Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Підвищення точності маршруту мобільних роботів за допомогою адаптованих кривих Безьє
(тема)

Виконав:
здобувач 2 року навчання,
групи КТРСм-24-2

Дмитро МАЦЬКО

Спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Тип програми Освітньо-професійна

Освітня програма Комп`ютеризовані та робототехнічні системи

Керівник доц. Наталія ДЕМСЬКА

Допускається до захисту
Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ

2025р.

Я, Мацько Дмитро В’ячеславович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу з академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

16 грудня 2025 р. Мацько Д. В.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти другий (магістерський)
Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР _____
(підпис)
« ___»_____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Мацьку Дмитру В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Підвищення точності маршруту мобільних роботів за допомогою адаптованих кривих Безьє

Затверджена наказом по університету від 10.11.2025 р. № 1018 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 08.12.2025 р.

3. Вихідні дані до роботи
3.1 Алгоритм пересування мобільного робота з використання адаптованих Кривих Безьє;
3.2 Для розробки алгоритму була використана мова програмування Python;
3.3 Для розробки та тестування алгоритму використовувалося програмне середовище Webots;
3.4 Для обчислення математичних функцій та малювання графіків використовувалося середовище Matlab.
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ;
4.2 Аналіз існуючих методів побудови маршруту пересування мобільних роботів;
4.3 Розробка методу підвищення точності пересування мобільного робота;
4.4 Розробка програмного забезпечення пересування робота за розробленим методом;
4.4 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Денмонстраційний матеріал представленний у вигляді презентації – 15 арк. ф. А 4

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	Приміт-ка
1	Аналіз існуючих методів побудови маршруту пересування мобільних роботів	01.09.25-24.09.24	виконано
2	Підготовка навчального набору даних та налаштування середовища для навчання моделі	24.09.25-26.10.25	виконано
3	Розробка методу підвищення точності пересування мобільного робота	26.10.25-24.11.25	виконано
4	Розробка програмного забезпечення пересування робота за розробленим методом	24.11.25-14.12.25	виконано
5	Подання роботи на перевірку щодо дотримання академічної доброчесності	15.12.25	виконано
6	Оформлення пояснювальної записки	21.12.25	виконано
7	Подання роботи на рецензію	20.12.25	виконано
8	Подання роботи на підпис зав. кафедри	21.12.25	виконано
9	Подання кваліфікаційної роботи в ЕК	22.12.25	виконано

Дата видачі завдання 01.09.2025 р.

Здобувач _____Дмитро МАЦЬКО

Керівник роботи _____ доц. Наталія ДЕМСЬКА

РЕФЕРАТ

Пояснювальна записка: 71 с., 7 табл., 26 рис., 3 дод., 10 джерел.

АЛГОРИТМ ПЕРЕСУВАННЯ, МОБІЛЬНИЙ РОБОТ, ПІДВИЩЕННЯ ТОЧНОСТІ.

Об’єкт дослідження – метод пересування мобільного робота з використанням адаптованих кривих Безьє.

Предмет дослідження – підвищення точності маршруту мобільних роботів з урахуванням багатьох факторів, таких як специфіка різних видів роботів, ситуацій, в яких вони можуть використовуватися.

Мета роботи – моделювання роботи програмного забезпечення робота з метою підвищення точності траєкторії пересування робота.

Методи дослідження – математичне моделювання адаптованих кривих Безьє в середовищі Matlab, та симуляція роботи програмного забезпечення в середовищі Webots.

В кваліфікаційній роботі проведено аналіз існуючих алгоритмів побудови оптимального шляху до цілі. Розроблено алгоритм керування роботом за яким буде працювати робот. Розроблено програмне забезпечення за описаним алгоритмом. Проведено симуляцію роботи розробленого програмного забезпечення в середовищі Webots.

У ході роботи було розроблено алгоритм пересування мобільного робота з використанням адаптованих кривих Безьє для побудови траєкторії. Розроблений алгоритм забезпечує високу надійність роботи та точність пересування.

ABSTRACT

Explanatory note: 71 p., 7 tabl., 26 fig., 3 app., 10 sources.

ACCURACY IMPROVEMENT, MOBILE ROBOT, MOTION ALGORITHM.

The object of the study is a method of mobile robot movement using adapted Bezier curves.

The subject of the study is increasing the accuracy of the route of mobile robots, taking into account many factors, such as the specifics of different types of robots, situations in which they can be used.

The purpose of the work is to simulate the operation of the robot software in order to increase the accuracy of the robot's movement trajectory.

Research methods are mathematical modeling of adapted Bezier curves in the Matlab environment, and simulation of the software in the Webots environment.

In the qualification work, an analysis of existing algorithms for constructing the optimal path to the goal was carried out. A robot control algorithm was developed, according to which the robot will work. Software was developed according to the described algorithm. The operation of the developed software was simulated in the Webots environment.

During the work, an algorithm for moving a mobile robot using adapted Bezier curves for constructing the trajectory was developed. The developed algorithm provides high reliability of operation and accuracy of movement.

ЗМІСТ

Перелік скорочень 6
Вступ 7
1 Аналіз існуючих методів побудови маршруту пересування мобільних роботів 9
1.1 Основні відомості про криві безьє 9
1.2 Аналіз існуючих методів побудови оптимального шляху 12
1.3 Аналіз існуючого програмного забезпечення для симуляції роботи мобільного робота 20
1.4 Висновки до першого розділу 23
2 Розробка методу підвищення точності пересування мобільного робота 25
2.1 Опис розробленого методу 25
2.3 Висновки до другого розділу 30
3 Розробка програмного забезпечення пересування робота за розробленим методом 31
3.1 Розробка програмного забезпечення пересування робота 39
3.2 Проведення симуляції роботи розробленого алгоритму 47
3.3 Розрахунок стійкості системи 51
3.4 Питання охорони праці та безпеки життєдіяльності 54
Додаток А Лістинг програмного забезпечення контролера робота в середовищі Webots 60
Додаток Б Демонстраційний матеріал 68

ПЕРЕЛІК СКОРОЧЕНЬ

ІЧ – інфрачервоні;

МР – мобільний робот;

ПЗ – програмне забезпечення;

САПР – системи автоматизованого проектування;

ШІМ – широтно імпульсна модуляція;

IDA* – Iterative Deeping A*;

RRT– Rapidly exploring random tree;

ROS – Robot Operating System.

ВСТУП

Стрімкий розвиток робототехніки та автономних систем упродовж останніх десятиліть зумовив зростання вимог до точності, надійності та адаптивності пересування мобільних роботів. Такі системи знаходять широке застосування в промисловості, логістиці, медицині, сільському господарстві, а також у середовищах з обмеженим або небезпечним доступом для людини. У більшості практичних задач мобільний робот повинен не лише досягати заданої цільової точки, а й рухатися по оптимальній, плавній та безпечній траєкторії, уникаючи перешкод і мінімізуючи похибки позиціонування [1].

Використання адаптованих кривих Безьє в побудові оптимального шляху пересування робота є складним завданням, яке вимагає врахування багатьох факторів, таких як специфіка різних видів роботів, ситуацій, в яких вони можуть використовуватися [2].

Проаналізувавши предметну область та тему роботи можна дійти висновку що предметом розробки є розробка програмного забезпечення пересування робота з використанням адаптованих кривих Безьє. Це програмне забезпечення повинно забезпечувати достатній рівень точності пересування, реагувати на перешкоди та мати можливість адаптуватися до динамічної середи.

Метою кваліфікаційної роботи є моделювання роботи програмного забезпечення робота з метою підвищення точності траєкторії пересування робота.

Об'єктом дослідження є метод пересування мобільного робота з використанням адаптованих кривих Безьє.

Для досягнення мети планується розв’язання наступних задач:

– аналіз існуючих методів визначення оптимального між двома точками;

– розробка алгоритму пересування робота з використанням адаптованих кривих Безьє;

– розробка програмного забезпечення для мобільного робота;

– розглянути питання охорони праці.

Робота виконана згідно [3-4]. Результати роботи опубліковані в [5].

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ПОБУДОВИ МАРШРУТУ ПЕРЕСУВАННЯ МОБІЛЬНИХ РОБОТІВ

1.1 Основні відомості про криві Безьє

На відміну від звичайних поліномів, криві Безьє мають такі переваги як геометрична інтуїтивність, тобто будуються по точкам що дозволяє мати плавність руху, кращу чисельну стійкість та легкість розділення та об’єднання кривих.

У багатьох методах побудови тривимірних кривих отримувана крива проходить через всі задані точки, тобто ці методи є методами інтерполяції. В багатьох випадках вони дозволяють отримати добрі результати. Але ці методи мають ряд недоліків, що роблять їх малопридатними для систем побудови кривих в діалоговому режимі. Це пояснюється тим, що задання форми кривої з допомогою таких числових характеристик, як напрямок і модуль дотичних векторів, не забезпечує інтуїтивного відчуття правильності побудови кривої, тобто не завжди існує очевидний зв’язок між числами і формою кривої. Крім того, методи підгонки тривимірними кривими в результаті дають криву одного і того самого порядку, який не змінюється. Для того, щоб збільшити кривизну, потрібно вводити більшу кількість точок [6].

Безьє розробив інший метод зображення кривої, який створює у користувача більш природнє сприйняття [7]. Крива Безьє визначається вершинами багатокутника, який задає форму кривої. Кривій належить перша та остання вершина, в той час як інші вершини характеризують похідні, порядок та вид кривої. Таким чином, крива задається з допомогою відкритого багатокутника (ламаної) сформованого заданими точками, як показано на рис 1.1.

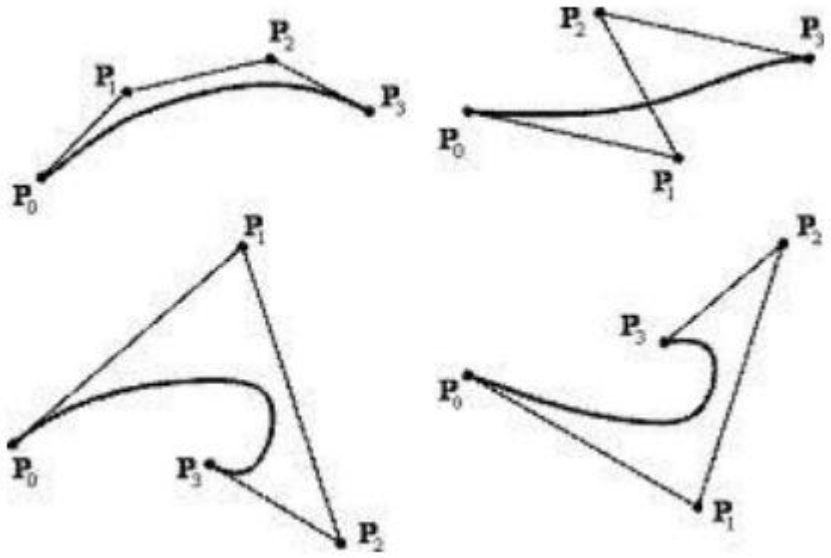


Рисунок 1.1 – Приклади кривих Безьє

В [8] крива Безьє ступеня n представлена як:

$$P_{[t_0,t_1]}(t) = \sum_{i=0}^{n} B_i^n(t) P_i, \tag{1.1}$$

де $P_i \ldots n$ – контрольні точки, такі як $P(t0) = P_0$, $P_{(t1)} = P_n$ а B_i^n;

(t) – поліном Бернштейна, заданий формулою:

$$B_i^n = \binom{n}{i}\left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i}\left(\frac{t - t_0}{t_1 - t_0}\right)^{i},\ i \in \{0, 1, \ldots, n\}, \tag{1.2}$$

де t – позначає змінну часу.

Крива Безьє має такі властивості для планування шляху:

– крива Безьє завжди проходить через P_0 та P_n;

– крива Безьє завжди дотична до ліній, що з'єднують P_0, P_1 та P_n, P_{n-1} у точках P_0 та P_n відповідно;

– крива Безьє завжди лежить всередині опуклої оболонки, що складається з її контрольних точок.

Щодо кривих Безьє для планування траєкторії мобільного робота(МР), алгоритм Кастельжо є актуальним. Алгоритм Кастельжо описує рекурсивний метод для поділу кривої Безьє $P[t_0,t_2](t)$ на два сегменти $P_{[t0,t1]}(t)$ та $_{[t1,t2]}(t)$.

Нехай, позначаючи через $P_0^0, P_1^0 \dots P_n^0$ контрольні точки $P[t_0,t_2](t)$. Контрольні точки $P_{[t0,t1]}(t)$ та $P_{[t1,t2]}(t)$ можна обчислити за допомогою:

$$P_i^j = (1-\tau)P_i^{j-1} + \tau P_{i+1}^{j-1}, j \in \{1,\dots,n\}, i \in \{0,\dots,n-j\}, \quad (1.3)$$

де $\tau = \frac{t_1-t_0}{t_2-t_0}$, тоді $P_0^0, P_1^0 \dots P_n^0$ – контрольні точки $P_{[t0,t1]}$ та $P_0^n, P_1^{n-1} \dots P_n^0$ – контрольні точки $P[t_1,t_2]$.

Крива Безьє $P_{[t0,t2]}$ завжди проходить через точку $P(t_1) = P_n^0$, якщо застосувати алгоритм Кастельо для поділу себе на $P_{[t0,t1]}$ та $P_{[t1,t2]}$. Більше того, вона завжди дотична до P_0^{n-1}, P_1^{n-1} та $P(t_1)$.

Криві Безьє поділяють в залежності від кількості точок через які будується крива. Загалом їх поділяють на лінійні криві Безьє, квадратичні та кубічні.

Лінійні криві Безьє при n = 1 де крива є відрізком від точки P0 до точки P1.Зовнішній вигляд лінійної кривої Безьє показано на рис 1.2, а. Крива задається як:

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2, \quad t \in [0,1]. \quad (1.4)$$

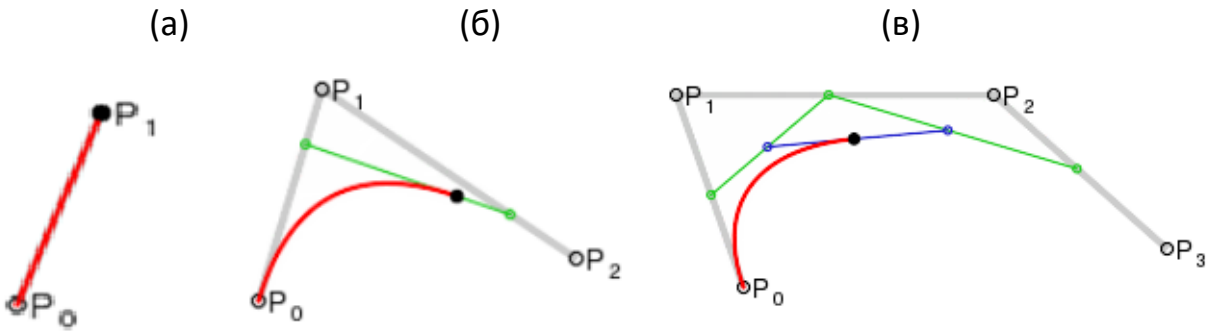
Квадратична крива Без'є (n = 2) задається трьома опорними точками, P0, P1 та P2. Зовнішній вигляд лінійної кривої Безьє показано на рис 1.2, б. Формула квадратичної кривої:

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2, \quad t \in [0,1]. \tag{1.5}$$

Криві Безьє через 4 опорні точки P_0, P_1, P_2 та P_3, називаються кубічними кривими задані в 2-х чи 3-мірному просторі визначають форму кривої.

Лінія починається в точці P_0 направляється до P_1 і закінчується в точці P_3 підходячи до неї з боку точки P_2. Тобто крива не проходить через точки P_1 та P_2, їх використовують для напрямку руху. Зовнішній вигляд лінійної кривої Безьє показано на рис 1.2, в. Формула описуюча кубічні криви Безьє:

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3, \quad t \in [0,1]. \tag{1.6}$$



а – лінійна крива; б – квадратична крива; в – кубічна крива

Рисунок 1.2 – Зовнішній вигляд різних кривих Безьє

1.2 Аналіз існуючих методів побудови оптимального шляху

За принципом побудови методи побудови оптимального шляху можна поділити на:

– методи на основі графів;
– методи на основі евристик та пошуку;
– методи динамічного програмування;

– методи на основі оптимізації траєкторій у просторі.

Загальна схема методів побудови оптимального шляху показана на рисунку 1.3.

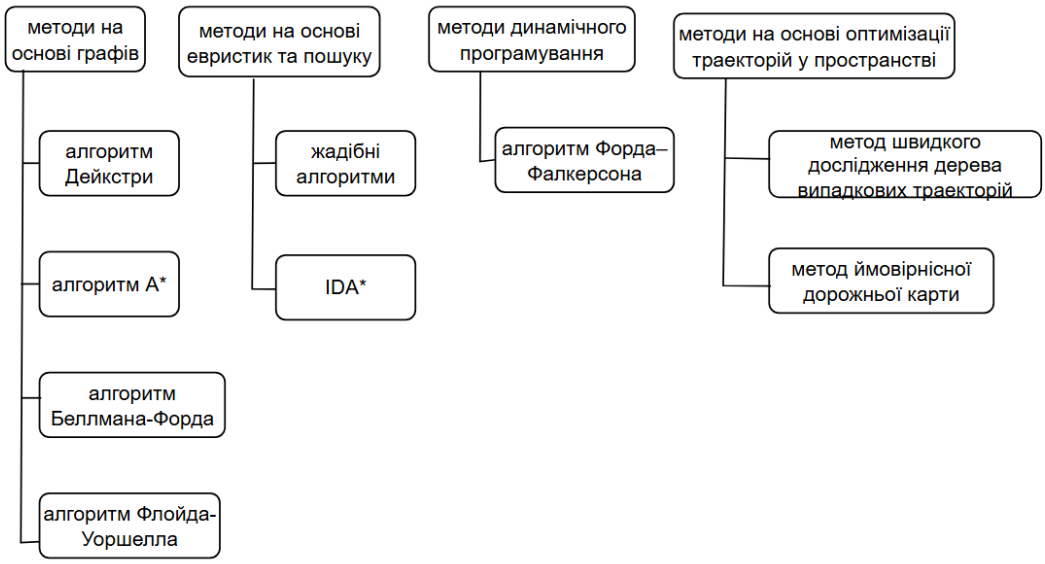


Рисунок 1.3 – Схема організації методів оптимізації шляху

Прикладами методів на основі графів можуть бути:

– алгоритм Дейкстри;

– алгоритм A*;

– алгоритм Беллмана-Форда;

– алгоритм Флойда-Уоршелла.

Алгоритм Дейкстри дозволяє знаходити найкоротший шлях у зважених графах з позитивними ребрами.

Алгоритм Дейкстри, також відомий як метод, який дозволяє знайти найбільш ефективний шлях від а початковий вузлів до всіх інших вузлів у зваженому графі. Цей графік повинен мати ваги жодного негативу на його краях, оскільки алгоритм не призначений для обробки від’ємних значень.

Основна ідея за алгоритмом є безперервний облік коротші відстані від початкового вузла до кожного вузла на графі. У міру просування алгоритм

оновлює ці відстані щоразу, коли знаходить коротший шлях. Прикладом зовнішнього вигляду графу для алгоритму Дейкстри показано на рисунку 1.4.

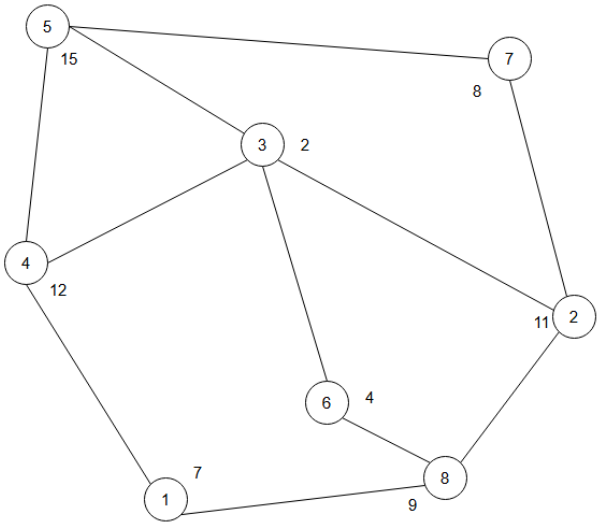


Рисунок 1.4 – Приклад зовнішнього вигляду графу алгоритму Дейкстри

Кінцевим результатом є а дерево найкоротших шляхів, який з'єднує початковий вузол з усіма іншими. Цей підхід корисний у різноманітних програмах, від навігаційних систем GPS до аналізу мережі та логістичного планування маршруту.

Алгоритм пошуку A* належить до евристичних алгоритмів пошуку. Використовується для пошуку найкоротшого шляху між двома вершинами графу з додатними вагами ребр. Докладніше про нього можна найти в [9].

Алгоритм використовує допоміжну функцію аби скеровувати напрям пошуку та скорочувати його тривалість. Алгоритм повний в тому сенсі, що завжди знаходить оптимальний розв'язок, якщо він існує.

Алгоритм A* спершу відвідує ті вершини, які ймовірно ведуть до найкоротшого шляху до мети. Аби розпізнати такі вершини, кожній відомій вершині x співставляється значення f(x), яке дорівнює довжині найкоротшого шляху від початкової вершини до кінцевої, який пролягає через обрану вершину. Вершини з найменшим значенням обираються в першу чергу.

Алгоритм Беллмана-Форда – це алгоритм, який обчислює найкоротші шляхи від однієї вихідної вершини до всіх інших вершин в зваженому орієнтованому графі. Безумовно, він являється повільнішим, ніж алгоритм Дейкстри для тієї ж задачі, але більш універсальний, оскільки здатний обробляти графи, в яких вага деяких ребер приймає від'ємного значення. Приклад графу алгоритма Беллмана-Форда показано на рисунку 1.5.

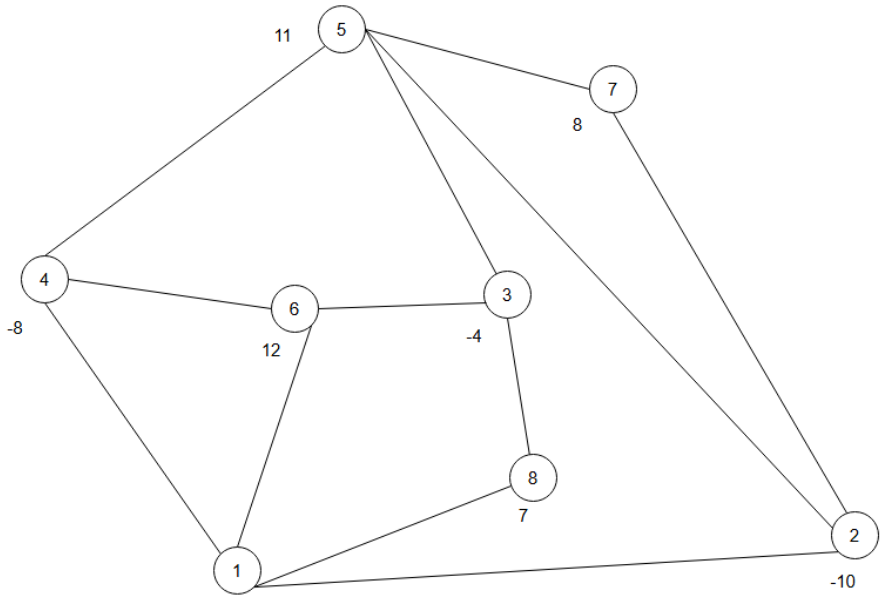


Рисунок 1.5 – Приклад графу алгоритму Беллмана-Форда

Як уже зазначалося вище, алгоритм Беллмана-Форда підходить для роботи з графами, що мають ребра з від'ємною вагою. Однак, якщо граф містить «від'ємний цикл», тобто цикл, сума ваги ребер якого дорівнює від'ємному значенню, тоді, для даного графа, не існує дерева найкоротших шляхів (будь-який шлях такого типу може бути покращений ще одним проходом через ребра, що утворюють від'ємний цикл) [10]. В такому випадку алгоритм Беллмана-Форда може виявити цикли від'ємної довжини і повідомити про їх існування, але він не може дати правильну відповідь, тобто знайти найкоротший шлях, якщо від'ємний цикл досяжний з вершини джерела.

Алгоритм Флойда-Уоршелла схожий с алгоритмом Беллмана-Форда. Як і алгоритм Беллмана-Форда він повертатиме матрицю відстаней між усіма

парами вершин. Він не зберігає деталі про найкоротші шляхи а лиш зберігає саме шляхи. Він подібний до алгоритму Дейкстри але знаходить найкоротші шляхи між усіма точками а не двома.

До методів на основі евристик та пошуку можливо віднести:

– жадібні алгоритми;

– пошук ітеративного заглиблення A*(IDA*).

Жадібним називається алгоритм, який на кожному кроці шукає локально-оптимальне рішення, припускаючи, що кінцеве загальне рішення є суперпозицією локальних також буде оптимальним.

У багатьох завданнях жадібна стратегія не дає оптимального рішення, проте, може призвести до локально-оптимальних рішень, які апроксимують глобально-оптимальне рішення за розумний час.

Жадібний алгоритм містить 5 компонентів:

Набір кандидатів, із якого створюється рішення,

– функція вибору, яка обирає найкращого кандидата для додавання до рішення,

– функція техніко-економічного обґрунтування, що використовується для визначення, чи може кандидат використовуватись для сприяння рішенню,

– цільова функція, яка надає значення рішенню чи частковому рішенню;

– функція, яка вказує, коли буде знайдено повне рішення.

Ітеративне поглиблення A* (IDA*) – це алгоритм обходу графів та пошуку шляхів, який може знайти найкоротший шлях між призначеним стартовим вузлом та будь-яким елементом набору цільових вузлів у зваженому графі [11].

Це варіант ітеративного поглиблюючого пошуку з пріоритетом у глибину, який запозичує ідею використання евристичної функції для консервативної оцінки вартості досягнення мети, що залишилася, за допомогою алгоритму пошуку A*. Оскільки це алгоритм пошуку, орієнтований на глибину, його використання пам'яті нижче, ніж в A*, але, на відміну від звичайного ітеративного пошуку, він зосереджений на вивченні

найбільш перспективних вузлів і, отже, не досягає однакової глибини скрізь у дереві пошуку.

На відміну від A*, IDA* не використовує динамічне програмування і тому часто досліджує ті самі вузли багато разів.

У той час як стандартний ітеративний пошук із заглибленням у глибину використовує глибину пошуку як поріг для кожної ітерації, IDA* використовує більш інформативне рівняння:

$$f(n) = g(n) + h(n) \tag{1.7}$$

де $g(n)$ – вартість переходу від кореня до вузла n;

$h(n)$ – це евристична оцінка вартості подорожі з конкретним завданням n до мети.

До методів динамічного проектування можливо віднести алгоритм Форда-Фалкерсона. Алгоритм Форда-Фалкерсона це жадібний алгоритм, що обчислює максимальний потік мережі потоків. Іноді його називають "методом", а не "алгоритмом", оскільки підхід до пошуку шляхів доповнення в залишковому графі не повністю визначений, або він заданий у кількох реалізаціях з різним часом виконання.

Якщо від джерела (стартового вузла) до поглинача (кінцевого вузла) є шлях з доступною пропускною спроможністю по всіх ребрах шляху, ми відправляємо потік по одному з шляхів. Потім ми знаходимо інший шлях, і таке інше. Шлях із доступною пропускною спроможністю називається доповнюючим шляхом.

До методів на основі оптимізації траєкторій у просторі відносяться:

– метод швидкого дослідження дерева випадкових траєкторій;

– метод ймовірнісної дорожньої карти.

Швидко досліджуване дерево випадкових траєкторій (RRT) – алгоритм, розроблений для ефективного пошуку неопуклих високовимірних просторів шляхом випадкового побудови дерева, що заповнює простір. Дерево

будується поступово, з вибірок, взятих випадковим чином із простору пошуку, і за своєю суттю має схильність розвиватися до великих недосліджених областей проблеми.

RRT можна розглядати як метод генерації траєкторій відкритих контурів для нелінійних систем із обмеженнями стану. RRT також можна розглядати як метод Монте-Карло для зміщення пошуку у найбільших областях графа в просторі конфігурацій. Деякі варіації навіть можна вважати стохастичними фракталами [12].

RRT вирощує дерево з коренем у початковій конфігурації, використовуючи випадкові вибірки з простору пошуку. Приклад зовнішнього вигляду подібного дерева показано на рисунку 1.6.

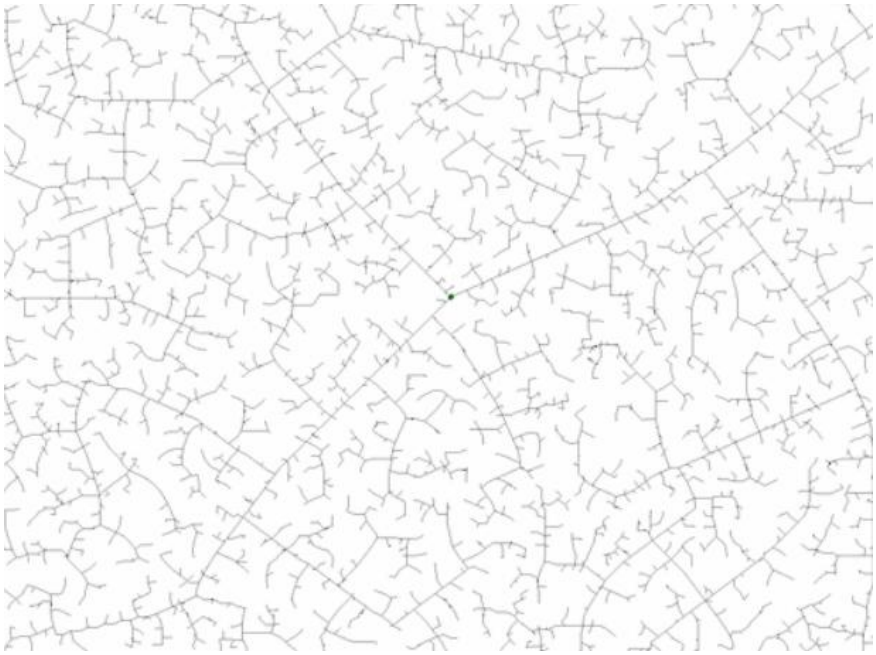


Рисунок 1.6 – Приклад зовнішнього вигляду дерева випадкових траєкторій

Під час кожного вибірку намагається встановити зв'язок між ним і найближчим станом у дереві. Якщо зв'язок можливий (проходить повністю через вільний простір і виконує будь-які обмеження), це призводить до додавання нового стану до дерева.

Довжина з'єднання між деревом і новим станом часто обмежується коефіцієнтом росту. Якщо випадкова вибірка знаходиться далі від найближчого стану в дереві, ніж дозволяє ця межа, використовується новий стан на максимальній відстані від дерева вздовж лінії до випадкової вибірки замість самої випадкової вибірки. Випадкові вибірки можна розглядати як такі, що контролюють напрямок росту дерева, а коефіцієнт росту визначає його швидкість. Це підтримує упередженість RRT, що заповнює простір, обмежуючи розмір інкрементального зростання [13].

Метод ймовірнісної дорожньої карти – це алгоритм планування руху в робототехніці, який розв'язує задачу визначення шляху між початковою конфігурацією робота та конфігурацією цілі, уникаючи зіткнень.

Основна ідея RRT полягає в тому, щоб взяти випадкові вибірки з конфігураційного простору робота, перевірити, чи знаходяться вони у вільному просторі, і використати локальний планувальник для спроби пов'язати ці конфігурації з іншими сусідніми конфігураціями. Додаються початкові та цільові конфігурації, а до отриманого графа застосовується алгоритм пошуку за графом для визначення шляху між початковою та цільовою конфігураціями.

Ймовірнісний планувальник дорожньої карти складається з двох етапів: етапу будівництва та фази запиту. На етапі будівництва будується дорожня карта (графік), яка наближає рухи, які можуть здійснюватися в навколишньому середовищі. Спочатку створюється випадкова конфігурація. Потім він з'єднується з деякими сусідами, зазвичай або k найближчими сусідами, або з усіма сусідами на відстані менше певної заздалегідь визначеної відстані. Конфігурації та з'єднання додаються до графіка, доки дорожня карта не стане достатньо щільною.

1.3 Аналіз існуючого програмного забезпечення для симуляції роботи мобільного робота

Аналіз існуючих програмних засобів симуляції руху мобільного робота є важливим етапом проєктування алгоритму так як дозволяє симулювати мобільного робота, прописавши йому принцип роботи без фізичного створення моделі робота.

Одним із прикладів програмного забезпечення симулюючого рух мобільного робота є Gazebo.

Gazebo – набір бібліотек розробки з відкритим кодом інкапсулює всі основні функції, такі як поширені математичні типи даних, ведення журналу, керування 3D-сітками та асинхронна передача повідомлень. Він має просунутий симулятор робота для досліджень, проектування та розробки, великий набір сенсорних і шумозаглушливих моделей, інтерфейс на основі плагінів до фізичних рушіїв, інтерфейс на основі плагінів для двигунів рендерингу [14].

Він має кросплатформенну підтримку, що дозволяє використовувати бібліотеки Gazebo на Linux і MacOS.

Він має нескладну інтеграцію Robot Operating System(ROS) що спрощує взаємодію між датчиками робота та його актуаторами.

Однак він значні обчислювальні вимоги та обмежену гнучкість для спеціалізованих досліджень траєкторій, що робить його менш зручним для швидкого прототипування алгоритмів планування шляху.

На рисунку 1.7 показано інтерфейс програми Gazebo.

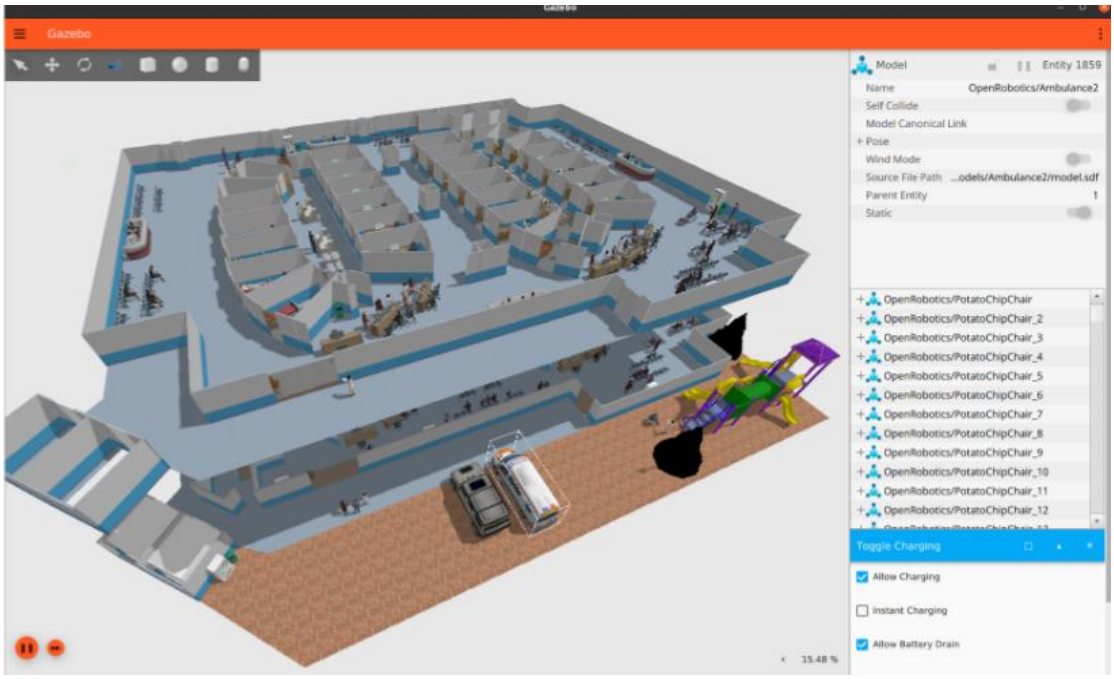


Рисунок 1.7 – Інтерфейс програмного забезпечення Gazebo

Ще одним прикладом для симуляції пересування мобільних роботів може бути програмне забезпечення Webots.

Webots – багатоплатформний десктопний додаток з відкритим вихідним кодом, який використовується для моделювання роботів. Він забезпечує повне середовище розробки для моделювання, програмування та симуляції роботів.

Він був розроблений для професійного використання та широко використовується в промисловості, освіті та наукових дослідженнях. Cyberbotics Ltd. підтримує Webots як свій основний продукт безперервно з 1998 року.

Він дозволяє легко виконувати симуляцію робототехніки, використовуючи вмонтовону бібліотеку активів Webots, яка включає роботів, датчики, актуатори, об'єкти та матеріали. Дозволяє імпортувати наявні моделі систем автоматизованого проектування (САПР).

Ядро Webots базується на поєднанні сучасного графічного інтерфейсу (Qt), фізичного движка (ODE fork) та механізму рендерингу OpenGL 3.3 (wren). Він працює на Windows, Linux і macOS. Симуляції Webots можна експортувати як фільми, інтерактивні HTML-сцени або анімацію або

навіть передавати потоком у будь-який веб-браузер Використання WebGL та веб-сокетів.

Робот може бути запрограмований на C, C++, Python, Java, MATLAB або ROS за допомогою простого API, що покриває всі основні потреби робототехніки.

Webots має відкриту документацію що дозволяє адаптувати його до майже до будь-якого проекту. Приклад зовнішнього вигляду вікна Webots показано на рис 1.8.

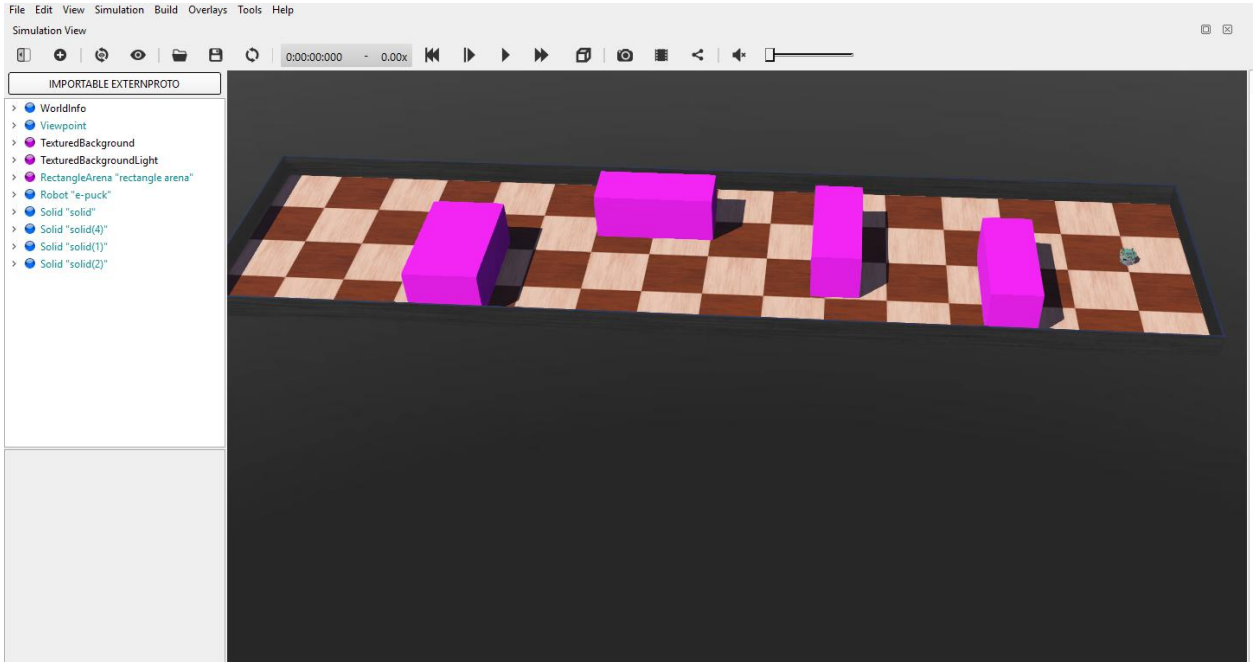


Рисунок 1.8 – Приклад вікна у програмі Webots

Ще одним із прикладів програмного забезпечення може бути CoppeliaSim. Симулятор робототехніки CoppeliaSim заснований на архітектурі розподіленого управління, де кожним об'єктом або моделлю можна індивідуально керувати за допомогою.

CoppeliaSim використовується для швидкої розробки алгоритмів, моделювання автоматизації виробництва, швидкого створення прототипів і верифікації, освіти, пов'язаної з робототехнікою, віддаленого моніторингу, подвійної перевірки безпеки, як цифровий двійник і багато іншого.

Вона має 5 фізичних двигунів (MuJoCo, Bullet Physics, ODE, Newton and Vortex Dynamics) для швидких і настроюваних динамічних розрахунків, для моделювання реальної фізики та взаємодії об'єктів (реакція зіткнення, хапання, м'які тіла, струни, мотузки, тканини тощо).

Може проводити розрахунки прямої/оберненої кінематики для будь-якого типу механізмів (розгалужених, замкнутих, надлишкових, що містять вкладені петлі тощо). Доступна вбудована версія алгоритмів IK/FK. Має потужну, реалістичну і точну імітацію об'ємного датчика наближення, виконує точний розрахунок мінімальної відстані в межах настроюваного обсягу виявлення. Працює на сітках, котрі та хмарах точок [15]. Симуляція датчиків зору з безліччю варіантів обробки зображень, які повністю налаштовуються та розширюються (наприклад, за допомогою плагіна).

1.4 Висновки до першого розділу

У розділі розглянуто існуючі методи побудови оптимального шляху від однієї точки до іншої та розглянуто існуюче програмне забезпечення для симуляції роботи мобільних роботів.

Методи на основі графів дають змогу знаходити найкоротший шлях від однієї точки до іншої в представленні графу де існує багато контрольних точок які з’єднані між собою.

Методи на основі евристик та пошуку є більш вдосконалою версією методів на основі графів.

Було розглянуто побудову кривих Безьє, їх види та характеристики. На цій основі було обрано для побудови траєкторії робота квадратичну криву Безьє, так як з наявністю більшої кількості контрольних точок можливо детальніше будувати траєкторію.

Було розглянуто різні види програмних середовищ для симуляції роботи мобільних роботів, їх плюси та мінуси. На основі чого було обрано програмне

середовище Webots через його велику гнучкість та простоту встановлення та використання.

2 РОЗРОБКА МЕТОДУ ПІДВИЩЕННЯ ТОЧНОСТІ ПЕРЕСУВАННЯ МОБІЛЬНОГО РОБОТА

2.1 Опис розробленого методу

Задля коректного представлення методу підвищення точності пересування мобільного робота за допомогою адаптованих кривих Безьє потрібно представити загальну формулу за якою буде будуватися плавна траєкторія пересування робота [16]. Загалом цю формулу можливо представити як формулу квадратичної кривої Безьє:

$$B = (1-t)^3 \cdot P_0 + 3(1-t)^2 \cdot t \cdot P_1 + 3(1-t) \cdot t^2 \cdot P_2 + t^3 \cdot P_3 \quad (2.1)$$

Нижче на рисунку 2.1 показано приклад графічного відображення описаної формулу на прикладі побудови послідовних квадратичних кривих. Графік було побудовано з використанням середи MATLAB.

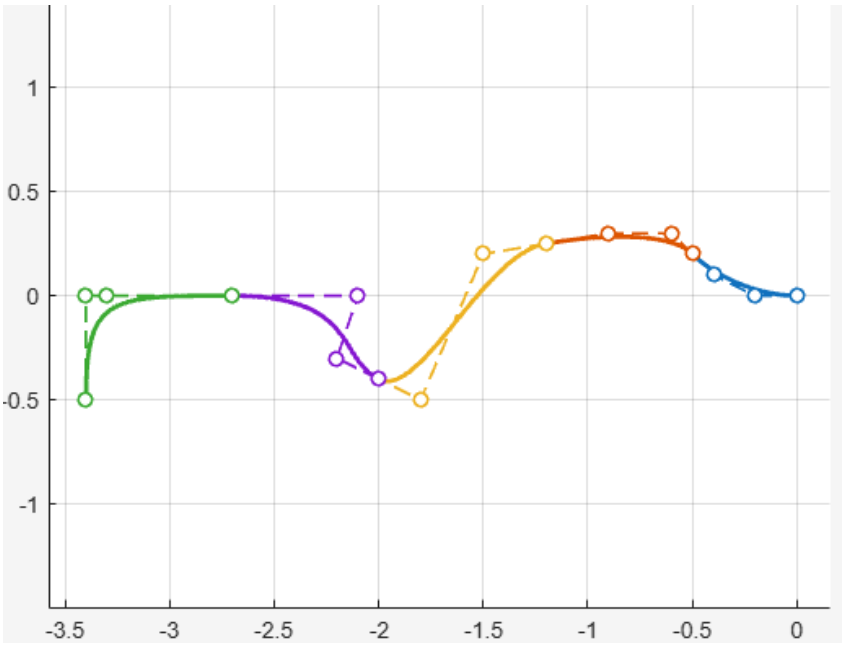


Рисунок 2.1 – Відображення п’яти кривих побудованих за (2.1)

На рис 2.1 показано 5 безперервних кривих які йдуть одна за одною з використанням додаткових контрольних точок. Кожна крива позначена своїм кольором.

До вхідних даних методу відносяться стартова точка, 2 контрольні точки які та кінцева точка. На рисунку 2.2 показано будування однієї кривої з використанням 4-х точок(початкової, кінцевої, та 2-х додаткових).

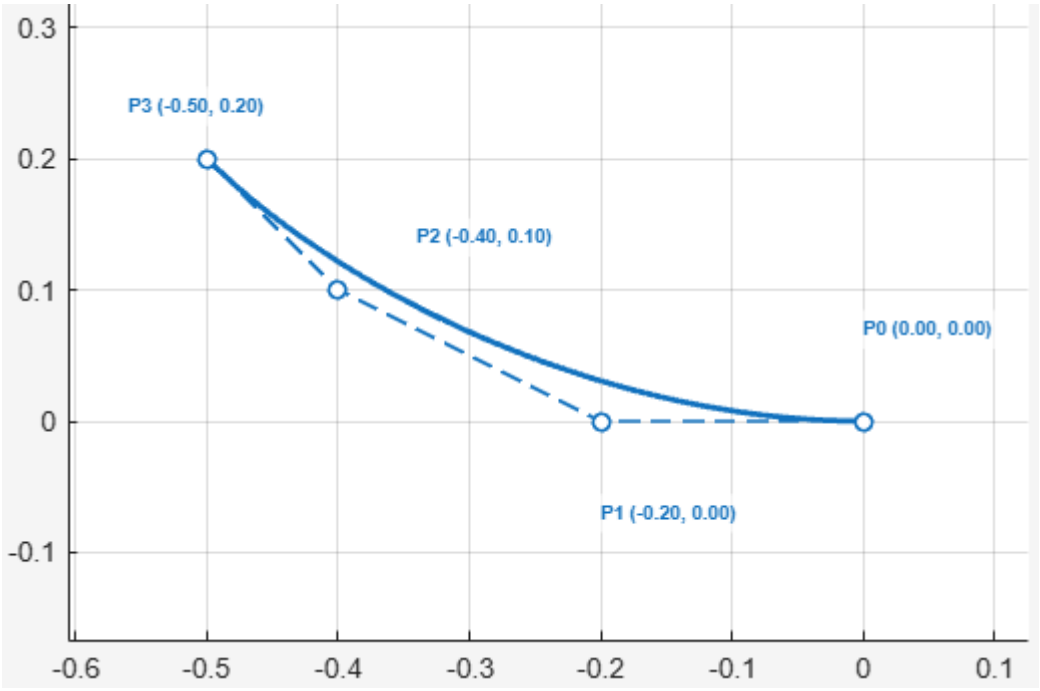


Рисунок 2.2 – Побудова кривої

Як видно з рисунку 2.2 побудова починається з 0ї точки та закінчується в 3-й точці. Також видно що крива не проходить через точки T1 та T3 а використовує їх як доповнюючи точки для побудови плавної кривої. Нижче на рисунку 2.3 показано побудову 2-х безперервних кривих Безьє.

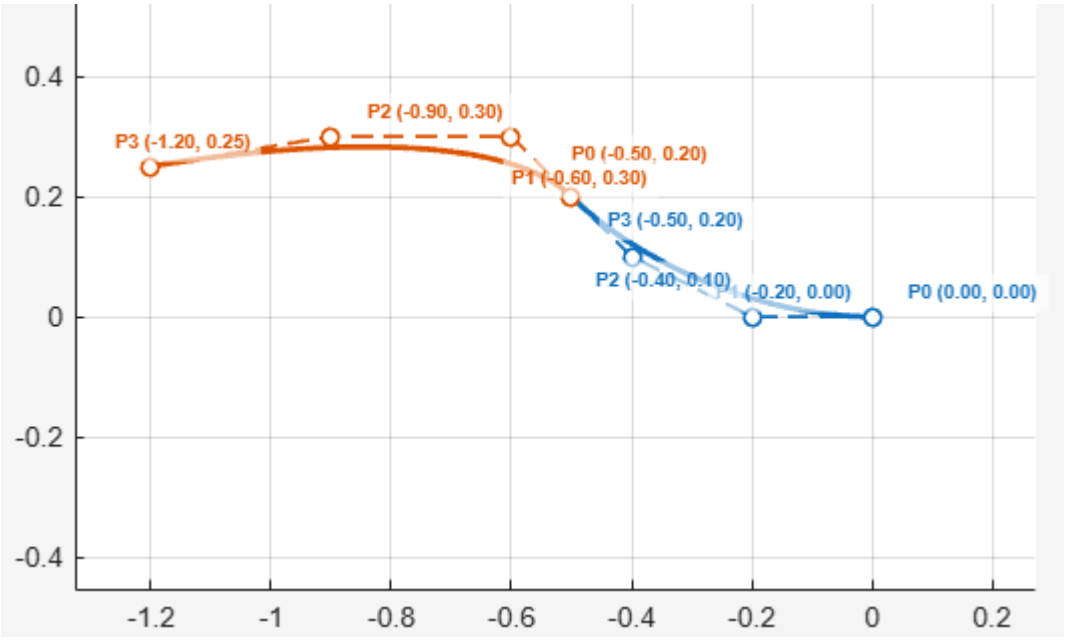


Рисунок 2.3 – Побудова 2-х кривих Безьє

Як видно з рис 2.3 крива 2 починається та продовжує криву 2 що допомогає зберігати плавність руху та його безперервність.

До плюсів цього метода можливо віднести:

– його простоту побудови, так як він будується на звичайній кривій Безьє;

– швидкість роботи методу завдяки його простоти;

– легкість його адаптації до різних машин.

До мінусів методи відносяться:

– потреба в заданні всіх контрольних точок що ускладнює реалізацію автоматичного огинання перешкод;

– потреба адаптації методу до кожного окремого випадку.

2.2 Розробка схеми алгоритму пересування робота за допомогою адаптованих кривих Безьє

При розробці методу пересування робота було розроблено алгоритм, за яким буде розроблено програмне забезпечення. Завдяки цьому алгоритму

програмне забезпечення можливо буде розробити для майже будь-якої схеми приладів. Алгоритм основи програмного забезпечення показано на рис. 2.4.

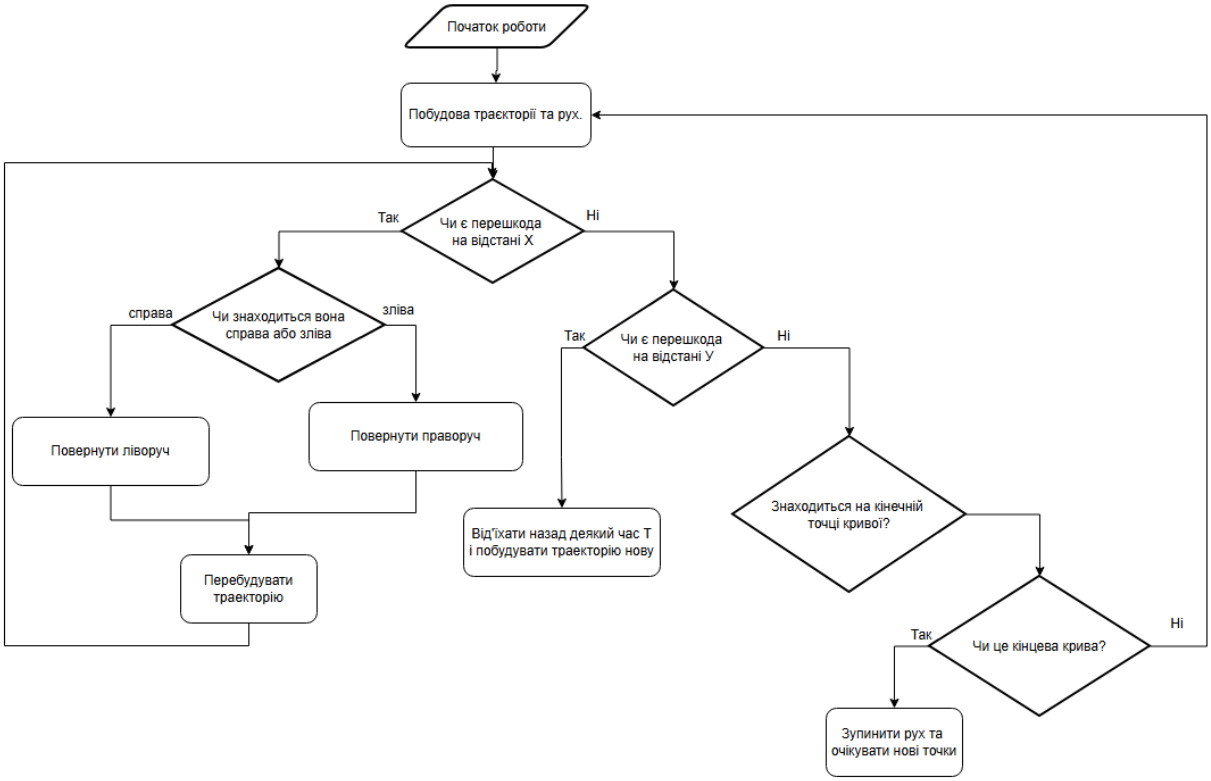


Рисунок 2.4 – Алгоритм роботи програмного забезпечення

ПЗ працює в циклічному режимі до кінцевої точки останньої кривої. Після початку роботи програми починається будуватися траєкторія руху робота та мобільний робот починає рух по цій траєкторії. Траєкторія руху робота буде будуватись за (2.1). Приклад зовнішнього вигляди можливої траєкторія показано на рисунку 2.5.

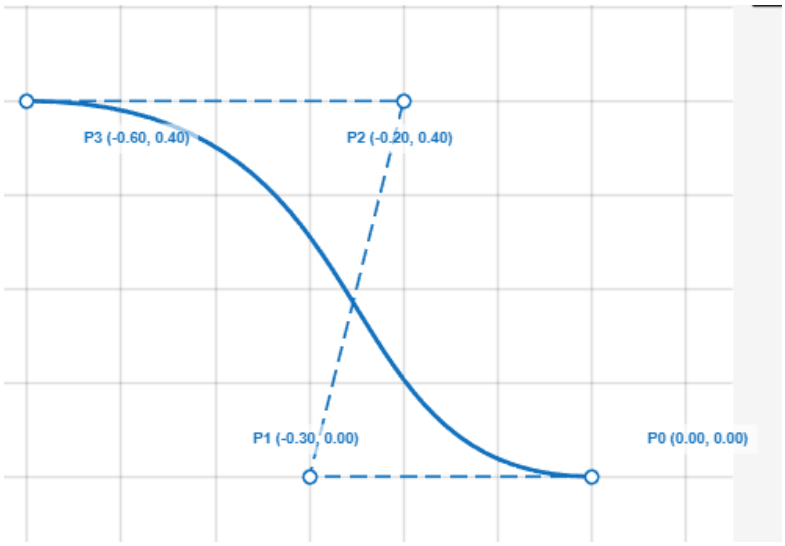


Рисунок 2.5 – Приклад побудови кривої по точкам

Як видно з рис 2.5 для побудови плавної траєкторії потрібно задавати всі точки, як початкову так і кінцеву. В даному рисунку побудова кривої починається в точці P0 з координатами 0 за x, та 0 за y(далі P0(0,0)). Тоді як крива закінчується у точці P3(-0.6, 0.4). Для побудови кривої використовуються точки P1(-0.3,0) та P2(-0.2,0.4). Після побудови однієї кривої потрібно задати точки наступної кривої з вказанням початкової точки як кінцевої точки минулої кривої. Одним із головних потреб в побудові кривої є зберігання плавності траєкторії незалежно від кількості кривих, тобто одна крива плавно переходила в наступну.

Після побудови траєкторії мобільний робот повинен почати рух по цій траєкторії. В цьому випадку робот повинен буде постійно перевіряти, чи є перешкода на відстані, за якою він встигає її оминути, і якщо вона є то він буде повинен повернути у протилежну сторону від перешкоди, після чого змінити траєкторію та продовжити рух.

В разі якщо перешкода знаходиться занадто близько використовується інший алгоритм дій, в якому робот повинен зупинитись, від’їхати на безпечну відстань та продовжити рух до потрібної точки.

В разі якщо на шляху немає перешкод і робот знаходиться в кінцевій точці кривої він повинен продовжити рух до наступної кривої. В разі якщо криві закінчилися робот повинен зупинитися та очікувати на передачу

наступних точок для побудови наступної кривої.

2.3 Висновки до другого розділу

У розділі описано формулу, за якою буде будуватися траєкторія пересування мобільного робота, за основу якої взято квадратичну формулу кривої Безьє.

Розглянуто декілька прикладів побудови подібної траєкторії по описаній формулі з використанням середовища MATLAB.

Розглянуто основні потреби для роботи програмного забезпечення мобільного робота. Було описано спрощену схему алгоритму роботи програмного забезпечення пересування робота, за яким буде працювати робот.

Було описано принцип пересування робота, тобто в яких випадках він буде змінювати траєкторію а в яких кардинально змінювати шлях руху.

Математично було описано принцип побудови кривої.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПЕРЕСУВАННЯ РОБОТА ЗА РОЗРОБЛЕНИМ МЕТОДОМ

Розробка алгоритму базується на описаному в 2.1 методі та реалізує пересування робота в різній середі. В ході виконання симуляції потрібно розробити алгоритм на базі розробленого методу пересування робота.

В алгоритмі потрібно описати можливість обминання перешкод та аналізу найближчого навколишнього середовища [17].

Першим етапом розробки алгоритму пересування робота є визначення його основних вимог та критеріїв. Основними вимогами є:

– збереження плавності траєкторії пересування;
– можливість оминання перешкод.

Після опису критерій вимог потрібно обрати середу програмування, мову програмування та об'єкт який буде використовуватися для реалізації програмного забезпечення (ПЗ).

Після аналізу існуючих методів проведення симуляції роботизованих систем, проведеного у пункті 1.2 було обрано симулятор Webots через його відносну простоту та можливості реалізації проєкту.

Мовою програмування для написання ПЗ було обрано мову програмування Python через його широкі можливості реалізації ПЗ та його легкість.

Описаний у раніше описаних розділах алгоритм може бути реалізований на великій кількості схемах. Найпростішим із них може бути мікросхема на платі Arduino з використанням певної кількості датчиків наближення, моторів та саме схеми.

Мікросхема Arduino є найпростішою мікросхемою для реалізації даного алгоритму через наявність всіх потенційно потрібних блоків та багатої варіації подачі на схему оперативних даних для продовження руху робота.

Існує багата кількість різновидів схем Arduino, деякі з котрих підходять до потреб описаного алгоритму. А саме:

– для підключення коліс та двигунів потрібно використовувати драйвер на який подається посилений сигнал широтно-імпульсної модуляції (ШІМ) та через який буде виконано більш детальний контроль швидкості обертання коліс;

– аналогові входи для підключення датчиків відстані або цифрові для використання датчиків наближення.

З описаних вище потреб можливо зрозуміти що на етапі реалізації отримання інформації про наближення до перешкоди є декілька рішень.

Одним з простих рішень є цифрові ІЧ датчики. Одним з їх плюсів є потреба в підключенні до цифрових входів на платі яких більше ніж аналогових та нижча ціна в порівнянні з іншими видами датчиків наближення. Однак одним з їх недоліків є те що вони не мають декількох значень а лише передають значення логічної 1 на вхід плати в разі наближення до перешкоди більше обраного значення. Прикладом цифрових ІЧ датчиків є ІЧ-відбивач TCRT5000 (рис. 3.1).

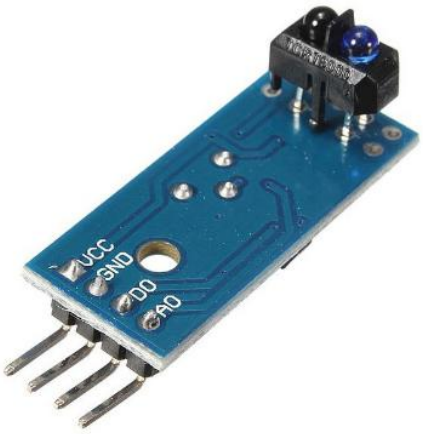


Рисунок 3.1 – ІЧ-відбивач TCRT5000

Датчик TCRT5000 працює в інфрачервоному діапазоні. Дозволяє впевнено визначати наявність перешкоди на відстані від 1 мм до 25 мм.

Наявність компаратора дозволяє встановити граничне значення спрацьовування.

Його характеристики описано в таблиці 3.1.

Таблиця 3.1 – Характеристики ІЧ-відбивача TCRT5000

Характеристика	Значення
Впевнене визначення перешкоди	1 до 25 мм
Робоча напруга	від 3,3 до 5 В
Розміри	3.2x1.4 см

Також можливо використовувати більш складні ІЧ-датчики відстані.

Головним їх плюсом є можливість отримувати детальну інформацію про дистанцію до перешкоди. Головним же мінусом є набагато більша ціна в порівнянні з датчиками наближення. Їх використання більш доцільне при реалізації описаного алгоритму тому як показано на рис 2.1 видно, що є потреба в отриманні даних про відстань до перешкоди. Простим прикладом подібного датчику відстані є лазерний датчик відстані TOF (time of flight) UART/I2C від Waveshare (рис. 3.2).

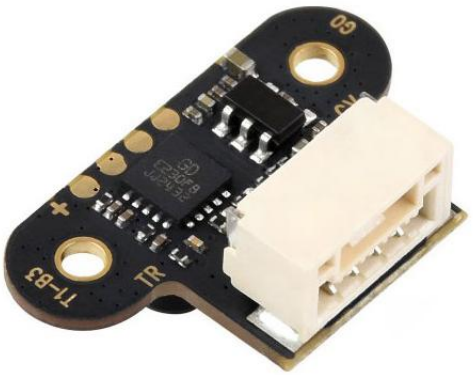


Рисунок 3.2 – Зовнішній вигляд датчику відстані UART/I2C

Це лазерний датчик відстані на основі TOF (час прольоту) з вбудованим мікроконтролером і алгоритмом вимірювання, діапазон вимірювання якого становить до 7.8 м що набагато більше потреб алгоритму, а точність вимірювання - до ±4 см. Він підтримує комунікаційні шини UART або I2C, відрізняється великою відстанню вимірювання і більш високою стійкістю до світлових перешкод зовнішніх умов. А його стійкість до навколишнього освітлення складає до 100 тис. Люкс. Характеристики датчику UART/I2C показано на таблиці 3.2.

Таблиця 3.2 – Характеристики датчика відстані UART/I2C

Характеристика	Значення
Типовий діапазон виміру	від 0.02 м до 7.8 м
Довжина хвилі випромінювача	940 нм
Поле зору (Field Of View FOV)	2 ° ~ 3 °
Напруга живлення	від 4.3 В до 5.2 В
Потужність	100 мВт
Розміри	18.8 мм × 12 мм × 10.3 мм (Д× Ш × В)

Двигуни робота повинні мати можливість динамічно змінювати свою швидкість тож потрібно буде використовувати драйвер на який подається ШІМ сигнал, який буде перетворювати команди з плати на команди для двигунів. Драйвери допомогають перетворювати ШІМ сигнал з плати на команди для керування двигунами робота. Прикладом такого драйвера є драйвер двоканальний двигунів на TB6612FNG (рис. 3.3).

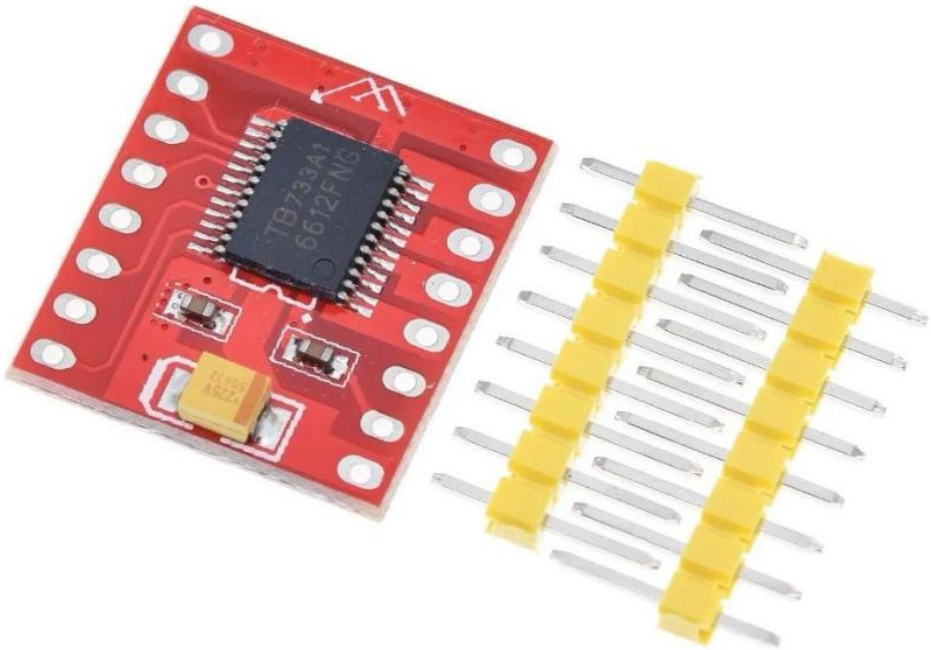


Рисунок 3.3 – Драйвер двоканальний двигунів на TB6612FNG

Модуль драйвера керування двома колекторними двигунами на мікросхемі TB6612FNG містить усі необхідні компоненти для керування двома колекторними двигунами та не вимагає використання додаткових компонентів. Розміри плати 21 мм х 18 мм х 3 мм.

Драйвер підтримує такі режими роботи: Вперед, Назад, Гальмування та Зупинка. Регулює швидкість обертання кожного з двигунів за допомогою ШІМ-регулювання з максимальною частотою 100КГц.

Після обирання можливих використовуємих блоків потрібно розглянути різні види плат Arduino, порівняти їх характеристики та обрати плату яка підходить під описані вище блоки а саме, плата потребує мінімум 17 цифрових входів/виходів та 2 ШІМ а також можливість підключення живлення. Нижче розглянуто декілька плат та їх характеристики.

Arduino Mega є одною з мікросхем яка базується на мікроконтролері ATmega2560 (рис. 2.4. Вона має 54 цифрових входів/виходів, з яких 14 можуть використовуватися для ШІМ, а також має 16 аналогових входів і 4 послідовних порти UART. Крім цього, плата містить кварцовий генератор на 16 МГц, USB-конектор, роз'єм живлення, роз'єм ICSP та кнопку перезавантаження.

Є можливість підключення комп'ютера через USB-кабель або забезпечити живлення за допомогою адаптера AC/DC чи акумулятора.

Зовнішній вигляд Arduino Mega показано на рисунку 3.4. Основні характеристики наведено в таблиці 3.3.

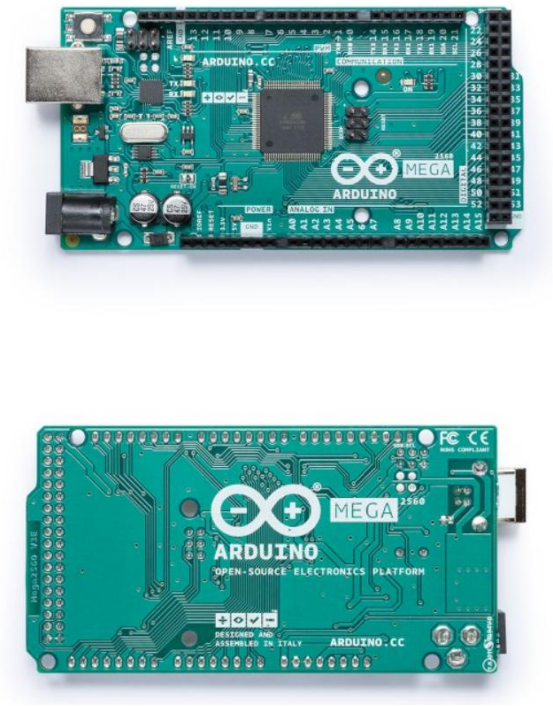


Рисунок 3.4 – Зовнішній вигляд Arduino Mega

Таблиця 3.3 – Основні характеристики Arduino Mega

Мікроконтролер	ATmega2560
Кількість цифрових входів/виходів	54
Кількість аналогових входів	16
Робоча напруга	5В
Тактовая частота	16 МГц
ОЗУ	8 Кб
Флеш-пам’ять	256 Кб
Розмір	10.15 см х 5.33 см

Мікроконтролер Arduino Nano, яка базується на ATmega328 або ATmega168 , на відміну від інших видів Arduino має невеликі розміри та меншу кількість пам’яті.. Основні відмінності полягають у відсутності роз'єму для живлення постійного струму та використанні кабелю Mini-B USB для підключення. Платформа Nano була розроблена та продається компанією Gravitech. Зовнішній вигляд Arduino Nano показано на рисунку 3.5. Основні характеристики наведено в таблиці 3.4.

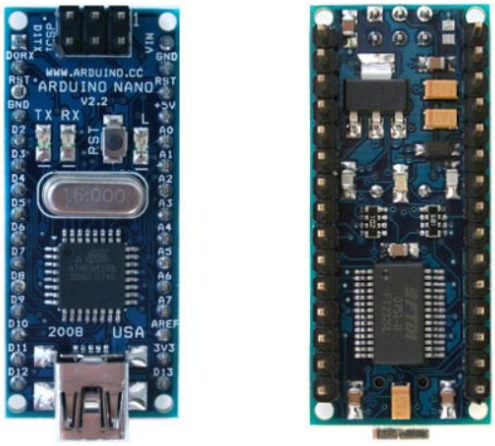


Рисунок 3.5 – Зовнішній вигляд Arduino Nano

Таблиця 3.4 – Основні характеристики Arduino Nano

Мікроконтролер	Atmel ATmega168 або ATmega328
Кількість цифрових входів/виходів	14
Кількість аналогових входів	8
Робоча напруга	5В
Тактовая частота	16 МГц
Флеш-пам’ять	16 Кб (ATmega168) або 32 Кб (ATmega328)
ОЗУ	1 Кб (ATmega168) або 2 Кб (ATmega328)
Розмір	1.85 см x 4.2 см

Мікроконтролер Arduino Uno, який базується на ATmega328. Має 14 цифрових входів/виходів, з яких 6 можуть використовуватися для ШІМ, і 6 аналогових входів. Він працює частоті 16 МГц, оснащений роз'ємом USB для підключення до комп'ютера, силовим роз'ємом для живлення, роз'ємом ICSP для програмування та кнопкою перезавантаження (рис. 3.6). Основні характеристики наведено в таблиці 3.5.

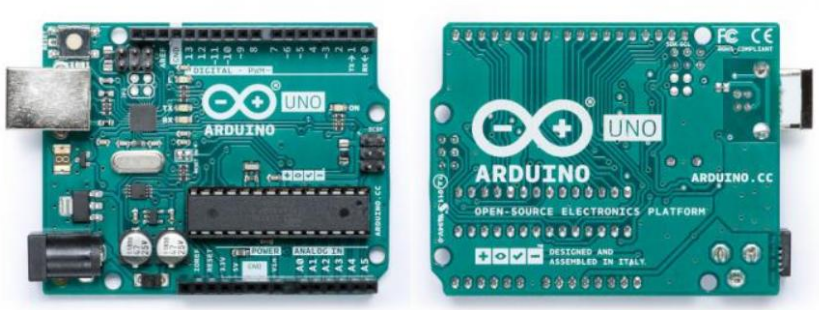


Рисунок 3.6 – Arduino UNO

Таблиця 3.5 – Основні характеристики Arduino UNO

Мікроконтролер	ATmega328
Кількість цифрових входів/виходів	14
Кількість аналогових входів	6
Робоча напруга	5В
Тактовая частота	16 МГц
ОЗУ	2 Кб
Флеш-пам’ять	32 Кб
Розмір	6.8 см х 5.3 см

Після аналізу описаних плат видно що з описаних плат підходить плата Arduino Mega через наявність у неї великої кількості аналогових входів/виходів та підтримки ШІМ сигналу.

3.1 Розробка програмного забезпечення пересування робота

Як об’єкт який може буди використовуватися для проведення симуляції було обрано існуючого робота e-puck через наявність в ньому потрібних датчиків та його простоти що дозволить сконцентруватися саме на розробці алгоритму.

E-puck – це невеликий мобільний робот, розроблений у Швейцарському федеральному технологічному інституті (EPFL) для освітніх та дослідницьких цілей. Це диференціальний робот із двома незалежно керованими колесами.

Зовнішній вигляд робота e-puck показано на рисунку 3.7.

Рисунок 3.7 – Зовнішній вигляд робота e-puck

Основні характеристики робота:

– розміри: 75 мм x 50 мм;

– вага: від 150 г до 200 г;

– датчики: 8 ІЧ-датчиків відстані по периметру;

– процесор: dsPIC 30F6014A @ 60 MHz;

– зв'язок: Bluetooth, ZigBee;

– акумулятор Li-Po (3.7V, 1000mAh).

В ході розробки алгоритму пересування зв’язок по Bluetooth, ZigBee не буде використовуватися так як на етапі тестування всі криві, їх контрольні точки та шлях пересування роботу будуть задаватися за замовчуванням.

Робот e-puck має 2 датчика переду, по 1му датчику з боків та 4 датчика позаду. Для зчитування найближчих перешкод будуть використовуватися 8 ІЧ датчиків. Їх розташування на роботі показано на рисунку 3.8.

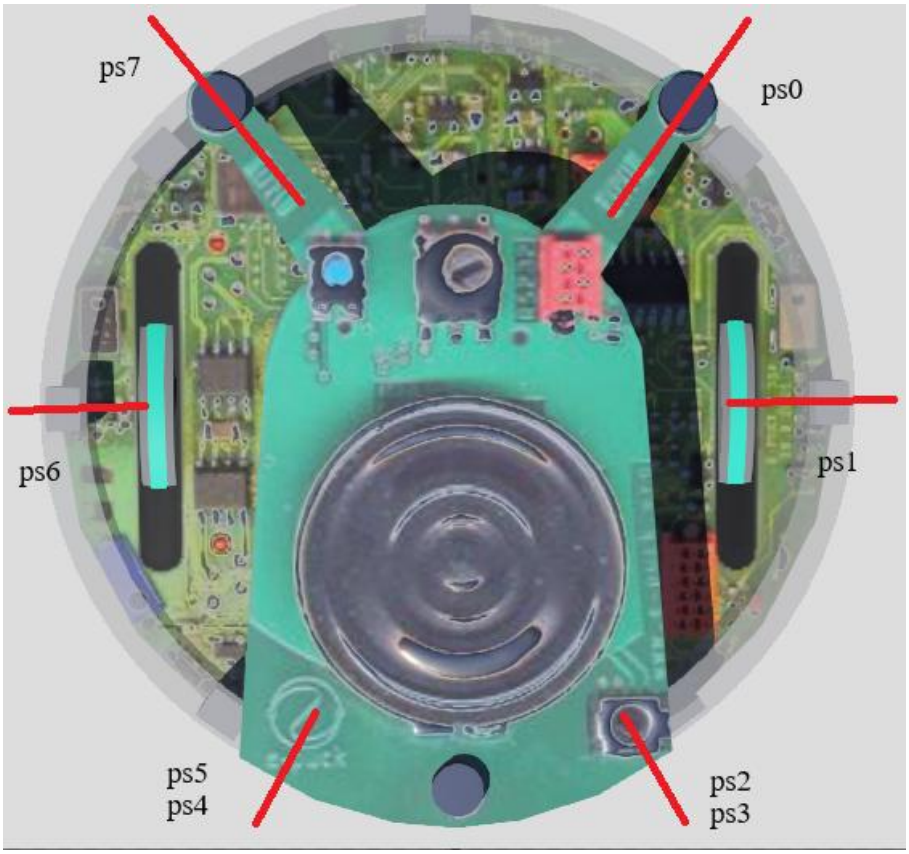


Рисунок 3.8 – Розташування ІЧ датчиків на роботі e-puck

Як видно з рис 3.8 попереду та позаду є сліпа зона між датчиками на що потрібно буде звернути увагу при написанні алгоритму пересування робота.

Основними плюсами робота, завдяки було його обрано є:

– простота конструкції, він керується двома колесами що спрощує реалізацію пересування робота;

– він має відкриту документацію що дозволяє полегшити реалізацію ПЗ

пересування робота.

Проаналізувавши основні критерії та потреби та обраного робота було розроблено ПЗ, яке складається з таких блоків:

– підключення потрібних бібліотек і класів;

– клас створення змінних та об’єктів класів;

– клас визначення контрольних точок траєкторії;

– функція завдання плавності переходів між кривими;

– функція побудови траєкторії з використанням кривих Безьє;

– функція отримання та обробки даних з датчиків;

– функція керування рухом;

– функція циклу навігації.

Підключення потрібних бібліотек описує використані бібліотек та об’єкти при написанні ПЗ які показані в таблиці 3.6.

Таблиця 3.6 – Імпортовані бібліотеки та об’єкти.

Назва бібліотеки чи об’єкта	Опис та призначення
Robot	Об’єкт який дозволяє отримати доступ до всіх пристроїв робота.
DistanceSensor	Клас ІЧ датчиків e-puck через який отримуються данні з датчиків в конкретний момент часу.
Motor	Клас двигунів що дозволяє керувати швидкістю та позицією коліс.
math	Стандартна бібліотека Python для математичних обчислень. Використовується для обчислення кута до цілі, повороту робота та побудови траєкторії.

Продовження таблиці 3.6.

Назва бібліотеки чи об’єкта	Опис та призначення
time	Стандартний модуль Python, який використовується для невеликих затримок, отримує поточний час, якщо потрібно виміряти тривалість маневрів (наприклад, від’їзд назад).

Клас задання використовуємих підкласів складається з створення об’єктів та задання їм базових значень для елементів які будуть використовуватися в подальшому. Створені змінні та їх використання описано в таблиці 3.7.

Таблиця 3.7 – Створі змінні для ПЗ

Змінна	Опис та	Значення за замовчуванням
self.robot	Об’єкт класа робот який потрібний для доступу до елементів робота.	
self.ps_names	Масив елементів датчиків робота, завдяки якому буде отриманий доступ до значення відстані до найближчої перешкоди.	
self.left_motor, self.right_motor	Об’єкти двигунів завдяки яким буде реалізовуватися рух робота	

Продовження таблиці 3.7.

Змінна	Опис та	Значення за замовчуванням
self.wheel_radius , self.axle_length, self.max_speed	Змінні які описують колеса та максимальну швидкість двигунів.	0.0205, 0.052, 6.28
self.path_points self.current_path_index self.current_curve_index self.estimated_pos self.estimated_angle self.last_time self.robot.getTime	Описання змінних, які будуть використовуватися в майбутньому	

Змінна self.ps_names являє собою масив елементів ІЧ датчиків робота, кожен з яких відповідає за свій напрям. Данні з датчиків будуть отримуватися у вигляді числового значення. Отримання датчиків виконується через функцію об’єкта класа Robot «self.robot.getDevice». Так як потрібно задавати значення при якому буде змінено траєкторію через завелике наближення до перешкоди було створено додаткові 2 змінні:

– self.safety_distance, значення за замовчуванням дорівнює 400. Змінна яка описує потрібну відстань при якій робот буде виконувати об’їзний маневр для оминання перешкоди, тобто якщо об’єкт знаходиться ближче ніж 400 одиниць датчику то робот буде вирішувати в яку сторону йому повертати щоб оминути перешкоду;

– self.critical_distance, значення за замовчуванням дорівнює 200. Змінна яка описує критичне значення дистанції в якому робот розуміє що він не може плавно оминути перешкоду. В цьому разі він від’їде назад по дузі і продовжне рух далі до точки.

Змінні двигунів аналогічно змінним датчиків отримуються через функцію об’єкта класа Robot «self.robot.getDevice». Але на відміну від датчиків в функцію «self.robot.getDevice» передаються назви коліс. Після отримання об’єктів колес їм задається стартова позиція для більш легкого змінювання позиції.

Змінна self.wheel_radius являє собою радіус одного колеса робота e-puck (в метрах). Використовується щоб перевести швидкість обертання двигуна в лінійну швидкість робота.

Змінна self.axle_length, це між центрами двох коліс робота (в метрах). Потрібна щоб обчислити, наскільки робот повертає за різниці швидкостей коліс. Без цього робот зможе правильно розраховувати свій поворот і кут.

Змінна self.max_speed є максимальною кутовою швидкістю моторів у Webots (рад/с). Обмежує команди на мотори, щоб робот не прискорювався вище за допустиме. Використовується при розрахунку базової швидкості руху. Без обмеження робот намагався б їхати швидше, ніж дозволяє механіка.

Після опису створення змінний йде блок класу визначення потрібних траєкторій. В цьому блоці використовується змінна self.curve_sequence в яку записується послідовність контрольних точок за якими будуть будуватися криві. Найбільшим мінусом є те що всі потрібні точки задаються вручну в коді.

В блоці функції задання плавності переходів між кривими до функції передаються контрольні точки минулої та поточної кривої, після чого перша точка поточної кривої дорівнює остання точка минулої кривої. Далі перша точка змінюється за (3.1):

$$P_{1\text{пот}} = P_{0\text{пот}} + (P_{0\text{пот}} - P_{2\text{поп}}), \tag{3.1}$$

де $P_{1\text{пот}}$ – друга точка поточної кривої;

$P_{0\text{пот}}$ – перша точна поточної кривої;

$P_{2\text{поп}}$ – третя точка попередньої кривої.

В блоці функції побудови траєкторії з використанням кривих Безьє створюється траєкторія за якою буде пересуватися робот. Блок схема алгоритму роботи цієї функції показана на рисунку 3.9.

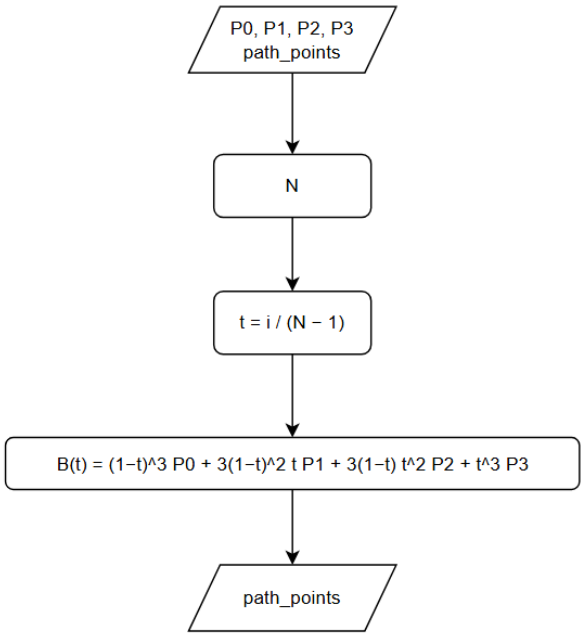


Рисунок 3.9 – Блок схема алгоритму роботи функції побудови траєкторії

Спочатку у вхідних даних у функцію передається 4 контрольні точки, після чого визначається кількість дискретних точок N, яка дорівнює 4 так як траєкторія будується по квадратичній кривій Безьє. Після чого створюється пустий список в який записується координати точки на кубічній кривій Безьє по (2.1) в залежності від часу. Після обчислення траєкторії вже повний масив точок координат на кубічній кривій повертається вихідними даними.

Функція отримання та обробки даних з датчиків є основною функцією, завдяки якій виконується зміна траєкторії робота зволікаючи на появу об’єктів на перешкоди на шляху робота. Саме ця функція на основі отриманих даних з датчиків вирішує, чи потрібно оминати перешкоду чи від’їзджати назад з подальшою перебудовою траєкторії.

Вирішування дій виконується шляхом порівняння даних, отриманих з датчиків відстані з значеннями критичної відстані та безпечної відстані.

В разі більшого значення з окремих датчиків ніж значення безпечної відстані вирішується повернути роботом в протилежну від датчика сторону. Тобто в разі більшого значення з датчика попереду справа робот вирішує повернути ліворуч і навпаки.

В разі же значення більшого ніж критичне запускається інша частина алгоритму яка відповідає за частковий від'їзд назад з подальшою перебудовою траєкторії для подальшого руху до кінцевої точки.

Функція керування рухом відповідає за забезпечення рухом робота за заданою траєкторією, передачі інформації з датчиків дистанції на функцію обробки даних з датчиків. Саме ця функція запитує отримує дані траєкторії з функції побудови траєкторії та забезпечує рух по потрібній траєкторії. Вона розраховує кут, на який потрібно повернутись роботу та швидкість обертання двигунів.

Функція циклу навігації є основною функцією, яка взаємодіє з усіма функціями, зв'язаними з керуванням руху, контролю за обминанням перешкод, обміном даних між функціями та їх обробкою. Також для більш простого контролю виконуємих функцій в процесі тестування ПЗ виводить у консоль інформацію про поточний стан пересування між кривими та помилки які трапляються під час руху.

3.2 Проведення симуляції роботи розробленого алгоритму

Як було описано раніше, для симуляції роботи написаного програмного забезпечення використовується програма Webots та робот e-puck. Через невеликий розмір робота e-puck всі кімнати було зменшено в декілька разів для зменшення часу симуляції. В ході виконання симуляції було умовну кімнату з перешкодами у програмному середовищі Webots яка описує коридор з декількома перешкодами що дозволить протестувати плавність побудованої траєкторії та можливість обминання перешкод роботом та можливість перебудови траєкторії в випадку зустрічі з перешкодою. Відображення кімнати показано на рисунку 3.10.

Рисунок 3.10 – Кімната коридор з перешкодами

Умовне відображення теоретичної траєкторії руху в даній кімнаті показано на рисунку 3.11.

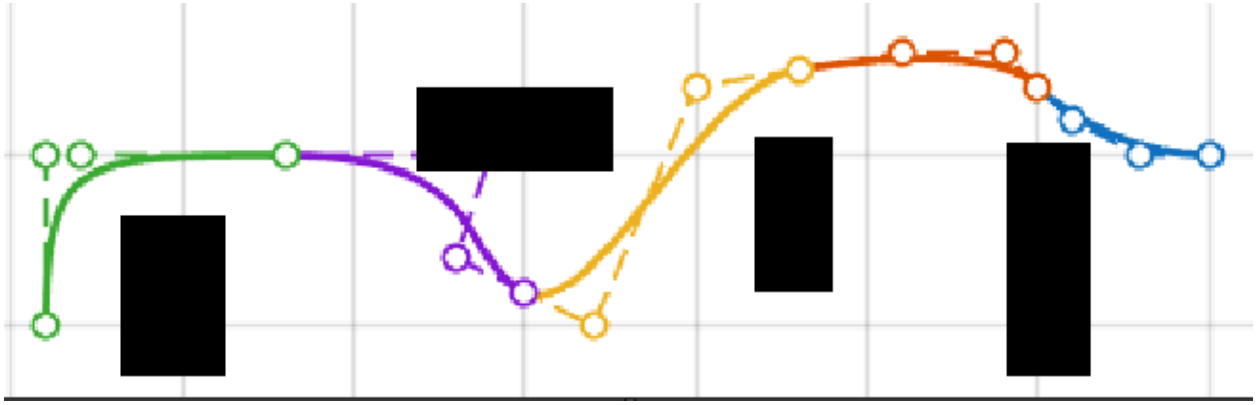


Рисунок 3.11 – Приблизна траєкторія руху робота

На рисунку 3.11 показано траєкторію руху від початку(синя крива) до кінця(зелена крива) при тестуванні під час симуляції першої кімнати.

Траєкторія пересування робота показана кривими різних кольорів, кожна з яких відповідає за свою ділянку руху, коли як чорними прямокутниками показано перешкоди в кімнаті.

Після початку руху робот приїде у позицію над першою перешкодою. Після повороту за неї він перейде до другої кривої і дале слідуючи за нею робот буде переміщатися до кінцевої кривої. Кінцеві точки розташування робота між кривими показано на рисунках 3.12 – 3.14.

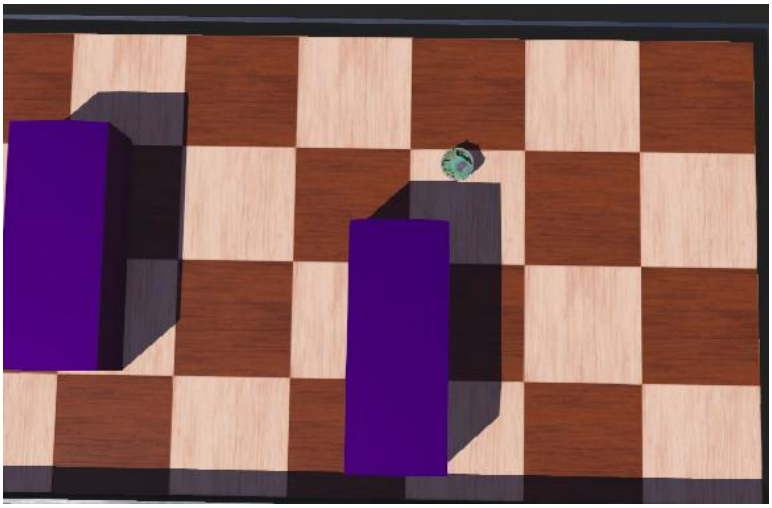


Рисунок 3.12 – Розташування робота на кінцевій точці 1-ї кривої

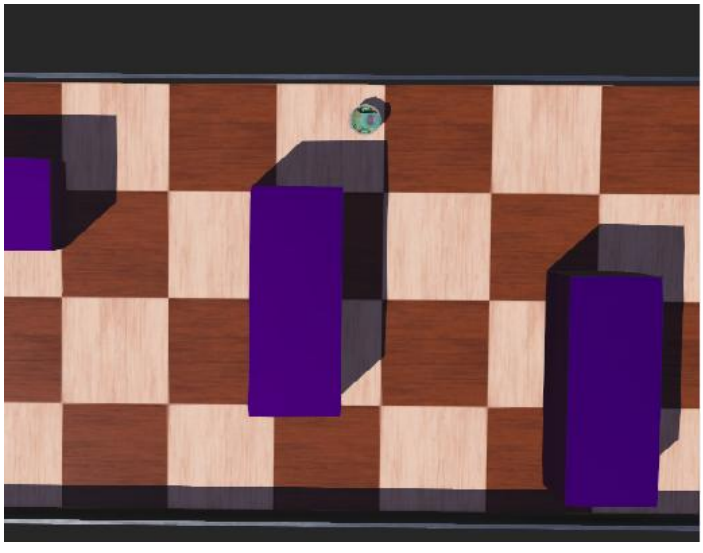


Рисунок 3.13 – Розташування робота на кінцевій точці 2-ї кривої

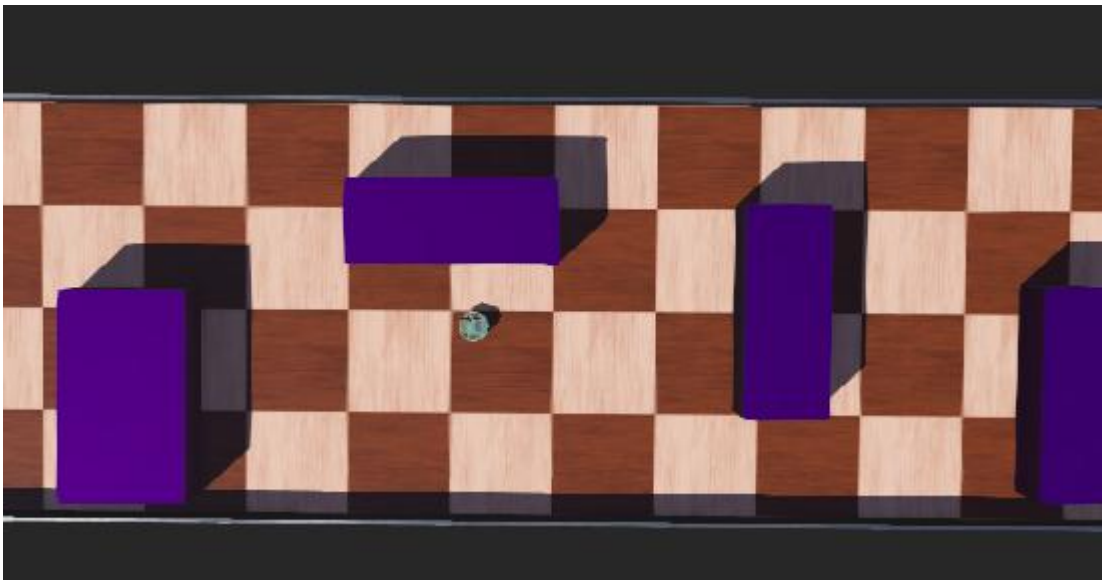


Рисунок 3.14 – Розташування робота на кінцевій точці 3ї кривої

Для біль коректної симуляції роботи алгоритму пересування при заданні точок кривих було навмисне вказано неправильні точки щоб перевірити можливість робота перебудовувати траєкторію у випадках зіткнення з перешкодами. Момент потенційного зіткнення з перешкодою показано на рисунку 3.15.

Рисунок 3.15 – Момент до зіткнення з перешкодою

Після зупинки перед зіткненням з перешкодою робот зупиняється, від’їжає назад та продовжує рух до кінцевої точки на кривій, що показано на рисунку 3.16.

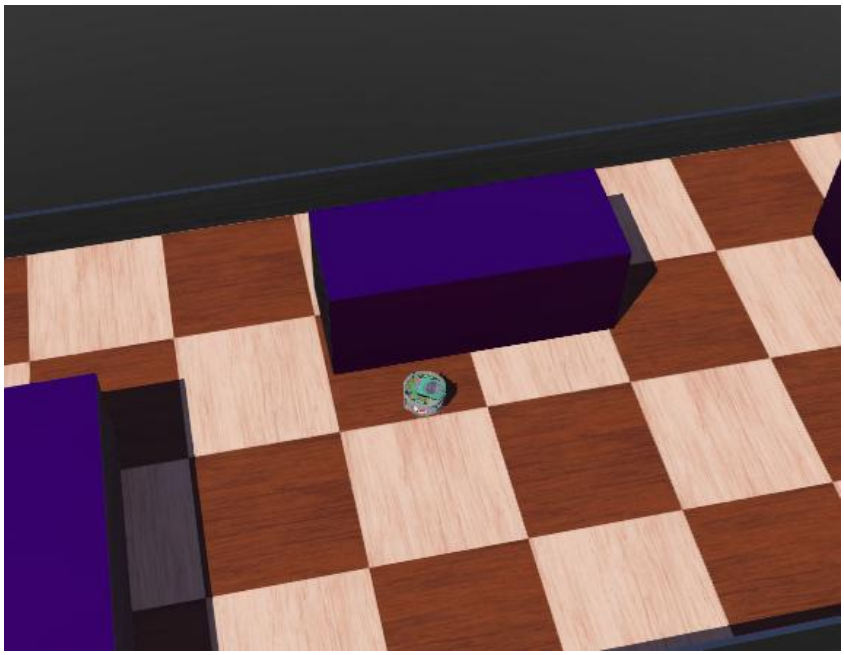


Рисунок 3.16 – Продовження руху до кривій за описаною траєкторією

Після зміни траєкторії мобільний робот перейде до кінцевої траєкторії та доїде до кінця кривих (рис. 3.17) після чого зупиниться.

Рисунок 3.17 – Кінцева точка останньої кривої

На протязі всієї роботи ПЗ в консоль виводився поточний прогрес по кривій, номер поточної кривої, її процент завершення та в разі перешкод, зміни траєкторії виводиться команда, яка виконується. На рисунку 3.19 показано прогрес кривої 1 відображений у консолі.

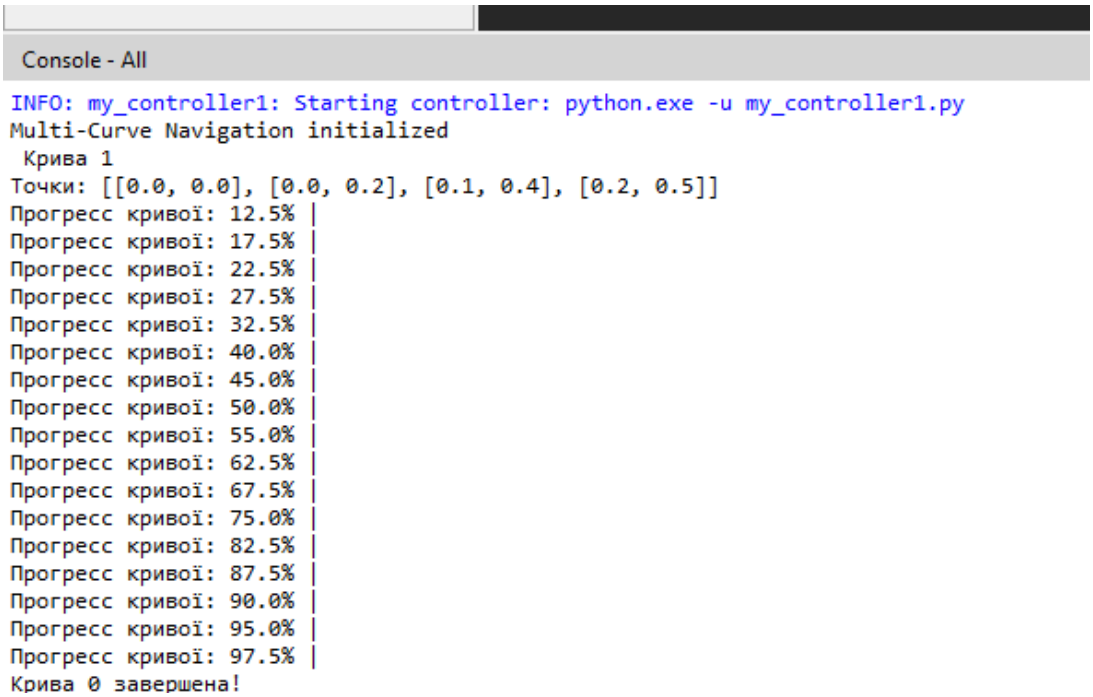


Рисунок 3.18 – Відображення прогресу кривих

На рисунку 3.19 показано консоль в випадку помилки при виконанні маршруту.

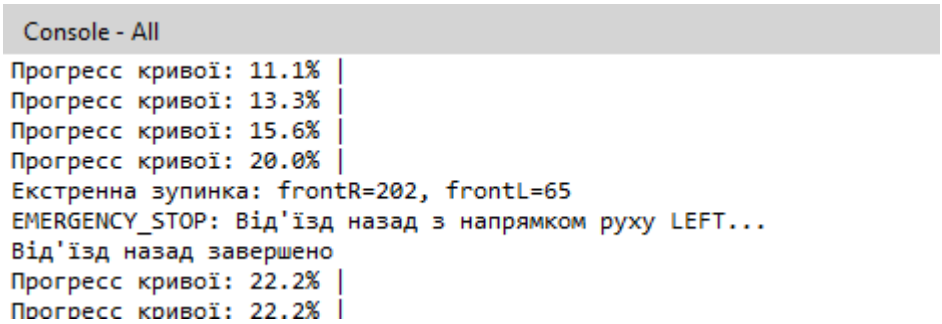


Рисунок 3.19 – Відображення помилки в консолі

3.3 Розрахунок стійкості системи

Стійкість методу підвищення точності пересування мобільних роботів за допомогою адаптованих кривих Безьє забезпечується здатністю системи забезпечувати достатній рівень точності в умовах динамічного навколишнього середовища.

При проведенні симуляції було виявлено деяку неточність пересування

робота до заданої траєкторії. Це може бути зв’язано за некоректною передачею команд на двигуни або через саму слизькість двигунів робота e-puck.

Задня коректної передачі заданої швидкості обертання робота можливе використання ПІД-регулятора де буде додано датчик який буде зчитувати абсолютний кут, на який був повернутий робот та дані з якого будуть зчитуватися та оброблятися задля корекції руху робота.

Для розрахування потрібно дізнатися основні характеристики ПІД регулятора. Він складається з 3-ох коефіцієнтів: P (пропорційний), I (інтегральний) та D (диференційний). Від них залежать характеристики системи час її відгуку та точність. Спочатку потрібно налаштувати ці коефіцієнти [10].

Одним із найпопулярнішим методом налаштування Під-регулятору є метод Зіглера-Ніколаса. Для налаштування параметрів потрібно задати коефіцієнти K_I та K_D рівними 0. Після потрібно поступово збільшувати K_P того моменту, поки система не почне коливатися, це значення називається критичним коефіцієнтом(K_{cr}), потім зменшуємо на 50%.

Поступово збільшуємо K_I до моменту, поки система не усуне постійну похибку.

Збільшуємо K_D до мінімального рівня, де система стане стабільною і не буде мати великих коливань. Після дослідження критичний коефіцієнт K_{cr} = 3.3 а період коливань T = 2. Далі потрібно розрахувати коефіцієнт ПІД-регулятора за формулами 3.2-3.4:

Розрахунок $\boldsymbol{K_P}$ за формулою 3.2:

$$K_P = 0{,}6 \cdot K_{cr}, \tag{3.2}$$

$$K_P = 0{,}6 \cdot 3{,}3 = 1{,}98.$$

Розрахунок $\boldsymbol{K_1}$ за формулою 3.3:

$$\boldsymbol{K_I = \frac{2 \cdot K_P}{T}}, \tag{3.3}$$

$$\boldsymbol{K_I = \frac{2 \cdot 1,98}{2} = 1,98}.$$

Розрахунок $\boldsymbol{K_1}$ за формулою 3.3:

$$\boldsymbol{K_D = \frac{K_P \cdot T}{8}}. \tag{3.4}$$

$$\boldsymbol{K_D = \frac{1,98 \cdot 2}{8} = 0,495}.$$

Отримані параметри:

$$\boldsymbol{K_P = 1,98},$$

$$\boldsymbol{K_I = 1,98},$$

$$\boldsymbol{K_D = 0,495}.$$

Для використання ПІД регулятору в ПЗ керуванням робота потрібно буде змінити принцип роботи функції зміни траєкторії руху пересуванням робота з урахуванням даних, отриманих з датчику зміни кута повороту.

Сам же датчик може бути змінений на датчик повороту, який буде зчитувати швидкість обертання двигунів та змінювати їх в залежності віт даних з датчика. Будуть додано змінні які будуть відповідати за коефіцієнти ПІД регулятору та зрівнювання даних отриманих з датчиків робота. Додавання ПІД регулятору до цього блоку забезпечить високу точність пересування робота та зменшить неточності в його повороті до мінімуму.

3.4 Питання охорони праці та безпеки життєдіяльності

Охорона праці є важливим аспектом будь-якої виробничої діяльності, спрямованої на забезпечення безпечних умов праці, захисту здоров'я та життя працівників, а також на запобігання травматизму та професійним захворюванням.

Питання охорони праці робота використовуючого метод побудови шляху пересування робота залежить насамперед від сфери використання цього робота.

В випадку використання робота в закритих приміщеннях та керуванням точок пересування робота одним із основних факторів охорони праці при роботі мобільного робота є питання охорони праці оператора, який буде передавати дані на робота та питання безпеки життєдіяльності в приміщенні, в якому працює робот. В цьому приміщенні повинна бути забезпечена пожежна безпека в разі займання електронних компонентів робота чи інших приладів.

Приміщення повинно бути сухим, опалюваним та добре вентильованим, температура повітря повинна знаходитись у межах від 18 °С до 24 °С, відносна вологість від 40 % до 60 %, рівень шуму в приміщенні не повинен перевищувати
50 дб. Приміщення повинно мати достатню площу для вільного розміщення робочого місця оператора, комп’ютерного обладнання та допоміжних пристроїв.

Усі електричні пристрої (комп’ютер, мережеве обладнання, блоки живлення) повинні бути заземлені відповідно до вимог електробезпеки. Кабелі живлення та передачі даних повинні мати ізоляцію без пошкоджень та бути прокладені таким чином, щоб уникнути механічних навантажень і спотикання.

Приміщення повинно бути оснащене первинними засобами пожежогасіння (вуглекислотний або порошковий вогнегасник). Забороняється використання пошкоджених розеток, подовжувачів та кабелів. Робоче місце

повинно бути розташоване таким чином, щоб забезпечити вільний доступ до евакуаційних виходів.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи на кафедрі комп’ютерно-інтегрованих технологій, автоматизації та робототехніки Харківського національного університету радіоелектроніки була проведена наукова діяльність, спрямована на розробку алгоритму пересування робота з метою підвищення точності пересування мобільних роботів за допомогою адаптованих кривих Безьє.

За результатами розроблено програмне забезпечення для робота з використанням адаптованих кривих Безьє для побудови траєкторії. Використано середовище Webots для моделювання роботи програмного забезпечення.

Було розроблене програмне забезпечення яке відповідає всім вимогам алгоритму пересування. Воно включало в себе можливість плавного переходу між траєкторіями, можливість обминання перешкод.

Було розраховано ПІД регулятор для вдосконалення роботи алгоритму пересування робота для підвищення точності пересування робота.

В результаті проведеної роботи був створений алгоритм пересування робота, який відповідає всім описаним вимогам. Розроблене програмне забезпечення забезпечує надійне та автономне керування роботом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Demska, N., Yevsieiev, V., Maksymova, S., & Ababneh, J. (2025). Decision-Making Model For Controlling A Collaborative Robot-Manipulator Based On The Sensor Fusion Method And Cnn Approach To Rule

Formation. *Multidisciplinary Journal of Science and Technology*, *5*(6), 846-859.

2. Demska, N., Yevsieiev, V., Maksymova, S., & Alkhalaileh, A. (2025). Analysis of Methods, Models and Algorithms for a Collaborative Robots Group Decentralized Control. *ACUMEN: International journal of multidisciplinary research*, *2*(2), 235-249. Șomîtcă, I. A., Brad, S., Florian, V., & Deaconu, Ș. E. (2022). Improving path accuracy of mobile robots in uncertain environments by adapted Bezier curves. *Electronics*, *11*(21), 3568.

3. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2024. 57 с.

4. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП «УкрНДНЦ». 2016. 30 с

5. Wang, D., Liu, Q., Yang, J., & Huang, D. (2024). Research on path planning for intelligent mobile robots based on improved A* algorithm. *Symmetry*, *16*(10), 1311.

6. Duraklı, Z., & Nabiyev, V. (2022). A new approach based on Bezier curves to solve path planning problems for mobile robots. *Journal of computational science*, *58*, 101540.

7. Lai, R., Wu, Z., Liu, X., & Zeng, N. (2023). Fusion algorithm of the

improved A* algorithm and segmented bezier curves for the path planning of mobile robots. *Sustainability*, *15*(3), 2483.

8. Xu, Y. (2024). Improved A* algorithm based on a dynamic parameter and the Bezier curve. In *AIP Conference Proceedings* (Vol. 3144, No. 1, p. 050008). AIP Publishing LLC.

9. Chatzisavvas, A., & Dasygenis, M. (2025). Enhancing the A* Algorithm for Efficient Route Planning in Agricultural Environments with a Hybrid Heuristic Approach and Path Smoothing. *Technologies*, *13*(9), 389.

10. Zheng, Z. A., Xie, S., Ye, Z., Zheng, X., & Yu, Z. (2025). Research on path smoothing optimisation based on improved RRT-Connect algorithm and third-order Bezier curve. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *239*(7), 2544-2561.

11. Machmudah, A., & Parman, S. (2021). Bezier curve collision-free route planning using meta-heuristic optimization. *Int. J. Artif. Intell. Robot*, *3*(1), 1-14.

12. Min Lin, X., Xia, L., Ge, Z., Hong, X., Zhao, R., & Jalalnezhad, M. (2025). Designing the optimal path curve based on spline functions for mobile robot using the combination of bee colony algorithm and genetic algorithm. *Journal of Vibration and Control*, *31*(7-8), 1359-1376.

13. Qin, T., Chu, H., Wang, J., Ren, P., Fu, M., & Wu, X. (2024, July). Mobile Robot Path Planning Method Based on Weight Coefficient Improved A* Algorithm. In *International Conference on Intelligent Robotics and Applications* (pp. 85-95). Singapore: Springer Nature Singapore.

14. Chen, T., Cai, Y., Chen, L., & Xu, X. (2022). Trajectory and velocity planning method of emergency rescue vehicle based on segmented three-dimensional quartic Bezier curve. *IEEE Transactions on Intelligent Transportation Systems*, *24*(3), 3461-3475.

15. Long, C., Shang, L., & Gao, H. (2024, November). Path Smoothing for USVs Using Adaptive Quadratic Bézier Curves. In *2024 4th International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)* (pp. 360-364). IEEE.

16. Yusuf, Z., Mohamad, S., & Ibrahim, W. S. W. (2025). Performance Comparison of A* and Dijkstra Algorithms with Bézier Curve in 2D Grid and OpenStreetMap Scenarios. *Journal of Applied Engineering Design and Simulation*, *5*(2), 79-89.

17. Zhu, W., & Chen, Z. (2025). Research on Path Planning for Mobile Charging Robots Based on Improved A* and DWA Algorithms. *Electronics*, *14*(12), 2318.

18. Li, L., Fu, Y., Yu, K., Alwakeel, A. M., & Alharbi, L. A. (2024). Optimal trajectory UAV path design based on bezier curves with multi-hop cluster selection in wireless networks. *Wireless Networks*, *30*(6), 5021-5032.

19. Основи охорони праці та безпеки життєдіяльності, Уманський державний педагогічний університет імені Павла Тичини, Баличева Н. В. УМАНЬ 2023, 273 с.

20. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» Навчальний посібник / І. Ш. Невлюдов, В.А. Андрусевич, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. – Харків : Видавництво Іванченка І. С., 2022. 151 с.

21. Марченко А.А., Гулий В.С. Теорія автоматичного керування. – Київ 2022. 30с.