Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматики
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА РОБОТА, ЩО РУХАЄТЬСЯ ПО ЗАДАНІЙ ТРАЄКТОРІЇ

Виконав: студент 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Шамрай М.В.
(прізвище та ініціали)
Керівник: к.т.н., доцент кафедри КН
Колодний В.В.
(прізвище та ініціали)
«04» червня 2025 р.
Рецензент: к.т.н., доцент кафедри АПТ
Барабан М.В.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра Комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
"05" травня 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шамраю Максиму Віталійовичу

1. Тема роботи: Розробка робота, що рухається по заданій траєкторії.
керівник роботи: Колодний Володимир Володимирович, к.т.н., доцент
затверджені наказом вищого навчального закладу "20" березня 2025 року №97

2. Строк подання студентом роботи 30 травня 2025 р

3. Вихідні дані до роботи :
Траєкторія руху – довільна, задана наперед;
Кількість колісних пар – 1од;
Кількість давачів лінії – 2шт;
Мова програмування – об'єктно-орієнтована;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
вступ; обґрунтування доцільності розробки робота, що рухається по заданій траєкторії; розробка апаратної частини робота, що рухається по заданій траєкторії; розробка програмної частини робота, що рухається по заданій траєкторії; список використаних джерел; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
схема алгоритму роботи робота, що рухається по заданій траєкторії; монтажна схема робота, що рухається по заданій траєкторії; загальна структурна схема функціонування робота, що рухається по заданій траєкторії.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Колодний В.В., к.т.н., доцент		

6. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1.	Обґрунтування доцільності розробки робота, що рухається по заданій траєкторії	05.05.25	07.05.25	
2.	Розробка апаратної частини робота, що рухається по заданій траєкторії	08.05.25	13.05.25	
3.	Розробка програмної частини робота, що рухається по заданій траєкторії	14.05.25	18.05.25	
4.	Розробка інструкції користувача.	19.05.25	29.05.25	
5.	Висновки та аналіз отриманих результатів	30.05.25	01.06.25	
6.	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ Шамрай М.В.
(підпис) (прізвище та ініці

Керівник проєкту (роботи) _____ Колодний В.В
(підпис) (прізвище та ініці

АНОТАЦІЯ

УДК 621.374.415

Шамрай М.В. Розробка робота, що рухається по заданій траєкторії. Бакалаврська кваліфікаційна робота складається з 76 сторінки формату А4, на яких є 42 рисунків, 4 таблиці, список використаних джерел містить 20 найменуваннь.

В роботі обґрунтовано доцільність розробки робота, що рухається по заданій траєкторії. Такий робот може працювати, коли задано завчасне програмування точок маршруту або лінійних сегментів, по яких робот рухається з високою точністю, забезпечуючи повторюваність і стабільність виконання завдання. Це дозволяє досягати високої ефективності, безпеки та якості робіт у різних галузях.

Програмна частина реалізована на базі створеного алгоритма руху по запрограмованій траекторії. Програмна реалізація виконана на мові програмування Wiring для середовища Arduino IDE.

Ключові слова: Arduino, траекторія руху, лінія, Arduino IDE, датчик лінії, Wiring.

ABSTRACT

Shamray M. Development of work moving along a given trajectory. Bachelor's qualification work consists of 76 pages of A4 format, which has 42 figures, 4 tables, the list of sources used contains 20 names.

The work substantiates the feasibility of developing a robot moving along a given trajectory. Such a robot can work when premature programming of route points or linear segments, on which the robot moves with high accuracy, ensuring the repetitiveness and stability of the task. This allows you to achieve high efficiency, safety and quality of work in various fields.

The software part is implemented on the basis of the created algorithm of traffic. The software implementation is made in Wiring programming for Arduino Ide.

Keywords: Arduino, traffic, line, Arduino Ide, line sensor, Wiring.

ЗМІСТ

ВСТУП

Актуальність теми. На сьогоднішній день ми можемо спостерігати, як швидко розвиваються технології, і поступово комп’ютер починає заміняти людину в різних сферах діяльності. Адже людина не в змозі так швидко оперувати великими об’ємами даних та на їх основі приймати правильні рішення. Хоча досить часто від цих рішень залежить життя інших людей. Робот – це машина, яка програмується людиною за допомогою комп'ютера, та здатна автоматично виконувати визначений ряд дій. Роботи широко застосовуються на виробництві для виконання багато повторних простих дій, що зазвичай виконувались людьми [1].

Перші такі роботи були маніпулятивними, тобто керування здійснювалось людиною за допомогою зовнішнього керуючого пристрою. Такі машини призначені для виконання одноманітної роботи, такої як складання, переміщення, тощо.

Автономний робот – це інтелектуальна машина, здатна виконувати певну поведінку або комплекс завдань із високим рівнем самостійності, тобто без безпосереднього втручання людини чи зовнішньої системи управління. Така здатність дозволяє роботам орієнтуватися в навколишньому середовищі, приймати рішення на основі сенсорних даних, адаптувати свою поведінку до змін у зовнішньому середовищі та навчатися з досвіду.

Автономна робототехніка — це міждисциплінарна галузь, яка поєднує знання та технології з кількох сфер: штучного інтелекту (ШІ), робототехніки, інформаційної інженерії, обробки сигналів, машинного навчання, механіки та електроніки. Застосування алгоритмів ШІ дозволяє автономним роботам не лише аналізувати отриману інформацію, а й приймати оптимальні рішення для виконання завдань у динамічному середовищі.

Сучасні автономні роботи знаходять застосування в широкому спектрі сфер: від побутових приладів (наприклад, роботів-пилососів) до складних промислових, військових і космічних систем. Вони можуть виконувати завдання, які є надто небезпечними, складними або рутинними для людей, зменшуючи ризики та підвищуючи ефективність виконання роботи. Таким чином, розвиток автономної

робототехніки є ключовим чинником у технологічному прогресі та цифровій трансформації сучасного суспільства [2].

Робота можна вважати автономним, якщо він виконує наступні вимоги:

- отримує інформацію про зовнішнє довкілля;
- уникає ситуацій, шкідливих для себе, майна або людини;
- працює без втручання людини;
- переміщує себе повністю або частково в межах свого робочого простору без втручання людини.

Для автономного робота також властива адаптація до нового середовища, та знаходження нових шляхів розв’язання поставлених перед ним задач.

Саме це і є причиною того, чому автономні автомобілі зараз настільки актуальні, популярні та розвиваються з неймовірною швидкістю.

Метою бакалаврської кваліфікаційної роботи є розширення функціональних можливостей програмного забезпечення управління роботом, що рухається по заданій траєкторії.

Об’єктом дослідження є процес управління роботом, що рухається по заданій траєкторії.

Предметом дослідження є програмні та апаратні засоби управління роботом, що рухається по заданій траєкторії.

Для досягнення поставленої мети необхідно виконати такі задачі:

- обґрунтувати доцільність розробки робота, що рухається по заданій траєкторії;
- розробити апаратну частину робота, що рухається по заданій траєкторії;
- розробити програмну частину робота, що рухається по заданій траєкторії;
- виконати тестування робота, що рухається по заданій траєкторії та провести аналіз отриманих результатів.

Результати дослідження впроваджено в роботу ТОВ «ІТІ».

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ РОБОТА, ЩО РУХАЄТЬСЯ ПО ЗАДАНІЙ ТРАЄКТОРІЇ

1.1 Аналіз основних способів розробки мобільних роботів

Сьогоденний ринок вимагає від бізнесу постійного вдосконалення та адаптації до нових технологій. Особливо це помітно у сфері виробництва та торгівлі, де стрімкий розвиток виробництва диктує нові правила гри. Зростаючий обсяг онлайн-замовлень вимагає автоматизації небезпечних операцій і оптимізації доставок. Розробка роботів, здатних до автономного переміщення за заданою траєкторією, стає ключовим елементом вирішення цих завдань, відкриваючи шлях до кращої ефективності та конкурентоспроможності.

Такі роботи мають значний потенціал для оптимізації бізнес-процесів, особливо в сфері виробництва та складського господарства. Автоматизація рутинних операцій з переміщення товарів дозволить суттєво підвищити швидкість та точність їх виконання, як наслідок зменшуються витрати. З огляду на це, розробка та впровадження таких роботів є економічно доцільним та перспективним напрямком інвестицій для сучасного бізнесу.

Впровадження робототехніки в логістику та виробництво дозволяє компаніям отримати значні конкурентні переваги, підвищуючи ефективність бізнесу та збільшуючи прибуток. Зниження витрат на оплату праці, підвищення продуктивності, оптимізація логістики - це лише деякі з економічних переваг, які роблять розробку та виготовлення роботів, що рухаються по заданій траєкторії, економічно виправданим та перспективним напрямком для інвестицій.

На сьогодні автономні мобільні роботи застосовуються в різних сферах діяльності, ось декілька найпопулярніших сфер в яких вони вже добре зарекомендували себе:

- марсоходи, які працюють в автономному режимі, досліджуючи різні планети, та передаючи інформацію людині;

- навантажувачі на фабриках та великих підприємствах;

- сільськогосподарські роботи для автоматичного збору/посіву врожаю;
- роботи - кур’єри;
- військові роботи;
- автопілоти.

Це спричинило стрімкий ріст популярності різного роду змагання серед автономних мобільних роботів. Саме тому проєкт по створенню автономного мобільного робота, здатного рухатися заданою траєкторією без втручання людини є актуальним.

Сьогодні провідні автомобільні компанії направили свої зусилля на розробку та будівництво машин, що являються автономними мобільними роботами, керування якими відбувається за допомогою штучного інтелекту на основі машинного навчання.

Особлива роль у вирішенні завдань управління автономними роботами належить інформаційно-сенсорній системі, яка повинна самостійно аналізувати поточну ситуацію, планувати свої дії і при цьому взаємодіяти з людиною–оператором мовою, близькою до природної мови. Вона повинна самостійно шукати і виявляти небезпечні предмети, вільно переміщатися в своєму просторі, в якому можуть знаходитися і інші рухомі об'єкти. При втраті зв'язку з оператором робот повинен самостійно, використовуючи отриману інформацію про зовнішній світ, повернутися назад на вихідну позицію.

Управління автономним роботом з боку оператора набуває нового характеру. Це вже не безпосереднє управління рухом, а постановка завдань. Оскільки умови виконання завдань не завжди дотримуються, управління набуває характеру діалогу між людиною і інтелектуальною системою управління. Такого роду робото-технічні системи називають системами кооперативного управління.

Область застосування автономних роботів дуже широка. Це пошук та знешкодження небезпечних об'єктів, завдання радіаційної та хімічної розвідки, робота в зоні техногенних і природних катастроф. Такі робототехнічні системи знаходять застосування і в цивільній сфері в якості сервісної робототехніки.

Сервісні роботи вже з'явилися і успішно виконують функції обслуговування відвідувачів в музеях, аеропортах, магазинах.

Автономні мобільні роботи (АМР) швидко увійшли та впевнено посіли нішу логістичної сфери підприємств. Майже у всіх сферах діяльності автономні мобільні роботи починають заміняти застаріле транспортне устаткування. Автономні мобільні роботи збільшують продуктивність у логістичній сфері підприємства, здатні функціонувати без участі персоналу та швидко окупають витрачені на модернізацію кошти.

Автономні мобільні роботи класифікуються по трьом основним характеристикам:

- середовище, у якому вони переміщуються;
- пристрій переміщення;
- вид навігації.

По типу середовища у якому переміщуються роботи існує 3 класи автономних мобільних роботів:

- наземні - крокуючі, колісні та гусеничні мобільні роботи, планетарні марсоходи і місяцеходи, транспортні роботи вантажники, безпілотні автомобілі;
- морські - підводні, надводні, автономні роботи-батискафи, катери;
- повітряні - гелікоптери з великим показником вантажопідйомності, а також компактні дрони, керовані автопілотом.

Системи навігації також діляться на:

- пасивні - коли передача сигналів іде від зовнішнього джерела або маркера;
- активні - коли визначення місцезнаходження вираховується роботом за допомогою GPS-навігації, тощо.

1.2 Поняття мобільних роботів

Робот, як термін, можна розуміти як багатофункціональний автомат для виконання механічної роботи, схожу на ту, що виконує людина. Думка про створення робота з'явилася після прагнення замінити людину на робота в важких і

небезпечних для життя ситуаціях. З плином часу поняття робот стало ширше, під ним стали представляти будь-яку автоматичну машину, яка може замінити людину, і яка буде нагадувати його розумну поведінку.

Мобільний робот вміє пересуватися в робочому просторі відповідно до програми керування. Такі роботи необхідно програмувати заздалегідь. Повинна бути присутня здатність самостійно орієнтуватися в навколишньому середовищі і здійснювати виконання завдання, спираючись тільки на власний штучний інтелект. Тому ці роботи і мають назву – мобільні, тому що вони не прив'язані до оператора.

Мабуть, найпоширеніші сьогодні і корисні кожній домогосподарці роботи - це пилососи. Інфрачервоні датчики стежать, як за типом поверхні (паркет або килим), щоб вибрати відповідну щітку або режим роботи, так і за тим, щоб пилосос не скотився вниз сходами. Можна придбати робот-пилосос ROBOKING від LG (рис.1.2).

Рисунок 1.2 – Робот пилосос ROBOKING

Свій варіант пропонує і Panasonic (рис 1.3). Або, скажімо, Trilobit від Electrolux (рис. 1.4).

Рисунок 1.3 – Робот пилосос Panasonic

Рисунок 1.4 – Робот пилосос Trilobit

Робот-пилосос орієнтується в просторі так само, як кажани, - за допомогою ультразвуку: покритий тонкою золотою пластиною акустичний локатор з великою кількістю мікрофонів відчуває вібрації на частоті 60кГц. А напівкругла форма дає роботу кут огляду 1800. Таким чином, Trilobite швидко визначає стіни, ніжки стільців, дверні отвори та сходи. Він легко долає проводи, кабелі і границю "підлога - килим". Зовні Trilobite найбільше схожий на круглу диванну подушку, а його невеликі розміри (висота 13 см і діаметр 35 см) дозволяють йому з легкістю прибирати навіть під ліжками та столами. Робот рухається на двох колесах, кожне з яких має свій двигун, а у Trilobite — чотири двигуни. Максимальна швидкість збирання - 40 квадратних сантиметрів за секунду. При тому, що людина з пилососом насправді прибирає пил лише з 60% доступної площі, Trilobite охоплює 95%. Під час збирання в електронних "мізках" робота-пилососа формується певна "карта місцевості". Крім того, робот обчислює, скільки часу потрібно для прибирання. Коли акумулятори робота "сідають", Trilobite сам знаходить зарядний пристрій (charging station) і їде заряджатися. Якщо батареї "сіли" раніше, ніж робот закінчив чистку, Trilobite - як тільки зарядиться - автоматично відновлює прибирання з того самого місця, де він її припинив. На жаль, робот не може сам вивантажити зібраний пил і, хоча мішки для збору пилу не потрібні - є "коробка багаторазового використання для пилу" (reusable dust box) - її господарям доведеться витрушувати самим. Керувати роботом-пилососом потрібно за допомогою LCD-дисплея - можна вибрати одну з трьох програм прибирання: звичайне, швидке та "місцеве", тобто на площі до двох квадратних метрів.

1.3 Класифікація мобільних роботів за функціональним призначенням

Розвиток технології штучного інтелекту передбачає широке використання роботів у різних галузях промисловості, інфраструктури та життєдіяльності людини. Прийняття рішень у виборі або розробці тієї чи іншої робототехнічної системи базується на аналізі та використанні системного підходу. Першим етапом є систематизація існуючих рішень, виділення значущих ознак та визначення переваг та недоліків таких рішень.

Відповідно до Згуровського М.З. [5] – сьогодні відбувається інформаційна революція: перехід суспільства від індустріального до інформаційного, а надалі очікується його перетворення на суспільство знань. За різним оцінкам, 30–60 % робочих місць замінять робототехнічними системами до 2035 року.

У зв'язку з цим, необхідно вивчати та розвивати напрями, пов'язані з робототехнікою, зокрема – мобільними робототехнічними системами. Для вибору того чи іншого робота необхідно провести аналіз та класифікацію існуючих рішень, а також виявити їх переваги та недоліки (табл. 1.1).

Таблиця 1.1 – Класифікація мобільних роботів.

Група	Базові вимоги	Базовий варіант використання
Спеціального призначення	Компактність, безшумність	Автономне, один канал передачі відеоінформації
Для військових і воєнізованих застосувань	Надійність, простота в управлінні, стандартизована корисне навантаження	У складі розвідувальних, ударних і охоронних комплексів, багатоканальні системи передачі різноманітних даних
Для екстремальних ситуацій, наукових досліджень, кінематографічних застосувань	Стійкість до несприятливих зовнішніх впливів, універсальність по відношенню до корисного бортового навантаження	Автономне, багатоваріантність реалізації каналів передачі даних
Для спортивних, промислових і побутових застосувань	Простота в управлінні, економічність, надійність	Автономне, один канал передачі відеоданих

1.4 Узагальнена структура мобільного робота

По конструкції багатофункціональні роботи представляють собою самохідні засоби, обладнані апаратурою розвідки, набором робочого обладнання, який можна замінити і інструментом. Вони мають на увазі наявність дистанційного керування і можливість вести відеоспостереження.

Рух мобільного робота в приміщенні, план якого заздалегідь невідомий має свої особливості. У приміщенні є як статичні перешкоди (стіни, столи, стільці), так і рухливі (люди, інші роботи). Мобільний робот оснащений скануючим лазерним далекоміром, який отримує стан рельєфу навколишніх об'єктів в площині, паралельній поверхні. Необхідно в режимі реального часу визначати положення мобільного робота в системі координат, пов'язаної з приміщенням (завдання локалізації), а також побудувати карту даного приміщення, що відображає рельєф стін і нерухомих об'єктів. Такого роду системи управління відомі як системи SLAM (System of Localization and Mapping). Функціональна структура навігаційної системи мобільного робота показана на рисунку 1.5.

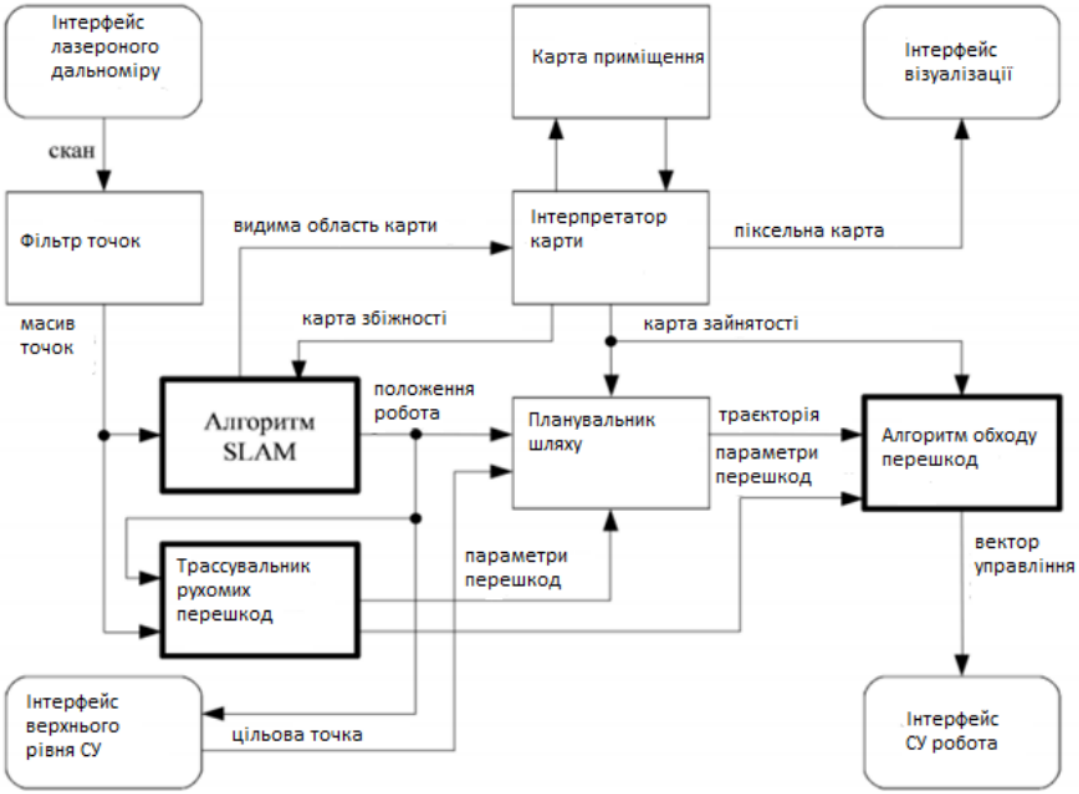


Рисунок 1.5 – Функціональна схема навігаційної системи мобільного робота

Особливістю запропонованої структури є незалежність від типу шасі мобільного робота, а також від наявності та типу датчиків одометра, що дозволяє використовувати розроблювану навігаційну систему на всіх типах мобільних роботів, які працюють в приміщенні.

Для будь-яких мобільних роботів, орієнтування на місцевості невід'ємна частина їхнього існування. Уникнення небезпечних ситуацій, таких як зіткнення з перешкодами та ризикованих умов (високі або низькі температури, потрапляння у воду чи радіація), але при розробці роботів враховуються умови в яких система буде використовуватись та враховуються можливі ситуації. В даній роботі будуть розкриті способи керування та побудови навігації для роботизованих систем, типи навігації, висвітлені їхні недоліки та запропоновані шляхи їх вирішення.

Навігація роботизованих систем включає в себе здатність визначення власного положення відносно стартового положення та побудова шляху до заданої точки в навколишньому середовищі, для цього їм потрібна мапа цього середовища. Тому на даний час це дуже актуальна тема дослідження, для розвитку сфери розробки роботизованих систем, яка скоро ввійде в нашу буденність та буде допомагати в нашому повсякденному житті.

Навігація мобільних роботів охоплює великий діапазон різних технологій і застосувань. Вона спирається як на дуже старі технології, так і на самі новітні досягнення науки і техніки.

Окремим завданням системи ближньої навігації є слідування маршруту. До її складу входять планування послідовності значних змін маршруту, подолання пересічених і викривлених ділянок, крутих спусків і підйомів дороги, а також, забезпечення навігації при наявності інших об'єктів. Таким чином, ця задача основна для всієї навігації робота та була пов'язана з першими етапами розробки навігаційних систем мобільних роботів.

Лінійна навігація також використовує орієнтири для позиціонування. Даний спосіб навігації можна виділити, як навігація по безперервним орієнтирам, але такий вид має дуже значний недолік в тому, що датчик фіксування орієнтиру повинен знаходитись в безпосередній близькості до лінії маршруту. Даний тип навігації

зазнав широкого застосування на довгі роки в задачах промислової автоматизації, пристрої які працювали в такий спосіб зазвичай називали Автоматично Керовані Пристрої. Однак, такий вид навігації був не достатньо детально вивчений, в наслідок цього спосіб виключає самостійну побудову маршрутів переміщення робота.

Основні способи для реалізації лінійної навігації:

- електромагнітне керування;

- керування відбиваючою або оптичною стрічкою;

- керування магнітною стрічкою, де використовується феріто-магнітне напилення;

- керування по термальним маркерами.

1.5 Висновок до розділу 1

В даному розділі проаналізовано основні види рухомих мобільних роботів. Наведено класифікацію мобільних роботів за функціональним призначенням. Створено узагальнену схему управління мобільним роботом, що рухається по заданій траєкторії.

2 РОЗРОБКА РОБОТА, ЩО РУХАЄТЬСЯ ПО ЗАДАНІЙ ТРАЄКТОРІЇ

2.1 Використання лінійної навігації для керування рухомим роботом

В даному способі навігації для побудови шляху руху використовуються маршрути у вигляді ліній, які наносяться на підлогу, ці лінії можуть бути виконані у вигляді яскравих білих ліній або ж можна нанести на підлогу магнітні лінії. Для реалізації такого способу компанія “RoboteQ” розробляє спеціальні датчики які застосовуються в лінійній навігації, платформа керування обладнується датчиком “RoboteQ MGS1600C”, який фіксує магнітне поле, (рис. 2.1).

Рисунок 2.1 – Давач, який фіксує магнітну лінію

Порівняння декількох способів побудови маршрутів для лінійної навігації використовуючи датчики фіксування кольорових ліній, магнітних ліній та індукційних кабелів наведено в таблиці 2.1

Таблиця 2.1 – Порівняння способів побудови маршрутів

Назва	Легко наносити	Легко змінити	Безпечне	Стійке до бруду	Непомітне
Магнітна стрічка	Так	Так	Так	Так	Так
Індукційний кабель	Ні	Ні	Ні	Так	Так
Оптична стрічка	Так	Так	Так	Ні	Ні

З порівняння отримуємо, що найкращим способом є використання магнітних стрічок, які легко можна монтувати на підлозі, та легко побудувати маршрути переміщення для роботів, до того ж вони не використовують електроенергію та стійкі до забруднення, вони не помітні та їх можна розташувати під покриттям на підлозі, за умови якщо покриття підлоги зроблене з матеріалів, які не перешкоджають проходженню магнітних потоків.

Такий спосіб має досить значні недоліки, він вносить свої обмеження в форму роботизованої платформи, в різноманітність маршрутів, та неможливість уникати перешкод, які виникатимуть на шляху роботизовних пристроїв. Так як цей спосіб широко застосовується в індустріальній сфері де приміщення мають велику площу та працює невелика кількість людей то цей спосіб можна застосувати. Відповідно до кожної платформи виробник рекомендує розміщувати датчик магнітних ліній виключно як зображено на рисунку 2.2.

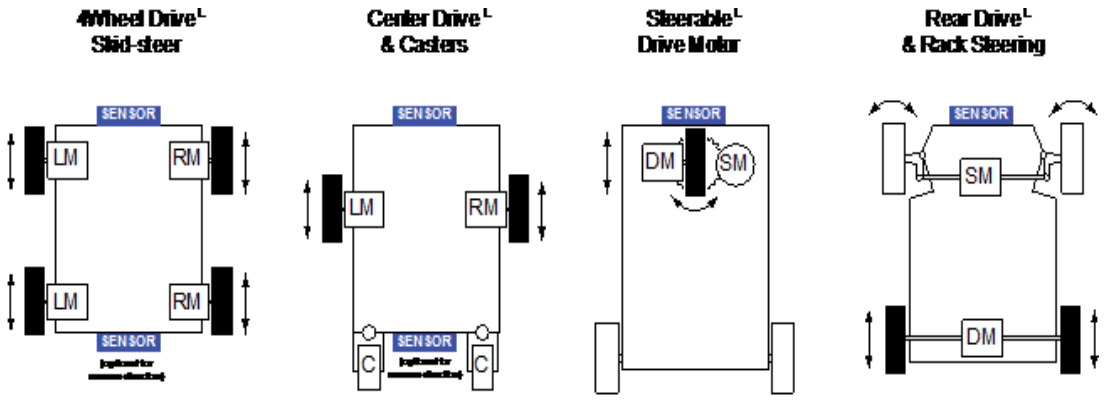


Рисунок 2.2 – Типи роботів, що використовують лінійну навігацію

Керування курсом робота дуже просте, йому потрібно просто слідувати лінії та зчитувати команди в деяких місцях, датчик фіксування магнітних ліній повертає значення, які відповідають відстані до центру лінії. Ця інформація потім використовується для корегування курсу роботизованої платформи. Якщо робот слідує паралельно лінії то значення з датчиків дорівнюють 0. Для найкращої точності та мінімальної затримки, алгоритм керування може бути повністю замінений ПІД регулятором.

Керування прискоренням робота залежить від середовища використання, але в даному випадку робот буде рухатись коли побачить стрічку, повертатиме ліворуч або праворуч і зупиниться в точно визначеному місці відповідно до того, як будуть нанесені маршрути переміщення в просторі. Робот відновить свій рух через деякий час самостійно, або коли оператор натисне кнопку. Робот постійно відслідковує наявність ліній на землі та відразу ж зупиниться, якщо лінія випадково закінчиться або буде перервана, такий спосіб більш безпечний ніж виконувати пошук самостійно.

Для надсилання команд роботу використовуються магнітні маркери, які є частиною магнітної стрічки з оберненою полярністю та розташовуються зліва і справа від центру основної лінії. Маркери – це досить простий та ефективний спосіб для ідентифікації особливих точок вздовж маршруту.

Такі маркери наносяться з лівого чи правого боку від основної лінії для подання платформі інформації, що попереду буде зміна напрямку, відповідно поворот наліво чи направо. Маркери, які розміщені по обидві сторони, сигналізують про місце де потрібно зробити зупинку (рис. 2.3).

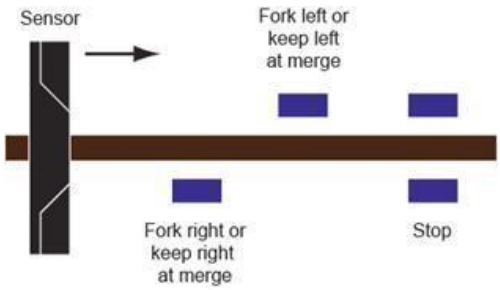


Рисунок 2.3 – Приклад розміщення командних маркерів вдовж стрічки

Приклад простого тестового маршруту зі всіма доступними для системи командами зображено на рисунку 2.4.

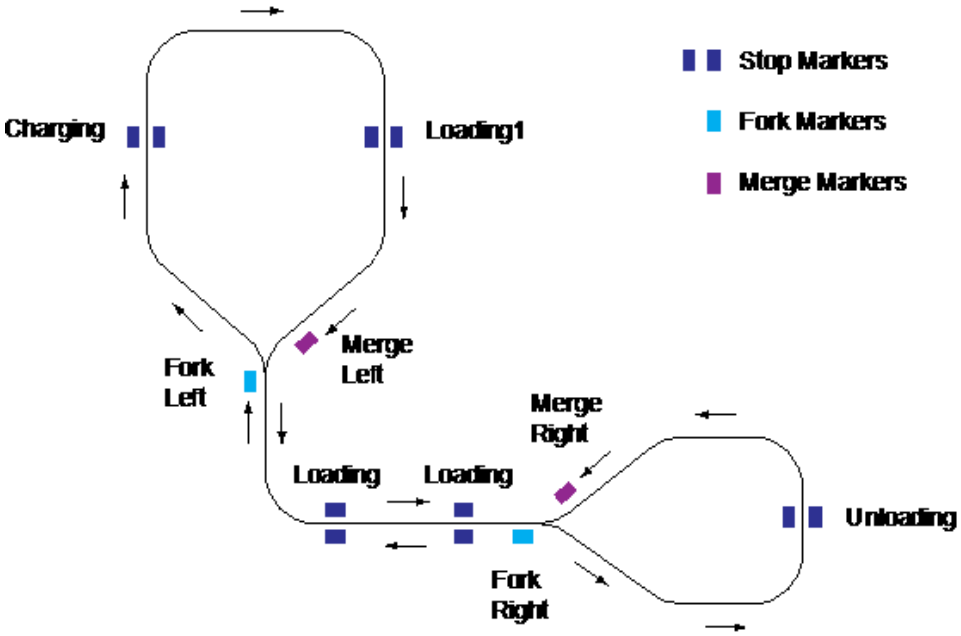


Рисунок 2.4 – Маршрут руху робота по заданій траєкторії

Такий же підхід організовано, якщо робот буде рухатися по кольоровій стрічці. Зазвичай стрічка обирається черного кольору, яка не пропускає УФ-випромінювання. А маркери управління можуть бути чорними або іншого кольору, які теж розміщуються по обидві сторони від основної лінії.

2.2 Вимоги до обладнання та вибір елементів для мобільного робота

Для реалізації робота, що рухається по заданій траєкторії знадобиться ряд компонентів для електричної частини:

- мікроконтролер Arduino – 1шт;
- двигун з прохідним валом, редуктором та колесом – 2шт;
- драйвер двигуна – 1шт;
- оптичний давач лінії KY-033 – 2шт;
- кнопка – 1шт;
- контролер TP4056 – 1шт;

- акумулятори 18650 3,7В – 2шт;

- монтажна плата – 1 шт.

Для механічної частини робота необхідна тільки платформа, на якій буде зібрано всіх необхідні елементи.

Розглянемо детальніше ці компоненти.

Плата Arduino − це ефективний засіб розробки програмованих електронних пристроїв, які, на відміну від персональних комп'ютерів, орієнтовані на тісну взаємодію з навколишнім світом. Arduino − це відкрита програмована апаратна платформа для роботи з різними фізичними об'єктами і являє собою просту плату з мікроконтролером, а також спеціальне середовище розробки для написання програмного забезпечення мікроконтролера [5].

Arduino може використовуватися для розробки інтерактивних систем, керованих різними датчиками і перемикачами. Такі системи, в свою чергу, можуть управляти роботою різних індикаторів, двигунів та інших пристроїв. Проєкти Arduino можуть бути як самостійними, так і взаємодіяти з програмним забезпеченням, що працює на персональному комп'ютері (наприклад, додатками Flash, Processing, MaxMSP). Будь-яку плату Arduino можна зібрати вручну або ж купити готовий пристрій. Середовище розробки для програмування такої плати має відкритий вихідний код і повністю безкоштовна.

Існує безліч інших мікроконтролерів і мікропроцесорних пристроїв, призначених для програмування різних апаратних засобів: Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard і багато інших. Всі ці пристрої пропонують схожу функціональність і покликані звільнити користувача від необхідності заглиблюватися в дрібні деталі внутрішнього устрою мікроконтролерів, надавши йому простий і зручний інтерфейс для їх програмування.

Контролери сімейства Arduino також спрощують процес роботи з мікроконтролером, але на відміну від інших систем надають ряд переваг для викладачів, студентів і радіоаматорів:

- низька вартість. У порівнянні з схожими апаратними платформами, плати Arduino мають відносно низьку вартість: готові модулі Arduino коштують не

дорожче 50 $, а можливість зібрати плату вручну дозволяє максимально заощадити кошти і отримати Arduino за мінімальну ціну;

- кросплатформеність. Програмне забезпечення Arduino працює на операційних системах Windows, Macintosh OSX і Linux, в той час, як більшість подібних систем орієнтовані на роботу тільки в Windows;

- просте та зручне середовище програмування. Середовище програмування Arduino зрозуміле і просте для початківців, але при цьому досить гнучке для досвідчених користувачів. Воно основане на середовищі програмування Processing, що може бути зручно для викладачів. Завдяки цьому, студенти, які вивчають програмування в середовищі Processing, зможуть легко освоїти Arduino;

- розширюване програмне забезпечення з відкритим вихідним кодом. Програмне забезпечення Arduino має відкритий вихідний код, завдяки цьому досвідчені програмісти можуть змінювати і доповнювати його. Можливості мови Arduino можна також розширювати за допомогою C++ бібліотек. Завдяки тому, що він заснований на мові AVR C, просунуті користувачі, охочі розібратися в технічних деталях, можуть легко перейти з мови Arduino на C або вставляти ділянки AVR- C коду безпосередньо в програми Arduino.

- розширюване відкрите апаратне забезпечення. Пристрої Arduino побудовані на базі мікроконтролерів Atmel ATmega8 і ATmega168. Завдяки тому, що всі схеми модулів Arduino опубліковані під ліцензією Creative Commons, досвідчені інженери і розробники можуть створювати свої версії пристроїв на основі існуючих. І навіть звичайні користувачі можуть збирати дослідні зразки Arduino для кращого розуміння принципів їх роботи і економії коштів [6].

Розглянемо детальніше різні плати сімейства Arduino.

Arduino Nano — це повнофункціональний мініатюрний пристрій на базі мікроконтролера ATmega328 (Arduino Nano 3.0) або ATmega168 (Arduino Nano 2.x), адаптований для використання з макетної плати. За функціональністю пристрій схожий на Arduino Duemilanove, і відрізняється від нього розмірами, відсутністю роз'єму живлення, а також іншим типом (Mini-B) USB-кабелю. Arduino Nano (рис. 2.5) розроблено і випускається фірмою Gravitech.

Рисунок 2.5 – Зовнішній вигляд контролера Arduino Nano

Головна відмінність цієї мініатюрної плати полягає у відсутності гнізда для зовнішнього джерела живлення, натомість використовуються VIN. Коли йдеться про створення мініатюрного пристрою, розмір Arduino Nano v3 ATmega328 / ATmega168 відіграє вирішальну роль при виборі платформи.

Технічні характеристики контролера Arduino Nano (рис. 2.6):

- мікроконтролер: ATmega328;
- тактова частота: 16 МГц;
- напруга логічних рівнів: 5 В;
- вхідна напруга живлення: 7-12 В;
- портів введення-виведення загального призначення: 22;
- максимальний струм з піна введення-виведення: 40 мА;
- максимальний вихідний струм піна 3.3V: 50 мА;
- максимальний вихідний струм піна 5V: 800 мА;
- портів з підтримкою ШІМ: 6;
- портів, підключених до АЦП: 8;
- розрядність АЦП: 10 біт;
- Flash-пам'ять: 32 КБ;
- EEPROM-пам'ять: 1 КБ;
- SRAM-пам'ять: 2 КБ;
- габарити: 18×45 мм.

Arduino Nano може живитися через кабель Mini-B USB, від зовнішнього джерела живлення з нестабілізованою напругою 6÷20В (через вивід 30) або зі стабілізованою напругою 5 В (через вивід 27). Пристрій автоматично вибирає джерело живлення з більшою напругою.

Напруга на мікросхему FTDI FT232RL подається тільки в разі живлення Arduino Nano через USB. Тому при живленні пристрою від інших зовнішніх джерел (не USB), вихід 3,3В (формований мікросхемою FTDI) буде неактивний, в результаті чого світлодіоди RX і TX можуть мерехтіти при наявності високого рівня сигналу на виводах 0 і 1.

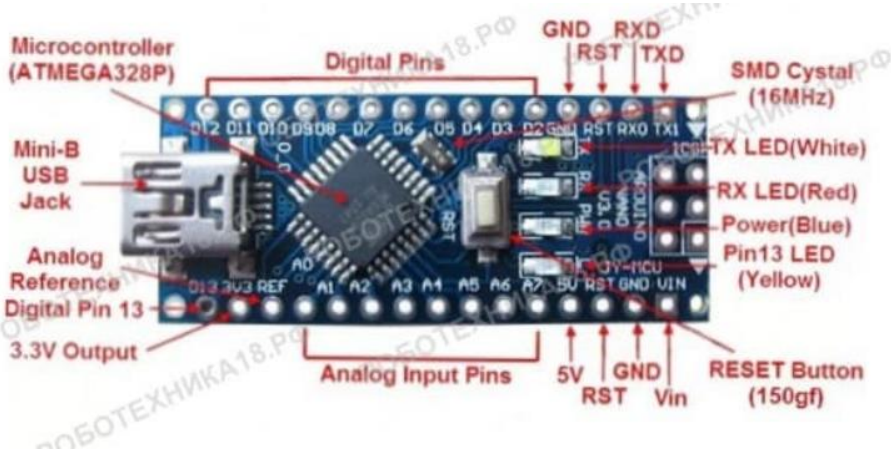


Рисунок 2.6 – Структура контролера Arduino Nano

Arduino Nano надає ряд можливостей для здійснення зв'язку з комп'ютером, ще одним Arduino або іншими мікроконтролерами. У ATmega168 і ATmega328 є приймач UART, що дозволяє здійснювати зв'язок з послідовним інтерфейсів за допомогою цифрових виводів 0 (RX) і 1 (TX). Мікросхема FTDI FT232RL забезпечує зв'язок приймача з USB-портом комп'ютера, і при підключенні до ПК дозволяє Arduino визначатися як віртуальний COM-порт (драйвера FTDI включені в пакет програмного забезпечення Arduino). У пакет програмного забезпечення Arduino також входить спеціальна програма, що дозволяє зчитувати і відправляти на Arduino прості текстові дані. При передачі даних комп'ютера через USB на платі будуть мигати світлодіоди RX і TX.

Плата Arduino Pro Mini - це пристрій на базі мікроконтролера ATmega 128. В його складі входить: 14 цифрових входів/виходів (з них 6 можуть використовуватися в якості ШІМ-виходів), 8 аналогових входів, кварцевий резонатор, кнопка Reset та контактні майданчики для підключення роз'ємів. Шестиконтактний роз'єм може служити для живлення та взаємодії з платою через USB (рис. 2.7).

Рисунок 2.7 - Плата Arduino Pro Mini

Arduino Uno – це пристрій на основі мікроконтролера ATmega328. У його склад входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (6 з них можуть використовуватися в якості ШІМ-виходів), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування (ICSP) і кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від AC/DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою USB-кабелю (рис. 2.8).

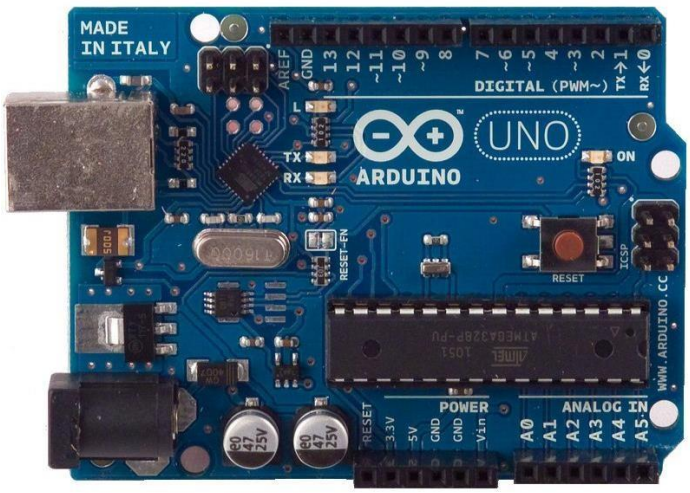


Рисунок 2.8 - Плата Arduino UNO

На відміну від всіх попередніх плат Arduino, Uno в якості перетворювача інтерфейсів USB-UART використовує мікроконтролер ATmega16U2 (ATmega8U2 до версії R2) замість мікросхеми FTDI.

Характеристики Arduino UNO:

- мікроконтролер: ATmega328;
- робоча напруга: 5 В;
- напруга живлення (рекомендована): 7-12 В;
- напруга живлення (гранична): 6-20 В;
- цифрові входи / виходи: 14 (з них 6 можуть використовуватися в якості ШІМ-виходів);
- аналогові входи: 6;
- максимальний струм одного виводу: 40 мА;
- максимальний вихідний струм виводу: 3.3V 50 мА;
- Flash-пам'ять: 32 КБ (ATmega328) з яких 0.5 КБ використовуються завантажувачем;
- SRAM: 2 КБ (ATmega328);
- EEPROM: 1 КБ (ATmega328);
- тактова частота: 16 МГц.

Arduino Uno може живитися від USB або від зовнішнього джерела живлення - тип джерела вибирається автоматично.

У якості зовнішнього джерела живлення (не USB) може використовуватися мережевий AC/DC-адаптер або акумулятор/батарея. Штекер адаптера (діаметр - 2.1мм, центральний контакт - позитивний) необхідно вставити у відповідний роз'єм живлення на платі. У разі живлення від акумулятора/батареї, її провід необхідно під'єднати до виводів Gnd і Vin роз'єму POWER.

Напруга зовнішнього джерела живлення може бути в межах від 6 до 20 В. Однак, зменшення напруги живлення нижче 7В призводить до зменшення напруги на виводі 5V, що може стати причиною нестабільної роботи пристрою. Використання напруги більше 12В може призводити до перегріву стабілізатора напруги і виходу плати з ладу. З огляду на це, рекомендується використовувати джерело живлення з напругою в діапазоні від 7 до 12В.

Для того, щоб наш робот міг рухатися, йому потрібен двигун з певним діапазономрегулювання швидкості, точність підтримки швидкості обертання приводу. Звідси випливає, що необхідним є застосування двигуна постійного струму (ДПС) з редуктором.

Невеликий колекторний двигун постійного струму з редуктором на DC3V-6V є одним із ключових компонентів для створення різних аматорських робототехнічних проєктів. Завдяки своїй універсальності та ефективності, він широко застосовується в моделях роботів, особливо в тих, що призначені для самостійного складання та навчання.

Цей двигун має гарний крутний момент, що дозволяє йому забезпечувати необхідну силу для руху роботів навіть під значним навантаженням. Його компактний розмір і зручність у використанні роблять його ідеальним вибором для ентузіастів робототехніки, які хочуть створювати мобільні платформи, роботизовані маніпулятори, або інші пристрої, що потребують точної і надійної роботи.

Редуктор, який використовується в цьому двигуні, оснащений пластиковими шестернями. Це робить конструкцію легшою та зменшує загальну вагу робота, що є важливим фактором для мобільних роботизованих систем. Водночас, пластикові

шестерні мають свої обмеження. При проєктуванні та експлуатації роботів потрібно враховувати можливі навантаження, оскільки пластикові компоненти можуть бути менш стійкими до великих механічних впливів порівняно з металевими.

Двигуни цього типу зазвичай входять до комплекту більшості платформ-шасі для самостійного складання роботів на колесах. Це дозволяє аматорам швидко та зручно інтегрувати двигун у свій проект без необхідності додаткових модифікацій. Такі комплекти часто включають не лише двигун, але й інші необхідні компоненти, що полегшує процес складання та налагодження роботизованої системи.

Таким чином, невеликий колекторний двигун постійного струму з редуктором на DC3V-6V є важливим і корисним елементом для всіх, хто захоплюється робототехнікою та бажає створювати ефективні й функціональні роботизовані системи своїми руками (рис. 2.9).

Рисунок 2.9 – Колекторний двигун постійного струму з редуктором та колесом

Основні характеристики:

- напруга живлення: 3-12В (рекомендована 3-6В);
- струм холостого ходу: 70 мА (при напрузі 3В);
- струм під навантаженням (максимальний): 250 мА (при напрузі 3В);
- момент, що крутить: 800 г/см (при напрузі 3В);
- швидкість обертання без навантаження 150-180 об/хв (при напрузі 6В);

- передаточна кількість редуктора: 1:48;

- діаметр осі: 5мм;

- габарити: 64х20х20 мм.

Драйвер двигунів MX1508 — це аналог драйвера двигунів L298N. Пристрій створений на основі мікросхеми MX1508, а принцип роботи драйвера досить простий: всередині пристрою знаходиться два транзистора, а при зміні полярності напруги відбувається зміна напрямку обертання двигунів. За допомогою такого драйвера можна створювати мобільних роботів, автономні автомобілі на основі мікроконтролерів, а також інші роботизовані пристрої з механічними модулями.

Модуль двигунів MX1508 (рис 2.10) має стандартний інтерфейс, за допомогою якого легко та швидко можна підключитися до пристроїв, створених на основі Arduino або інших мікроконтролерів. Живлення модуля здійснюється за допомогою керуючого пристрою або від зовнішньої батареї. Дана плата також дозволяє можливість ШІМ-управління для плавної зміни швидкості обертання двигуна. MX1508 може управляти двома двигунами постійного струму або 4-дротяними двофазними кроковими двигунами, в тому числі дозволяє регулювати швидкість обертання чи міняти напрям обертання, здійснювати реверс.

Характеристики:

мікросхема: MX1508;

напруга живлення модуля: 2-10 В;

напруга вхідного сигналу: 1.8-7 В;

робочий струм: 1.5 А, піковий до 2.5 А;

розміри: 25х21х6 мм;

вага: 3 г.

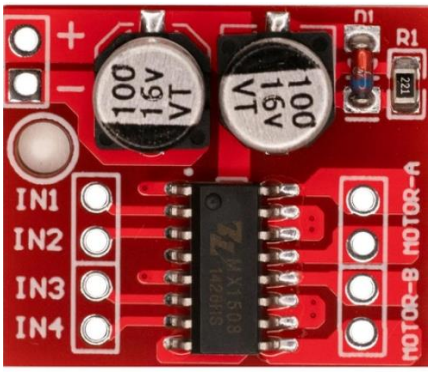


Рисунок 2.10 – Драйвер двигунів MX1508

Драйвер двигуна L298N для Arduino (рис. 2.11) використовується для управління двома малопотужними колекторними двигунами постійного струму або малопотужним 4 - х дротяним двофазним кроковим двигуном. Практичне застосування: управління двигунами невеликих колісних роботів або двигунами пересувних іграшок.

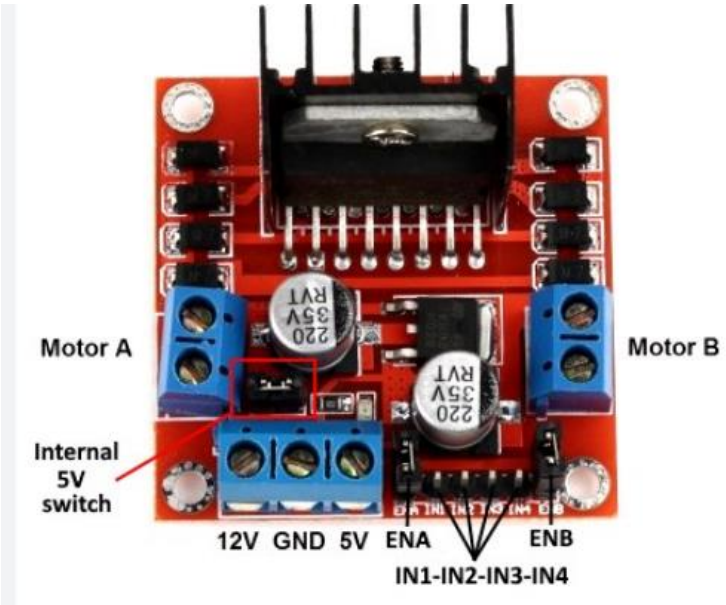


Рисунок 2.11 – Драйвер двигунів L298

Для закріплення модуля на плоскій поверхні в платі передбачено один монтажний отвір.

Управління драйвером L298N здійснюється або від Arduino контролера, або від іншого мікропроцесорного керуючого пристрою. Мікросхема модуля L298N

працює за принципом Н-моста і використовуються для зміни полярності живлення мотора, що дає можливість реверсу.

Живлення драйвера здійснюється або від Arduino контролера, або іншого мікропроцесорного керуючого пристрою, або зовнішнього джерела живлення (блоку живлення, батареї). Напруга живлення становить 2 - 9 В постійного струму.

Характеристики:

- мікросхема: L298N;
- особливість: Н-міст, можливість реверсу;
- напруга живлення: 2 - 9 В постійного струму;
- керуючий сигнал: 1,8 - 7 В постійного струму;
- максимальний споживаний струм двигунів: до 1,5 А;
- розміри: 25 х 21 х 6 мм;
- вага: 3 г.

Кнопки для Arduino (рис. 2.12) складаються з невеликої текстолітової плати, на якій розпаяні: тактова кнопка 12х12 мм, притягуючий резистор і 3 піна для підключення до Arduino. Також в комплект входять ковпачки для кнопок різних кольорів. Всього в комплекті 5 модулів кнопок та 5 різнокольорових ковпачків.

Рисунок 2.12 – Кнопки для Arduino

Контролер заряду літій-їонних акумуляторів TP4056 (рис. 2.13) - містить контролер заряду Li-Ion акумуляторів TP4056. Мікросхема має індикацію процесу заряду та сама відключає акумулятор при досягненні напруги на ньому 4,2В. У момент заряду світиться червоний світлодіод, коли батарея буде повністю заряджена, засвітиться зелений світлодіод, червоний при цьому згасне.

Процес заряджання акумулятора ідентичний зарядці мобільного телефону, яскраві світлодіоди сповіщають про закінчення заряджання акумулятора.

Модуль підходить для зарядки літій-іонних і літій-полімерних (Li-Ion, Li-Po) акумуляторів на 3,7В. Будь то акумулятори від мобільного телефону або батареї типорозмір 18650, які застосовуються в батареях для ноутбуків.

Подати напругу на пристрій можна двома способами: через роз'єм, міні USB, або шляхом паяння дротів, минаючи роз'єм міні USB.

Характеристики:

- вхідна напруга: 4.5В-5.5В;
- напруга повного заряду: 4.2В;
- струм заряду: 1А;
- вхідний роз'єм: mini USB (+ місця для підпаювання проводів);
- робоча температура: -10..+85;
- розміри модуля: 5 х 19 х 25 мм.

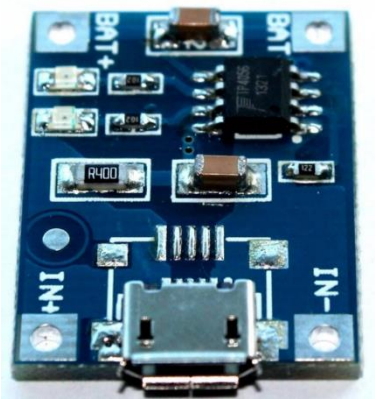


Рисунок 2.13 – Контролер заряду літій-їонних акумуляторів TP4056

Акумулятор 18650 Li-Ion LiitoKala (рис. 2.14) призначений для забезпечення працездатності портативних пристроїв різного призначення з високим постійним споживанням струму, зокрема в електроінструментах, потужних ліхтарях, електронних сигаретах, дронах тощо.

Ці акумулятори мають тривалий термін роботи та зберігають свої властивості в екстремальних умовах із кількістю циклів перезаряджання понад 500. Акумулятор 18650 Li-ion завдяки високій щільності струму забезпечує гарантовану стабільну

напругу впродовж циклу розряджання, одночасно має низький саморозряд 5—7%. Без "ефекту пам'яті", це дає можливість заряджати його без попереднього розряджання.

Рисунок 2.14 – Літій-їонний акумулятор 18650

Для нашого робота ще знадобиться тримач на 2 акумулятора типорозміру 18650 (рис. 2.15) та макетна плата (рис. 2.16) для розміщення всіх електронних компонентів.

Рисунок 2.15 – тримач на 2 акумулятора типорозміру 18650

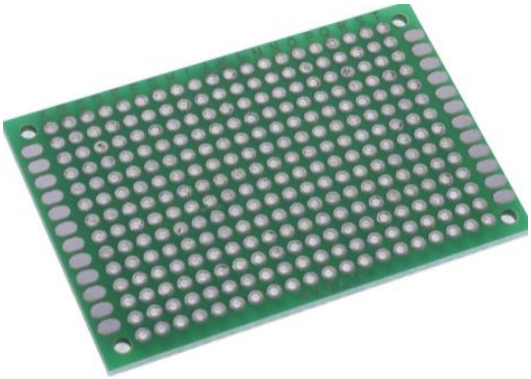


Рисунок 2.16 – Макетна плата

Для нашого робота ще знадобиться оптичний давач лінії KY-033 (рис. 2.17).

Рисунок 2.17 – Оптичний давач лінії KY-033

Оптичний давач лінії KY-033 складається з інфрачервоного випромінювача та фототранзистора, який приймає відбитий сигнал від об'єкта, розташованого на невеликій відстані – 1-20мм.

Давач можна використовувати для відстеження та руху робота по чорній лінії, так як від чорного кольору сигнал відображається погано, а при втраті роботом лінії та відображенні від білого кольору сигнал відображається добре.

Для налаштування дальності спрацьовування датчика є змінний резистор.

При спрацьовуванні світиться світлодіод на платі і на виході “S” з'являється лог. 0.

Характеристики сенсора:

- інфрачервоний датчик відображення TCRT5000;
- впевнене визначення перешкоди від 1 до 25мм;
- цифровий вихід компаратора з максимальним навантаженням 15мА;
- регулювання чутливості потенціометром;
- робоча напруга від 3.3 до 5В;
- розміри модуля: 40 х 11мм;
- пороговий компаратор напруги LM393.

Також цей оптичний сенсор можна застосовувати для детектування предметів, що відбивають ультрафіолетове випромінювання, або для вимірювання обертів двигуна.

Для виготовлення корпусу робота необхідні пластикові комплектуючі, показані на рисунку 2.18. Ці комплектуючі включають в себе кілька основних частин, які забезпечують структурну цілісність та функціональність корпусу

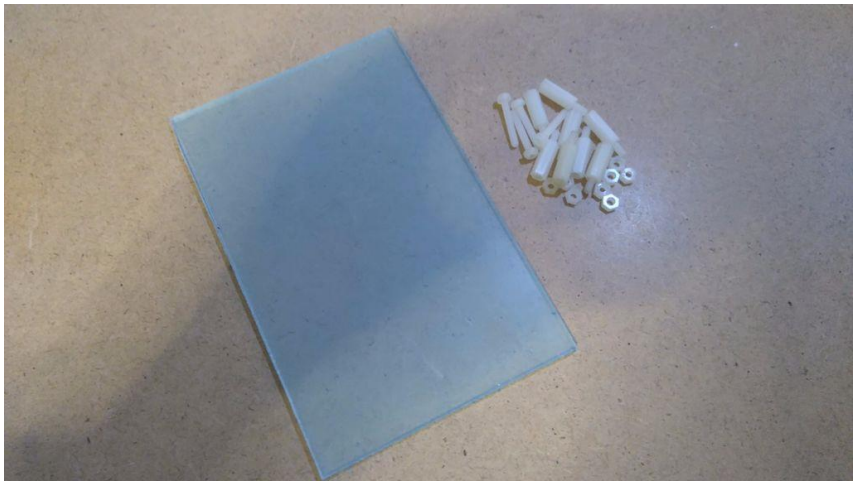


Рисунок 2.18 – Пластикові деталі корпусу робота

Отже, для початку потрібно зробити платформу на якій все буде кріпитись. Вона може бути будь-якої форми. Головне, щоб вона вміщала всі компоненти, які необхідні для функціонування робота, що рухається по заданій траєкторії. На пластику розмічуємо місце кріплення плати Arduino Uno, двох двигунів з колесами а також третього коліщатка для балансування робота (рис 2.19).

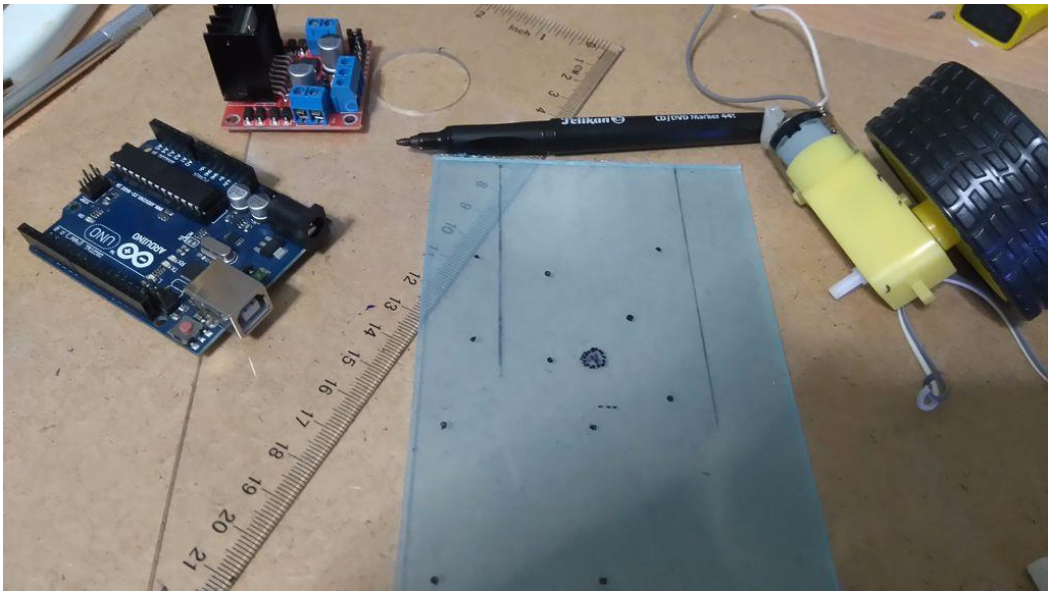


Рисунок 2.19 – Розмічування корпусу робота

Далі, попередньо прикріплюємо двигуни з колесами (рис. 2.20).

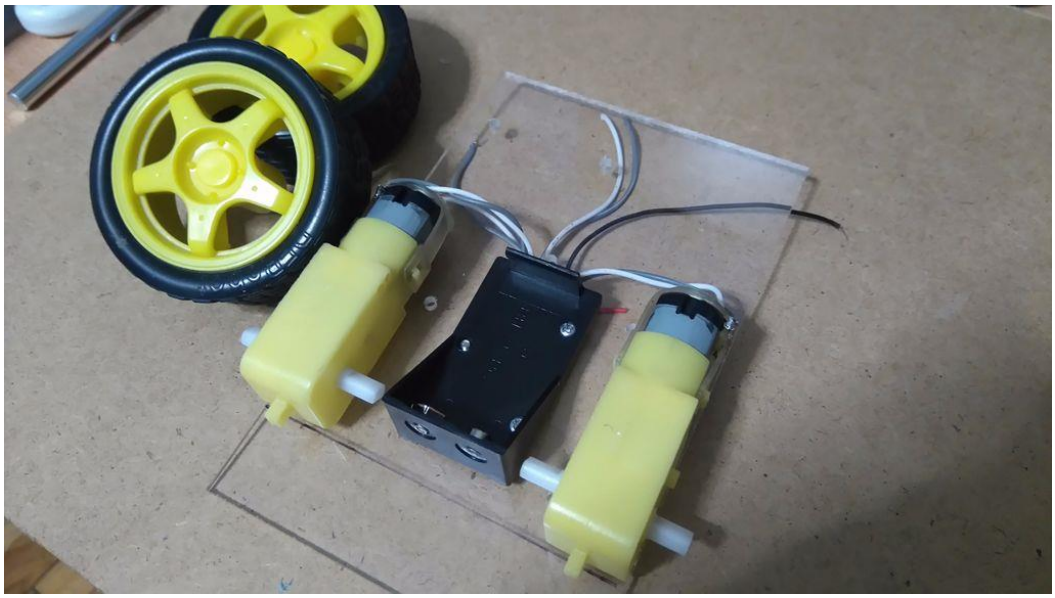


Рисунок 2.20 – Кріплення двигунів на корпус робота

Далі, виконуємо монтаж електронних компонентів нашого робота: контролера двигунів, плати Arduino (рис. 2.21). Електронні компоненти кріпимо на спеціальні пластикові чи металеві стійки, щоб забезпечити електричний захист зворотньої сторони плат контролерів.

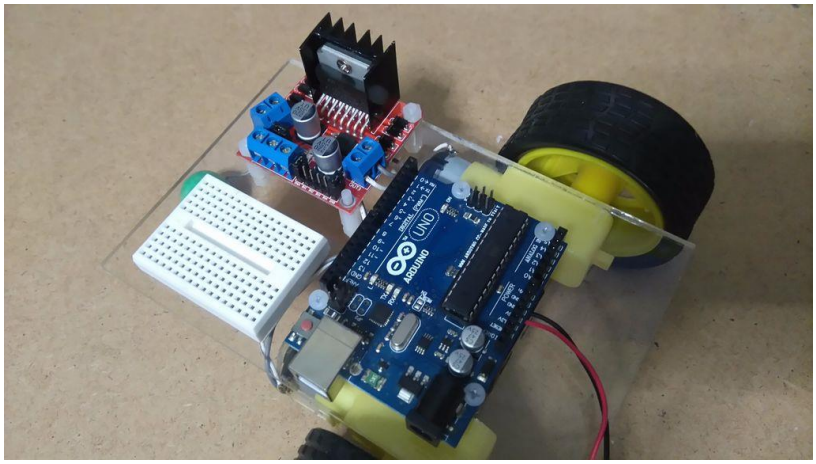


Рисунок 2.21 – Монтаж електронних компонентів на корпус робота

Прикріплюємо знизу по центру третє коліщатко для балансування робота (рис. 2.22). Воно необхідне для виконування маневрів розвертання чи повертання корпусу робота відносно центральної лінії руху.

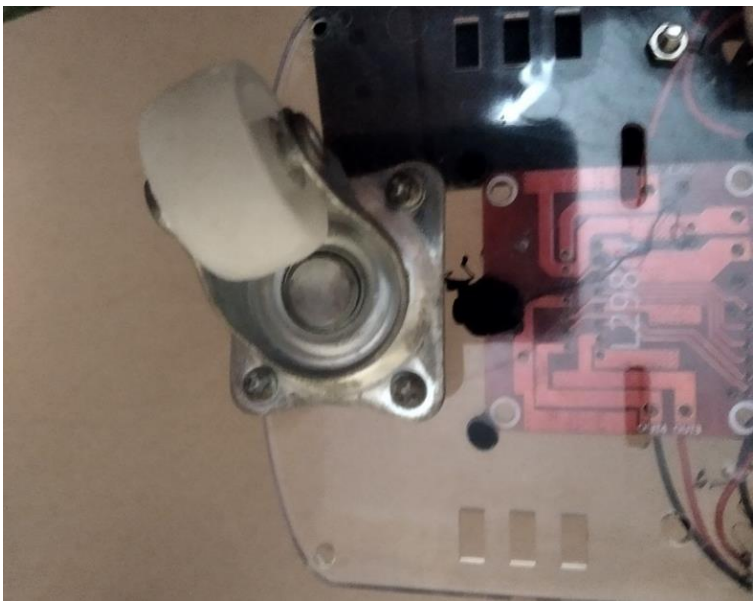


Рисунок 2.22 – Монтаж третього коліщатка для балансування робота

Виконуємо кріплення оптичних давачів лінії в нижній передній частині корпусу нашого робота (рис. 2.23).

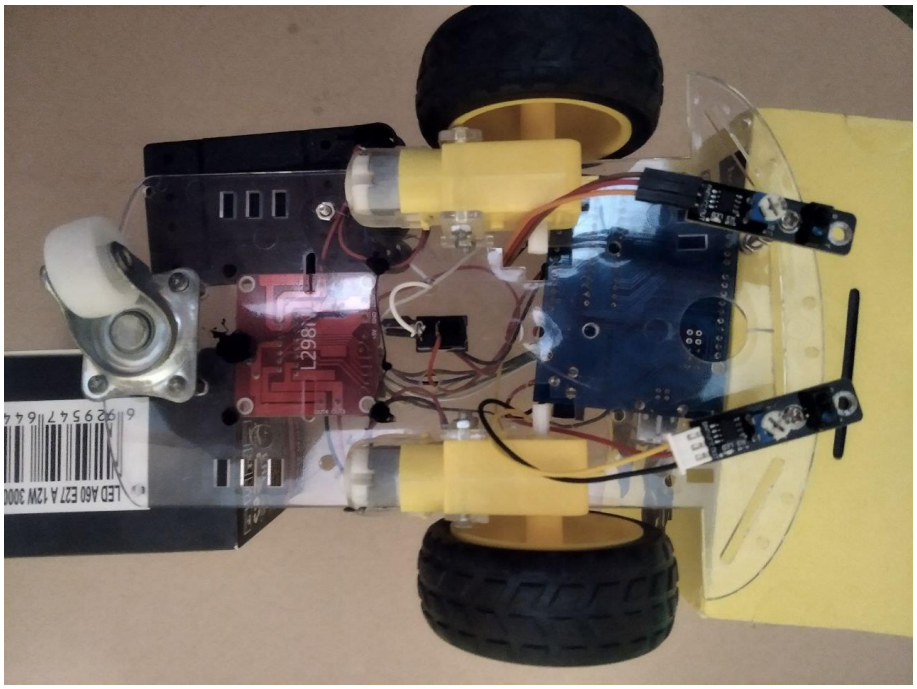


Рисунок 2.23 – Монтаж оптичних давачів лінії на корпус робота

Вибір місця під кріплення оптичних давачів лінії на корпус робота виконується емпіричним шляхом. Від цього буде залежати якість руху робота по заданій лінії. Процес визначення оптимального розташування оптичних давачів є

критично важливим, оскільки правильне їх розміщення забезпечує точне відстеження траєкторії та стабільність руху.

Визначається попередня позиція для давачів, виходячи з теоретичних міркувань щодо оптимального кута огляду та відстані до поверхні. Зазвичай, давачі розміщуються поблизу передньої частини робота для забезпечення швидкої реакції на зміну траєкторії.

2.3 Розробка схеми робота, що рухається по заданій траєкторії

Дана схема демонструє основні компоненти та їх взаємодію, забезпечуючи роботу системи навігації та управління робота.

Основні компоненти функціональної схеми включають:

- контролер Arduino - центральний елемент системи, який обробляє сигнали від датчиків та керує всіма іншими компонентами робота. Контролер виконує обчислення та приймає рішення на основі даних, отриманих від давачів лінії та інших сенсорів;

- оптичні давачі лінії - розміщені на нижній частині корпусу робота, сканують поверхню і визначають положення робота відносно заданої лінії. Вони передають інформацію про наявність або відсутність лінії контролеру, який використовує ці дані для корекції траєкторії руху;

- привідні мотори - забезпечують рух робота. Вони отримують команди від контролера для регулювання швидкості та напрямку руху кожного колеса. Контролер постійно коригує роботу моторів, щоб робот точно слідував за лінією;

- акумулятор - джерело живлення для всіх електричних компонентів робота. Акумулятор забезпечує необхідну енергію для роботи контролера, давачів, моторів та інших компонентів.

Протягом руху контролер також відстежує стан акумулятора та інших систем, щоб забезпечити безперервну роботу робота. У разі виявлення проблем, робот може автоматично зупинитися або сповістити користувача про необхідність втручання.

Ця функціональна схема забезпечує ефективну роботу робота, дозволяючи йому точно слідувати за заданою лінією та виконувати поставлені завдання. Емпіричний вибір місця кріплення оптичних давачів, згаданий раніше, є ключовим для забезпечення високої точності та стабільності роботи системи.

Функціональна схема робота, що рухається по заданій траєкторії наведена на рисунку 2.24.

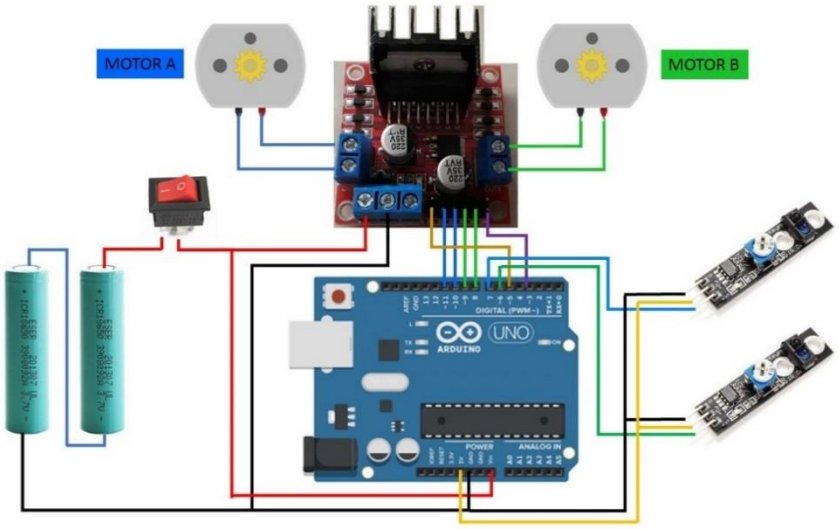


Рисунок 2.24 – Функціональна схема робота, що рухається по заданій траєкторії

Отже після зборки механічної частини робота та приєднання до неї електричних компонентів маємо пристрій, як показано на рисунках 2.25-2.26.

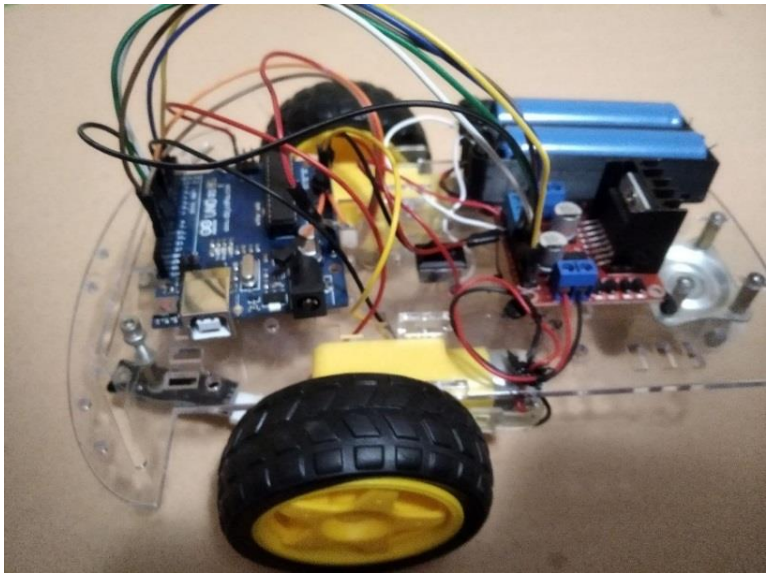


Рисунок 2.25 – Робот, що рухається за заданою траєкторією

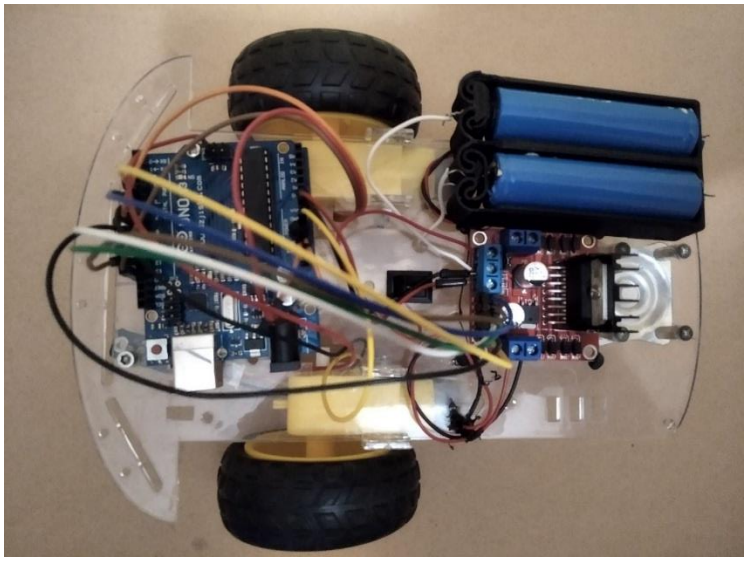


Рисунок 2.26 – Робот, що рухається за заданою траєкторією

2.4 Розробка алгоритму управління роботом, що рухається по заданій траєнторії

Узагальнену схему алгоритму управління роботом, що рухається по заданій траєкторії представлено на рисунку 2.27. Дана схема представляє опис узагальнених кроків, які виконує робот при навігації по заданій траекторії руху. Робот переміщується по лінії, представленій певною темною смужкою на світлому тлі. Щоб збільшити контраст кольору, краще вибрати колір білого фону та чорний колір дороги.

Для контролю руху мобільного робота використовуються кольорові датчики (вибір конкретного продажу датчиків наведено нижче), які дають інший результат залежно від місця розташування датчиків - над темною або легкою областю. У випадку використання цифрових датчиків - два стани відповідають стану логічного 0 та логічного 1, у випадку використання аналогових датчиків - необхідно визначити відхилення поточних показань щодо вибраного значення підтримки.

Кількість датчиків може змінюватись, отже, алгоритм руху зміниться. У цьому завданні будуть використані два кольорові датчики, розташовані на відстані p один від одного. Траєкторія – це лінія певної товщини h, де $h<p$, $p-h > 0$. Таке

співвідношення дозволяє контролювати найменше відхилення горизонтального направляючого робота від заданого шляху.

Для переміщення робота потрібна певна колісна платформа. Існують різні варіанти: двоколесні платформи з підтримкою, чотири-колесні платформи, що характеризуються різними масою та мобільністю. У контексті цього завдання достатньо використовувати двоколісну платформу з підтримкою у вигляді кульки, щоб забезпечити стабільність структури.

Таким чином, загальний алгоритм виглядає так: робот рухається по траєкторії, представленій чорною лінією на білому тлі, так що датчики в нормальному стані знаходяться з боків зображеної лінії. У випадку будь-якого переміщення роботу відносно основного шляху показання датчиків змінюються: значення темної поверхні відрізняються від тих, які були отримані над яскравим (білим) простором.

Як інтелектуальна система, яка утворює команди управління для керування роботом на основі аналізу даних, отриманих із зовнішніх пристроїв (датчиків), використовується мікроконтролер. Мікроконтролер під час руху постійно опитує датчики. Як тільки покази датчиків відхилились на певну величину від початкового стану, мікроконтролер після зчитування нових значень, дає керуючі сигнали на двигуни таким чином, щоб робот стабілізується відносно зазначеного шляху.

На основі вибраного типу колісної платформи необхідно:

- для зміщення робота праворуч: зменшити швидкість обертання правого колеса, залишаючи швидкість лівого без змін;

- для зміщення робота ліворуч: зменшити швидкість обертання лівого колеса, залишаючи швидкість правого без змін;

- для обертання робота на місці, необхідно повністю зупинити один з двигунів.

Коли пристрій рухається по лінії без перешкод, така ситуація неможлива, що обидва датчики одночасно будуть над чорною лінією, оскільки при найменшому відхиленні робот буде стабілізований, а товщина лінії розроблена так, щоб під час нормального руху датчики розташовані суворо з по бокам лінії.

Зупинка робота реалізується за допомогою лінійної перпендикулярної лінії основного шляху. Таким чином, якщо обидва датчики одночасно дають значення,

що відповідають чорній поверхні, контролер дає сигнали зупинки обертання двигунів, робот зупиняється.

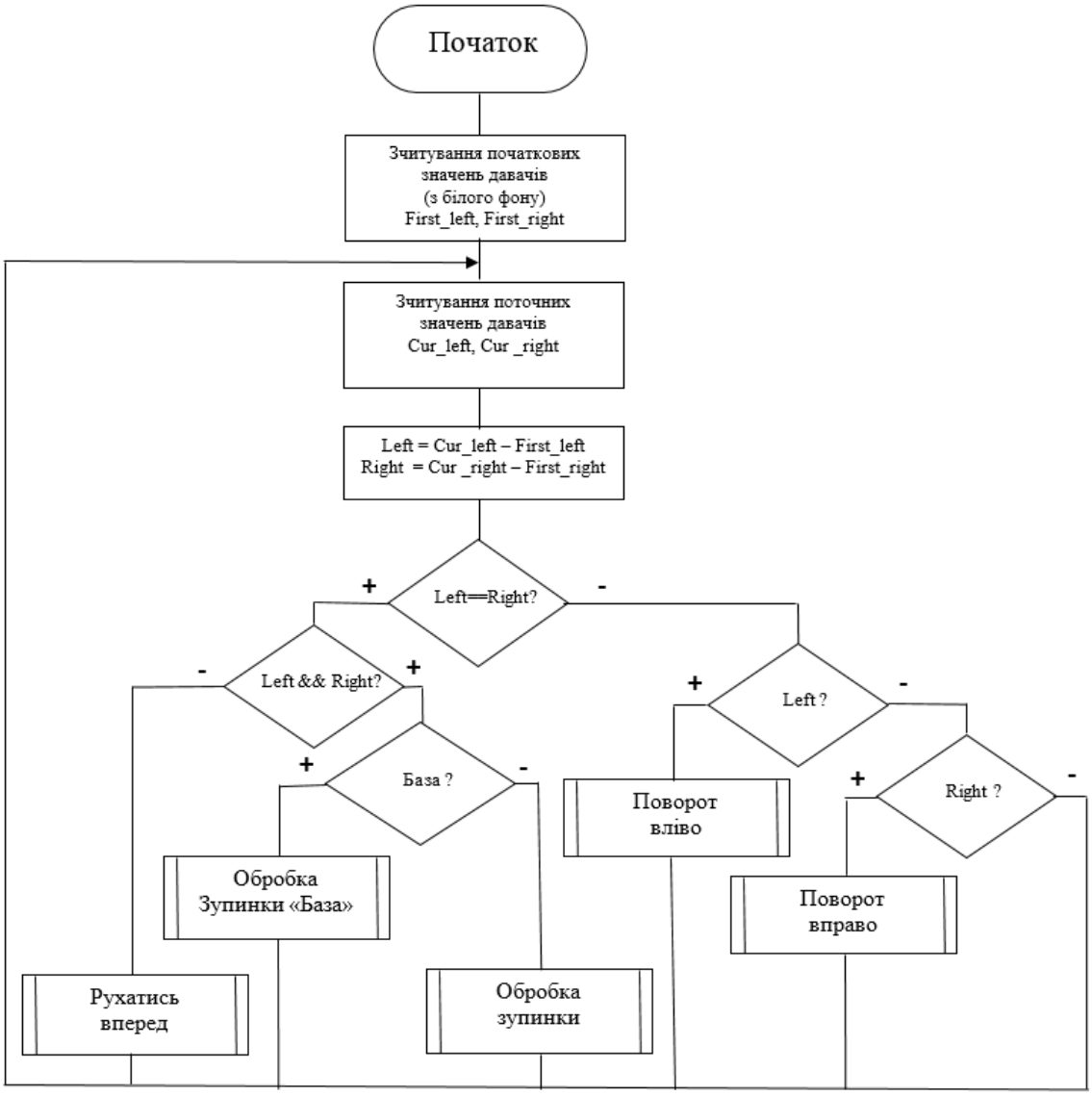


Рисунок 2.27 – Узагальнена схема алгоритму управління роботом, що рухається по заданій траєкторії

2.5 Висновок до розділу 2

В даному розділі розроблено систему управління рухомим роботом, що рухається по заданій траєкторії. Обгрунтовано вимоги до обладнання та вибору елементної бази для створюваного пристрою. Розроблено алгоритм управління робототехнічним пристроєм при подоланні перешкод.

3 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ РОБОТА, ЩО РУХАЄТЬСЯ ПО ЗАДАНІЙ ТРАЄКТОРІЇ

3.1 Обгрунтування вибору середовища розробки

Для апаратної реалізації робота було обрано контролер Arduino Nano. А отже і програмування даного контролера ми будемо виконувати за допомогою спеціалізованого середовища Arduino IDE.

Перш ніж почати писати програму та програмувати контролер, потрібно виконати деяку попередню підготовку. На персональному комп'ютері потрібно установити драйвер для віртуального COM-порту (послідовний інтерфейс) і середовище програмування / розробки для Arduino.

Спочатку потрібно встановити драйвер для мікросхеми перетворювача USB–UART. Якщо використовується оригінальна плата Arduino, або високоякісний клон – необхідно встановити драйвер мікросхеми FT232RL, хоча дану операцію можна і не проводити, оскільки цей драйвер вже входить до комплекту інсталятора середовища розробки Arduino. У випадку використання сумісних плат, необхідно встановити драйвер перетворювача мікросхеми, що встановлена в ній (здебільшого використовуються мікросхеми CH340, CP2102 або PL2303). Мікроконтролер перетворювача інтерфейсів запрограмований таким чином, що в менеджері пристроїв плата позначена як віртуальний COM-порт (віртуальний послідовний інтерфейс). При підключенні плати комп'ютер отримує додатковий COM-інтерфейс. ОС Windows не робить різниці між фізичним та віртуальним COM-інтерфейсом. Новий інтерфейс отримує наступне вільне ім'я, наприклад, COM7.

Тепер встановимо середовище програмування Arduino, яке базується на програмі Processing. Processing – це просте інтегроване середовище розробки, розроблене спеціально для користувача, який не глибоко володіє програмуванням, але, тим не менше, потребує написання власної програми. Останню стабільну версію середовища можна завантажити з офіційного сайту проєкту https://www.arduino.cc/ у вигляді встановлювача, або ж архіву з розпакованою версією, яка не потребує

встановлення (портативна версія). Важливо регулярно виконувати оновлення програмного забезпечення, оскільки в середовище постійно додаються нові бібліотеки, драйвери а також виправляються наявні помилки.

Інтегрована середовище розробки (IDE – Itegrated Development Environment) Arduino – це кросплатформний додаток на Java, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату.

Середовище розробки спроєктоване для програмування новачками, які не знайомі близько з розробкою програмного забезпечення. Мова програмування аналогічна використовуваному в проєкті Wiring. Строго кажучи, це C/C ++, доповнена деякими бібліотеками та з певними обмеженнями (наприклад недоступні можливості багатопоточного програмування та відсутня об'єктна модель C++). Програми обробляються з допомогою препроцесора, а потім компілюються за допомогою вільно розповсюджуваного компілятора AVR-GCC.

3.2 Установка середовища Arduino IDE

Після встановлення або копіювання з архіву програми Arduino можна запустити файл Arduino.exe. Для зручності виклику програми можна створити ярлик на робочому столі. В Arduino-IDE знаходяться різні інструменти і настройки , які полегшують спілкування з програмою Arduino (рис. 3.1).

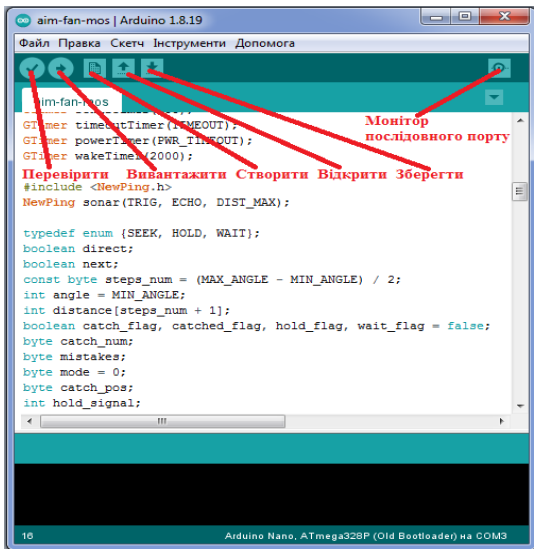


Рисунок 3.1 - Середовище розробки Arduino IDE

Розглянемо докладніше середовище розробки Arduino IDE. З меню можна викликати всі функції Arduino. Під головним меню знаходиться панель інструментів, де розташовані такі команди:

- перевірка – виконується перевірка на наявність помилок в програмному коді;
- завантаження – формування файлу, який буде переданий в плату мікроконтролера та його завантаження;
- новий – створення нового файлу Arduino;
- відкрити – відкриття тексту програми;
- зберегти – збереження тексту програми;
- монітор COM порту – відкриття вбудованого командного терміналу.

До початку роботи необхідно задати вихідні установки. Для цього потрібно вибрати потрібну модель плати Arduino і використовуваний інтерфейс. У даному випадку вказана плата Arduino Nano. Якщо потрібна інша, то слід вибрати відповідну плату зі списку (рис. 3.2).

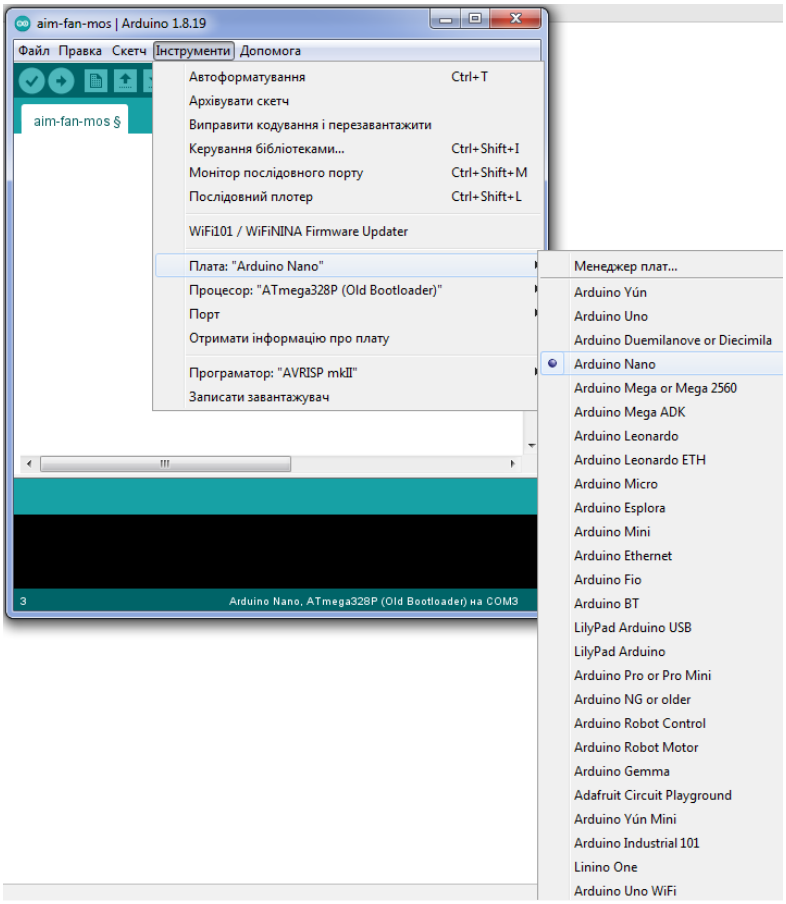


Рисунок 3.2 - Вибір плати мікроконтролера Arduino

3.3 Вибір мови програмування

Програму для функціонування робота, що рухається за заданою траєкторією написано на мові Wiring, яка дуже схож з мовою C++.

C++ було розроблено як універсальну мову зі статичними типами даних, ефективністю та переносимістю мови C.

C++ розроблена так, щоб безпосередньо та всебічно підтримувати безліч стилів програмування (процедурне програмування, абстракцію даних, об'єктно-орієнтоване програмування та узагальнене програмування).

Вона розроблена так, щоб давати програмісту свободу вибору, навіть якщо це дає можливість вибирати неправильно.

C розроблена так, щоб максимально зберегти сумісність із C, тим самим уможливлюючи легкий перехід від програмування на C.

Уникає таких особливостей, які залежать від платформи чи не є універсальними.

Не накладає жодного надлишкового навантаження на програму, яка не використовує жодних можливостей.

Розроблено так, щоб не вимагати надто ускладненого середовища програмування. C++ вважається однією з дуже швидких мов програмування.

Переваги C++ над C:

- велика безпека;
- можливість писати узагальнений код за допомогою шаблонів;
- можливість використовувати об'єктно-орієнтований підхід;
- управління ресурсами за допомогою RAII;
- спрощення коду за рахунок перевантаження функцій та операторів;
- простіша обробка помилок за рахунок винятків.

Недоліки:

- довга компіляція;
- більший обсяг згенерованого машинного коду.

3.4 Встановлення драйвера CH340 Arduino Nano

Для початку - мікросхема CH340G використовується для передачі даних USB -> UART та в повній функціональності замінила мікросхему ATmega16U2 зберігаючи надійність та швидкість при передачі даних.

На платах перетворювач CH340 для Arduino має такий вигляд, як на рис. 3.3.

Рисунок 3.3 – Мікросхема перетворювача CH340 для Arduino

Для інсталювання драйвера операційної системі Windows 10, Windows 8, Windows 7, треба перейти за посиланням - "Завантажити драйвер V3.5 03-2019". --- ZIP-архів.

Після того як закінчилось завантаження файлу, необхідного його відкрити та натиснути на кнопку "INSTALL (рис. 3.4)

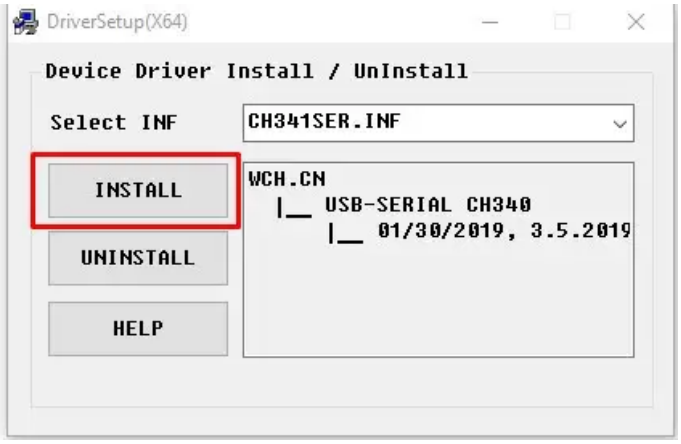


Рисунок 3.4 – Інсталювання дрейвера перетворювача CH340 для Arduino

Треба зачекати кілька секунд і драйвер буде встановлений, про що свідчить дане сповіщення (рис. 3.5)

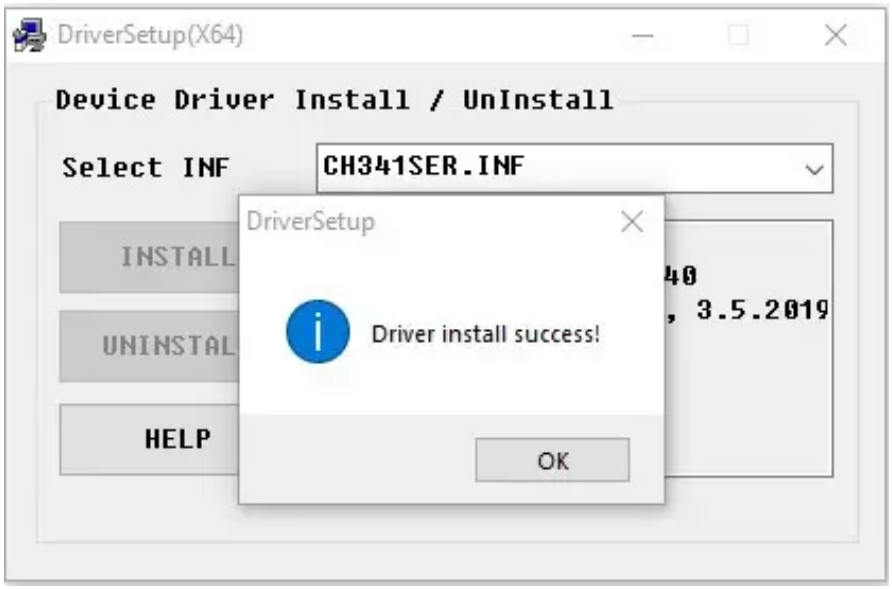


Рисунок 3.5 – Успішне встановлення дрейвера перетворювача CH340 для Arduino

Після підключення плати драйвер зробить емуляцію COM порта та починає передачу даних між комп'ютером та платою Arduino (рис. 3.6).

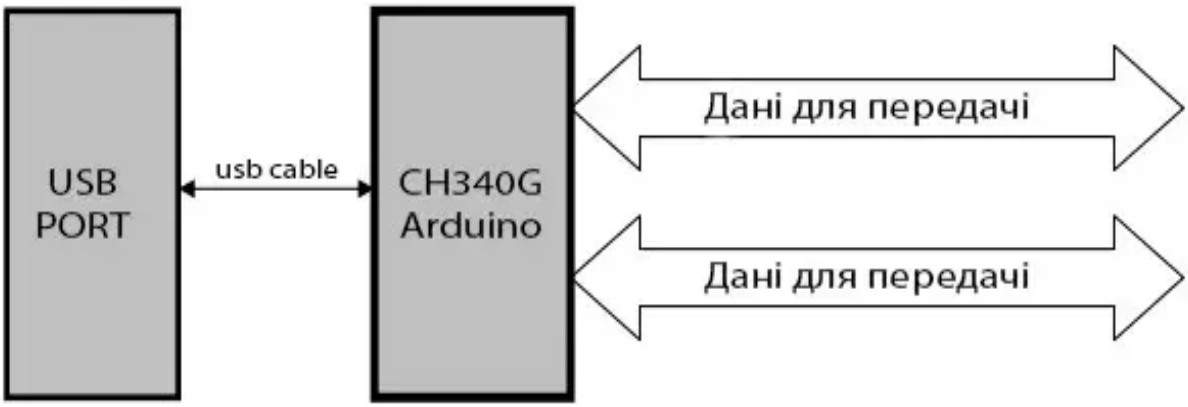


Рисунок 3.6 – Схема роботи драйвера перетворювача CH340 для Arduino

Який саме номер займає встановлений COM-порт, відображається в Диспетчері пристроїв (рис. 3.7).

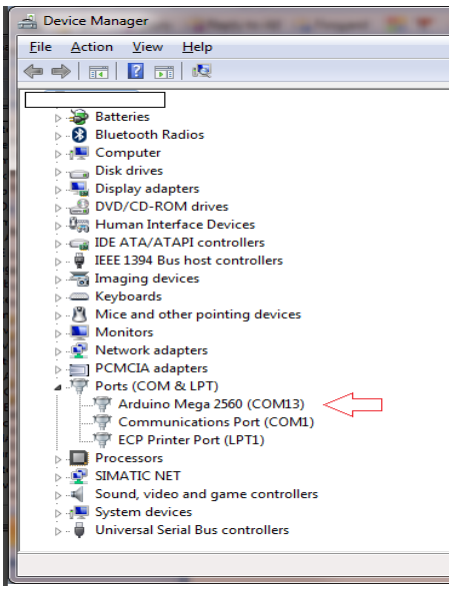


Рисунок 3.7 – Відображення плати Arduino в Диспетчері пристроїв

Отже, даний драйвер та бібліотека CH340 інсталювалися вірно і можливо починати програмувати плати Arduino. Але після підключення плати та запуску середовища програмування Arduino IDE необхідно змінити COM порт (рис. 3.8).

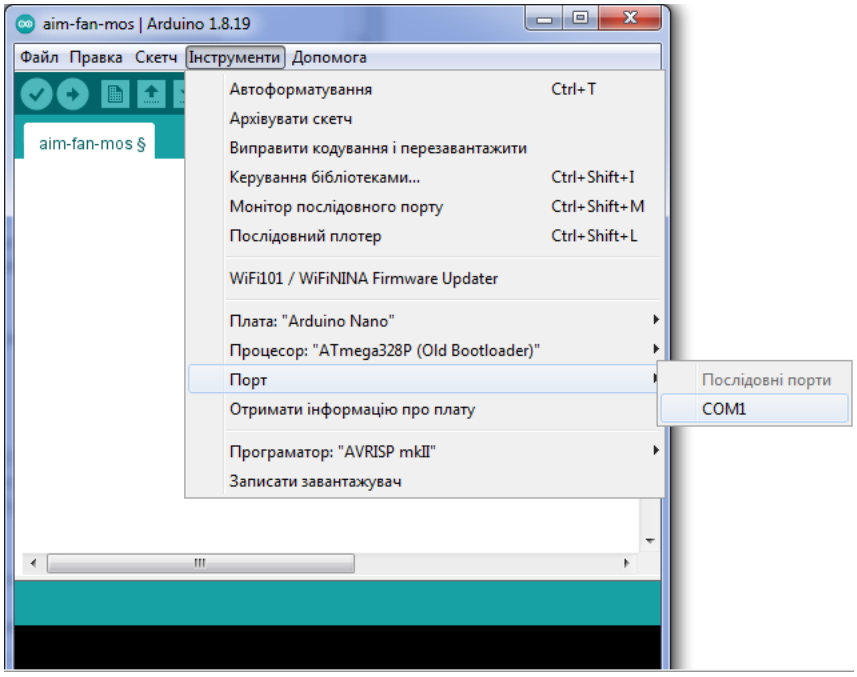


Рисунок 3.8 – Налаштування COM порта в середовищі Arduino IDE

3.5 Програмування контролера Arduino

Розглянемо склад основних компонентів редактора Arduino IDE

1. Головне меню.

Меню редактора включає в себе наступні основні елементи: файл, правка, скетч, сервіс і довідка. Розглянемо докладніше кожен з них (рис. 3.9).

У пункті Файл можна знайти команди, що відповідають за створення нової програми, читання старої, збереження її змін, а також команди для завантаження програми на мікроконтролер.

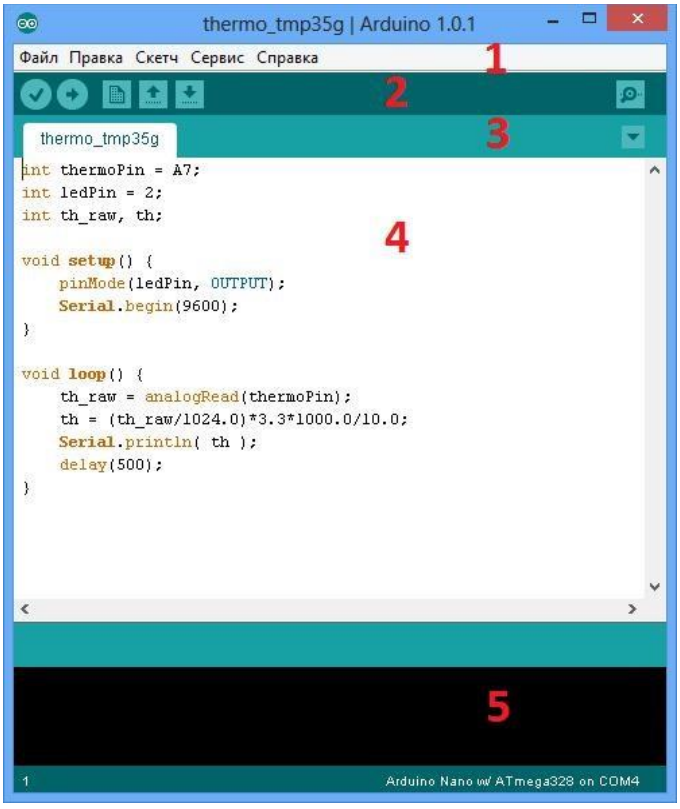


Рисунок 3.9 – Зовнішній вигляд редактора Arduino IDE

Створити - створити нову програму (скетч).

Відкрити - відкрити існуючу програму.

Папка зі скетчами - відкрити програму із заданої папки.

Приклади - відкрити приклад програми; Закрити - закрити поточне вікно.

Зберегти - зберегти зміни в раніше збережену програму.

Зберегти як - зберегти нову програму, із зазначенням імені.

Завантажити - завантажити програму в Arduino.

Завантажити за допомогою програматора - завантажити програму за допомогою програматора.

Налаштування друку – налаштування принтера.

Друк - вивід на друк коду програми.

Налаштування – налаштування редактора.

Вихід - вихід з Arduino IDE.

Пункт меню Правка містить команди, пов'язані з редагуванням тексту програми, включаючи копіювання, вставку, настройку відступів і пошук за ключовим словом.

У розділі Скетч розміщуються команди для контролю за процесом компіляції програми.

Перевірити / Компілювати - компілювати програму.

Показати папку скетчів - відкрити системну папку з програмами.

Додати файл - додати до проекту файл з даними або програмою.

Імпортувати бібліотеку - підключити до програми бібліотеку зі списку встановлених.

Пункт меню Сервіс включає в себе допоміжні функції для роботи з самим мікроконтролером.

Автоформатування - автоматична розстановка відступів, переносів рядків і т.п .

Архівувати скетч - архівація папки з програмою, і збереження архіву в вказане місце.

Виправити - кодування і перезавантажити.

Монітор порту - відкрити вікно для обміну даними з мікроконтролером.

Плата - вибір поточної плати (в даному випадку Arduino Uno).

Послідовний порт - вибір порту, до якого підключений.

Нарешті, меню Довідка містить докладний опис всіх функцій самого редактора Arduino IDE, а також всілякі команди і прийоми роботи з платформою Arduino.

2. Меню іконок:

Перевірити / Компілювати програму.

Завантажити програму в Arduino.

Створити нову програму.

Відкрити існуючу програму.

Зберегти програму.

Монітор послідовного порту.

3. Вкладки

Кожна програма для Arduino може складатися з декількох файлів. Для перемикання між цими файлами служить система вкладок в редакторі. Там же, можна створити нову вкладку, і асоціювати з нею файл в папці з проектом.

4. Вікно програми

Безпосередньо, текст програми створюється і редагується в головному вікні редактора. По суті, вікно редактора являє собою типовий текстовий редактор, з підсвічуванням конструкцій коду.

5. Вікно повідомлень

У самому низу редактора Arduino IDE є невелике вікно, що служить для виведення повідомлень про проблеми, які виникають в процесі компіляції програми, або під час завантаження програми в мікроконтролер.

Програмування Arduino здійснюється на мові C ++. Нижче приведені основні конструкції і умовності мови, необхідні для успішного проходження даного курсу.

1. Кожен вираз закінчується символом; крапка з комою. наприклад:

a = b + c;

Тіло функцій і складових операторів (if, else, for, while) відокремлюється фігурними дужками(Аналогічно BeginEnd в мові Pascal). наприклад:

```
if ( a>0 )
{
   b = a+1;
}
```

Рядки відокремлюються звичайними подвійними лапками ".

Приклад: println ("some text");

Символи відокремлюються одинарними лапками: symbol = 'a';

Підключення бібліотек здійснюється за допомогою конструкції:

#include <math.h>

Комметарії в програмі починаються з символів // два прямих слеша. Приклад:

// це моя програма

Оголошення змінної в мові с ++ здійснюється за допомогою конструкції виду:

тип_перемінної ім'я_змінної;

Наприклад:

int x, y; // оголошені дві змінні x і y, які мають цілий тип

Цілі числа:

byte від 0 до 255;

int від 32 768 до 32 767;

word від 0 до 65535;

long від 2 147 483 648 до 2 147 483 647.

Дробові числа:

float від 3.4028235E + 38 до 3.4028235E + 38;

double еквівалетно float в поточній версії Arduino.

Масиви

Масиви в C ++ задаються конструкцією типу:

тип_елемента імя_массіва [розмір];

Наприклад:

int numbers [10]; // задає масив з десяти цілих чисел

Рядки і символи:

char символ;

Рядки в C ++ є масиви з елементами типу char.

Наприклад:

char my_str [10]; // рядок з десяти символів

Інші типи:

void порожній тип;

boolean false або true (брехня або істина).

Оператори порівняння:

== рівність

! = Нерівність

< менше

<= Менше, або дорівнює

> більше

> = Більше, або дорівнює

Умови

```
if (a> 0) { ...
   команди, що виконуються в разі істинності умови
} else { ...
   команди, що виконуються в іншому випадку
}
```

Цикли

```
for (k = 0; k <3; k = k + 1) { ...
   команди, що виконуються на кожному кроці циклу
}
```

У дужках послідовно вказується:

- початкове значення ітератора k = 0;

- умова продовження циклу k <3 (поки ітератор менше трьох);

- дію над ітератором під час кожного кроку k = k + 1 (збільшуємо на одиницю на кожному кроці).

Функції;

- тип_функції імя_функциі (аргументи)

{ команди, що виконуються в рамках функції return результат_функції;

}

тип_функції - тип значення.

Наприклад, стандартна функція sin має тип значення, що повертається float.

- ім’я_функції - будь-який рядок, що починається з букви, і містить тільки літери і символи підкреслення. аргументи - перелік аргументів, які функція використовує для своїх дій.

- результат_функціі - пременися або число, що визначає значення, що повертається функції.

Нижче наведено приклад функції, що підсумовує два цілих числа.

```
int sum (int a, int b) { int result; result = a + b; return result; }
```

Мінімальна програма, яка може запустити на Arduino складається всього з двох функцій:

```
void setup () {
}
void loop () { }
```

Перша функція setup викликається тільки один раз, після перезапуску Arduino. Друга loop, викликається нескінченне число разів під час роботи Arduino.

Для зв'язку з платою Arduino можна використовувати спеціальну програму моніторингу послідовного порту (Serial Monitor), вбудовану в програмне забезпечення Arduino . Монітор послідовної шини відображає дані, що посилаються в плату Arduino через віртуальний послідовний порт за допомогою USB. Для відправки даних вибирається швидкість передачі зі списку, відповідна значенню Serial.begin в скетчі. Потім необхідно ввести текст і натиснути кнопку Send або Enter.

Плата Arduino Nano має один послідовний порт (також відомий як UART або USART): Serial, що використовується для зв'язку з комп'ютером через USB. Він пов'язаний з цифровими виводами 0 (RX) і 1 (TX). Таким чином, під час роботи послідовного порту, виводи 0 і 1 не можуть використовуватися в якості цифрових входів або виходів.

Для забезпечення зв'язку плати Arduino з комп'ютером або іншими пристроями використовується клас Serial. Клас - це абстрактний тип даних. За

допомогою класу описується деяка сутність (її характеристики і можливі дії). Наприклад, клас може описувати змінні, параметри і функції послідовного порту. Клас Serial містить близько 20 функцій. Розглянемо деякі.

Функції для роботи з послідовним портом плати Arduino:

1. if (Serial)

Параметри – ні.

Значення, що повертаються – boolean: повертає true, якщо вказаний послідовний порт готовий до роботи.

Опис: дозволяє перевірити готовність певного послідовного порту.

2. Serial.begin(speed) Serial.begin(speed, config)

Параметри:

speed: швидкість в бітах на секунду (бодах) - long

config: задає кількість біт даних, перевірку парності и стопові біти:

SERIAL_5N1

SERIAL_6N1

SERIAL_7N1

SERIAL_8N1 (за замовченням)

SERIAL_5N2

SERIAL_8E2

SERIAL_7O2

Опис: задає швидкість передачі даних по послідовному інтерфейсу в бітах в секунду (бодах). Для взаємодії з комп'ютером слід використовувати одну з попередньо встановлених швидкостей обміну: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200.

Другий аргумент цієї функції необов'язковий. Він дозволяє налаштувати кількість біт даних, перевірку парності і степових біти. За замовчуванням, посилка складається з 8 біт даних, без перевірки парності, з одним стоповим бітом.

3. Serial.end ()

Параметри – ні.

Значення, що повертаються – немає.

Опис: функція розриває послідовний зв'язок, після чого виводи RX і TX знову можна використовувати як виводи загального призначення. Для відновлення послідовного з'єднання необхідно використовувати функцію

Serial.begin ().

4. Serial.available()

Параметри – ні.

Значення, що повертаються: кількість байт, доступних для зчитування.

Опис: повертає кількість байт (символів) доступних для зчитування з буфера послідовного порту. Під символами розуміються дані, які вже прийняті і зберігаються в послідовному приймальному буфері (який може зберігати максимум 64 байта).

5. Serial.read()

Параметри – ні.

Значення, що повертаються: перший байт прийнятих даних (або -1, якщо таких нема) – int.

Опис: зчитує дані, що надходять по послідовному інтерфейсу.

6. Serial.print(val)

Serial.print(val, format)

Параметри: val: значення, яке необхідно вивести - будь-який тип даних; format: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою точкою).

Значення, що повертаються: кількість виведених байт. Зчитування цього значення не є обов'язковим.

Опис: функція виводить через послідовний порт заданий ASCII текст у вигляді, зрозумілому для людини. Ця команда може мати кілька різних форм.

При виведенні числа кожній його цифрі відповідає один ASCII-символ. Дробові числа теж виводяться у вигляді ASCII-цифр, при цьому після коми за замовчуванням залишається два десяткових знака. Байти виводяться у вигляді окремих символів, а символи і рядки виводяться без змін – «як є».

Приклад:

Serial.print(78) - виведе "78"

Serial.print(1.23456) - виведе "1.23"

Serial.print('N') - виведе "N"

Serial.print("Hello world.") - виведе "Hello world."

Другий параметр необов'язковий. Він задає формат виведення; цей параметр може набувати таких значень: BIN (двійкова система з основою 2), OCT (восьмерична система з основою 8), DEC (десяткова система з основою 10), HEX (шістнадцятирічна система з основою 16). Для числа з плаваючою точкою цей параметр визначає кількість десяткових знаків після коми.

Приклад:

Serial.print (78, BIN) - виведе "1001110"

Serial.print (78, OCT) - виведе "116"

Serial.print (78, DEC) - виведе "78" Serial.print (78, HEX) - виведе "4E"

Serial.println (1.23456, 0) - виведе "1"

Serial.println (1.23456, 2) - виведе "1.23"

Serial.println (1.23456, 4) - виведе "1.2346"

Отже, далі завантажуємо та розпаковуємо середовище Arduino IDE.

Підключаємо мікрокомп'ютер Arduino до роз’єму USB. Драйвер (послідовний інтерфейс CH340) встановлюється автоматично. Якщо цього не сталося, то в папці Arduino IDE є тека Drivers з усім необхідним.

Запускаємо Arduino IDE та відкриваємо файл з прошивкою (рис. 3.10).

Завантажуємо програму в Arduino. Програма сама проінформує, коли завантаження буде завершено (тривалість - кілька секунд).

Тепер потрібно відключити контролер Arduino Nano від шини USB.

Отже, з урахування всього сказаного, розробимо алгоритм руху робота за заданою траєкторією.

Припустимо, у нас є біле поле, і на ньому чорним кольором намальовано трек для нашого робота. Використовувані давачі лінії видають логічний нуль, коли бачать чорне поле і одиницю, коли бачать біле.

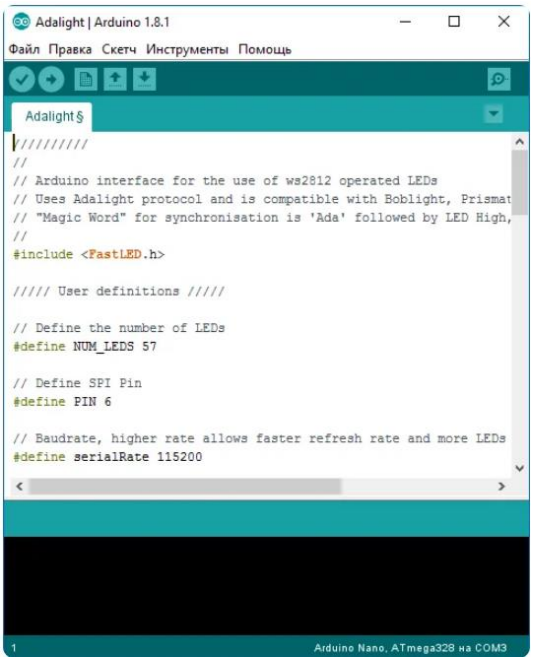


Рисунок 3.10 – Скетч для управління роботом

На прямій робот повинен пропускати трек між сенсорами, тобто обидва сенсори повинні показувати одиниці.

При повороті траєкторії праворуч, правий сенсор наїжджає на трек і починає показувати логічний нуль. При повороті ліворуч нуль показує лівий сенсор. Таким чином отримуємо просту систему з трьома станами:

- State_forward – їдемо вперед;

- State_right – повертаємо вправо;

- State_left – повертаємо вліво.

На вхід системи надходить інформація із сенсорів. І ми отримуємо ось таку логіку переходів (табл. 3.1).

Таблиця 3.1 – Логіка корекції руху робота

Лівий	Правий	Цільовий стан
0	0	State_forward
0	1	State_right
1	0	State_left
1	1	State_forward

Програмно це виглядає ось так:

```
#define SPEED_LEFT       6
#define SPEED_RIGHT      5
#define DIR_LEFT         7
#define DIR_RIGHT        4
#define LEFT_SENSOR_PIN  8
#define RIGHT_SENSOR_PIN 9

// Швидкість руху вперед (0-255)
#define SPEED            35

// Коефіцієнт корекції в скільки раз треба загальмувати
// одне з коліс для повороту
#define BRAKE_K          4

#define STATE_FORWARD    0
#define STATE_RIGHT      1
#define STATE_LEFT       2

int state = STATE_FORWARD;

void runForward()
{
   state = STATE_FORWARD;

   // Для регулювання швидкості `SPEED` може бути від 0 до 255,
   analogWrite(SPEED_LEFT, SPEED);
   analogWrite(SPEED_RIGHT, SPEED);

   // Якщо в DIR_LEFT чи DIR_RIGHT пишемо HIGH,  то двигун буде рухати відповідне колесо
```

```
    // вперед, якщо LOW - назад.
    digitalWrite(DIR_LEFT, HIGH);
    digitalWrite(DIR_RIGHT, HIGH);
}

void steerRight()
{
    state = STATE_RIGHT;

    // Гальмуємо праве колесо відносно лівого,
    // щоб почати поворот
    analogWrite(SPEED_RIGHT, SPEED / BRAKE_K);
    analogWrite(SPEED_LEFT, SPEED);

    digitalWrite(DIR_LEFT, HIGH);
    digitalWrite(DIR_RIGHT, HIGH);
}

void steerLeft()
{
    state = STATE_LEFT;

    analogWrite(SPEED_LEFT, SPEED / BRAKE_K);
    analogWrite(SPEED_RIGHT, SPEED);

    digitalWrite(DIR_LEFT, HIGH);
    digitalWrite(DIR_RIGHT, HIGH);
}
```

```
void setup()
{
   // Налаштовуємо піни плати 4,5,6,7 на виведення сигналів
   for(int i = 4; i <= 7; i++)
      pinMode(i, OUTPUT);

   // Їдемо вперед
   runForward();
}

void loop()
{
   boolean left = !digitalRead(LEFT_SENSOR_PIN);
   boolean right = !digitalRead(RIGHT_SENSOR_PIN);

   int targetState;
   if (left == right) {
      // под сенсорами все біле чи все чорне
      // їдемо вперед
      targetState = STATE_FORWARD;
   } else if (left) {
      // лівий сенсор в'їхав  в трек
      // повертаємо вліво
      targetState = STATE_LEFT;
   } else {
      targetState = STATE_RIGHT;
   }

   if (state == targetState) {
      return;
```

```
    }

    switch (targetState) {
      case STATE_FORWARD:
        runForward();
        break;

      case STATE_RIGHT:
        steerRight();
        break;

      case STATE_LEFT:
        steerLeft();
        break;
    }

    // не дозволяємо сильно хилятися на прямій
    delay(50);
}
```

Однак якщо виставити швидкість двигунів більше, ми зіткнемося з наступною проблемою: наш робот вилітатиме з треку, не встигаючи відреагувати на поворот. Це пов'язано з тим, що наші двигуни не вміють гальмувати миттєво.

У цьому легко переконатися, поставивши наступний експеримент: із заданою швидкістю робот рухатиметься по поверхні, і в деякий момент буде встановлена нульова швидкість і виміряний гальмівний шлях робота. Нехай робот розганяється по монотонній поверхні та гальмується при фіксуванні імпровізованої стоп-лінії.

Таким чином, починаючи з деякого моменту у нашого робота немає жодної можливості встигнути зреагувати та залишитися на треку.

Що можна зробити? Після того, як давачі вловлюють поворот, можна зупинитися і рухатися назад на деяку відстань, яка залежить від швидкості перед зупинкою.

Для розширення функціональних можливостей рухомих роботів, які використовують лінійну навігацію, було запропоновано впровадження адаптивних алгоритмів керування, що дозволяють здійснювати більш гнучку та точну орієнтацію в просторі. Лінійна навігація, яка базується на слідуванні за визначеним маршрутом (наприклад, лінією на підлозі або магнітною стрічкою), забезпечує просту реалізацію та високу точність руху в контрольованих умовах. Робот постійно відслідковує наявність ліній на землі та відразу ж зупиниться, якщо лінія випадково закінчиться або буде перервана, такий спосіб більш безпечний ніж виконувати пошук самостійно.

Для надсилання команд роботу під час руху по лінії запропоновано використовувати стрічкові маркери, що розташовуються зліва і справа від центру основної лінії.

Маркери – це досить простий та ефективний спосіб для ідентифікації особливих точок вздовж маршруту.Такі маркери наносяться з лівого та правого боку від основної лінії для сигналізування роботу про місце де потрібно зробити зупинку.

Отже, ставимо нашого робота на задану траєкторію, подаємо живлення. Все. Робот готовий до роботи.

3.6 Висновок до розділу 3

В даному розділі обгрунтовано вибір мови програмування та середовища розробки робота, що рухається по заданій траєкторії. Побудована структура робота, що рухається по заданій траєкторії. Здійснено програмну реалізацію розробленого алгоритма керування робота, що рухається по заданій траєкторії. Виготовлено лабораторний макет робота, що рухається по заданій траєкторії.

ВИСНОВКИ

Під час виконання бакалаврської кваліфікаційної роботи було розроблено робота, що рухається по заданій траекторії.

В роботі проаналізовано основні види рухомих мобільних роботів. Наведено класифікацію мобільних роботів за функціональним призначенням. Створено узагальнену схему управління мобільним роботом, що рухається по заданій траєкторії. Розроблено систему управління рухомим роботом, що рухається по заданій траєкторії. Обгрунтовано вимоги до обладнання та вибору елементної бази для створюваного пристрою. Розроблено алгоритм управління робототехнічним пристроєм при подоланні перешкод. Обгрунтовано вибір мови програмування та середовища розробки робота, що рухається по заданій траєкторії. Побудована структура робота, що рухається по заданій траєкторії. Здійснено програмну реалізацію розробленого алгоритма керування робота, що рухається по заданій траекторії. Виготовлено лабораторний макет робота, що рухається по заданій траекторії з урахуванням місць примусової зупинки. Такі місця організовуються за допомогою нанесення по обидві сторони лінійної траекторії маркерів для зупинки робота.

Отже всі поставлені задачі виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Arduino. URL: https://uk.wikipedia.org/wiki/Arduino.
2. Stadler Form Aktiengesellschaft. URL: https://stadlerform.ua/about/
3. Bresser. URL: http://www.bresser-ukraine.com.ua/company/main/
4. Mobile robotics (Robot). Module 1. [Электр. ресурс] / M. Dunn // URL: https://learn.open2study.com/mod/youtube/view.php?id=79916 .
5. Siegward, R Introduction to Autonomous Mobile Robots / R. Siegward, 10. R. Nourbakhsh - Cambridge MA: MIT Press 2004. – 321 p.
6. First. URL: https://vencon.ua/ua/brands/first
7. Uniel. URL: https://www.uniel.com/
8. UML для бізнес-моделювання. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html
9. Arduino. URL: https://vseplus.com/ua/article/cto-takoe-arduino-783
10. Модуль RGB-світлодіода. URL: https://arduino.ua/ru/prod2966-modylrgb-svetodioda-ky-016-dlya-arduino (дата звернення: 29.05.2025).
11. Інфрачервоний датчик руху HC-SR501. URL: https://arduino.ua/ru/prod193-ik-datchik-dvijeniya-dlya-arduino-hc-sr501 (дата звернення: 29.05.2025)
12. Мікроконтролери і мікропроцесорні пристрої. URL: https://elprivod.nmu.org.ua/ua/interesting/what_is_mp_mc_plc.php
13. Arduino інтерфейси. URL: https://geekmatic.in.ua/ua/arduino_lesson_112
14. Arduino Nano. URL: https://uk.wikipedia.org/wiki/Arduino_Nano
15. C++. URL: http://www.znannya.org/?view=Cpp_basics
16. Основні оператори. URL: http://tinker.uamper.com/docs/programuvannya-arduino.html
17. Основи мікропроцесорної техніки. URL: https://ela.kpi.ua/bitstream/123456789/27992/1/OMPT_laboratorni.pdf (дата звернення: 29.05.2025).
18. Платформа Arduino. URL: https://www.duet.edu.ua/uploads/DocS/9st9.pdf
19. IDE (інтегроване середовище розробки). URL: https://apixdrive.com/ua/blog/

useful/ide-i-sdk-strashnye-terminy-prostymi-slovami

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).

ДОДАТОК А (обов'язковий)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка робота, що рухається по заданій траєкторії

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 19,97 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Колодний В.В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Шамрай М.В.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (довідниковий)
Інструкція користувача

1. Нанести розмітку чорного кольору на підлогу, по якій має рухатися робот, що виористовує лінійну наіігацію.

2. Розташувати по обидва боки від лінії руху маркери примусової зупинки рухомого робота.

3. Встановити робота на підлогу таким чином, щоб лінія траекторія знаходилась між оптичними давачами лінії.

4. Увімкнути робот (рис. Б.1).

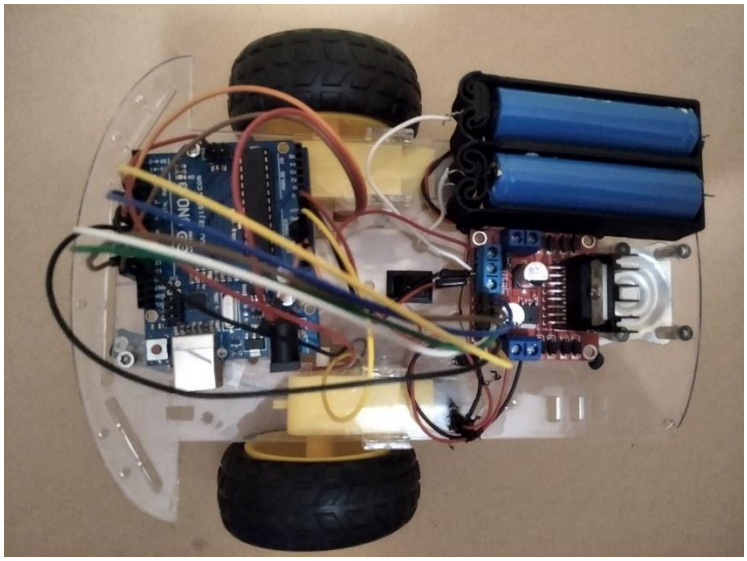


Рисунок Б.1 – Робот, що рухається по заданій траекторії

ДОДАТОК В (обов'язковий)
Лістинг програми

```
#define SPEED_LEFT       6
#define SPEED_RIGHT      5
#define DIR_LEFT         7
#define DIR_RIGHT        4
#define LEFT_SENSOR_PIN  8
#define RIGHT_SENSOR_PIN 9

// Швидкість руху вперед (0-255)
#define SPEED            35

// Коефіцієнт корекції в скільки раз треба загальмувати
// одне з коліс для повороту
#define BRAKE_K          4

#define STATE_FORWARD    0
#define STATE_RIGHT      1
#define STATE_LEFT       2

int state = STATE_FORWARD;

void runForward()
{
   state = STATE_FORWARD;

   // Для регулювання швидкості `SPEED` може бути від 0 до 255,
   analogWrite(SPEED_LEFT, SPEED);
   analogWrite(SPEED_RIGHT, SPEED);

   // Якщо в DIR_LEFT чи DIR_RIGHT пишемо HIGH,  то двигун буде рухати
відповідне колесо
   // вперед, якщо LOW - назад.
   digitalWrite(DIR_LEFT, HIGH);
   digitalWrite(DIR_RIGHT, HIGH);
}
```

```
void steerRight()
{
   state = STATE_RIGHT;

   // Гальмуємо праве колесо відносно лівого,
   // щоб почати поворот
   analogWrite(SPEED_RIGHT, SPEED / BRAKE_K);
   analogWrite(SPEED_LEFT, SPEED);

   digitalWrite(DIR_LEFT, HIGH);
   digitalWrite(DIR_RIGHT, HIGH);
}

void steerLeft()
{
   state = STATE_LEFT;

   analogWrite(SPEED_LEFT, SPEED / BRAKE_K);
   analogWrite(SPEED_RIGHT, SPEED);

   digitalWrite(DIR_LEFT, HIGH);
   digitalWrite(DIR_RIGHT, HIGH);
}



void setup()
{
   // Налаштовуємо піни плати 4,5,6,7 на виведення сигналів
   for(int i = 4; i <= 7; i++)
      pinMode(i, OUTPUT);

   // Їдемо вперед
   runForward();
}

void loop()
```

```
{
   boolean left = !digitalRead(LEFT_SENSOR_PIN);
   boolean right = !digitalRead(RIGHT_SENSOR_PIN);

   int targetState;
   if (left == right) {
      // под сенсорами все біле чи все чорне
      // їдемо вперед
      targetState = STATE_FORWARD;
   } else if (left) {
      // лівий сенсор в'їхав  в трек
      // повертаємо вліво
      targetState = STATE_LEFT;
   } else {
      targetState = STATE_RIGHT;
   }
    if (state == targetState) {
      return;
   }
    switch (targetState) {
      case STATE_FORWARD:
         runForward();
         break;

      case STATE_RIGHT:
         steerRight();
         break;

      case STATE_LEFT:
         steerLeft();
         break;
   }

   // не дозволяємо сильно хилятися на прямій
   delay(50);
}
```

ДОДАТОК Г (обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«РОЗРОБКА РОБОТА, ЩО РУХАЄТЬСЯ ПО ЗАДАНІЙ ТРАЕКТОРІЇ»

Виконав: студент 4 курсу, групи 4КН-21б
спеціальності 122 – Комп’ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Шамрай М.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри КН
Колодний В.В.
(прізвище та ініціали)

Вінниця ВНТУ - 2025 рік

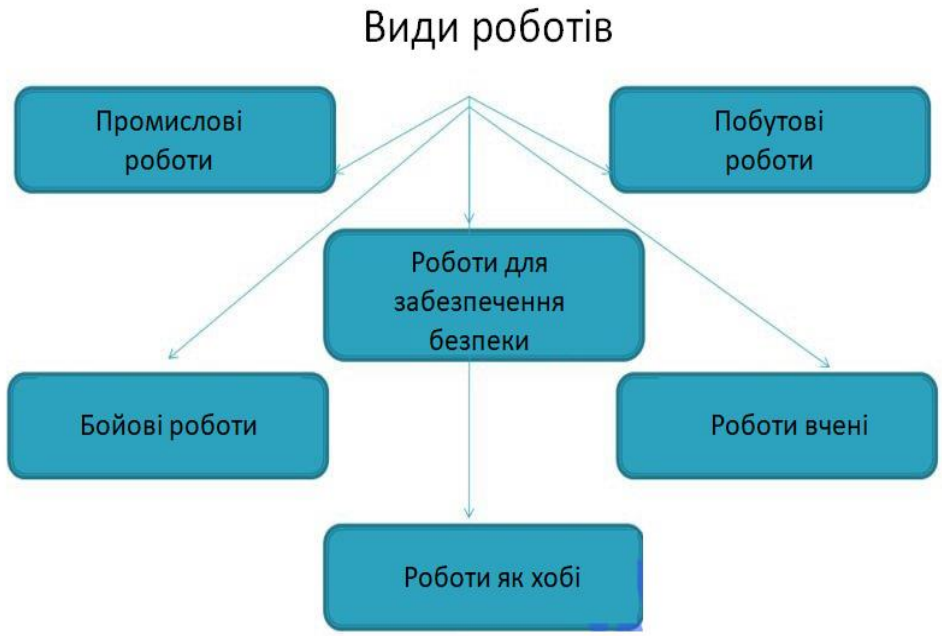


Рисунок Г.1 – Класифікація роботів

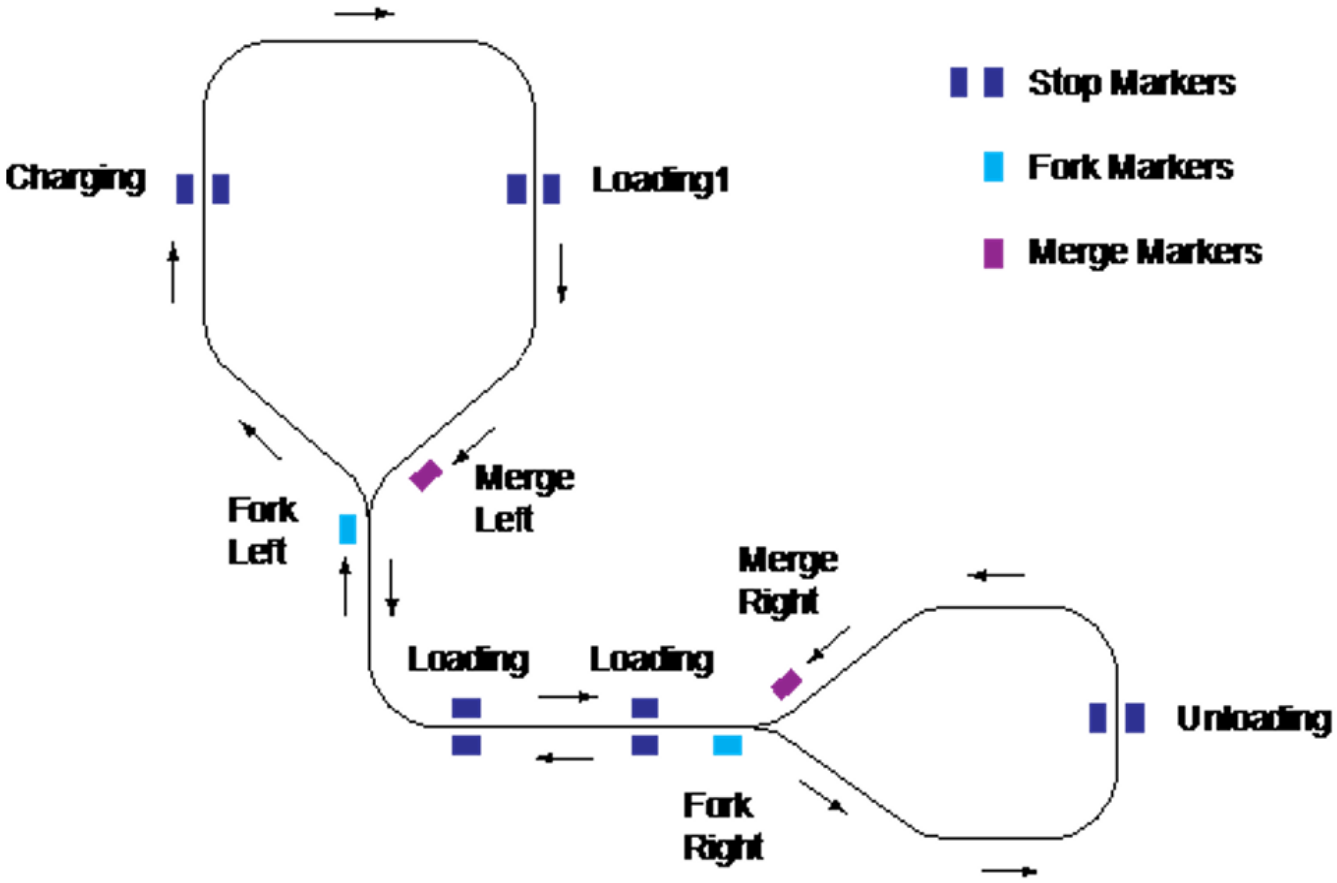


Рисунок Г.2 − Використання лінійної навігації для керування рухомим роботом

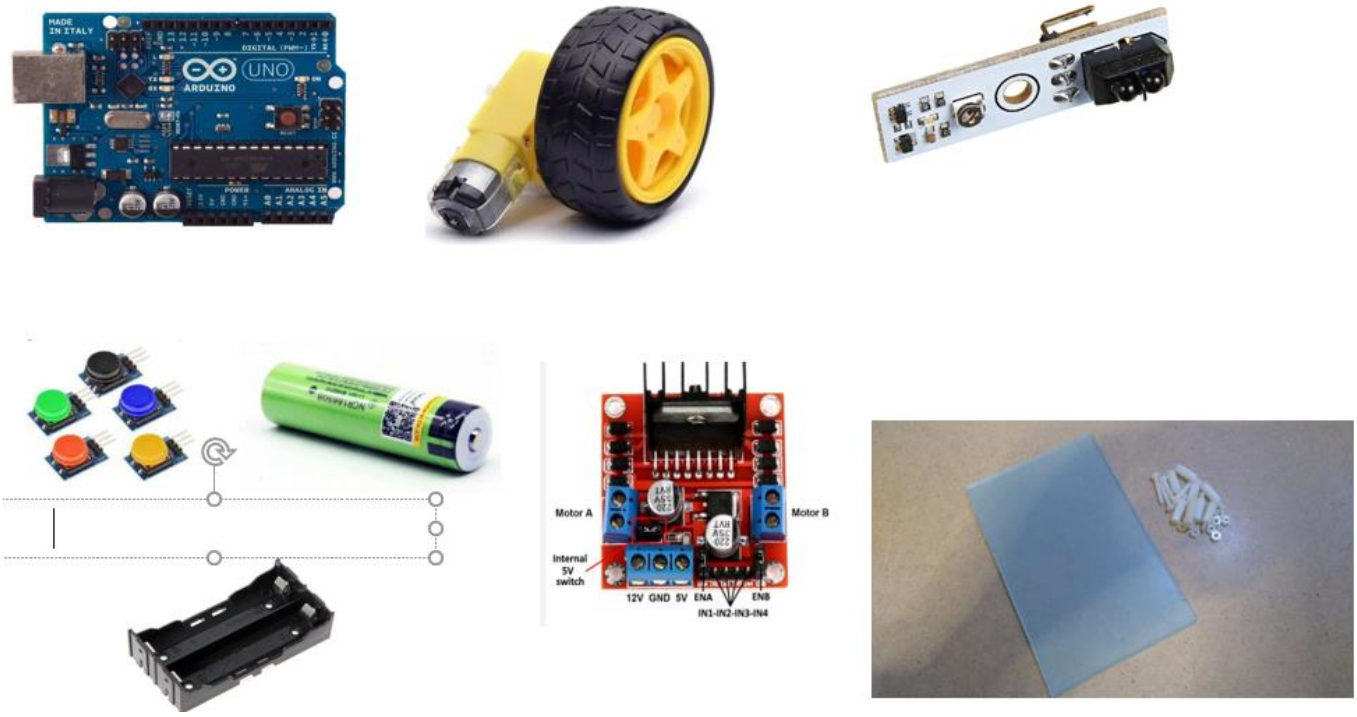


Рисунок Г.3 – Компоненти для реалізації робота

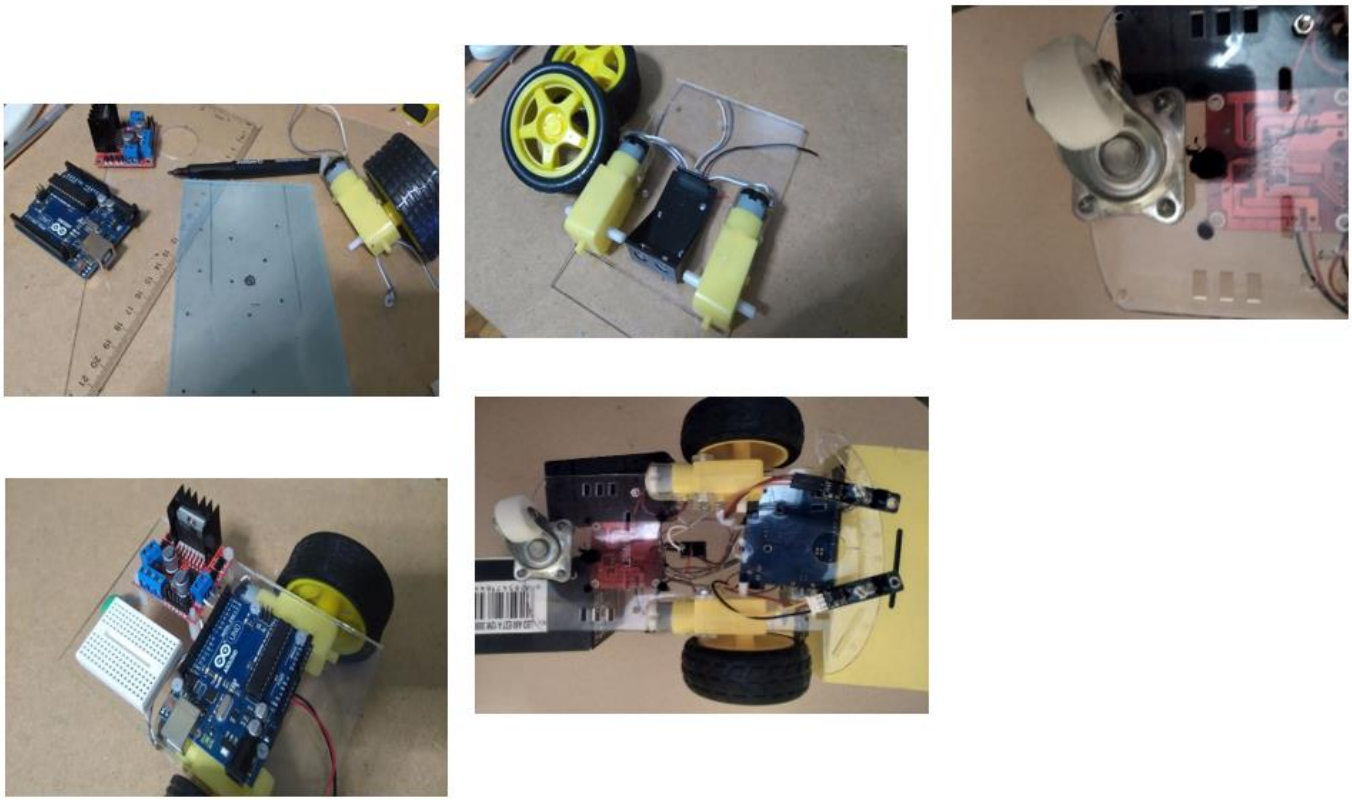


Рисунок Г.4 – Процес виготовлення робота

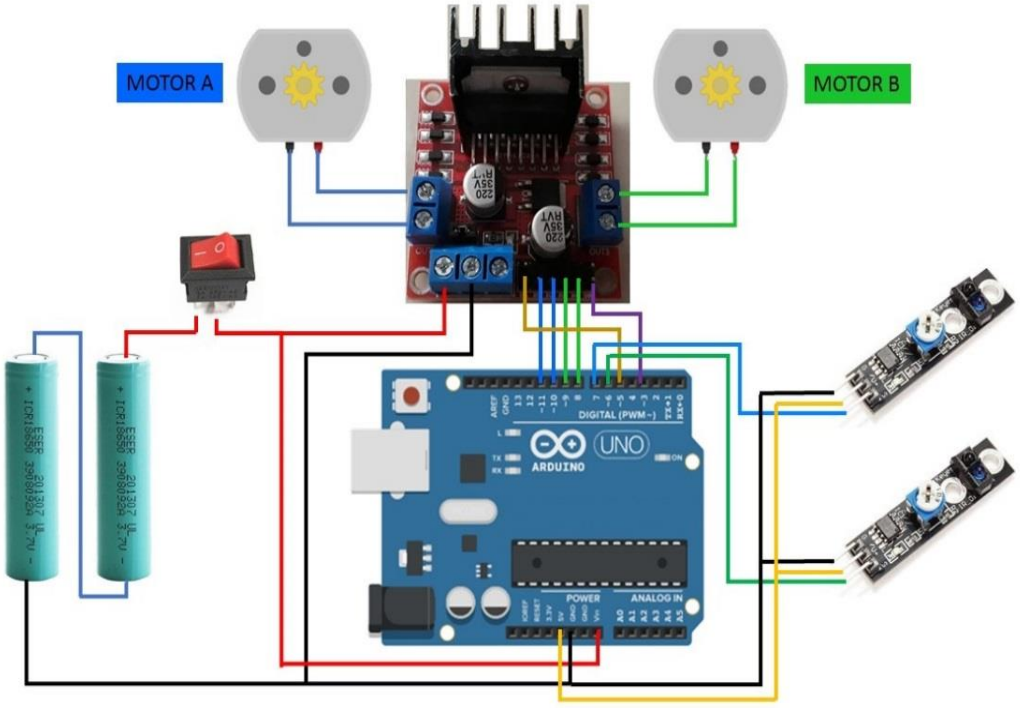


Рисунок Г.5 − Монтажна схема робота, що рухається по заданій траєкторії

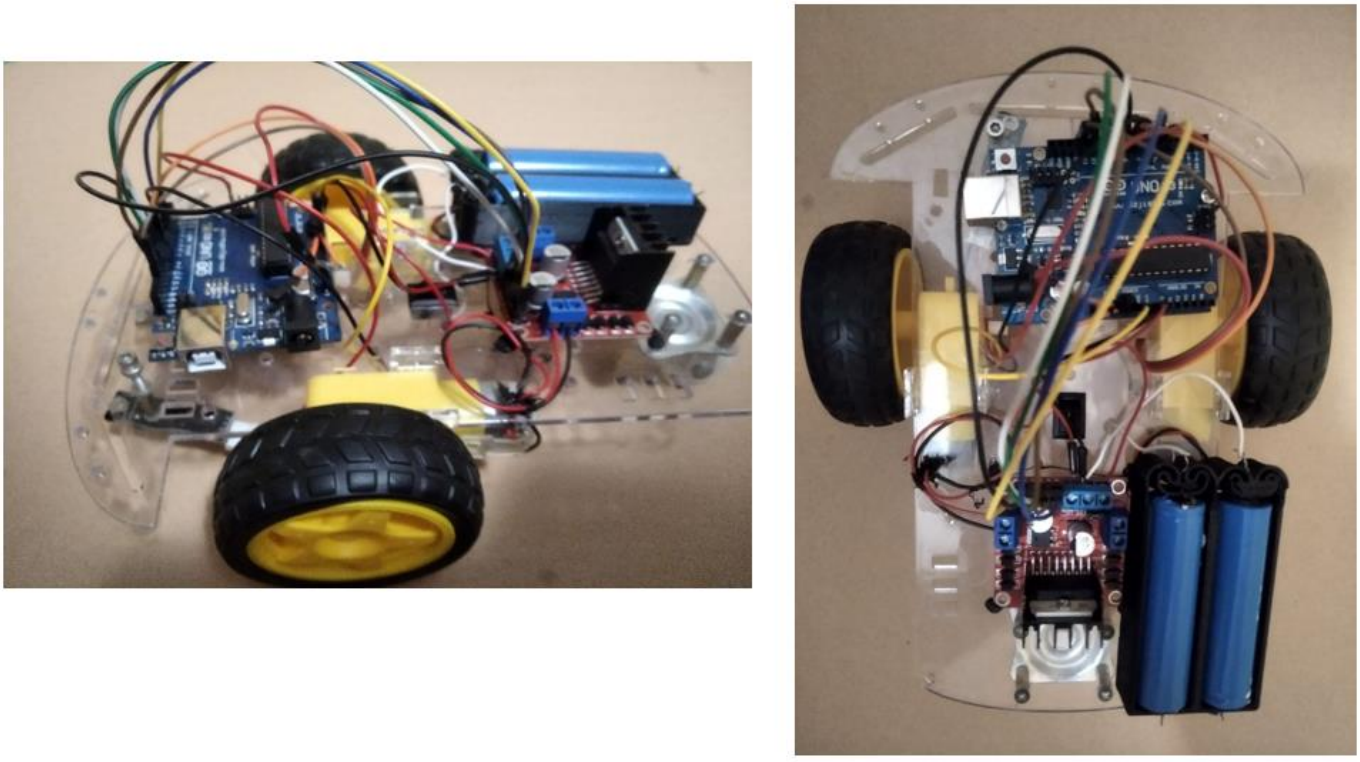


Рисунок Г.6 − Готовий макет робота

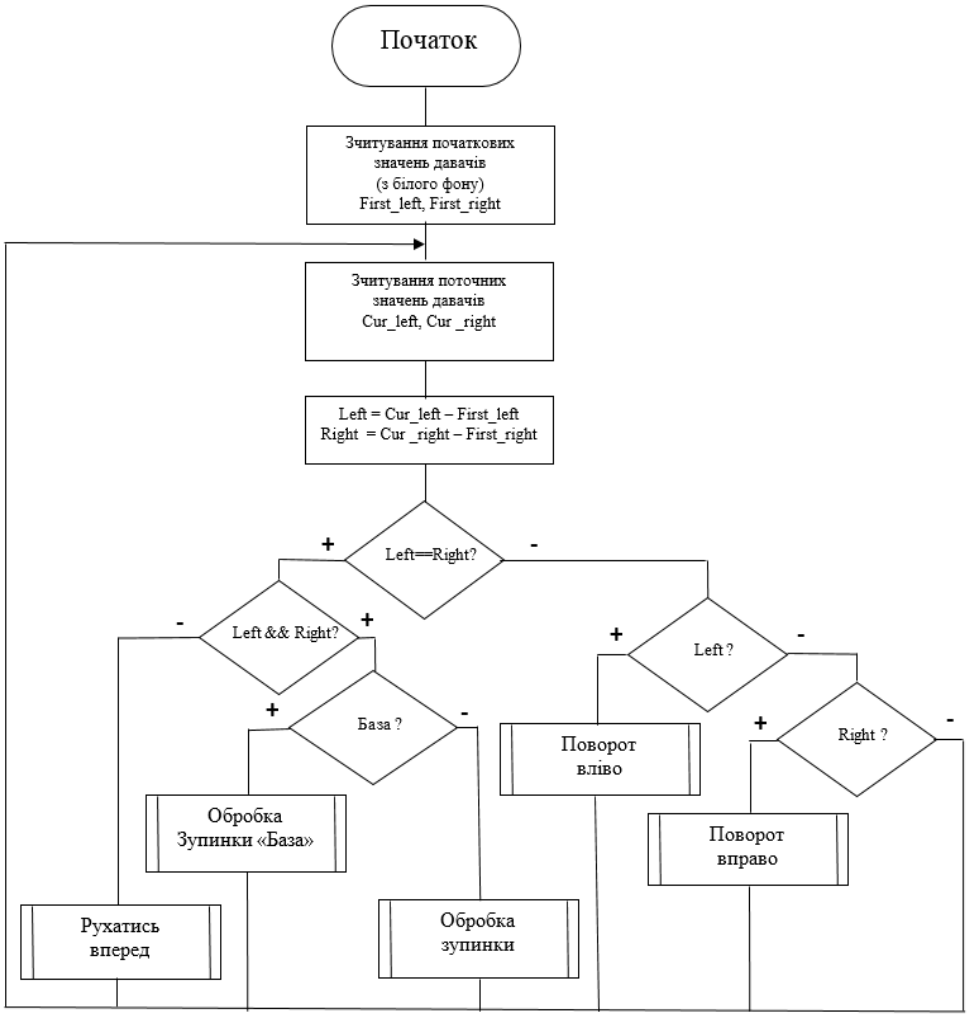
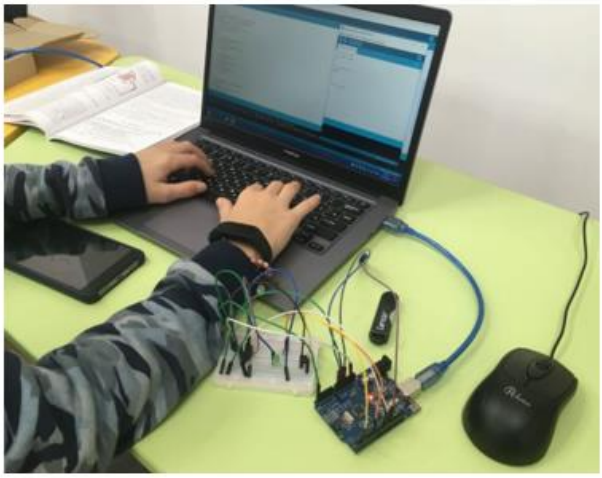


Рисунок Г.7 − Алгоритм управління роботом, що рухається по заданій траєкторії



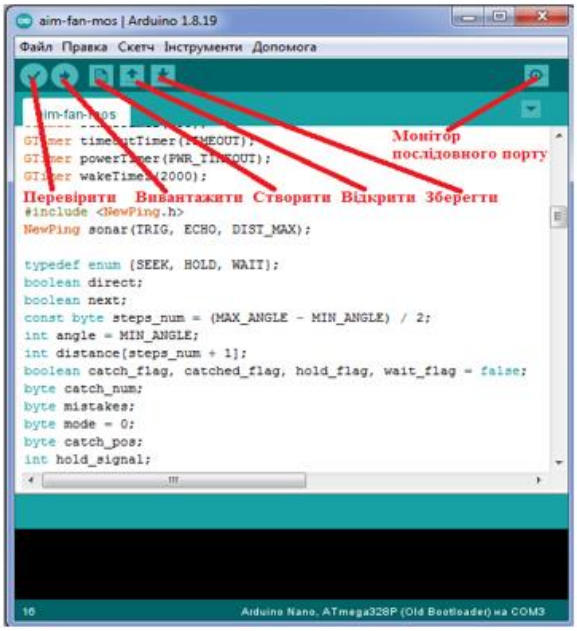


Рисунок Г.8 – Процес програмування робота, що рухається по заданій траєкторії

ДОДАТОК Д (довідниковий)

Довідка про впровадження

МАЛЕ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "ТОВ "ІТІ"
Україна, 21021, м.Вінниця, вул. Келецька, 56

№ 54 від "6" 06 2025 р.

ДОВІДКА

Дана Шамраю Максиму Віталійовичу в тому, що результати, одержані ним в процесі виконання бакалаврської кваліфікаційної роботи, а саме алгоритм роботи робота, що рухається по аданій траекторії, планується використати в розробках ТОВ «ІТІ».

Зам. директора ТОВ «ІТІ» «ІТІ» Бодяк В.М.