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

рівень вищої освіти перший (бакалаврський)

Розробка програмного забезпечення для розрахунку маршруту мобільного робота-кур'єра при доставці замовлень
(тема)

Виконав:
здобувач 3 року навчання,
прискореної групи АКТАКІТу-22-1
Руслан ІЩЕНКО
(власне ім'я прізвище)
Спеціальності 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Автоматизація та комп`ютерно-інтегровані технології
(повна назва освітньої програми)
Керівник доцент Рауф АЛЛАХВЕРАНОВ
(посада, власне ім'я прізвище)

Допускається до захисту

Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я прізвище)

2025р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп`ютерно-інтегровані технології
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР _____
(підпис)
« 28 » квітня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Іщенко Руслану Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення для розрахунку маршруту мобільного робота-кур'єра при доставці замовлень
затверджена наказом по університету від " 19 " травня 2025р. № 405 Ст.
2. Термін подання здобувачем роботи " 25 " червня 2025 р.
3. Вихідні дані до роботи 3.1 Мобільний робот TurtleBot 4;
3.3 Хвильовий алгоритм;
3.4 Мова програмування – JavaScript;
3.5 Операційна система – Microsoft Windows 10.

4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ;
4.2 Аналіз технічного завдання та предметної області;
4.3 Розроблення маршруту переміщення робота-кур'єра;
4.4 Розроблення програмного забезпечення для розрахунку маршруту робота-кур'єра;
4.5 Охорона праці;
4.6 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 15 с. формату А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

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 технічного завдання та предметної області	28.04 – 04.05.25	виконано
2	Розроблення маршруту переміщення робота-кур'єра	05.05 – 14.05.25	виконано
3	Розроблення програмного забезпечення для розрахунку маршруту робота-кур'єра	15.05 – 31.05.25	виконано
4	Охорона праці	01.06 – 11.06.25	виконано
5	Оформлення пояснювальної записки	12.06 – 15.06.25	виконано
6	Подання роботи на перевірку Інтернет-системою StrikePlagiarism	16.06 – 18.06.25	виконано
7	Подання роботи на рецензію	19.06 – 21.06.25	виконано
8	Подання роботи на підпис зав. кафедри	22.06 – 24.06.25	виконано
9	Подання кваліфікаційної роботи в ЕК	25.06.25	виконано

Дата видачі завдання 28.04.2025р.

Здобувач _____ Руслан ІЩЕНКО
(підпис)

Керівник роботи _____ доцент Рауф АЛЛАХВЕРАНОВ
(підпис) (посада, власне ім'я прізвище)

Я, Іщенко Руслан Олександрович, як здобувач вищої освіти ХНУРЕ, розумію та підтримую політику закладу з академічної доброчесності. Я не надавав і неодержував 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.

«18» червня 2025 р.  Руслан ІЩЕНКО

РЕФЕРАТ

Пояснювальна записка: 68 с., 1 табл., 19 рис., 2 дод., 20 джерел.

МОБІЛЬНИЙ РОБОТ, РОБОТ-КУР’ЄР, TURTLEBOT 4, РОЗРАХУНОК, МАРШРУТ, ХВИЛЬОВИЙ АЛГОРИТМ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЗАМОВЛЕННЯ, ПЕРЕШКОДА.

Мета роботи – розроблення програмного забезпечення для розрахунку маршруту переміщення замовлення роботом-кур'єром з однієї точки в іншу з урахуванням перешкод.

Об’єкт розробки – орієнтація програмного забезпечення робота-кур’єра в певному просторі.

Предмет розробки – програмне забезпечення для розрахунку маршруту мобільного робота-кур’єра.

У кваліфікаційній роботі було проаналізовано існуючі види мобільних роботів. Як робота-кур'єра було обрано мобільного робота TurtleBot 4. Проаналізовано його технічні характеристики та розглянуто принципи керування даним мобільним роботом. Для імітації реальних умов було створено програмний макет середовища. Програмно реалізовано хвильовий алгоритм, за яким розраховується і будується маршрут робота-кур'єра. Розроблено програмне забезпечення для переміщення замовлення роботом-кур'єром з однієї точки в іншу з урахуванням перешкод. Розглянуто та опрацьовано питання з охорони праці.

Отримані результати роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», зокрема до пункту 9.4 «Розвиток високотехнологічного машинобудування».

ABSTRACT

Explanatory note: 68 pp., 1 tab., 19 figs., 2 appendices, 20 sources.

MOBILE ROBOT, COURIER ROBOT, TURTLEBOT 4, CALCULATION, ROUTE, WAVE ALGORITHM, SOFTWARE, ORDER, OBSTACLE.

The purpose of the work is to develop software for calculating the route of moving an order by a courier robot from one point to another, taking into account obstacles.

The object of development is the orientation of a mobile courier robot in a certain space.

The subject of development is software for calculating the route of a mobile courier robot.

In the qualification work, existing types of mobile robots were analyzed. The TurtleBot 4 mobile robot was chosen as the courier robot. Its technical characteristics were analyzed and the principles of controlling this mobile robot were considered. To simulate real conditions, a software model of the environment was created. A wave algorithm was implemented in software, according to which the courier robot route is calculated and built. Software was developed for moving an order by a courier robot from one point to another, taking into account obstacles. Occupational safety issues were considered and worked out.

The results of the work can be attributed to Sustainable Development Goal 9 “Industry, Innovation and Infrastructure”, in particular to paragraph 9.4 “Development of high-tech engineering”.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ДРП – дискретне робоче поле;

МР – мобільний робот;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

СУ – система управління;

ТЗ – технічне завдання.

ВСТУП

Розробка алгоритмів розрахунку маршруту мобільних роботів є актуальним завданням в сфері робототехніки. Актуальність роботи обумовлена широким розповсюдженням сімейства мобільних роботів в усіх сферах людської діяльності. Розроблене програмне забезпечення (ПЗ) цілком задовольняє потреби сучасних офісних мобільних роботів та є ефективним для застосування на практиці.

В даний час виконано велику кількість досліджень пов'язаних з розробкою управління та побудови маршруту мобільних роботів, в них входять як складання карти, так і побудова оптимального маршруту, проте мало уваги приділено траєкторіям з обмеженнями по швидкості в конкретних точках.

Складними обмеженнями для руху робота є перешкоди на його шляху, невпинний рух на поворотах, завдання різних швидкостей в точках маршруту робота.

Вибір теми кваліфікаційної роботи був зроблений на підставі актуальності, важливості та ефективності завдання для робототехніки в цілому. Завдання побудови маршруту мобільного робота з урахуванням координат точок маршруту робота і обмежень на довжину побудованого маршруту може мати широкий спектр практичних застосувань. Розроблене ПЗ може використовуватися у всіх існуючих офісних роботах задля покращення ефективності та простоти використання останніх.

Об’єкт розробки – орієнтація програмного забезпечення робота-кур’єра в певному просторі.

Предмет розробки – програмне забезпечення для розрахунку маршруту мобільного робота-кур’єра.

Мета роботи – розроблення програмного забезпечення для розрахунку маршруту переміщення замовлення роботом-кур'єром з однієї точки в іншу з урахуванням перешкод.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– проаналізувати методи алгоритму руху мобільних роботів;

– обрати мобільного робота;

– розробити детерміновану карту;

– розробити алгоритм програми розрахунку маршруту мобільного робота;

– розробити програмне забезпечення для розрахунку маршруту мобільного робота;

– розробити заходи та технічні з охорони праці для забезпечення безпеки праці працюючого персоналу лабораторії, де виконувалась кваліфікаційна робота.

Пояснювальну записку кваліфікаційної роботи оформлено згідно з ДСТУ 3008:2015 [1], а також з рекомендаціями з підготовки і оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти [2-3], отримані результати роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», зокрема до пункту 9.4 «Розвиток високотехнологічного машинобудування».

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ І ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сучасних мобільних роботів-кур'єрів

На сьогодні кур'єрська доставка в міських умовах стає актуальним завданням. Багато компаній приступили до розробки мобільних роботів-кур'єрів, здатних доставляти замовлення з однієї точки в іншу. У США великі компанії, такі як Amazon, Starship, CargoPod та інші, вже розробили власних мобільних роботів-кур'єрів. У Європі, зокрема у Франції, також уже розроблено кілька мобільних роботів-кур'єрів, це роботи: Ez-Pro автомобільного концерну Renault і TwinswHeel. Італійська компанія Piaggio, відома як розробник культових скутерів Vespa розробила робота-кур'єра Gita. Деякі китайські компанії також приступили до виробництва такого роду мобільних роботів-кур'єрів, наприклад, компанія DJ.com розробила робота JD, а компанія Segwa робота LoomoGo.

Безумовно, більшість компаній розробили мобільних роботів-кур'єрів, які працюють у міських умовах. Основні труднощі при розробці таких роботів пов'язані з впровадженням автономного управління, оскільки умови експлуатації передбачають безліч зовнішніх впливів, вплив яких важко передбачити. З цієї причини наявні рішення автономного керування такими роботами можуть застосовуватися тільки на гладких і рівних поверхнях.

Повна автоматизація процесів транспортування замовлень допоможе розв'язати проблеми, що виникають під час їхньої доставки, і значно скоротить матеріальні витрати, які несе кур'єрська служба [4].

Для подальших досліджень розглянемо найбільш популярні мобільні роботи-кур'єри провідних виробників, оберемо один із них як досліджуваний зразок згідно з ТЗ.

1.1.1 Мобільний робот-кур'єр Amazon Scout

Amazon Scout – це роботизований кур'єр, розроблений американською фірмою Amazon для автономної доставки посилок [5]. Amazon Scout являє собою шестиколісний візок, здатний долати перешкоди і доставляти посилки на адреси клієнтів (рисунок 1.1).

Рисунок 1.1 – Мобільний робот-кур'єр Amazon Scout

Amazon Scout використовує штучний інтелект і сенсори для самостійної навігації та обходу перешкод.

Роботи-кур'єри Scout використовуються для доставки товарів клієнтам Amazon, особливо в районах зі щільною забудовою і складною інфраструктурою.

Робот-кур'єр Scout здатний самостійно рухатися за заданим маршрутом, доставляти посилки, а також спілкуватися з клієнтами (наприклад, для отримання коду доступу до посилки).

Amazon також працює над іншими проектами в галузі робототехніки, включно з системами всередині складу для автоматизації сортування та обробки замовлень

Багато хто визнає, що надання послуг роботом-кур'єром Scout буде набагато оперативнішим і якіснішим, ніж взаємодія з кур'єром-людиною. Для покупців Amazon отримання замовлення не викликає складнощів. Для цього необхідно:

– вибрати товар;

– отримати PIN-код на смартфон і на електронну пошту;

– через пару годин ви отримаєте сигнал про приїзд робота;

– ввести PIN-код;

– відкрити слот і забрати посилку.

У тому випадку, якщо нікого не виявилося вдома, то нікому буде ввести PIN-код, клієнту надають можливість вибрати опцію «Залишити посилку біля дверей» [5].

Щоб не завдавати незручностей пішоходам, Скаут переміщається із середньою швидкістю ходьби. Він навіть зможе підлаштовуватися під швидкість потоку, прискорюючи або сповільнюючи рух. Це допоможе людям якомога швидше звикнути до нових гаджетів на дорогах міст.

1.1.2 Мобільний робот-кур'єр Starship

Робот-кур'єр Starship – це розроблений компанією Starship Technologies автономний наземний робот, який використовується для доставки замовлень на вимогу [6]. Він являє собою шестиколісний транспортний засіб, який може переміщатися вулицями і тротуарами, надаючи кур'єрські послуги як приватним особам, так і підприємствам (рисунок 1.2).

Рисунок 1.2 – Мобільний робот-кур'єр Starship

Starship – це перспективна технологія, яка може змінити спосіб, у який ми отримуємо товари та послуги. Автономна доставка, висока безпека та

екологічність роблять мобільних роботів Starship привабливим рішенням для багатьох підприємств і споживачів [6].

Основні характеристики Starship:

– робот керується ПЗ і операторами, а не окремими людьми, що робить його безпечним і ефективним;

– роботи Starship спроектовані для безпечного переміщення, з урахуванням безпеки пішоходів та інших учасників руху;

– роботи можуть переміщатися на швидкості до 6 км/год і перевозити вантажі вагою до 10 кг;

– Starship може доставляти замовлення на відстані до 6 миль (близько 9,6 км);

– роботи Starship вважаються дешевшими, ніж традиційні кур'єрські послуги, і можуть знижувати вартість доставки;

– Starship має високу надійність і безпеку, що дає йому змогу долати великі відстані та доставляти замовлення в різні міста і країни.

Компанія Starship Technologies використовує роботів Starship для доставки замовлень з різних магазинів, ресторанів та інших підприємств. Замовник може зробити замовлення через мобільний застосунок, обрати адресу доставки та відстежувати пересування робота. Робот доставить замовлення за вказаною адресою, після чого клієнт може забрати його, використовуючи код, який отримає через додаток.

Переваги роботів-кур'єрів Starship:

– можуть доставити замовлення протягом 15-30 хвилин;

– не вимагає прямого контакту з клієнтом, що може бути корисним у деяких ситуаціях;

– є екологічно чистими, тому що не продукують викидів та не споживають багато енергії;

– можуть використовуватися для доставки різноманітних товарів, включно з харчовими продуктами, напоями, ліками та багатьом іншим.

1.1.3 Мобільний робот-кур'єр Gita

Італійська компанія Piaggio, відома як розробник культових скутерів Vespa, представила світу оновлений дизайн персональних вантажних роботів Gita, свого роду «автономних мобільних валіз» [7]. Пристрій, висота якого становить близько 66 см, здатний нести на борту до 18 кг вантажів (рисунок 1.3).

Рисунок 1.3 – Мобільний робот-кур'єр Gita

Gita слідує за спеціальним поясом, який носить його власник, - крім того, пристрій вчиться на ходу і формує власну карту місцевості. Камера в поясі та метод SLAM (Simultaneous Localization and Mapping) формують тривимірну карту просто під час руху власника Gita [7].

Користувачі зможуть розмічати точки маршруту і посилати Gita в автономну подорож – камери та ультразвукові далекоміри дадуть змогу роботу уникнути зіткнень із перешкодами.

Назва Gita перекладається з італійської мови як «пікнік» або «коротка вилазка» – «робот» цікавий тим, що підтримує власника і відкриває масу можливостей, пов'язаних з підвищенням мобільності.

Уперше розробники представили Gita у 2017 році – відтоді вони оновили дизайн пристрою і обмежили його швидкість, скоротивши її з 35 км/год до приблизно 10 км/год. Заряду батарей вистачить на 4 год роботи

(в режимі повільної прогулянки цей період можна буде розтягнути на 8 год, до того ж заряджання займе всього 3 год). У багажному відділенні робота є роз'єми для зарядки мобільних пристроїв. Невеликий екран на кришці люка показує рівень батареї, що залишився, або виводить інші необхідні повідомлення.

Девайс став виглядати ще симпатичніше, італійці безумовно розуміються на промисловому дизайні. Незрозумілим залишається, чи може Gita впоратися з підйомом сходами, що з нею робити, коли потрібно піднятися на ескалаторі.

Уявіть собі ситуацію: ви виходите з супермаркету з трьома сумками в кожній руці, і вже відчуваєте, що от-от щось проґавите. Здорово було б у такі моменти мати під рукою робота-помічника, який зміг би взяти весь ваш вантаж на себе. Апарат на ім'я Gita може бути таким варіантом.

Робот був створений Piaggio Fast Forward, дочірньою компанією Piaggio Group, яка є відомою моторолерами Vespa. Piaggio Fast Forward описує Gita як «автономний транспортний засіб», який допомагає людині нести важкі пакети додому. Цей апарат своїм дизайном трохи нагадує робота-кур'єра від Starship Technologies, але відрізняється тим, що є персональним помічником, який постійно слідує за людиною, навіть у приміщеннях [7].

Пристрій заввишки 65 см здатний перевозити до 20 кг вантажу зі швидкістю понад 35 км на годину. Робот може керуватися або від людини-оператора, або пересуватися автономно в місцях із підготовленим маркуванням. Клієнт одягає спеціальний білий пояс із камерою на передній частині. Використовуючи технологію SLAM (локалізація і картографія), ця система створює 3D хмари з точками на карті оточення користувача. За допомогою цієї карти робот знає, куди йому йти.

1.1.4 Мобільний робот-кур'єр TwinswHeel H04

Фирма TwinswHeel разрабатывает и производит дроиды, которые вносят свой вклад в будущее городской логистики. В первую очередь разработка сосредоточена для закрытых объектов, таких как крупные промышленные площадки, а вскоре и в городах с электронной коммерцией и магазинами у дома. Эти дроиды основаны на автономном транспортном средстве, чтобы преуспевать в самостоятельном бизнесе по доставке заказов.

TwinswHeel H04 – это умный, гибкий и эффективный дроид для круглогодичной доставки заказов 24/7 (рисунок 1.4).

Цей мобільний робот оснащений 4 колесами і спеціальною системою керування, яка дозволяє йому проходити через різні види місцевості та долати перешкоди, зокрема на тротуарах або між іншими об'єктами.

Рисунок 1.4 –Мобільний робот-кур'єр TwinswHeel H04

Основні характеристики включають:

– автономність: здатність працювати без людської участі, використовуючи технології GPS, сенсори та камери для орієнтування в середовищі.

– мобільність: через 4 колеса та маневрену конструкцію він може комфортно пересуватися в умовах міста.

– навантаження: здатний перевозити вантаж до 30 – 40 кг.

– безпека: робот оснащений системами для виявлення перешкод і уникнення зіткнень.

Це чудовий приклад того, як автоматизовані технології можуть змінювати сферу доставки, знижуючи навантаження на кур'єрів і підвищуючи ефективність.

1.1.5 Мобільний робот-кур'єр TurtleBot 4

TurtleBot 4 – це одна з останніх версій мобільного робота з відкритим кодом, розроблена для використання в різних дослідженнях, навчанні і розробці робототехнічних застосунків (рисунок 1.5). Цей мобільний робот активно використовується в дослідницьких лабораторіях, навчальних закладах та в розробці прототипів для різних галузей [9].

Рисунок 1.5 –Мобільний робот-кур'єр TwinswHeel H04

Основні характеристики мобільного робота TurtleBot 4 [9]:

– картографування та орієнтація. TurtleBot 4 оснащений високоточними сенсорами, такими як LiDAR, камери та IMU (інерціальна вимірювальна одиниця), що дозволяє йому створювати детальні карти навколишнього середовища за допомогою технологій, як SLAM (Simultaneous Localization and Mapping). Це означає, що він може автономно переміщатися по незнайомій території, скануючи простір і створюючи карту для орієнтації;

– автономна навігація. Використовуючи алгоритми планування шляху та уникнення перешкод, TurtleBot 4 може обирати найкращий шлях для досягнення цілі, навіть у складних умовах. Це робить його ідеальним для застосунків в робототехніці, таких як розвідка територій, тестування алгоритмів навігації і автономні доставки;

– моделі штучного інтелекту. TurtleBot 4 підтримує інтеграцію з різними бібліотеками та фреймворками для штучного інтелекту, наприклад, TensorFlow або PyTorch. Це дозволяє запускати моделі для розпізнавання об'єктів, планування дій на основі даних сенсорів або для навчання роботів в реальному середовищі;

– відкритий код і спільнота. Однією з найсильніших сторін TurtleBot 4 є велика спільнота розробників, які використовують ROS (Robot Operating System) для програмування. Оскільки це система з відкритим кодом, кожен може адаптувати робот до своїх потреб і розробляти нові функції;

– розширення та налаштування. TurtleBot 4 дає можливість додавати нові сенсори, камери, обчислювальні модулі та інші компоненти для розширення функціональності робота. Ця гнучкість робить його потужним інструментом для експериментів і розробок в області робототехніки.

Використання таких мобільних роботів для досліджень у сфері автономних систем, штучного інтелекту та робототехніки значно покращує можливості для наукових експериментів та практичних розробок.

Отже, проаналізувавши види мобільних роботів для подальших досліджень за темою класифікаційної роботи обираємо мобільний робот TurtleBot 4.

1.2 Платформа для розробки робототехнічних додатків

ROS (Robot Operating System) – це набір програмних бібліотек та інструментів для розробки програмного забезпечення для мобільних роботів. Це не операційна система в класичному розумінні цього слова, а швидше набір фреймворків та утиліт, які забезпечують абстракцію для низькорівневого доступу до апаратних пристроїв, управління процесами, комунікації між програмами (ноодами), а також для обробки даних з сенсорів та управління рухом мобільних роботів.

Мобільна база робота Turtlebot 4 повністю інтегрована з мінікомп'ютером Raspberry Pi 4, який працює під управлінням Ubuntu 20.04, ROS2 і вбудованих драйверів датчиків [9].

Платформа ROS 2 – це другий варіант відкритої платформи для розробки робототехнічних додатків. Це фреймворк, який забезпечує стандартизоване середовище для розроблення, тестування та запуску програм, пов'язаних з керуванням роботами, і надає поліпшену архітектуру та надійність порівняно з першим варіантом ROS.

Платформа ROS 2 базується на стандартах розподілу даних (DDS), що дає змогу обмінюватися даними між різними частинами системи мобільного робота, навіть якщо вони перебувають на різних комп'ютерах або в різних мережах. Підтримує мови програмування C++, Python, Java та інші, що робить його гнучким і зручним для різних розробників. Забезпечує надійну і передбачувану роботу, що важливо для критично важливих додатків, таких як автономна навігація роботів. ROS 2 має надійнішу та безпечнішу архітектуру, ніж ROS 1, що дає змогу розробляти стійкіші та надійніші робототехнічні додатки. Платформа є відкритим програмним забезпеченням,

що дає змогу користувачам вільно вивчати, модифікувати та розповсюджувати його, водночас має велику та активну спільноту розробників і користувачів, що забезпечує підтримку та допомогу під час розроблення проєктів.

Платформа ROS 2 використовується для розроблення різних робототехнічних застосунків, включно з:

– автономна навігація. ROS 2 дає змогу роботам самостійно переміщатися навколишнім середовищем, уникаючи перешкод і досягаючи мети.

– управління роботами. ROS 2 дає змогу керувати різними компонентами робота, як-от двигуни, датчики та камери.

– візуалізація та моніторинг. ROS 2 дає змогу візуалізувати і моніторити роботу робота, що допомагає розробникам і операторам відстежувати його стан і продуктивність.

– обробка зображень. ROS 2 дає змогу обробляти зображення і дані з камер мобільного робота, що дає йому змогу розуміти навколишнє середовище і взаємодіяти з ним.

– розробка різних робототехнічних систем. ROS 2 дає змогу розробляти різні робототехнічні системи, як-от роботи-пилососи, роботи-помічники та промислові роботи.

Платформа ROS2 є надзвичайно потужним фреймворком для створення програмного забезпечення для мобільних роботів, що дозволяє абстрагувати складні аспекти робототехніки та інтегрувати різні компоненти.

1.3 Планування маршруту в реальному часі

Маршрут мобільного робота TurtleBot 4 можна планувати та коригувати за допомогою різних методів, зокрема використовуючи алгоритми навігації та планування шляху.

Мобільний робот TurtleBot 4 може створювати карту навколишнього середовища за допомогою алгоритмів SLAM (Simultaneous Localization and Mapping), що дозволяє роботу самостійно орієнтуватися в нових, незнайомих для нього просторах.

Після того як карта створена, робот використовує алгоритми планування маршруту для того, щоб знайти найкращий маршрут від поточного місця до заданої точки. Алгоритм планування шляху вважає різні фактори, такі як наявність перешкод, безпечні шляхи та можливі точки для маневрів.

Під час руху робот TurtleBot 4 постійно сканує оточення за допомогою сенсорів (LiDAR, камери, ультразвукові сенсори) для виявлення перешкод. Алгоритми, такі як Dynamic Window Approach (DWA), допомагають роботу уникати зіткнень, вибираючи найкращі шляхи для руху [9].

Робот аналізує дані від сенсорів та, якщо на шляху виникають перешкоди, вносить коригування в маршрут або змінює напрямок.

Якщо TurtleBot 4 працює на відкритій місцевості або в зовнішніх умовах, GPS може бути використаний для більш точного визначення місця розташування робота. Внутрішні сенсори, такі як IMU (інерціальна вимірювальна одиниця), допомагають стабілізувати позицію робота під час руху.

За допомогою платформи ROS можна налаштувати різні етапи маршруту:

– навігаційний стек ROS включає в себе пакет move_base, що дозволяє визначити кінцеву точку маршруту і на основі карти спланувати шлях;

– визначення цілі: задається точка призначення в координатах (наприклад, за допомогою rviz), і робот починає планувати шлях до цієї точки.

Після того як маршрут визначено, робота можна протестувати в реальному середовищі. Це допомагає виявити будь-які можливі проблеми, наприклад, несумісність з фізичними перешкодами або некоректне

планування через точність сенсорів. Після тестування зазвичай вносяться коригування в програму або картографування.

Якщо мобільний робот TurtleBot 4 повинен орієнтуватися і прокладати маршрут у невизначеному середовищі, де карта чи структура навколишнього простору ще не відома, то для цього використовується поєднання кількох ключових технологій, реальне планування шляху та обробка сенсорних даних в реальному часі.

У невизначеному середовищі першою задачею робота є створення карти навколишнього середовища, щоб він міг орієнтуватися в просторі. Оскільки середовище невизначене (без заздалегідь визначеної карти), TurtleBot 4 застосовує алгоритми SLAM [9].

Ці алгоритми дозволяють мобільному роботу одночасно і створювати карту, і визначати своє місце в ній.

Якщо середовище невизначене і змінюється, роботу необхідно активно уникати перешкоди. Для цього він використовує датчики на борту:

Якщо середовище постійно змінюється (наприклад, рухливі перешкоди або нові об'єкти), роботу доводиться постійно адаптуватися. Для цього TurtleBot 4 регулярно оновлює свою картографію та локалізацію в реальному часі.

Після того, як було налаштовано базове картографування і планування маршруту, можна протестувати систему в реальному середовищі, спостерігаючи за тим, як робот реагує на зміни навколишнього середовища і чи коректно він адаптує свої шляхи.

У деяких випадках можна інтегрувати штучний інтелект для кращого адаптування до невизначеного середовища. Наприклад, робот може використовувати розпізнавання об'єктів для ідентифікації перешкод або планування маршрутів, що враховують більш складні умови навколишнього середовища.

У невизначеному середовищі робот TurtleBot 4 буде активно використовувати SLAM для картографування, алгоритми навігації для

планування маршруту в реальному часі, сенсори для уникнення перешкод, а також штучний інтелект для покращення прийняття рішень. Все це разом дозволяє роботу адаптуватися до змін в оточенні і забезпечує ефективну автономну навігацію.

2 РОЗРОБЛЕННЯ МАРШРУТУ МОБІЛЬНОГО РОБОТА-КУР'ЄРА

2.1 Вибір алгоритму розрахунку маршруту мобільного робота-кур'єра

Пошук шляху (англ. Pathfinding) – термін в інформатиці та штучному інтелекті, який означає визначення комп'ютерною програмою найкращого, оптимального маршруту між двома точками. Еквівалентні шляхи представлені на рисунку 2.1.

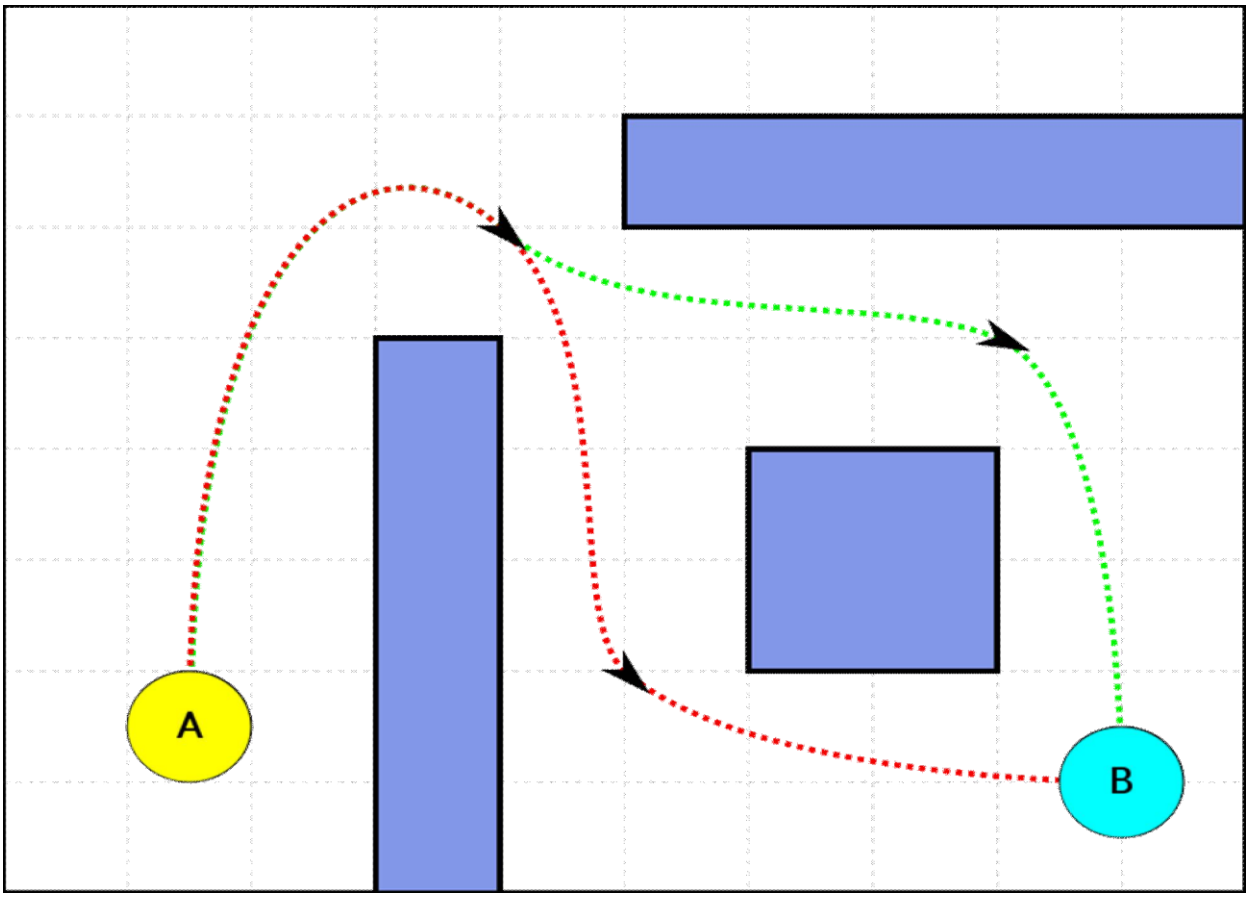
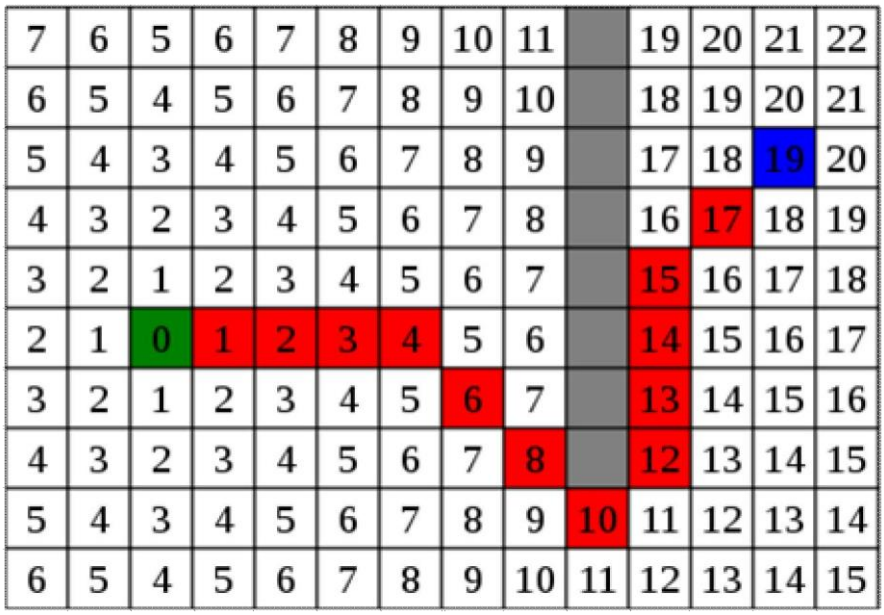


Рисунок 2.1 – Еквівалентні шляхи між A та B у двовимірному середовищі

За своєю природою алгоритм пошуку маршруту реалізує пошук на графі, починаючи з однієї (стартової) точки і досліджуючи суміжні вузли до тих пір, поки не буде досягнутий вузол призначення (кінцевий вузол). Крім того, в алгоритмі пошуку шляху в більшості випадків також закладена мета знаходження найкоротшого шляху. Деякі методи пошуку на графі, такі як пошук в ширину, можуть знайти шлях, якщо дана достатня кількість часу. Інші методи, які «досліджують» граф, можуть досягти точки призначення набагато швидше. Приклад пошукового алгоритму на графі представлений на рисунку 2.2.



червоний – шлях; синій – пункт призначення; сірий – перешкода

Рисунок 2.2 – Приклад пошукового алгоритму

Тут можна привести аналогію з людиною, що йде через кімнату. Людина може перед початком шляху заздалегідь досліджувати всі характеристики і перешкоди в просторі, обчислити оптимальний маршрут і тільки тоді почати безпосереднє рух. В іншому випадку людина може відразу рухатися в приблизному або передбачуваному напрямку мети і потім, вже під час шляху, робити коригування свого руху для уникнення зіткнень з перешкодами [10].

Маючи наступне дискретне середовище (рисунок 2.3), яке обмежене лінією та розділене квадратними осередками для зручності розрахунку маршруту мобільним роботом та користувачем, було обрано хвильовий алгоритм для розрахунку маршруту мобільного робота-кур’єра. Карта розробленої місцевості наведена на рисунку 2.3.

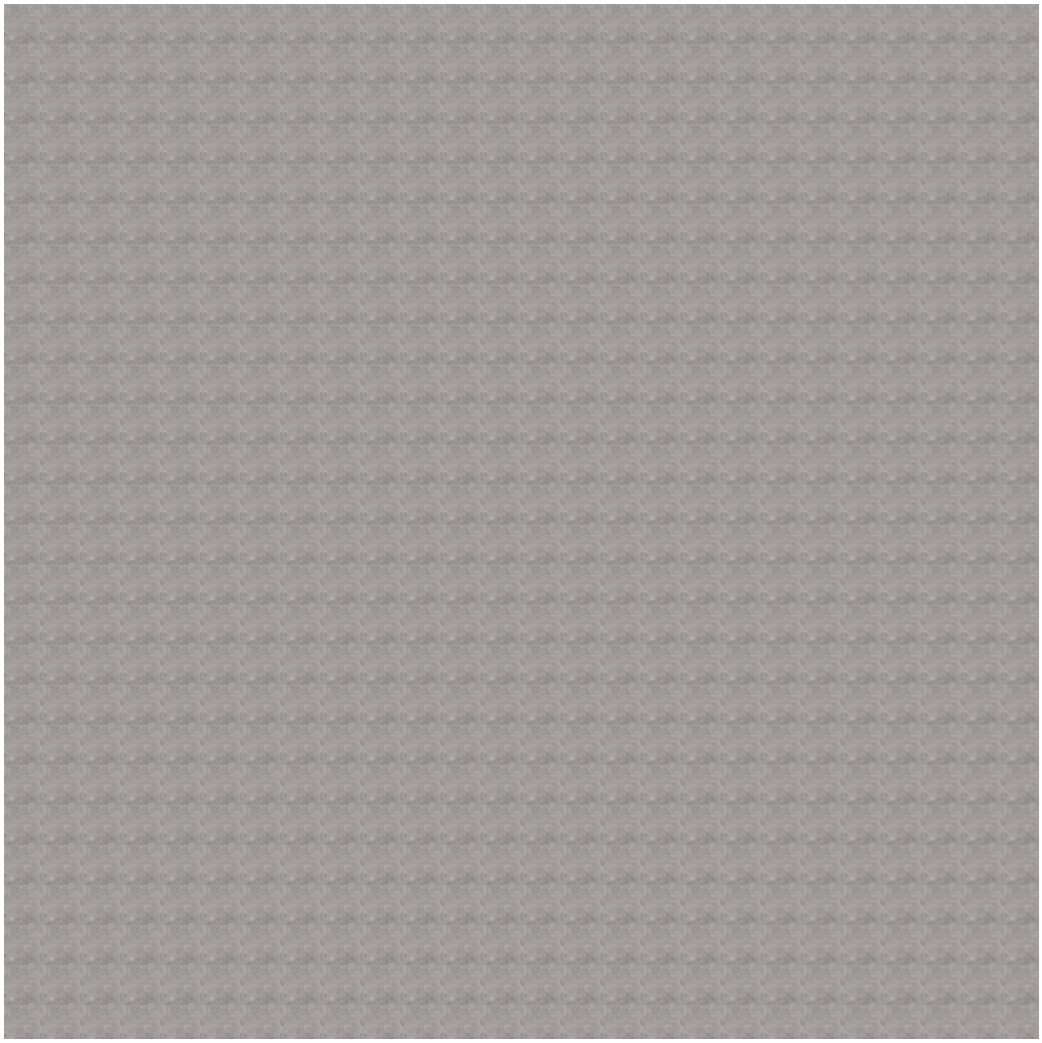


Рисунок 2.3 – Розділене квадратними осередками середовище місцевості

Алгоритм хвильового трасування (хвильовий алгоритм, алгоритм Лі) – алгоритм пошуку найкоротшого шляху на планарном графі. Належить до алгоритмів, заснованим на методах пошуку в ширину.

В основному використовується при комп'ютерному трасуванні (розводці) друкованих плат, з'єднувальних провідників на поверхні мікросхем. Інше застосування хвильового алгоритму – пошук найкоротшого відстані на карті в комп'ютерних стратегічних іграх [10].

Хвильовий алгоритм в контексті пошуку шляху в лабіринті був запропонований Е. Ф. Муром. Вчений відкрив цей алгоритм при формалізації алгоритмів трасування друкованих плат в 2002 році.

Алгоритм працює на дискретному робочому полі (ДРП), що представляє собою обмежену замкнутою лінією фігуру, не обов'язково прямокутну, розбитуна прямокутні осередки, в окремому випадку – квадратні. Безліч всіх осередків ДРП розбивається на підмножини: «прохідні» (вільні), тобто при пошуку шляху їх можна проходити, «непрохідні» (перешкоди), шлях через цей осередок заборонений, стартовий осередок (джерело) і фінішній (приймач). Призначають стартовий і фінішний осередки для вказівки пари осередків, між якими потрібно знайти найкоротший шлях.

Алгоритм призначений для пошуку найкоротшого шляху від стартової осередку до кінцевої комірки, якщо це можливо, або, за відсутності шляху, видати повідомлення про непрохідність.

Робота алгоритму включає в себе три етапи: ініціалізацію, поширення хвилі і відновлення шляху.

Під час ініціалізації будується образ безлічі осередків оброблюваного поля, кожному осередку приписуються атрибути прохідності/непрохідності, запам'ятовуються стартова і фінішна осередки.

Далі, від стартової осередку породжується крок до сусіднього осередку, при цьому перевіряється, прохідна вона, і чи не належить раніше поміченої в шляху осередку.

Сусідні комірки прийнято класифікувати двояко: в сенсі околиці Мура і околиці фон Неймана, який відрізняється тим, що в околиці фон Неймана сусідніми осередками вважаються тільки 4 осередки по вертикалі і горизонталі, в околиці Мура – всі 8 осередків, включаючи діагональні [8].

При виконанні умов прохідності і неналежність її до раніше позначеним в шляху осередкам, в атрибут осередки записується число, що дорівнює кількості кроків від стартової осередки, від стартового осередку на першому кроці це буде 1. Кожна клітинка, помічена числом кроків від стартової осередку стає стартовою і з неї породжуються чергові кроки в сусідні осередки. Результат роботи алгоритма відображений на рисунку 2.4. Очевидно, що при такому переборі буде знайдено шлях від початкової комірки до кінцевої, або черговий крок з будь-породженої в дорозі осередки буде неможливий.

9	10		10	9	8	9	10	11	12	13	14
8	9		9	8	7	8	9	10	11	12	13
7	8	9	8	7	6	7	8	9	10	11	12
6	7	8	7	6	5	6	7			10	11
5					4	5	6	7	8	9	10
4	3	2	1	2	3	4	5	6			11
3	2	1	0	1	2	3	4	5			10
4	3	2	1	2	3	4	5	6	7	8	9

Рисунок 2.4 – Результат роботи хвильового алгоритму
на дискретному середовищі (ортогональний шлях)

Алгоритм складається з кількох етапів, серед яких основними можна виділити наступні: підготовчий етап, етап поширення хвилі та етап побудови шляху.

На підготовчому етапі проводиться аналіз комірок, що присутні на карті, визначаються зайняті комірки. Решта комірок позначаються як вільні, їм присвоюється вага "0". В лічильник кількості кроків *K* записується 1 [11].

Етап поширення хвилі полягає у знаходженні точки фінішу шляхом перебору сусідніх комірок та присвоєння їм відповідної ваги. В першу чергу перевіряються всі комірки, суміжні з початковою. Якщо комірка має вагу "0", їй присвоюється значення з лічильника кроків. Комірки з іншою вагою (заповнені

на попередніх кроках), а також комірки, відмічені як зайняті, залишаються без змін. Якщо одна з комірок є точкою фінішу, алгоритм переходить на наступний етап – побудову шляху. В іншому випадку лічильник кроків збільшується на одиницю і описаний для початкової точки алгоритм повторюється для всіх точок з вагою *K*-1. Якщо кінцева точка не знайдена, і для всіх точок з вагою *K*-1 в околі не лишилось вільних комірок, то вважається, що шлях не існує [11].

На етапі побудови шляху (згортання хвилі) починаємо рух з кінцевої точки. Для цієї точки обирається комірка з її околу (вага цієї комірки буде *K*). Тоді для цієї комірки знову шукається суміжна з вагою *K*-1. Таким чином продовжується доти, поки не знайдено комірку з вагою 1, в околі якої знаходиться початкова точка (вага початкової точки 0). Таким чином маємо побудований шлях, який також відносить до множини шляхів мінімальної довжини.

Відновлення найкоротшого шляху відбувається в зворотномунапрямку: при виборі комірки від фінішного осередку до стартового на кожному кроці вибирається осередок, що має атрибут відстані від стартової на одиницю менше поточної комірки. Очевидно, що таким чином знаходиться найкоротший шлях між парою заданих осередків. Трас з мінімальною числовою довжиною шляху, як при пошуку шляху в околицях Мура, так і фон Неймана може існувати кілька. Вибір остаточного шляху в додатках диктується іншими міркуваннями, що знаходяться поза цього алгоритму [11].

2.2 Створення моделі середовища

Проаналізувавши поставлені вимоги до тематики, предметної області, технічного завдання та напрямку розробки кваліфікаційної роботи бакалавра, було обрано одноповерховий бізнес-центр як закрите (замкнуте) середовище, розміром 600×600 пікселів, вид якої представлений на рисунку 2.5.

Рисунок 2.5 – Модель (карта) одноповерхового бізнес-центра

Даний макет середовища моделює справжній поверх бізнес-центру з кімнатами без вхідних дверей в кожну для простоти моделювання руху робота-кур'єра та подальшого наочного розуміння процесу доставки замовлення.

Кожна кімната – замкнуте середовище зі своїми перешкодами у вигляді офісної фурнітури та приладами, наприклад, екран в кімнаті відпочинку.

Весь поверх має тільки один вихід, виїзд робота за межі поверху не передбачається.

Також у кожній кімнаті знаходяться столи, стільці та шафи у якості фізичних перешкод для робота-кур’єра. Кожна перешкода, як стіни, так і кожен об’єкт, мають свою власну координату розміщення на карті, для того щоб робот мав установи, в які точки він не може потрапити, які повинен огибати, а через які точки може здійснити рух для подальшої доставки.

Для наочності кожен квадратний блок на карті – це піксель 25×25, будемо вважати, що розмір кожного пікселю – 1×1 м.

Верхні кімнати – дві по 8×9 м., третя – 4×9 м. Нижні кімнати – 10×9 м. та 11×9 м.

Кімнати зверху зліва-направо: робочий кабінет, кімната для засідань та переговорів, кухня, знизу – кімната для відпочинку та робочий кабінет.

Мається коридор – 2 м в ширину.

Отже, перешкодами являються тільки вище названі предмети, рух співробітників не враховується. Адже задача стоїть в розрахунку найменшої маршруту доставки роботом.

2.3 Розробка алгоритму побудови маршруту мобільного робота-кур’єра

Як зазначалося, для моделювання руху робота був обраний хвильовий алгоритм. Адже з ним можна задати шлях до цілі і одразу слідувати до неї. Це як найкраще підходить для підприємств великих компаній, бізнес центрів. Для нового створення та розрахунку маршруту нового маршруту потрібно перераховувати кожного разу новий шлях.

Сформована карта місцевості і буде слугувати нашою областю застосування. На карті зазначені непрохідні зони, у вигляді перешкод, зрозуміло що мобільний робот неможе прокладати свій маршрут через ці відмічені в координатах зони, та прохідні – вільне місце для створення маршруту роботом.

Одна з основних функцій робота полягає у виконанні руху в задану точку в залежності від геометричних форм перешкод. Однак не

завжди трасування ставить перед собою мету знайти найкоротший шлях, найчастіше це просто неможливо. Програмування даної функції впирається в проблему, яка зводиться до алгоритму обходу перешкод.

В основупрограмтрасуванняможутьбутипокладеніалгоритми:

– з дискретним поданням зони руху;

– засновані на теорії графів;

– засновані на методах потенційних полів;

– засновані на евристичних моделях;

– з поданням інформації у вигляді рівнянь.

В данній кваліфікаційної роботі була розглянута наступна задача трасування: обхід двовимірних перешкод по заданій цифровій (растровій) карті місцевості у вигляді двовимірної площини, розділеної на квадратні клітини із точками, де мобільний робот-кур'єр повинен забрати вантаж та віддати його, які задаються користувачем.

Спочатку необхідно обрати стратегію обходу перешкоди. Ми можемо піти двома шляхами:

– перший – прокладати шлях «находу», ігноруючи перешкоди до зіткнення з ними;

– другий – заздалегідь спланувати шлях до початку переміщення. Скористалися другим шляхом.

Мобільний робот-кур'єр, отримавши координати точки, куди необхідно доставити вантаж, починає аналізувати карту, де бере в розрахунок координати перешкод та будує найкоротший маршрут огинання перешкод у вигляді стін та офісної фурнітури.

Слід зазначити, що пріоритетом вибору між двома однаковими по відстані траєкторій буде рух праворуч, у відповідності з прийнятим правостороннім рухом у нашій країні.

Саме тому має місце побудування алгоритму огинання перешкод, за яким мобільний робот-кур’єр буде реалізовувати розрахунок маршруту. Який представлений на рисунку 2.6.

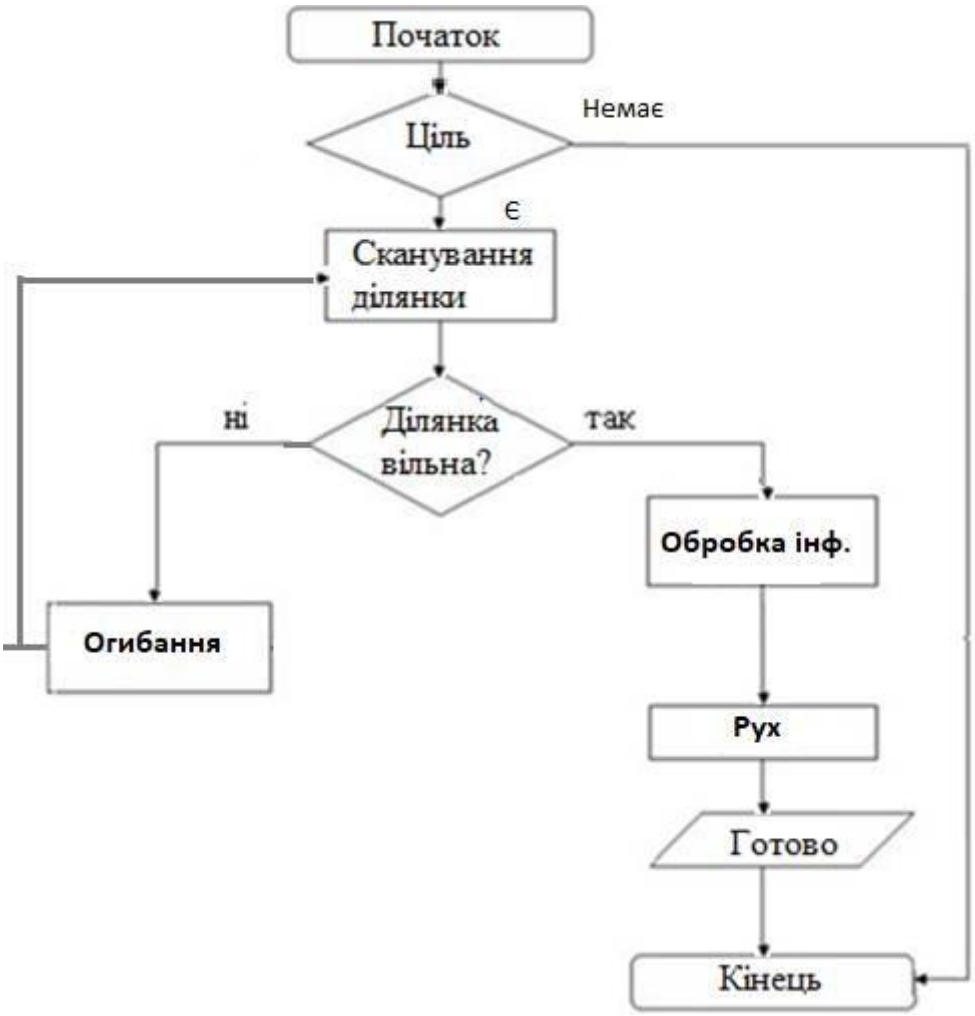


Рисунок 2.6 – Алгоритм огибання перешкод

Будемо вважати офіс бізнес центру початковим і єдиним середовищем для руху робота-кур’єра. С початку установлюємо точку, куди повинен прибути робот-кур’єр для отримання вантажу, траєкторія руху його з точки отримання вантажу до точки передачі вантажу позначається чорним кольором.

Кожна клітинка (блок) карти має свій статус, прохідна або непрохідна. Ці данні закладені до системи робота, тому при кожному прокладеному кроці

маршруту кожне коло буду перевірятися на свій статус, чи не належить йому атрибут непрохідного та навпаки.

В тому разі, якщо кожен крок (коло) прокладеного маршруту отримає підтвердження статусу прохідного, робот-кур’єр почне свій рух по прокладеній маршруту, записуючи інформацію про кожен крок, початкову та кінцеву точки руху [12].

В разі, коли хоча б одне коло маршруту опиниться непрохідним, вся траєкторія одразу стане не придатною для реалізації і цикл обробки маршруту запуститься знову.

Слід зауважити, що при побудові маршруту, система мобільного робота-кур’єра прокладає саме найкоротший шлях до своєї цілі методом підрахування кожного кроку до цілі та їх сумарну кількість. Зрозуміло, що робот почне свій рух тільки по найкоротшій прокладеній маршруту. Здюйснюється це в два етапи:

– з початкового елемента поширюється в 4-х напрямках хвиля (рисунок 2.7). Елемент в який прийшла хвиля утворює фронт хвилі. На рисунку цифрами позначені номери фронтів хвилі. Кожен елемент першого фронту хвилі є джерелом вторинної хвилі. Елементи другого фронту хвилі генерують хвилю третього фронту тощо. Процес триває до тих пір поки не буде досягнутий кінцевий елемент;

– будується сама траса. Її побудова здійснюється від кінцевого елемента до початкового.

Плюс алгоритму в тому, що він дозволяє знайти шлях в будь-якому лабіринті (за умови, що шлях існує). Але головний недолік цього алгоритму в тому, що при побудові траси потрібен великий обсяг пам'яті.

Існує два способи визначення кількості сусідніх клітин: вважати сусідніми клітини, доступні через загально межі (їх 4) і вважати сусідніми клітини, доступні через загальні межі і загальні кути (таких клітин буде 8). В роботі реалізований простіший варіант, який використовує 4 клітини. Одразу видно мінуси. Замість того, щоб йти по діагоналі, алгоритм йде по прямій. Відомо, що

сторона трикутника менша від суми двох інших сторін. Так що, для більш кращого пошуку, краще використовувати 8 клітин, що представлено на рисунку 2.7.

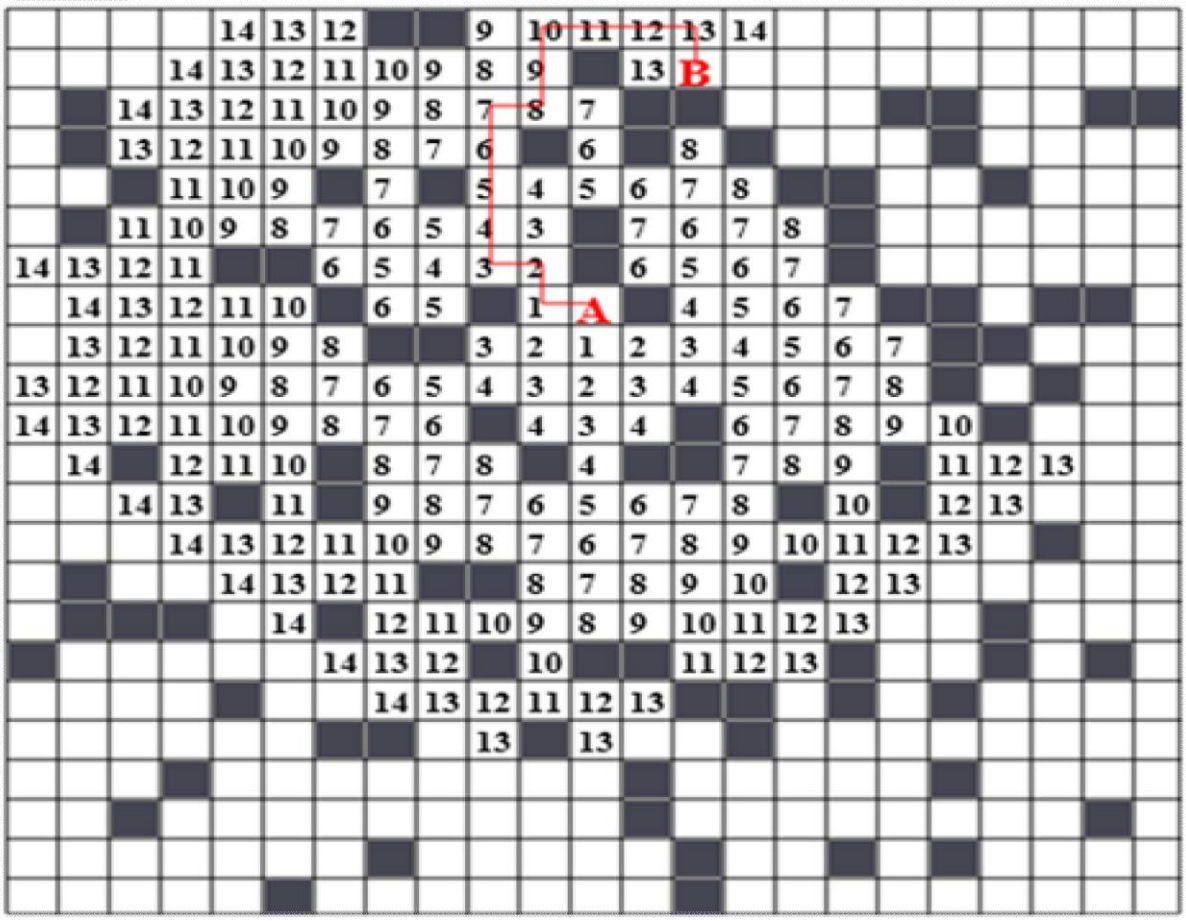


Рисунок 2.7 – Хвильовий алгоритм

Коли оптимальна траєкторія побудована, робот починає свій рух до цілі, в даному випадку в роботі рух відбувається через центр кожного блоку пікселя по діагоналі клітинки. Координати центру кожної клітинки наносяться на карту, та розраховуються за формулами (2.1) та (2.2):

$$X_i = X_{i-1} \pm \Delta S \cdot \cos\alpha; \quad (2.1)$$

$$Y_i = Y_{i-1} \pm \Delta S \cdot \sin\alpha. \quad (2.2)$$

де X_{i-1},Y_{i-1} – абсциса і ордината центру попереднього блоку;

ΔS – величина переміщення;

Δ – кут заданого курсу робота.

У формулі (2.1) вибирається знак «+», якщо *i*-та ділянка розташована правіше *Xi* > *i*-1 ізнак «–», якщо лівіше. У формулі (2.2) знак «+» вибирається, якщо *i*-й ділянка розташована вище, ніж *i*-1, і «–», якщо нижче. Також, поруч з цими координатами клітинок ставляться індекси зсуву для того щоб робот-кур'єр запам'ятав упорядкованість перешкод карти. Іншими словами, якщо робот ідентифікує блок зайнятим, то він повинен обійти його. Даний процес триває до тих пір, поки робот не дійде до цілі [10].

Коли роботу необхідно обійти перешкоди має сенс застосування наступного алгоритму.

Для кожного з напрямків руху робота-кур'єра, можливих в даний момент, складається повна вага P_i даного напрямку, яка визначається виразом (2.3):

$$P_i=\sum g_k P_{ki}, \tag{2.3}$$

де P_{ki} – оцінка *i*-го напряму блоком оцінки;

g_k – вага оцінки.

Як спрямовуючу силу чергового кроку вибирається напрямок, який отримав найбільшу вагу P_m. Пошук оптимального шляху представлений на рисунку 2.8.

Таким чином, завданням робота є дослідження середовища з метою складання її карти і знаходження мети. Як зазначалося вище, спочатку робот-кур'єр здійснює побудову карти зовнішнього середовища, рухаючись з початкової точки (рисунок 2.8) відповідно до запропонованої методики [12].

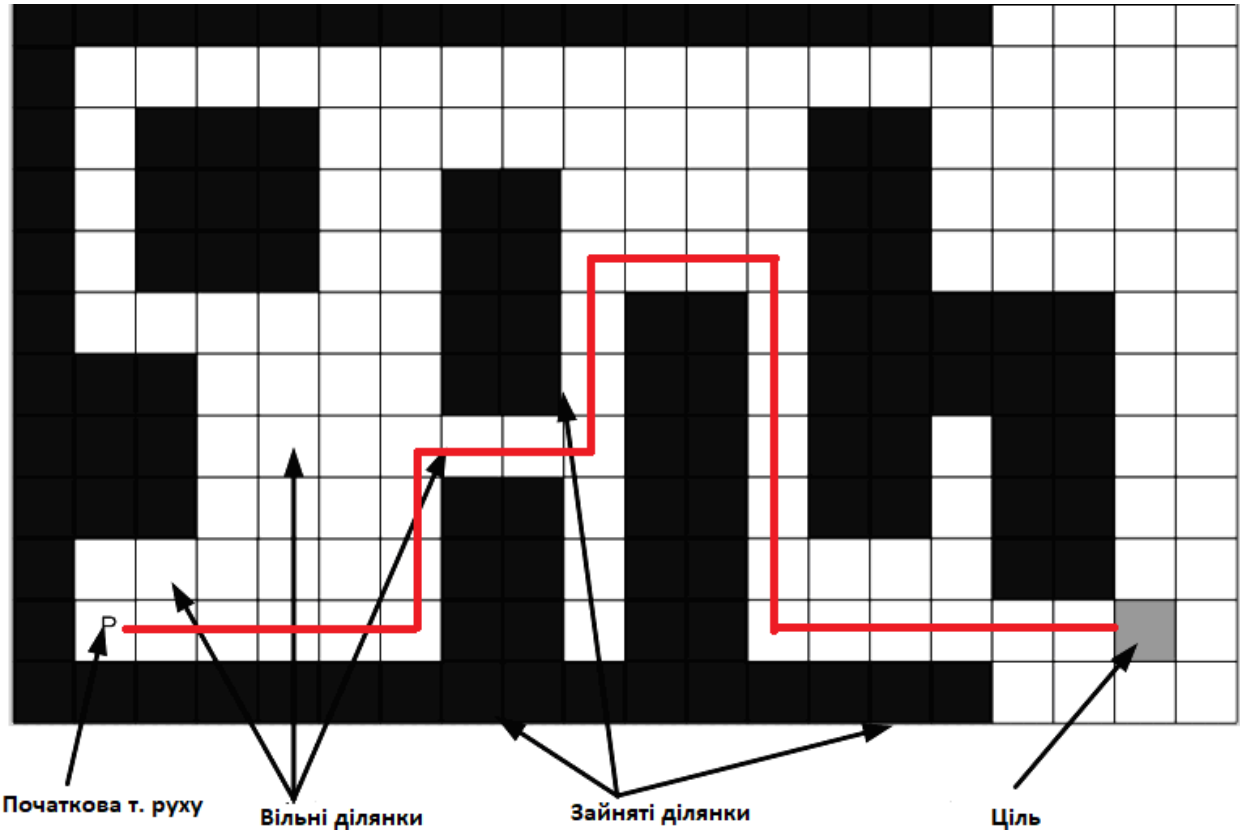


Рисунок 2.8 – Пошук оптимальної маршруту

На рисунку наглядно продемонстровано оптимальний вибір роботом-кур’єром ділянки маршруту, на якій відсутні перешкоди. Робот відібрав найменші значення координат осі абсцис та ординат, в результаті чого він буде рухатися по найкоротшому шляху за найменший проміжок часу, що підтверджує оптимальність та ефективність хвильового метода [12].

3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗРАХУНКУ МАРШРУТУ МОБІЛЬНОГО РОБОТА-КУР'ЄРА

3.1 Вибір мови програмування

Для розробки ПЗ була обрана мова JavaScript. JavaScript (JS) – динамічна, об'єктно-орієнтована мова програмування. Мова програмування є безпосередньою реалізацією стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією.

Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, наслідування функцій як об'єктів першогокласу [13].

Мова JavaScript використовується для:

– написання сценаріїв веб-сторінок для надання їм інтерактивності;

– створення односторінкових веб застосунків (ReactJS, AngularJS);

– програмування на стороні сервера (Node.js);

– стаціонарних застосунків (Electron, NW.js);

– мобільних застосунків (React Native, Cordova);

– сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);

– PDF-документів.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme [14].

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожим на Java, але при цьому легким для використання не програмістами. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних у веб-розробці.

Назва «JavaScript» є зареєстрованим товарним знаком компанії Oracle Corporation.

Компанія Nombas (згодом придбана Openwave) почала розробку вбудованої скриптової мови Cmm (Сі-мінус-мінус), яка, за задумом розробників, повиненна була стати досить потужною, щоб замінити макроси, зберігаючи при цьому схожість з Сі, щоб розробникам не становило жодних проблем вивчити його. Головною відмінністю від Сі була робота з пам'яттю.

У новому мовою все управління пам'яттю здійснювалося автоматично: не було необхідності створювати буфери, оголошувати змінні, здійснювати перетворення типів. В іншому мови сильно схожі один на одного: зокрема, Cmm підтримувала стандартні функції і оператори Сі. Cmm булла перейменована в ScriptEase, оскільки вихідна назва лунала надто негативно, а

згадка в ньому Сі «відлякувало» людей. На основі цієї мови був створений пропріетарний продукт CEnvi. В кінці листопада 2000 року Nombas розробила версію CEnvi, впроваджувану у веб-сторінки. Сторінки, які можна було змінювати за допомогою скриптової мови, отримали назву EspressoPages – вони демонстрували використання скриптової мови для створення гри, перевірки користувальницького введення в форми і створення анімації.

Espresso Pages позиціонувалися як демоверсія, покликана допомогти уявити, що трапиться, якщо в браузер буде впроваджений мову Cmm. Працювали вони тільки в 16-бітовому Netscape Navigator під управлінням ОС Windows.

JavaScript спочатку створювався для того, щоб зробити web-сторінки «живими». Програми на цій мові називаються скриптами. У браузері вони підключаються безпосередньо до HTML і, як тільки завантажується сторінка – зараз же виконуються [15, 16].

Програми на JavaScript – звичайний текст. Вони не вимагають якоїсь спеціальної підготовки.

В цьому плані JavaScript сильно відрізняється від іншої мови, яка називається Java.

Коли створювали мову JavaScript, у неї спочатку була інша назва: «LiveScript». Але тоді була дуже популярна мову Java, і маркетологи вирішили, що схожа назва зробить нову мову більш популярною [16].

Планувалося, що JavaScript буде таким собі «молодшим братом» Java. Однак, історія розпорядилася по-своєму, JavaScript сильно виросла, і зараз це абсолютно незалежна мова, зі своєю специфікацією, яка називається ECMAScript, і до Java не має ніякого відношення.

JavaScript може виконуватися не тільки в браузері, а де завгодно, потрібна лише спеціальна програма-інтерпретатор. Процес виконання скрипта називають «інтерпретацією».

Для виконання програм, не має значення якою мовою, існують два способи:

– компіляція – це коли вихідний код програми, за допомогою спеціального інструменту, іншої програми, яка називається «компілятор», перетворюється в іншу мову, як правило – в машинний код. Цей машинний код потім поширюється і запускається. При цьому вихідний код програми залишається у розробника.

– інтерпретація – це коли вихідний код програми отримує інший інструмент, який називають «інтерпретатор», і виконує його «як є». При цьому поширюється саме сам вихідний код (скрипт). Цей підхід застосовується в браузерах для JavaScript.

Сучасні інтерпретатори перед виконанням перетворять JavaScript в машинний код або близько до нього, оптимізують, а вже потім виконують. І навіть під час виконання намагаються оптимізувати. Тому JavaScript працює дуже швидко [16].

У всі основні браузери вбудований інтерпретатор JavaScript, саме тому вони можуть виконувати скрипти на сторінці. Але, зрозуміло, JavaScript можна використовувати не тільки в браузері. Це повноцінна мова, програми на якій можна запускати і на сервері, і навіть в пральній машинці, якщо в ній встановлений відповідний інтерпретатор [16].

Сучасний JavaScript – це «безпечна» мова програмування загального призначення. Вона не надає низькорівневих засобів роботи з пам'яттю, процесором, так як спочатку була орієнтований на браузери, в яких це не потрібно [16].

Що ж стосується інших можливостей – вони залежать від оточення, в якому запущена JavaScript. У браузері JavaScript вміє робити все, що відноситься до маніпуляції зі сторінкою, взаємодії з відвідувачем і, в якійсь мірі, з сервером:

– створювати нові HTML-теги, видаляти існуючі, змінювати стилі елементів, ховати, показувати елементи тощо;

– реагувати на дії відвідувача, обробляти кліки миші, переміщення курсора, натискання на клавіатуру тощо;

– посилати запити на сервер і завантажувати дані без перезавантаження сторінки (ця технологія називається "AJAX");

– отримувати і встановлювати cookie, запитувати дані, виводити повідомлення [17].

JavaScript – швидка і потужна мова, але браузер накладає на її виконання деякі обмеження [17].

Це зроблено для безпеки користувачів, щоб зловмисник не міг за допомогою JavaScript отримати особисті дані або якось нашкодити комп'ютеру користувача.

Цих обмежень немає там, де JavaScript використовується поза браузера, наприклад на сервері. Крім того, сучасні браузери надають свої механізми по установці плагінів і розширень, які володіють розширеними можливостями, але вимагають спеціальних дій по установці від користувача [17].

Більшість можливостей JavaScript в браузері обмежена поточним вікном і сторінкою.

JavaScript не може читати/записувати довільні файли на жорсткий диск, копіювати їх або викликати програми. Вона не має прямого доступу до операційної системи.

Сучасні браузери можуть працювати з файлами, але ця можливість обмежена спеціально виділеної Директорією – «пісочницею». Можливості щодо доступу до пристроїв також опрацьовуються в сучасних стандартах і частково доступні в деяких браузерах [17].

JavaScript, що працює в одній вкладці, не може спілкуватися з іншими вкладками і вікнами, за винятком випадку, коли він сам відкрив це вікно або декілька вкладок з одного джерела (однаковий домен, порт, протокол).

З JavaScript можна легко посилати запити на сервер, з якого прийшла сторінка. Запит на інший домен теж можливий, але менш зручний [17].

Є як мінімум три чудових особливості JavaScript:

– повна інтеграція з HTML/CSS;

– прості речі робляться просто;

– підтримується всіма поширеними браузерами і включений за замовчуванням.

Цих трьох речей одночасно немає більше ні в одній браузерної технології. Тому JavaScript і є найпоширенішим засобом створення браузерних інтерфейсів.

Сама мова JavaScript поліпшується. Сучасний стандарт ECMAScript 5 включає в себе нові можливості для розробки, ECMAScript 6 буде кроком вперед у поліпшенні синтаксису мови.

Сучасні браузери покращують свої движки, щоб збільшити швидкість виконання JavaScript, виправляють баги і намагаються слідувати стандартам.

Дуже важливо те, що нові стандарти HTML5 та ECMAScript зберігають максимальну сумісність з попередніми версіями. Це дозволяє уникнути неприємностей з уже існуючими додатками.

Втім, невелика проблема з «супер-сучасними речами» все ж є іноді браузери намагаються включити нові можливості, які ще не повністю описані в стандарті, але настільки цікаві, що розробники просто не можуть чекати.

Проте, з часом стандарт змінюється і браузерам доводиться підлаштовуватися до нього, що може привести до помилок в уже написаному, заснованому на старій реалізації, JavaScript-коді.

При цьому всі браузери сходяться до стандарту, і відмінностей між ними вже набагато менше, ніж всього лише кілька років тому.

Щоб додати JavaScript-код на сторінку, потрібно використовувати теги <script></ script>, вони є рекомендованими, але не обов'язково поміщати їх всередині контейнера <head>. В одному документі може бути скільки завгодно контейнерів <script>. Атрибут «type = 'text / javascript'» необов’язково вказувати, дане значення використовується за умовчанням.

HTML є мовою програмування, з її допомогою виконується розмітка документів. Більшість веб-сторінок написано на HTML. Текстові документи, що мають HTML розмітку (вони мають розширення .html або .htm), обробляються браузерами. Які в свою чергу відображають документ в його форматованому вигляді. Вони надають можливість користувачеві використовувати зручний інтерфейс для запиту веб-сторінок та їх перегляду, а також відправляти введені користувачем дані на сервер. Таких як текст, гіперпосилання або мультимедіа відображатимуться в браузері можна визначити за допомогою цієї мови [13-15].

З моменту свого створення HTML і пов'язані з нею протоколи порівняно швидко отримали визнання. Однак у перші роки існування цієї мови розмітки не було жодних чітких стандартів. Хоча її творці спочатку і задумували HTML як семантичну мову, позбавлену презентаційних можливостей, її практичне використання із різними браузерами призвело до додавання багатьох презентаційних елементів і атрибутів в HTML. Останні стандарти, пов'язані з HTML, відображають зусилля з подолання хаотичного розвитку мови і створення раціональної основи для розробки як змістовних, так і виразних документів. Щоб повернути HTML її роль семантичної мови, Консорціум Всесвітньої павутини розробив мови стилізування, такі як Каскадні таблиці стилів та Розширена мова таблиць стилів, аби перенести на них відповідальність за вигляд документа. У зв'язку з цим специфікація HTML повільно почала повертатися виключно до семантики.

Ймовірно, HTML – найуспішніша мова розмітки документів у всьому світі. Проте, коли світові представили XML, було вирішено створити нову версію HTML, похідну від XML. Адже з XML-заснованим HTML інші XML-мови могли би включати частини XHTML, а XHTML-документи могли бвключати частини інших мов розмітки. Також автори веб-документів могли б скористатися перевагами редизайну задля очищення деяких з найбільш неохайних частини HTML, а також додати деякі з нових необхідних функцій,

таких як покращені форми. Нижче зазначені деякі переваги використання XHTML замість HTML [11].

Якщо документ є лише чистим XHTML 1.0 (не включає інші мови розмітки), то різниця між XHTML та HTML майже не помітна. Проте, оскільки стають доступними все більше і більше XML-інструментів (наприклад, XSLT для перетворення документів), переваги використання XHTML стають все помітнішими. Наприклад, XForms дозволяє досить просто керувати редагуванням документів XHTML (або будь-яких інших видів документа XML). Семантичні веб-застосунки також зможуть скористатися документами XHTML за своїми потребами. Якщо документ більш ніж просто XHTML 1.0 (наприклад, у документі використовуються мови розмітки MathML, SMIL, або SVG), тоді переваги використання XHTML значно помітніші, адже HTML не підтримує такі комбінації мов розмітки в одному документі [11-13].

Семантичний HTML – спосіб написання HTML, що віддає перевагу підкресленню смислу закодованої інформації радше за її подання (зовнішній вигляд). Ще з самого початку свого розвитку HTML мав у складі елементи семантичної розмітки, проте також мав і елементи презентаційної розмітки.

Також HTML має семантично-нейтральні елементи span та div. З кінця 2000-х, коли Каскадні таблиці стилів почали належно працювати в більшості браузерів, авторам документів було рекомендовано уникати використання презентаційної розмітки HTML з метою розділення змісту.

Canvas – елемент HTML5, призначений для створення реєстрового двомірного зображення за допомогою скриптів, зазвичай на мові JavaScript.

Початок відліку блоку знаходиться зліва зверху. Від нього і будується кожен елемент блоку. Розмір простору координат не обов'язково відображає розмір фактичної інформації, що відображається площі. За замовчуванням його ширина дорівнює трьомстам пикселям, а висота ста п'ятдесяти.

Використовується, як правило, для відтворення графіків для статей і ігрового поля в деяких браузерних іграх. Але також може використовуватися для вбудовування відео в сторінку і створення повноцінного плеєра.

Використовується в WebGL для апаратного прискорення 3D графіки.

Компанією Google була випущена JavaScript-бібліотека explorercanvas, яка дозволяла працювати з Canvas в браузерах IE7 і IE8.

Canvas може ускладнити завдання роботам по розпізнаванню Капчі.

При використанні canvas з сервера ми завантажуємо не картинка, а набір точок (або алгоритм промальовування), за якими браузер промальовує картинку (капчу) [11].

Для роботи з Canvas необхідні базові знання HTML і JavaScript, але вона дуже проста. Він є елементом HTML5 та створює растрове двомірне зображення який за допомогою скриптів. Зліва зверху знаходиться початок відліку блоку. Від нього буде будуватися кожен блок-елемент. Розмір простору координат не завжди відображає розмір фактичної площі [11-13].

Відрисовка статей, графіків, а також ігрового поля в браузерних іграх є головною метою Canvas. Існує можливість його застосування для відрисовки графів і створення колажів або анімації.

Canvas має деякий недолік, він може ускладнити завдання роботам по розпізнаванню капчі. Коли використовується Canvas з сервера завантажуємо набір точок (або алгоритм промальовування), а не картинку, за якими браузер промальовує картинку (капчу) [11].

Розміщувати відео, картинку і текст він дозволяє. Існує можливість кольорової заливки і обведення контурів, а також додати градієнт. І нарешті він може відрисовувати фігури за допомогою вказівок контрольних точок.

При цьому змінюється ширина ліній, стиль з'єднань ліній і кисть малювання ліній.

Особливості Canvas:

– зміна висоти і ширини полотна;

– початок відліку (точка 0, 0) знаходиться в лівому верхньому кутку;

– 3D контексту немає, є окремі розробки, але вони не стандартизовані;

– колір тексту і розмір шрифту вказується як в CSS.

Недоліки Canvas:

– надмірно навантажується процесор і оперативна пам'ять;

– через обмеження збирача сміття немає можливості очистити пам'ять;

– необхідно самому обробляти події з об'єктами;

– погана продуктивність при високому дозволі;

– доводиться малювати окремо кожен елемент;

– можливість створення на сторінках спеціальних «маячків», у так званому Canvas Fingerprinting, для відстеження користувачів в мережі.

Переваги Canvas:

– на відміну від SVG зручніше мати справу з великим числом елементів;

– має апаратне прискорення;

– можна маніпулювати кожним пікселем;

– можна застосовувати фільтри обробки зображень;

– є багато бібліотек.

CSS (Cascading Style Sheets – каскадні таблиці стилів) – спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних [19].

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів [11-13].

Специфікації CSS були створені та розвиваються консорціумом всесвітньої мережі [13].

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі – сукупність правил CSS одного або більше рівнів, створені для окремих типів

пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо [13].

CSS прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки – розділення змісту сторінки (даних) та їхньої візуальної презентації.

CSS – формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

Переважно використовується як засіб опису, оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мов розмітки HTML і XHTML, але може також застосовуватися до будь-яких XML-документах, наприклад, до SVG або XUL [11-13].

CSS використовується творцями веб-сторінок для завдання кольорів, шрифтів, розташування окремих блоків і інших аспектів представлення зовнішнього вигляду цих веб-сторінок. Основною метою розробки CSS було розділення опису логічної структури веб-сторінки (яке проводиться за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї web-сторінки (яке тепер проводиться за допомогою формального мови CSS). Такий поділ може збільшити доступність документа, надати велику гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друковане подання, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при виведенні пристроями, що використовують шрифт Брайля.

Таблиці стилів дають змогу спростити процес створення сторінок і поліпшити їхній зовнішній вигляд. Концепція стилів подібна до ідеї стилів, яка реалізована в сучасних текстових редакторах — текст спочатку вводять, а потім форматують, користуючись стилями. Застосування стилів дає змогу вводити на сторінку потрібні тексти та інші елементи, не задумуючись над їхнім зовнішнім виглядом і розташуванням [11-13].

Таблиці стилів програміст зазвичай створює окремо від html-файлу. Під час створення html-файлу він концентрує увагу на змісті сторінки, а не на її зовнішньому вигляді, а під час створення таблиці стилів – навпаки. Отже, стилі дають змогу розмежувати етапи створення html-файлу й удосконалення зовнішнього вигляду сторінки.

Вважатимемо, що таблиця стилів уже створена. Тепер нам потрібно забезпечити взаємодію таблиці з html-файлом. Розглянемо три способи такої взаємодії: зв'язування, імпортування, вбудовування.

Правила CSS пишуться на формальній мові CSS і розташовуються в таблицях стилів, тобто таблиці стилів містять в собі правила CSS. Ці таблиці стилів можуть розташовуватися як в самому веб-документі, зовнішній вигляд якого вони описують, так і в окремих файлах, що мають формат CSS. По суті, формат CSS – це звичайний текстовий файл. У файлі .css не міститься нічого, крім переліку правил CSS і коментарів до них. [11-13].

Таблиці стилів містять в собі правила самої CSS. Дані таблиці можуть розташовуватися як в самому веб-документі, зовнішній вигляд якого вони описують, так і в окремих файлах, що мають формат CSS. По суті, формат CSS – це звичайний текстовий файл. У файлі не міститься нічого, крім переліку правил CSS і коментарів до них, тому таблиці можуть бути підключені або впроваджені в описуваний ними веб-документ чотирма різними способами [11-13]:

– коли таблиця стилів знаходиться в окремому файлі, вона може бути підключена до веб-документу за допомогою тега <link>, розташованого в цьому документі між тегами <head> і </ head>. (Тег <link> матиме атрибут href, що має значенням адресу цієї таблиці стилів). Всі правила цієї таблиці діють протягом усього документа;

– коли таблиця стилів знаходиться в окремому файлі, вона може бути підключена до веб-документу за допомогою директиви @import, що розташовується в цьому документі між тегами <style></ style, відразу після

тега <style>, яка також вказує на адресу цієї таблиці стилів. Всі правила цієї таблиці діють протягом усього документа;

– коли таблиця стилів описана в самому документі, вона може розташовуватися в ньому між тегами <style> і </ style>. Всі правила цієї таблиці також діють протягом усього документа;

– коли таблиця стилів описана в самому документі, вона може розташовуватися в ньому в тілі якогось окремого тега цього документа. Всі правила цієї таблиці діють тільки на вміст цього тега.

У перших двох випадках йдеться про те, що з документом застосовані зовнішні таблиці стилів, а в інших двох випадках – внутрішні таблиці стилів.

Як відомо, HTML-документи будуються на підставі ієрархії елементів, яка може бути наочно представлена в деревовидної формі. Елементи HTML один для одного можуть бути батьківськими, дочірніми, елементами-предками, елементами-нащадками, сестринськими [11].

Елемент є батьком іншого елемента, якщо в ієрархічній структурі документа він знаходиться відразу, безпосередньо над цим елементом.

Елемент є предком іншого елемента, якщо в ієрархічній структурі документа він знаходиться десь вище цього елемента.

Наприклад, в документі присутні два абзаци p, що включають в себе шрифт з напівжирним шрифтом b. Тоді елементи b будуть дочірніми елементами своїх батьківських елементів p та нащадками свого предка body.

У свою чергу, для елементів p елемент body буде тільки батьком. І крім того, ці два елементи p будуть сестринськими елементами, як такі, що одного з батьків-body [11].

В CSS можуть задаватися за допомогою селекторів не лише поодинокі елементи, але і елементи, які є нащадками, дочірніми або сестринськими елементами інших елементів [11-13].

3.2 Програмна реалізація алгоритму розрахунку маршруту мобільного робота-кур'єра

Перед безпосередньою розробкою системи розрахунку маршруту мобільного робота-кур'єра було створено предметне середовище у якості одноповерхового приміщення бізнес-центру, обраний мобільний робот та проаналізовано алгоритм пошуку маршруту.

Перший крок – створення карти місцевості.

Задаємо параметри карти:

```
const BLOCK_SIZE = 25;
const MAP_SIZE = 26;
let game = POINT_ROBOT$
```

Наступний крок – задаємо пустоту на карті:

```
for (let i = 0; i !== MAP_SIZE; i ++) {
for (let j = 0; j !== MAP_SIZE; j ++) {
    map [i][j] = FREE_SPACE;
```

Потім задаємо внутрішні елементи (рисунок 3.1).

```
map[1][1] = WALL;
map[1][2] = WALL;
map[1][3] = WALL;
map[1][4] = WALL;
map[1][5] = WALL;
map[1][6] = WALL;
map[1][7] = WALL;
map[1][8] = WALL;
map[1][9] = WALL;
map[1][10] = WALL;
map[1][11] = WALL;
map[1][12] = WALL;
map[1][13] = WALL;
map[1][14] = WALL;
map[1][15] = WALL;
map[1][16] = WALL;
map[1][17] = WALL;
map[1][18] = WALL;
map[1][19] = WALL;
map[1][20] = WALL;
map[1][21] = WALL;
map[1][22] = WALL;
map[1][23] = WALL;
map[1][24] = WALL;
map[2][1] = WALL;
map[2][3] = CHAIR;
map[2][8] = CHAIR;
map[2][10] = WALL;
map[2][19] = WALL;
map[2][20] = 9;
map[2][24] = WALL;
map[3][1] = WALL;
map[3][2] = 3;
map[3][3] = 3;
map[3][4] = 3;
map[3][7] = 3;
map[3][8] = 3;
map[3][9] = 3;
map[3][10] = WALL;
map[3][14] = CHAIR;
map[3][15] = CHAIR;
map[3][19] = WALL;
map[3][23] = CHAIR;
map[3][24] = WALL;
map[4][1] = WALL;
map[4][10] = WALL;
map[4][13] = CHAIR;
map[4][14] = 3;
map[4][15] = 3;
map[4][16] = CHAIR;
map[4][19] = WALL;
map[4][21] = CHAIR;
map[4][22] = 3;
map[4][23] = 3;
map[4][24] = WALL;
map[5][1] = WALL;
map[5][3] = CHAIR;
map[5][8] = CHAIR;
map[5][10] = WALL;
map[5][13] = CHAIR;
map[5][14] = 3;
map[5][15] = 3;
map[5][16] = CHAIR;
map[5][19] = WALL;
map[5][22] = 3;
map[5][23] = 3;
map[5][24] = WALL;
map[6][1] = WALL;
map[6][2] = 3;
map[6][3] = 3;
map[6][4] = 3;
map[6][7] = 3;
map[9][3] = 3;
map[9][4] = 3;
map[9][9] = 5;
map[9][10] = WALL;
map[9][11] = 6;
map[9][18] = 5;
map[9][19] = WALL;
map[9][20] = 4;
map[9][24] = WALL;
map[10][1] = WALL;
map[10][8] = 5;
map[10][9] = 5;
map[10][10] = WALL;
map[10][11] = 6;
map[10][12] = 6;
map[10][18] = 5;
map[10][19] = WALL;
map[10][20] = 4;
map[10][24] = WALL;
map[11][1] = WALL;
map[11][2] = WALL;
map[11][3] = WALL;
map[11][4] = WALL;
map[11][5] = FREE_SPACE;
map[11][6] = FREE_SPACE;
map[11][7] = WALL;
map[11][8] = WALL;
map[11][9] = WALL;
map[11][10] = WALL;
map[11][11] = WALL;
map[11][12] = WALL;
map[11][13] = WALL;
map[11][14] = FREE_SPACE;
map[11][15] = FREE_SPACE;
map[11][16] = WALL;
```

Рисунок 3.1 – Завдання координат для стін та перешкод

Далі прописуємо кожному елементу та стінам на карті свої координати. Створивши карту місцевості, задавши координати створення всіх перешкод та стін, отримуємо повноцінне середовище для розрахунку маршруту мобільного робота-кур'єра.

На рисунку 3.2 відображено карту місцевості.

Рисунок 3.2 – Карта місцевості

Побудуємо алгоритм розрахунку роботом маршруту (рисунок 3.3).

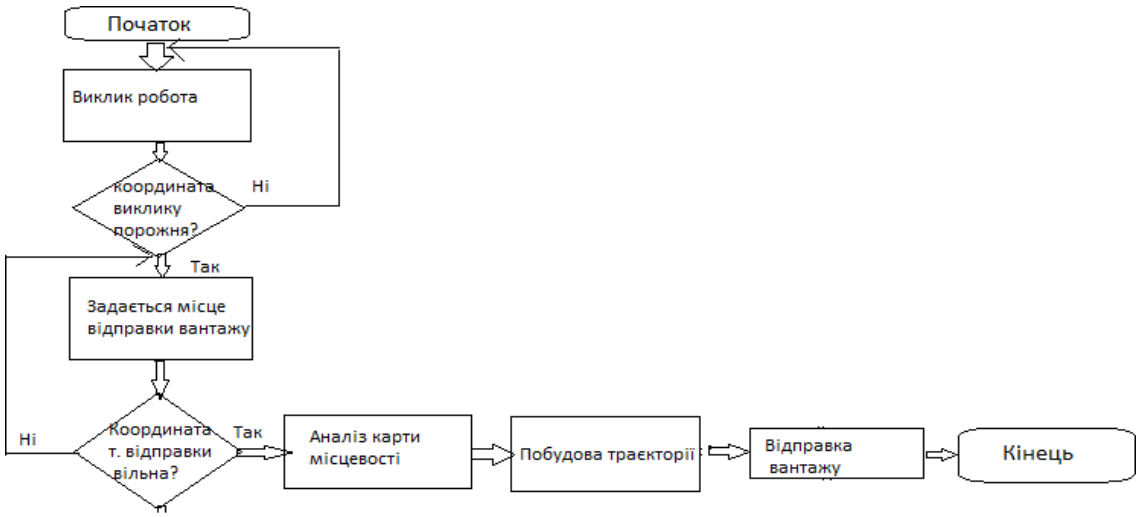


Рисунок 3.3 – Алгоритм роботи програми

Тепер реалізуємо алгоритм який дозволяє встановлювати вихідну позицію робота-кур'єра. У разі якщо комірка порожня, робот стає на позицію, інакше – виникає помилка. Також в даній частині коду реалізовані вікна повідомлень про загальний стан дій:

```
      functiongame_click(x, y) {
    letclickPosition=checkZeroPosition(x,  y,  map);
    switch (game) {
      casePOINT_ROBOT:
        if(false===clickPosition) {
          window.toastr.error('Тут неможливо розмістити робота-кур'єра, спробуйте знову');
        } else {
          game=POINT_FINISH;
      setRobotPosition(clickPosition[0],clickPosition[1]);
      renderGame(document.getElementById('game'),map);
      window.toastr.info('Куди робот-кур'єр повинен доставити посилку?');
        }
        break;
```

Аналогічним чином обирається місце, де робот-кур’єр буде отримувати вантаж, це може бути чашка, канцелярські вироби тощо. Головне, щоб вага вантажу не перевищувала 2 кг – максимально допустиму вагу підйому вантажу мобільним роботом TurtleBot4.

У разі завдання точки появи вантажу на стінах або перешкодах отримаємо помилку у вигляді випливаючого вікна та збросу алгоритму реалізацію розрахунку маршруту роботом що видно на рисунку 3.4.

Рисунок 3.4 – Помилка при заданні не правильної точки появи робота-кур’єра

Наступний етап – реалізація побудови алгоритму знаходження роботом-кур’єром маршруту, за хвильовим алгоритмом:

```
temNotUndefined(map_arr,i+1,j)) {
                    if(map_arr[i+1][j]===253)
                       {  x.splice(0,  x.length);
                       y.splice(0, y.length);
                       xx=
                       i+1;
                       yy
                       = j;
                       for(letii=0;ii!==Nk;ii++)          {
                          map_right = 999;
                          if(map_arr[xx+1][yy]!==undefined)       {
                             map_right = map_arr[xx + 1][yy];
                          }
                          map_left=999;
                          if(map_arr[xx-1][yy]!==undefined)       {
                             map_left = map_arr[xx - 1][yy];
                          }
                          map_up=999;
                          if(map_arr[xx][yy+1]!==undefined)       {
                             map_up = map_arr[xx][yy + 1];
```

```
                    }
                    map_down=999;
                    if(map_arr[xx][yy-1]!==undefined)        {
                       map_down = map_arr[xx][yy - 1];
                    }
                    volnaxy = getMinXY(map_right, xx + 1, yy, map_left, xx
- 1, yy, map_up, xx, yy + 1, map_down, xx, yy - 1);
                    xx1=volnaxy[
                    0];
                    yy1=volnaxy[
                    1];
                    volnaxy.splice(0,
                    volnaxy.length); x.push(xx1);
                    y.push(yy1);
                    if(map_arr[xx1][yy1]===0
                       { Ni = 999;
                       break;
                    }
```

Потім будуємо найкоротший шлях мобільного робота-кур’єра від місця підбору вантажу до місця, куди його необхідно доставити (рисунок 3.5).

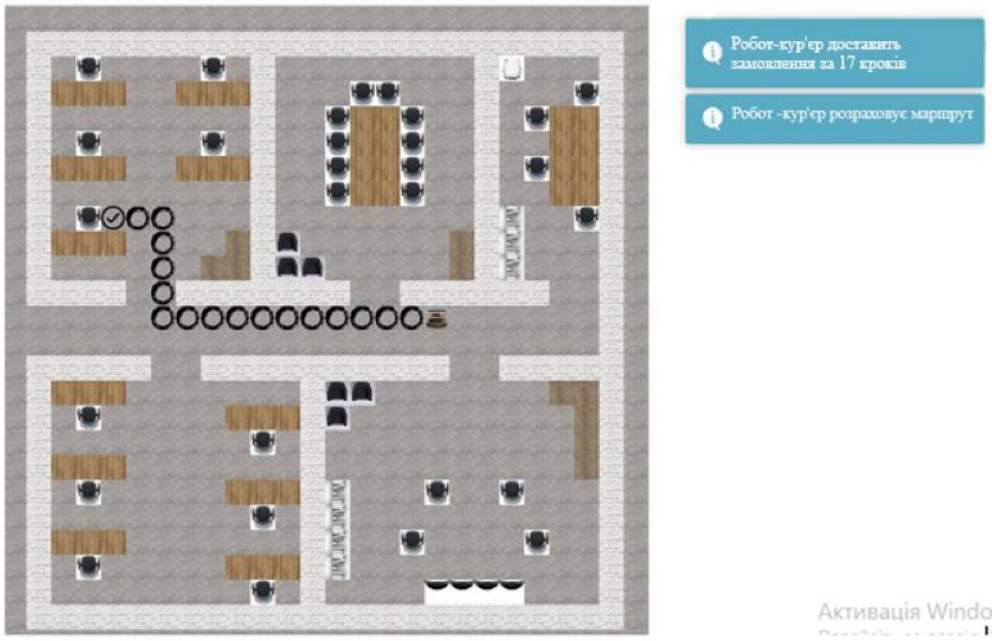


Рисунок 3.5 – Розрахунок маршруту мобільним роботом-кур’єром

Код програми, найкоротшого шляху робота-кур’єра від місця підбору вантажу до місця доставики:

```
way_full[0][0] = robot_x;
  way_full[0][1]=robot_y;
  for(vari=0;i<way[0].length;i++)
    { way_full[i + 1][0] =
    way[0][i];
    way_full[i+1][1]=way[1][i];
  }
  way_full[way[0].length + 1][0] =
  cargo_x; way_full[way[0].length +
  1][1] = cargo_y;
  for(vari=0;i<point_way[0].length;i++)
  {
    way_full[i+way[0].length+2][0]=point_way[0][i];
    way_full[i+way[0].length+2][1]=point_way[1][i];
  }
```

Рух мобільного робота у різні напрямки карти реалізований у наступній частині коду.

Рух робота вгору карти:

```
    if(way_full[gi][0]>way_full[gi+1][0]) {
      renderWay(map_canvas, 'dirt', way_full[gi + 1][0] * BLOCK_SIZE,
  way_full[gi][1] * BLOCK_SIZE, BLOCK_SIZE, BLOCK_SIZE);// Clear block
      renderWay(map_canvas, 'robot',way_full[gi][0] * BLOCK_SIZE – index
  + 1,way_full[gi][1]*BLOCK_SIZE+1,BLOCK_SIZE-2,BLOCK_SIZE-2);//
Clearblock
  }
```

Рух робота вниз по карті:

```
    if(way_full[gi][0]<way_full[gi+1][0]) {
      renderWay(map_canvas, 'dirt', way_full[gi + 1][0] * BLOCK_SIZE,
  way_full[gi][1] * BLOCK_SIZE, BLOCK_SIZE, BLOCK_SIZE);// Clear block
      renderWay(map_canvas,'robot',way_full[gi][0] * BLOCK_SIZE + index
  +1,way_full[gi][1]*BLOCK_SIZE+1,BLOCK_SIZE-2, BLOCK_SIZE-2);//
Clearblock
  }
```

Рух робота по карті праворуч:

```
if(way_full[gi][1]<way_full[gi+1][1]) {
    renderWay(map_canvas, 'dirt', way_full[gi][0] * BLOCK_SIZE, way_full[gi + 1][1] * BLOCK_SIZE, BLOCK_SIZE, BLOCK_SIZE);// Clear block
    renderWay(map_canvas, 'robot', way_full[gi][0] * BLOCK_SIZE + 1, way_full[gi][1]*BLOCK_SIZE+index+1,BLOCK_SIZE-2,BLOCK_SIZE-2);//Clearblock
}
```

Рух робота по карті ліворуч:

```
if(way_full[gi][1]>way_full[gi+1][1]){
    renderWay(map_canvas, 'dirt', way_full[gi][0] * BLOCK_SIZE, way_full[gi + 1][1] * BLOCK_SIZE, BLOCK_SIZE, BLOCK_SIZE);// Clear block
    renderWay(map_canvas,'robot',way_full[gi][0]*BLOCK_SIZE+1, way_full[gi][1]*BLOCK_SIZE-index+1,BLOCK_SIZE-2,BLOCK_SIZE-2);//Clearblock
}
```

Нижче програмний код завершення руху робота-кур’єра, коли він досягає кінцевої точки:

```
if(index>BLOCK_SIZE){//Nextblock
    index = 0;
    gi++;
  }
  setTimeout(wayGo,10);
}else{
  renderWay(map_canvas, 'dirt', point_x * BLOCK_SIZE, point_y * BLOCK_SIZE, BLOCK_SIZE, BLOCK_SIZE);
  renderWay(map_canvas, 'cargo', point_x * BLOCK_SIZE, point_y * BLOCK_SIZE, BLOCK_SIZE, BLOCK_SIZE);
  window.toastr.success('Чудово! Завдання виконане!');
}
}}
```

Повний текст програми приведено у Додатку А.

На рисунку 3.6 наведено повідомлення позитивного випадку реалізації всього алгоритму роботом-кур’єром.

Рисунок 3.6 – Позитивний алгоритм роботи програми

4 ОХОРОНА ПРАЦІ

4.1 Аналіз умов праці в лабораторії

Дослідження з кваліфікаційної роботи проводилися в лабораторії розміром: 4,2 м × 3,5 м × 3,2 м. У приміщенні працює 1 людина. Площа приміщення становить 14,7 м^2, об'єм – 47,04 м^3. На одне робоче місце припадає вся площа і об'єм приміщення, відповідає нормативному документу НПАОП 0.00-1.28-10 [19].

Було визначено, згідно з ДСТУ 2293-99, що в лабораторії існують небезпечні та шкідливі фактори:

– знижена рухливість повітря;

– недостатня освітленість робочої зони;

– підвищене значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини (небезпечний фактор).

Домінуючий шкідливий виробничий фактор – недолік штучного освітлення.

4.2 Промислова безпека в лабораторії

В лабораторії використовується трифазна чотирьохдротова мережа змінного струму напругою 220/380 В, частота 50 Гц, режим нейтралі – глухозаземлена. Згідно НПАОП 40.1-1.21-98, приміщення належить до класу приміщень без підвищеної небезпеки. Умови, які створюють підвищену та особливу небезпеку (підвищена вологість, струмопровідна пил, струмопровідні підлоги, можливість одночасного дотику до заземленим металоконструкцій будівлі і металевих поверхонь електроприладів), відсутні.

Для захисту працівників від ураження електричним струмом в лабораторії, згідно НПАОП 40.1-1.32-01, використовується система заземлення TN-C-S типу та захисне відключення.

Необхідно проводити контроль ізоляції відповідно до вимог ПУЕ-2011. Контроль проводити між нульовим та фазним провідниками, а також між фазами. Опір ізоляції не менше 500 кОм на фазу. Контроль проводити не рідше 1 разу на рік при відключеному електроживленні.

Електропроводка в приміщенні виконана з можливістю заміни: прихована в каналах будівельних конструкцій, що відповідає вимогам НПАОП 40.1-1.32-01.

4.3 Виробнича санітарія і гігієна праці

Роботи, що виконуються в приміщенні лабораторії, проводяться сидячи та не потребують систематичних важких фізичних навантажень, підняття та перенесення важких речей. В таких умовах енерговитрати працівників складають не більше 120 ккал/год. Отже, виконувані роботи відносяться до категорії 1а .

Згідно ДСН 3.3.6.042-99 та категорії виконуваних робіт, для приміщення встановлені наступні оптимальні норми мікроклімату: температура в літній період 23 – 25 °С, в зимовий період 22 – 24 °С, відносна вологість повітря 40 – 60 %, швидкість руху повітря ≤ 0,1 м/с.

Щоб забезпечити необхідні норми мікрокліматичних параметрів та чистоти повітря в лабораторії використовується кондиціонування. Підтримка зазначених параметрів в холодний період здійснюється системою опалення відповідно до СНиП 2.04.05-91.

Ще одним шкідливим фактором є розумове перенапруження. Основні заходи щодо захисту від розумового перенапруження людини:

– регулярно, через кожні 40 – 50 хв. робочого часу робити технологічні перерви, для розрядки розумового напруження;

– проведення фізичних вправ (зарядки, розминки) під час перерви.

Рівень загального штучного освітлення приміщення можна перевірити за допомогою методу питомої потужності.

Розрахунок штучного освітлення проводиться методом коефіцієнта використання світлового потоку. Мета перевірочного розрахунку – визначення фактичної освітленості в приміщенні. Основна розрахункова формула методу коефіцієнта використання світлового потоку:

$$F = \frac{E_f k_z S_z}{n\eta\gamma N}, \qquad (4.1)$$

де E_f – фактична освітленість, лк;

S – площа освітлюваного приміщення, що залежить від розмірів приміщення;

z – коефіцієнт нерівномірності освітленості (z=1,1);

k_z – коефіцієнт запасу, що враховує запилення світильників та знос джерел запасу світла в процесі експлуатації;

N – кількість світильників в ряду;

η – коефіцієнт використання світлового потоку ламп;

γ – коефіцієнт затінення (γ=0,8);

n – кількість рядів світильників;

У таблиці 4.1 наведено вихідні дані.

Таблиця 4.1 – Вихідні дані

F_l	$n_л$	N	Π	N	$k_з$
1600	2	2	0.5	4	1.5

Фактична освітленість складає:

$$E_f = \frac{Fn\eta\gamma N}{k_z S_z}, \tag{4.2}$$

$$F = F_i n_i. \tag{4.3}$$

$$F = 1600 \cdot 2 = 3200 \text{ Лк.}$$

$$E_f = \frac{3200 \cdot 2 \cdot 0{,}5 \cdot 4 \cdot 0{,}88}{1{,}5 \cdot 14{,}07 \cdot 1{,}1} = 441 \text{ Лк.}$$

За результатами розрахунку отримали освітленість складає 441 лк, що належить допустимому діапазону 200 – 500 лк. У приміщенні нестачі штучного освітлення немає.

4.4 Пожежна безпека лабораторії

Приміщення лабораторії за вибухо- та пожежною небезпекою відноситься до категорії В за НАПБ Б.03.002-2007, оскільки в приміщеннях лабораторії знаходяться тверді горючі матеріали [19]. Основними причинами виникнення пожежі можна вважати:

– випадкове пошкодження ізоляції струмоведучих провідників;

– незадовільний стан вилок, розеток;

– перенавантаження дротів живлення.

Лабораторія розташована в будівлі II ступеня вогнестійкості відповідно до ДБН В.1.1.7-2016 .

За пожежонебезпекою, згідно з ПУЕ-2011, лабораторія належить зоні П – IIа [19]. У ній присутні тверді горючі речовини, папір, меблі, одяг.

Заходи пожежної профілактики відповідно до ГОСТ 12.1.004-91. ССБТ, до яких входять: застосування системи запобігання пожежі, протипожежного захисту, виконання організаційних заходів.

Згідно НАПБ Б03.001-2004 в приміщеннях розміщені первинні засоби пожежогасіння – вуглекислотний вогнегасник ВВК-1,4 з розрахунку 1 вогнегасник на 3 ПК, але не менше 1 на приміщення [19].

Згідно ДБН В.2.5.56-2010 в приміщеннях встановлено точковий димової-пожежний сповіщувач ДІП-1, який контролює площу до 86 м^2.

ВИСНОВКИ

У межах кваліфікаційної роботи було здійснено комплексне дослідження в галузі мобільної робототехніки з акцентом на реалізацію функціоналу робота-кур'єра. Зокрема, було проаналізовано існуючі типи мобільних роботів, що дозволило обґрунтовано обрати TurtleBot 4 як базову платформу для реалізації поставленого завдання.

На основі аналізу технічних характеристик TurtleBot 4 було визначено його переваги та обмеження у контексті завдань автономної навігації. Розглянуто принципи керування мобільним роботом та особливості його інтеграції в симульоване середовище, що дозволило створити програмний макет умов, наближених до реальних.

Для розв’язання задачі побудови маршруту розроблено та реалізовано хвильовий алгоритм, який дозволяє ефективно визначати оптимальний шлях з урахуванням статичних перешкод. На основі цього алгоритму створено програмне забезпечення для автономного переміщення робота-кур’єра між заданими точками.

Також у роботі приділено увагу питанням охорони праці, що є важливим аспектом при роботі з технічними засобами та у процесі розробки програмного забезпечення.

Отримані результати можуть бути використані для подальших досліджень та вдосконалення систем автономної навігації мобільних роботів, зокрема в логістичних та сервісних сферах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015 Інформація та документація «Звіти у сфері науки і техніки». Структура та правила офомрлювання. / В. Земцева; Ю. Поліщук, канд. фіз.-мат. наук; Р. Санченко, канд. техн. наук; Л. Шрамко; А. Ямчук (науковий керівник) ДП «УкрНДНЦ» від 22 червня 2015р. № 61 з 2017- 07-01.

2. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І. Ш. Невлюдов, О. І. Филипенко, О. В. Токарєва, С. П. Новоселов, О. В Сичова. – Харків: ХНУРЕ, 2023. – 64 с.

3. 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.

4. Технічні засоби автоматизації: Підручник / І. Ш. Невлюдов, А. О. Андрусевич, О. І. Филипенко, Н. П. Демська, С. П. Новоселов. – Кривий Ріг : Криворізький коледж НАУ, 2019. – 366 с.

5. Робот Amazon Scout [Електронний ресурс]. – Режим доступу: https:// https://www.amazon.com/Moorebot-Scout-Model-Waterproof-Monitoring/dp/B0CQ6QTXH1 / (Дата звернення 29.04.2025). – Загл. з екрану.

6. Робот Starship [Електронний ресурс]. – Режим доступу: https://www. https://www.starship.xyz/ (Дата звернення 30.04.2025). – Загл. з екрану

7. Робот Gita [Електронний ресурс]. – Режим доступу: https://www.piaggio.com/us_EN/piaggio-world/about-us-piaggio/the-revolutionary-nature-of-gita-and-kilo/ (Дата звернення 02.05.2025). – Загл. з екрану.

8. Робот TwinswHeel H04 [Електронний ресурс]. – Режим доступу: https://www.twinswheel.fr/ (Дата звернення 04.05.2025). – Загл. з екрану.

9. Робот TurtleBot 4 [Електронний ресурс]. – Режим доступу: https://robotics.ua/news/prototypes/6112-turtlebot_2i_novyj_mobilnyj_robot/ (Дата звернення 06.05.2025). – Загл. з екрану.

10. Handbook of Industrial Robotics. 2nd ed / Ed. by S. Y. Nof. – New York: John Wiley & Sons, 2019. – 1378 p. – ISBN 978-0-471-17783-8. – P. 3-5.

11. Хазіна С. А. Комп’ютерне моделювання фізичного процесу у різних програмних середовищах / М-во освіти і науки України, Нац. пед. ун-т імені М. П. Драгоманова. – К. : НАУ, 2018. – С. 93-97.

12. Хвильовий алгоритм [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Хвильовий_алгоритм (Дата звернення 01.05.2025). – Загл. з екрану.

13. Опис хвильового алгоритму [Електронний ресурс]. – Режим доступу: https://habr.com/post/264189/ (Дата звернення 02.05.2025). – Загл. з екрану.

14. Robin Nixon. Building Dynamic Websites with PHP, MySQL, JavaScript, CSS and HTML5, 6th Edition / Publisher(s): O'Reilly Media, Inc., 2021. – 828 p.

15. Jurgen Wolf. HTML and CSS: The Comprehensive Guide / Publisher(s): Rheinwerk Computing, 2023. – 814 p.

16. Ben Frain. Responsive Web Design with HTML5 and CSS. Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition / Publisher(s): Wiley, 2020. – 408 p.

17. Mikael Olsson. Java 17 Quick Syntax Reference. A Pocket Guide to the Java SE Language, APIs, and Library. 3rd Ed. / Publisher(s): Apress, 2022. – 132 p.

18. Douglas Crockford. How JavaScript Works / Publisher(s): Douglas Crockford, 2021. – 280 c.

19. John Paxton, Adam D. Scott & Shelley Powers. JavaScript Cookbook: Programming the Web. 3rd Ed. / Publisher(s): O'Reilly Media, Inc., 2021. – 650 p.

20. Комплекс навчально-методичного забезпечення навчальної дисципліни «Організація керування умовами праці» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [Електронний ресурс] / ХНУРЕ; розроб.: Т.Є. Стиценко, Г.В. Пронюк, Н.М. Сердюк. – Харків, 2017. – 108 с.

21. 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.

22. Nevliudov, I., & et al. (2022). Object Recognition for a Humanoid Robot Based on a Microcontroller. In 2022 IEEE XVIII International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), IEEE, 61-64.

23. Nevliudov, I., & et al.. (2020). Method of Algorithms for Cyber-Physical Production Systems Functioning Synthesis. International Journal of Emerging Trends in Engineering Research, 8(10), 7465-7473.

24. Невлюдов І.Ш. Виробничі процеси та обладнання об’єктів автоматизації. Збірник задач: Навчальний посібник / І.Ш. Невлюдов, А.О. Андрусевич, Г.В. Пономарьова, А.О. Функендорф. Кривий Ріг: КК НАУ. 2018. – 332 с.

25. Теорія автоматичного управління (збірник задач) [Текст]: навч.посіб. для студентів спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / І.Ш. Невлюдов, О.В. Токарєва; Харків. нац. ун-т радіоелектроніки. - Харків: Панов А.М., 2020. – 240 с.

26. Невлюдов І.Ш., Євсєєв В.В., Максимова С.С. BEAM робототехніка: Навчальний посібник. – Oktan Print – Prague.: 2024.- 276 с.

27. Невлюдов І.Ш. Комп’ютерно-інтегровані технології виробництва технічних засобів автоматизації. Частина 1: підручник. Харків: ФОП Панов А.М., 2021. – 604 с.

28. Основи наукових досліджень : підручник / І. Ш. Невлюдов, Ю. М. Олександров, А. О. Андрусевич, О. О. Чала ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Prague : OKTAN PRINT, 2024. – 468 с.

29. Технічне та програмне забезпечення розробки малогабаритного мобільного робота: монографія / І.Ш. Невлюдов, В.В. Євсєєв, Д.В. Гурін. Кривий Ріг: Криворізький фаховий коледж Державного некомерційного підприємства «Державний університет «Київський авіаційний інститут»,2025. –355с. DOI: https://doi.org/10.30837/978-617-8332-74-7