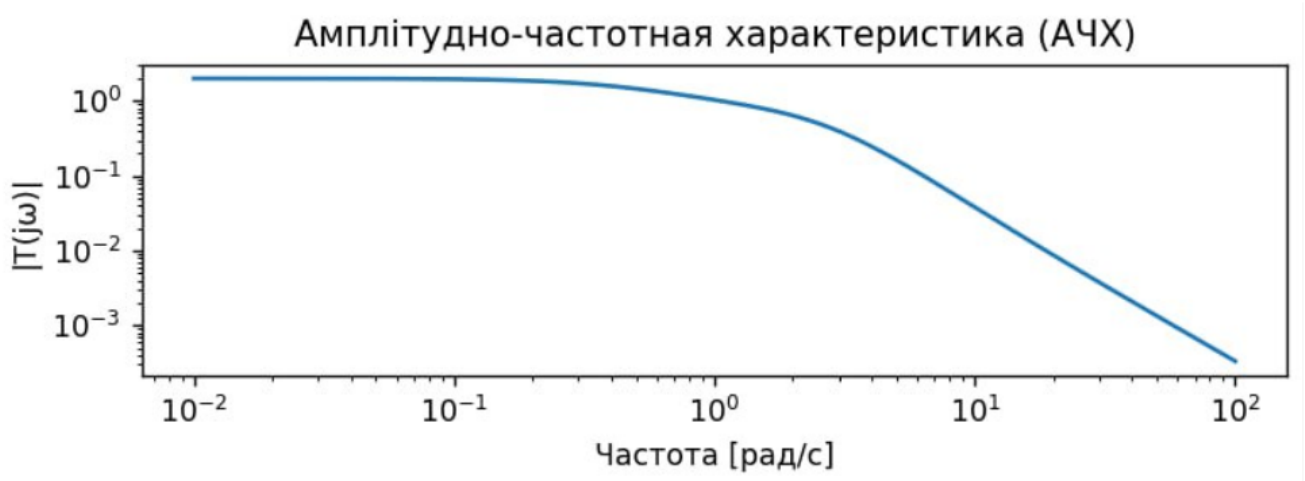

Рисунок 3.10 – Графік амплитудно-частотної характеристики

На рисунку 3.11 представлений графік ЛФЧХ, що демонструє зміну фази вихідного сигналу, залежно від частоти вхідного сигналу, що є важливим для аналізу фазової стійкості. [16].

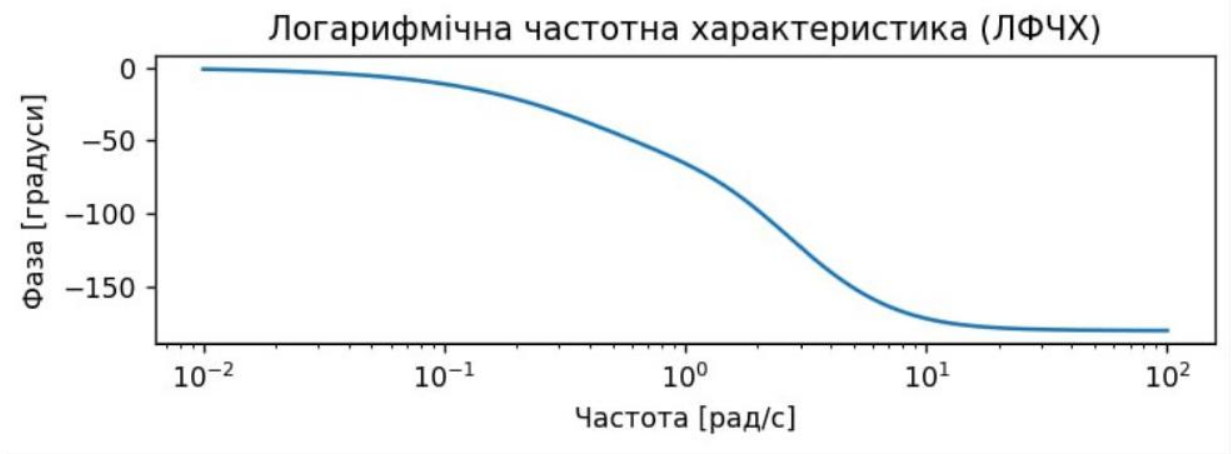

Рисунок 3.11 – Графік логарифмічної амплитудно-частотної характеристики

Ці графіки дають змогу в подальшому аналізувати та оптимізовувати систему, враховуючі отриманні дані і маючи уявлення про властивості системи.

3.4 Розробка програми керування мобільною платформою

Під час реалізації програмної частини системи керування було прийнято рішення розробити два варіанти рішення цієї задачі. Для порівняння і аналізу ефективності роботи мобільної платформи з ГА та без нього.

Традиційний варіант – це керування без використання ГА. У цьому варіанті мобільна платформа буде керуватися за допомогою традиційних методів, таких як уникнення перешкод використовуючи датчик ультразвуку та на основі заздалегідь визначених правил і параметрів. Цей підхід забезпечує стабільну та передбачувану поведінку платформи, але не враховує можливості адаптації до змінних умов середовища, що потребує постійних зупинок та аналізу середовища, що значно збільшує час досягення мети.

Інший варіант – це керування з використанням ГА. Цей варіант передбачає використання генетичного алгоритму для оптимізації параметрів керування мобільною платформою. ГА дозволяє системі навчатися та адаптуватися до різних умов середовища, знаходячи оптимальні стратегії руху. Це може включати налаштування швидкості, кутів повороту та інших параметрів на основі даних, зібраних під час роботи платформи, але також це включає в себе потребу проаналізувати середовище для створення першого покоління хромосом, що трохи затримую робота на початку виконання задачі.

Передбачається, що використання генетичного алгоритму покращить адаптивність та ефективність роботи мобільної платформи в порівнянні з традиційним підходом. Однак, ГА може вимагати більше обчислювальних ресурсів та часу на початковому етапі навчання.

Далі буде проведенно розгляд і порівняння двох програм. В процесі буде представлено детальний опис кожного фрагменту коду.

Розгляд роботи програми без генетичного алгоритму:

Почнемо розгляд з рухових функцій мобільної платформи:

Рух вперед:

```
void forward(bool debug = false) {
analogWrite(ENA, carSpeed);
analogWrite(ENB, carSpeed);
 digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
digitalWrite(IN3, LOW);
digitalWrite(IN4, HIGH);
 if (debug) Serial.println("Go forward!"); }
```

Коли ця функція викликається, то реалізовує рух вперед з визначенною швидкістю.

Рух назад:

```
void back(bool debug = false) {
analogWrite(ENA, carSpeed);
analogWrite(ENB, carSpeed);
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
if (debug) Serial.println("Go back!"); }
```

Коли ця функція викликається, то реалізовує рух назад з визначенною швидкістю.

Рух ліворуч:

```
void left(bool debug = false) {
analogWrite(ENA, carSpeed);
analogWrite(ENB, carSpeed);
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 digitalWrite(IN3, LOW);
digitalWrite(IN4, HIGH);
 if (debug) Serial.println("Go left!"); }
```

Коли ця функція викликається, то реалізовує рух вліво з визначенною швидкістю.

Рух праворуч:

```
void right(bool debug = false) {
analogWrite(ENA, carSpeed);
analogWrite(ENB, carSpeed);
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW); if (debug) Serial.println("Go right!"); }
```

Коли ця функція викликається, то реалізовує рух вправо з визначенною швидкістю.

Функція стоп:

```
void stop(bool debug = false) {
digitalWrite(ENA, LOW);
digitalWrite(ENB, LOW);
if (debug) Serial.println("Stop!"); }
```

Коли ця функція викликається, то робот зупиняється. Слід зазначити, що всі ці функції викликаються після якогось тригеру під час виявлення перешкод. Ці перешкоди і їх положення і є умовами для виклику функцій.

Функція для виявлення прешкод та розрахунок відстані до них:

```
int getDistance() {
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
return (int)
pulseIn(ECHO_PIN, HIGH) / 58; }
```

У цій функції проводиться аналіз середовища та виявлення за допомогою ультразвукового датчику.

Наступна функція повністю відображає весь основний алгоритм уникнення перешкод. В цьому фрагменті коду описано покроково аналіз роботом середовища навколо із подальшим викликом функцій руху.

Функція уникнення перешкод:

```
void obstacles_avoidance_mode() {
if (func_mode == ObstaclesAvoidance) {
servo.write(90); delays(500);
middleDistance = getDistance();
if (middleDistance <= 40) {
stop(); delays(500);
servo.write(10);
delays(500);
rightDistance = getDistance();
```

```
delays(500);
 servo.write(90);
 delays(500); servo.write(170);
 delays(500);
leftDistance = getDistance();
 delays(500);
 servo.write(90);
 delays(500);
 if (rightDistance > leftDistance) {
stop();
delay(100);
 back();
delay(200);
 right();
delay(300);
 } else if (rightDistance < leftDistance) {
stop();
delay(100);
back();
 delay(200);
left();
delay(300);
} else if ((rightDistance <= 40) || (leftDistance <= 40)) {
back();
delays(180);
} else {
forward(); }
} else {
forward(); } } }
```

Цей цикл являє собою основний прицнип роботи традиційного способу управління роботом.

Після розгляду традиційного методуповертаємося до методу, який використовує ГА, та розпишемо його основні функції і принцип їх роботи.

Функція генерації початкової популяції:

```
void initializePopulation() {
for (int i = 0; i < populationSize; i++) {
for (int j = 0; j < genomeLength; j++) {
population[i][j] = random(50, 255); } } }
```

У цьому фрагменті коду наглядно зображено приклад генерації початкової, але в данному коді в якості даних стоїть ронадомна генерація, яка під час реалізації проекту замінюється на значення швидкості, та параметри, отримані з ультразвукового датчику.

Функція вимірювання продуктивності:

```
float measurePerformance(int genome[ ]) {
float performance = 0.0;
for (int i = 0; i < genomeLength; i++) {
forwardWithSpeed(genome[i]);
 performance += getDistance();
delay(genome[i]); }
return performance; }
```

У цьому фрагменті коду реалізований першоетапний аналіз ефективності генів. Після чого починається взаємодія безпосередньо з генами, їх мутація та подальше схрещення.

Функція мутації:

```
void mutate(int genome[]) {
for (int i = 0; i < genomeLength; i++)
{ if (random(10) < 1) {
random(50, 255); } } }
```

Тут відображено структуру роботи процесу мутації гену. Це необхідно для формування більш різноманітної популяціїі подальшого покращення наступних поколінь шляхом наближення різномаїття даних до мінливого середовища.

Функція схрещення:

```
void crossover(int parent1[], int parent2[], int child[]) {
for (int i = 0; i < genomeLength; i++) {
if (random(2) == 0) {
child[i] = parent1[i];
} else {
child[i] = parent2[i]; } } }
```

У цій функції відображено вибір випадкового гена батьків для генерації нового покоління генів. Нове покоління генів буде більш пристасованним до середовища і кожне нове покоління буде більше наближати індивід до унікального рішення, яке під час адаптації робота було найліпшим.

Функція турнірного відбору генів:

```
int selectParent() {
int best = random(populationSize);
for (int i = 0; i < 3; i++) {
int contender = random(populationSize);
if (fitness[contender] > fitness[best]) {
best = contender; }
```

```
}
 return best;
}
```

У цій фукнції відображено базові необхідні дії, які відбуваються з генами після їх мутації та схрещення. Саме ця функція забезпечує еволюцію нових поколінь замість їх деградації.

Функція ГА:

```
void geneticAlgorithm() {
initializePopulation();
for (int generation = 0;
generation < 100;
 generation++) {
 evaluateFitness();
int newPopulation[populationSize][genomeLength];
 for (int i = 0; i < populationSize / 2; i++) { i
nt parent1Index = selectParent();
 int parent2Index = selectParent();
crossover(population[parent1Index], population[parent2Index],
newPopulation[i]); mutate(newPopulation[i]); }
for (int i = 0; i < populationSize; i++) {
for (int j = 0; j < genomeLength; j++) {
population[i][j] = newPopulation[i][j]; }
} } }
```

У цьому коді сконцетровано основний принцип роботи ГА для мобільної платформи, а саме відбір 50% найліпших генів, які покажуть себе найліпше і заміняє попередні покоління на нові для повторення операцій із мутаціями та схрещеннями. Цей алгоритм можна покращити шляхом «відбору із відбросу»,

мається на увазі те, що можна залишити 25-50% з тих генів, які були викинуті під час відбору. Це треба для того, щоб потенційно не втратити можливі найліпші та найдаптованіші гени під час схрещення, але також слід зазначити що це потребує більших обчислювальних ресурсів і займе більше часу на знайдення рішення індивідом

3.5 Проведення експеременті в

У цьому підрозділі буде відображено експерименти керування платформою за домогою традиційних методів управління та за допомогою ГА, вмонтованого в програму керування. Далі буде відображено фотозвіт проходження лінііїї перешкод на рис.3.12- 3.19.

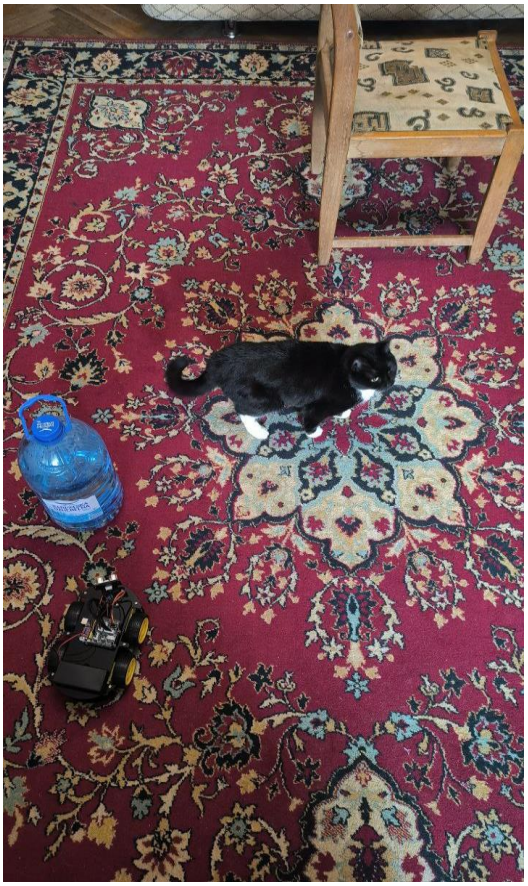

Рисунок 3.12 – Лінія перешкод

Стартова позиція робота, це білий візерунок знизу, а кінець шляху біля бильці дивану зверху фоторграфії на рисунку 3.13

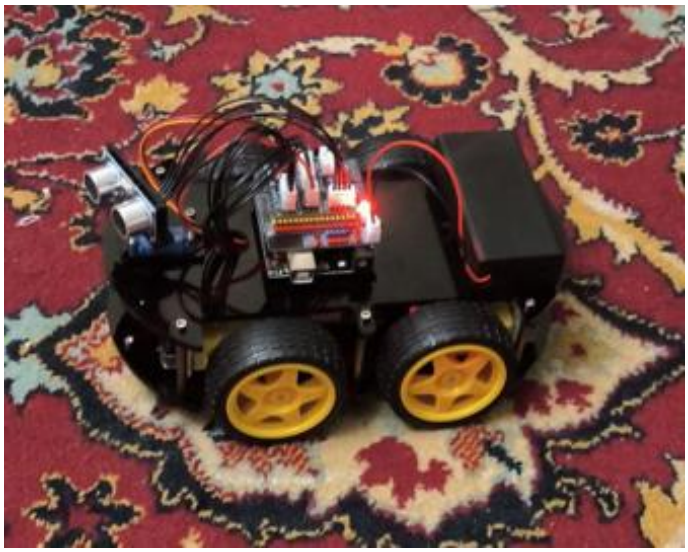

Рисунок 3.13 – Старт роботи програми

Далі робота програми продовжується до досягення мети з поетапним проходженням перешкод. Деякі перешкоди є мобільними та не постійними, як, наприклад, кот, який рухається і не стоїть статично на місці. Під час проходження маршруту проблем майже не виникло в обох випадках, але в випадку без ГА розвороти і реакція на перешкоди було довшими і іноді реакція була не своєчасною.

Рисунок 3.14 – Платформа відреагувала на мобільну перешкоду (код з ГА)

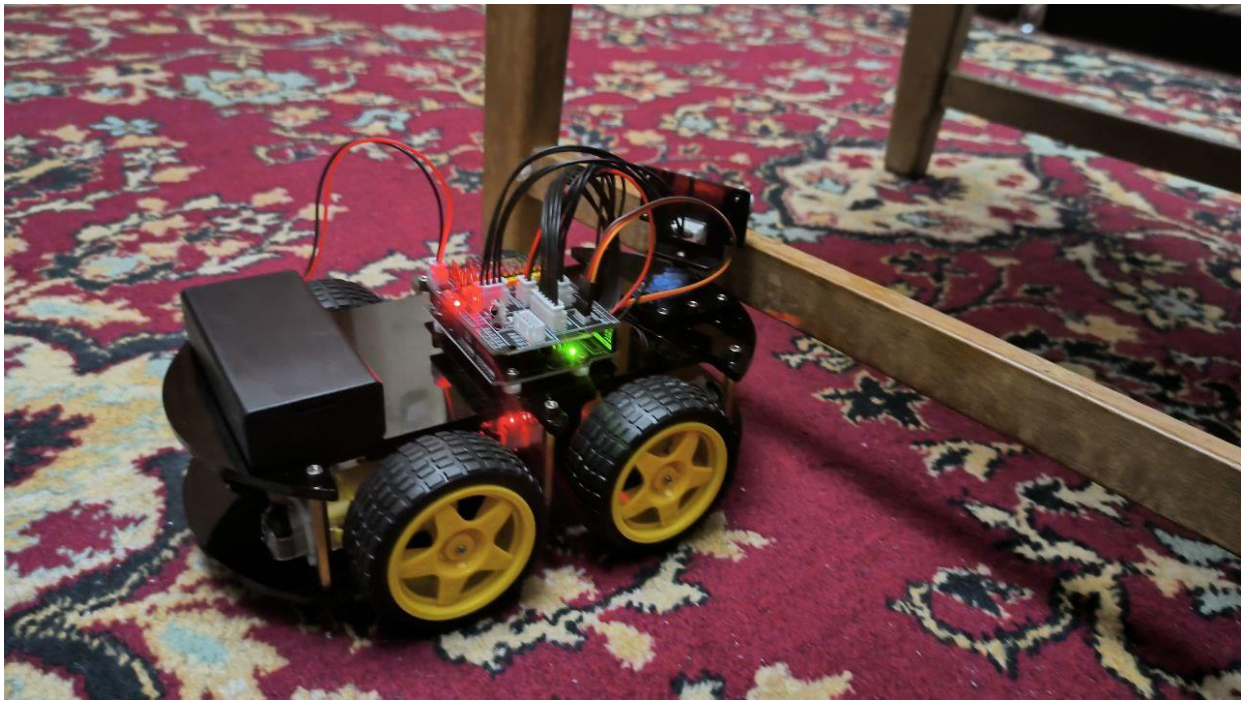

Рисунок 3.15 – Платформа не встигла відреагувати на перешкоду (код без ГА)

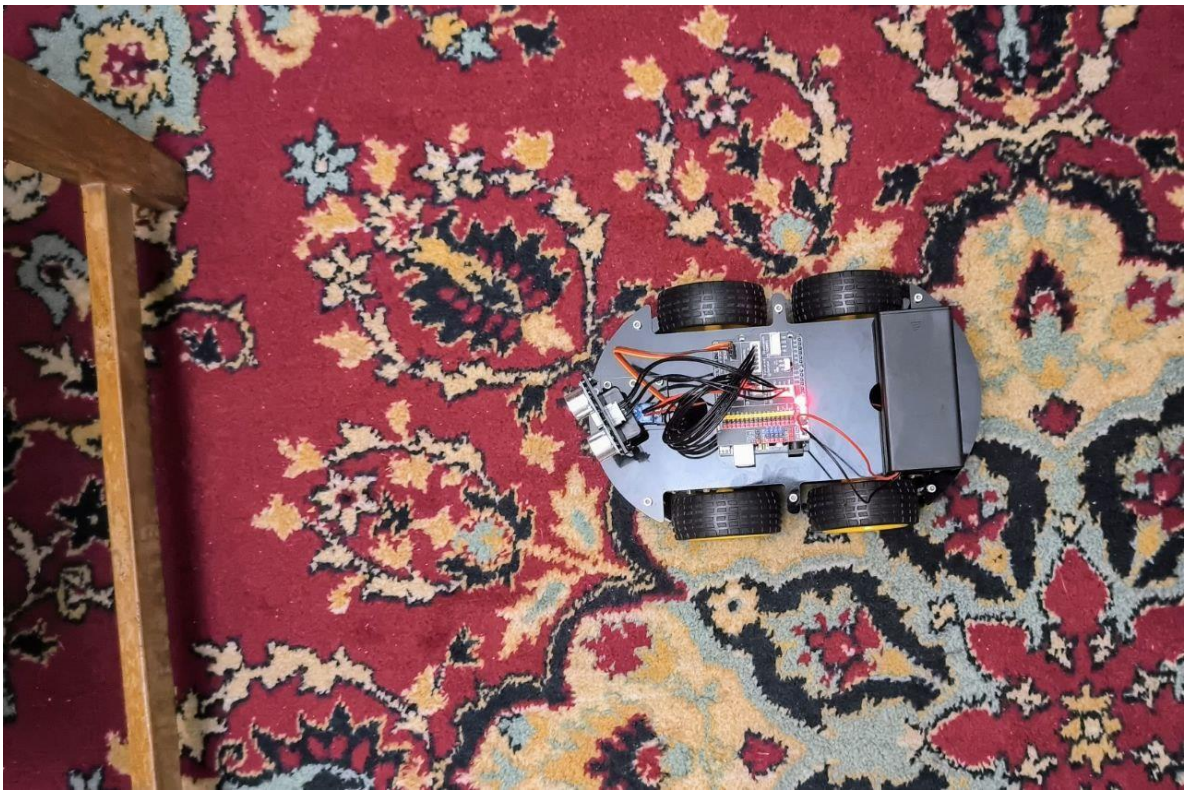

Рисунок 3.16 – Платформа від’їхала назад для розвороту після зіткнення з перешкодою (код без ГА)

Рисунок 3.17 – Платформа одразу відреагувала на перешекоду і пішла на розворот (код з ГА)

Рисунок 3.18 – Платформа продовжила рухатися повз стілець в обох випадках

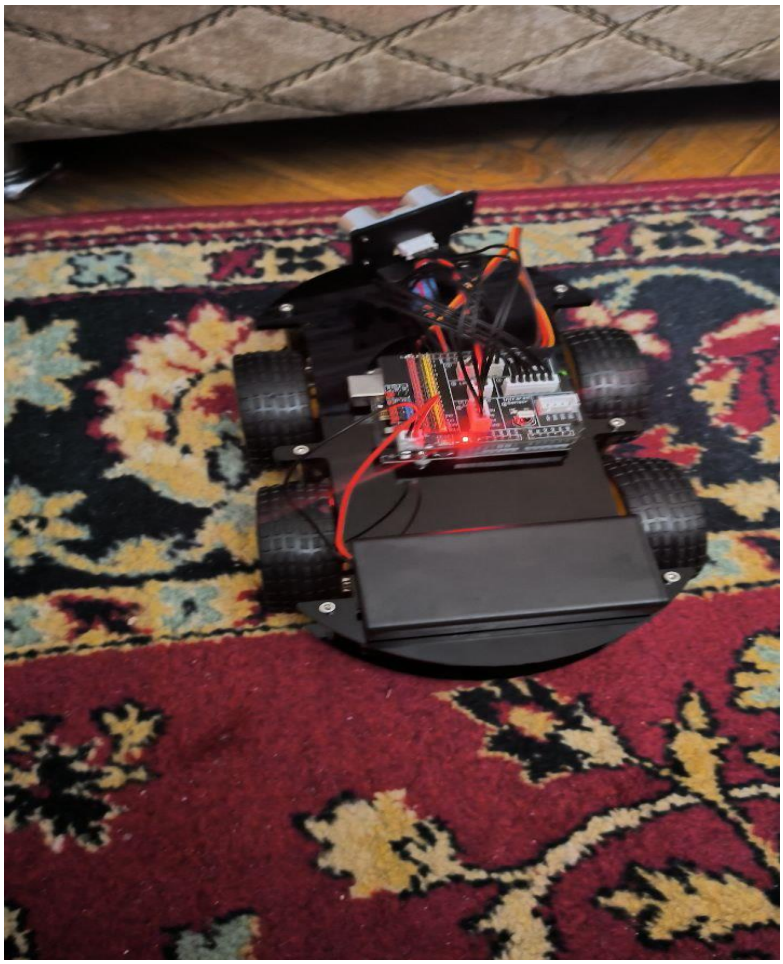

Рисунок 3.19 – Кінець роботи програми

Результати експерементів з фізичним макетом представлені у таблиці 3.1.

Таблиця 3.1 – Проміжки часу з експерементів

Номер експеременту	ГА	Без ГА
1	2:01	1:59
2	1:42	2:10
3	1:30	2:04
4	1:32	1:58

За підсумками експерементів можно зробити висновок що платформа с ГА реагує на перешкоди краще та швидше, ніж робот без ГА. Порівнюючи час проходження шляху отримаємо результат на користь платформи с ГА, одна

хвилина проти майже двох у випадку платформи без ГА. Слід зазначити, що платформі на основі генетичного алгоритму знадобилося попередньо «вивчити» трасу для того, щоб досягти найліпшого результати, навіть не дивлячись на мобільні перешкоди під час проходження шляху. Якщо брати загальний час, то можна сказати що цей час однаковий для обох випадків, бо попереднє вивчення траси також потребує часу, але подальші спроби пройти трасу вже швидші саме у робота, в основі якого лежить ГА.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи була розроблена віртуальна модель для дослідження навчання робота на основі досвіду, що вирізняється зручністю експлуатації та простотою розуміння. Ця модель дозволяє проводити експерименти з навчання робота у зазначеному середовищі та умовах, що робить її обмежено придатною для широкого спектру дослідницьких завдань, але дає змогу повністю і коректно зрозуміти принцип роботи ГА та вплив різних змінних на якість адаптації індивіда до умов.

Основною метою роботи стало створення віртуального макету, який дозволить моделювати процеси навчання робота з використанням ГА, а в подальшому і різних методів машинного навчання. Це дозволить значно розширити сферу застосування робототехнічних систем та підвищити їх ефективність у реальних умовах. Проведено аналіз різних методів машинного навчання, які можуть бути використані для навчання роботів.

Також було створено фізичний макет мобільної платформи, що дало змогу провести експеременти та дослідити вплив на якість керування платформою двох різних ПЗ, а саме с використання ГА та без його участі. За результатами можно побачити що покращення під час першої спроби пройти маршрут відсутнє, але в подальшому маршрут проходиться значно швидше саме платформою с ГА.

Отримані результати свідчать про:

– ефективність та надійність розробленої віртуальної моделі для дослідження навчання робота на основі досвіду;

– працездатність віртуального макету та його перспективність для застосування в різних сферах досліджень;

– величезний вплив в позитивну сторону генетичного алгоритму та, потенційно інших методів навчання, на керування фізичною моделлю.

Ці результати можуть бути використані для:

– подальшого вдосконалення віртуальної моделі та розширення її функціональних можливостей;

– розробки нових методів та алгоритмів навчання роботів;

– створення інтегрованих систем керування роботами з функцією адаптивного навчання на основі досвіду.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008: 2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. – 31 с.

2. Невлюдов І. Ш. Дипломне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» [Текст]: навч. посіб. / І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова – Київ-58, пр. Космонавта Комарова, 1, 2016 – 320 с.

3. Методичні вказівки до підготовки атестаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми: «Автоматизація та комп'ютерно-інтегровані технології» / упоряд.: І. Ш. Невлюдов, О. В. Токарєва, Г. В. Пономарьова. – Харків: ХНУРЕ, – 2019. – 36 с.

4. Шостя С. П. Машинне навчання // Новітні інформаційно-комунікаційні технології в освіті : матеріали VII Всеукраїнської науково-практичної Інтернет-конференції молодих учених та студентів (Полтава, 24-25 листопада 2021 р.). Полтава : ПП "Астрая", 2021. С. 146-148.

5. Yevsieiev, V., & Gurin, D. (2023). Comparative Analysis of the Basic Methods Used in Industry 4.0 and Industry 5.0. Collection of Scientific Papers «ΛΌГОΣ», (Bologna, Italy), 113–115. https://doi.org/10.36074/logos-29.09.2023.31

6. Кононюк А.Ю. К65 Нейроні мережі і генетичні алгоритми К.:«Корнійчук», 2008. – 446 с.

7. Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В., Новоселов С. П., Демська Н. П. Проектування мобільних маніпуляційних роботів: Монографія. – Х. :, 2022. – 427 с.

8. Невлюдов І. Ш. BEAM робототехніка : навч. посіб. / І. Ш. Невлюдов, В. В. Євсєєв, С. С. Максимова ; Харків. нац. ун-т радіоелектроніки, кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР). – Кривий Ріг : Видавець Чернявський Д. О., 2024. – 276 с. – ISBN 978-617-8045-79-1

9. Моделі та методи кіберфізичних виробничих систем в концепції Industry 4.0 : монографія / І. Ш. Невлюдов, В. В. Євсєєв, А. О. Андрусевич, С. С. Максимова ; – Oktan Print – Prague. 2023. – 321 с.

10. Andrieiev A. M., Kulynych A. H. Vykorystannia aparatno-prohramnoho kompleksu Arduino v innovatsiinii diialnosti maibutnikh uchyteliv fizyky ta uchniv. URL: http://ite.kspu.edu/webfm_send/943

11. Somenko D. V., Somenko O. O., Vykorystannia mozhlyvostei aparatno obchysliuvalnoi platformy Arduino v laboratornomu praktykumi z fizyky. Naukovi zapysky. Seriia: Problemy metodyky fizykomatematychnoi i tekhnolohichnoi osvity. Vypusk 9 (1), Kirovohrad — 2016, s. 173–184.

12. Vladyslav Yevsieiev, Nikolaj Starodubcev (2023). Development of a control algorithm for a small-sized mobile manipulation robot. Scientific Collection «InterConf», (140), P. 648-651.

13. Мокренко П.В., Ядловська В.В. Огляд розвитку робототехніки. Частина 1. (Робототехніка до XX століття). Автоматика, вимірювання та керування. 2020. Т. 2. № 1(2). С. 67–77. URL: https://doi.org/10.23939/amm2020.01.067.

14. Робототехніка. Штучний інтелект. Чернігівський обласний центр зацнятості. URL: https://chg.dcz.gov.ua/publikaciya/robototehnika-shtuchnyy-intelekt.

15. Бабич О., Бойко Я., Галін В., Чупринський О. Проектування інтелектуальних інформаційних систем на базі МК Raspberry Pi. Електроніка та інформаційні технології. 2019. Вип. 11. С. 61–72.

16. Невлюдов, І.Ш. Теорія автоматичного управління (збірник задач): навчальний посібник / І.Ш. Невлюдов, О.В.Токарєва. – Харків: ХНУРЕ, 2020. 240 с.

17. Nano | Adruino Documentation. Arduino Docs. URL: https://docs.arduino.cc/hardware/nano.

18.Драйвер для двигунів L298N. Arduino.ua. URL: https://arduino.ua/ru/prod406-draiver-dvyh-dvigatelei-na-l298n

19. Yevsieiev V. (2023) Development of a program for modeling the control of a mobile manipulation robot in the unity environment / Yevsieiev V., Starodubcev N. // Scientific Collection «InterConf», (141), P. 331-334.

20. Євсєєв В.В. Проектування мобільних роботів на базі одноплатних комп'ютерів (Raspberry Pi и мови Python 3.6) // Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В. Підручник. – Харків : 2020. С. 257.

21. Yevsieiev V. Some aspects of the development of the BEAM robot control scheme / V. Yevsieiev // In IV International Scientific and Theoretical Conference,Singapore, Republic of Singapore. - P. 79-81.