

Міністерство освіти і науки України  
Державний заклад  
«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

**Назаревич Леонід Юрійович**

**СИСТЕМА НАВІГАЦІЇ РОБОТИЗОВАНОГО ТРАНСПОРТНОГО  
ЗАСОБУ НА БАЗІ ARDUINO**

**кваліфікаційна робота**

**здобувача вищої освіти другого (магістерського) рівня**

**освітньої програми «Комп'ютерні мережі»**

**за спеціальністю 123 Комп'ютерна інженерія**

Особистий підпис \_\_\_\_\_ Леонід НАЗАРЕВИЧ

Науковий керівник \_\_\_\_\_ Володимир ДОНЧЕНКО,  
старший викладач кафедри  
інформаційних технологій та систем

Завідувач кафедри \_\_\_\_\_ Микола СЕМЕНОВ,  
кандидат педагогічних наук, доцент  
кафедри інформаційних технологій  
та систем

Полтава – 2025

## **АНОТАЦІЯ**

**Назаревич Л. Ю.**

**Тема:** Система навігації роботизованого транспортного засобу на базі ARDUINO.

**Спеціальність:** 123 «Комп'ютерна інженерія».

**Установа:** ЛНУ імені Тараса Шевченка, 2025 р.

**Магістерська робота містить:** 92 с., 57 рис., 2 табл., 2 додат., 59 джерел.

**Об'єктом дослідження** – система навігації роботизованого транспортного засобу.

**Предмет дослідження** – методи та алгоритми навігації роботизованого транспортного засобу, а також апаратно-програмне забезпечення навігації на базі Arduino.

**Мета дослідження** – створення системи управління мобільного колісного робота з використанням Arduino, розробка мобільного колісного робота.

**Результати роботи.** У магістерській роботі створено та реалізовано апаратно-програмний комплекс мобільного роботизованого транспортного засобу здатного орієнтуватися у просторі у режимі реального часу та уникати перешкоди, спираючись на: мікроконтролерну плату Arduino UNO, шилд драйверу двигунів Adafruit Motor Shield; використовується код, написаний у програмному забезпеченні Arduino IDE із застосуванням бібліотек AF Motor та Servo.

Отримані результати можуть бути корисними при подальших дослідженнях та розробках різних систем навігації мобільних роботів, завдання яких – пересуватися у невизначеному середовищі з численними перешкодами.

**Ключові слова.** НАВІГАЦІЯ, САМОХІДНА ТЕХНІКА, МОБІЛЬНИЙ РОБОТ, АЛГОРИТМИ УНИКНЕННЯ ПЕРЕШКОД, ДАТЧИКИ, УЛЬТРАЗВУКОВИЙ ДАТЧИК, ARDUINO.

## ABSTRACT

**Nazarevych Leonid**

**Theme:** ARDUINO-based robotic vehicle navigation system.

**Speciality:** 123 "Computer Engineering"

**Institution:** Luhansk Taras Shevchenko National University (LTSNU), 2025.

**Master's work of:** 92 pages, 57 Fig., 2 Table, 1 adj., 59 source.

**A research object is** robotic vehicle navigation system.

**The article of research is** creation of a control system for a mobile wheeled robot using Arduino, development of a mobile wheeled robot.

**An aim of work is** creation of a control system for a mobile wheeled robot using Arduino, development of a mobile wheeled robot.

**Job performances.** In the master's thesis, a hardware and software complex of a mobile robotic vehicle capable of navigating in space in real time and avoiding obstacles was created and implemented, based on: the Arduino UNO microcontroller board, the Adafruit Motor Shield motor driver shield; the code written in the Arduino IDE software using the AF Motor and Servo libraries is used.

The results obtained may be useful in further research and development of various navigation systems for mobile robots, the task of which is to move in an uncertain environment with numerous obstacles.

**Keywords.** NAVIGATION, SELF-DRIVING VEHICLE, MOBILE ROBOT, OBSTACLE AVOIDANCE ALGORITHMS, SENSORS, ULTRASONIC SENSOR, ARDUINO.

## ЗМІСТ

<b>ВСТУП.....</b>	<b>5</b>
<b>РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ</b>	<b>7</b>
1.1. Історія розвитку робототехніки.....	7
1.2. Сфери застосування колісних рухомих платформ .....	11
1.3. Огляд існуючих автоматизованих систем управління рухомими платформами.....	17
1.4. Платформа Arduino як інструмент для швидкого прототипування.....	27
1.5. Постановка задачі.....	31
Висновки до розділу .....	32
<b>РОЗДІЛ 2. АНАЛІЗ СУЧАСНИХ МЕТОДІВ АВТОНОМНОЇ НАВІГАЦІЇ МОБІЛЬНИХ РОБОТІВ З АКЦЕНТОМ НА АЛГОРИТМАХ УХИЛЕННЯ ВІД ПЕРЕШКОД.....</b>	<b>34</b>
2.1. Автономна навігація .....	34
2.2 Алгоритми навігації.....	41
2.3 Алгоритми планування маршруту.....	43
Висновки розділу .....	52
<b>РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА СТВОРЕННЯ ПРОГРАМОВАНОГО РОБОТА НА БАЗІ ARDUINO.....</b>	<b>54</b>
3.1. Апаратна база мобільного робота на основі Arduino Uno .....	54
3.2. Побудова мобільного робота на базі Arduino .....	71
3.3 Програмні засоби проєктування мобільного робота .....	76
3.4. Тестування мобільного робота .....	81
Висновки розділу .....	84
<b>ВИСНОВКИ .....</b>	<b>85</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b>	<b>87</b>
<b>ДОДАТОК А.....</b>	<b>93</b>
<b>ДОДАТОК Б .....</b>	<b>105</b>

## ВСТУП

Сьогодні рухомі платформи знаходять застосування в різних сферах діяльності людини. Вони використовуються для транспортування вантажів, виконання дистанційних операцій у небезпечних для людини умовах та інших завдань. Завдяки широкому спектру можливих застосувань, ці платформи привертають увагу науковців і розробників.

Одним із найпоширеніших типів рухомих платформ є колісні платформи, які можуть мати різну кількість коліс. Одноколісні та двоколісні платформи, хоча й забезпечують високу мобільність, є нестійкими нелінійними системами, що вимагають складних алгоритмів для автоматичного підтримання рівноваги. Через ці особливості найбільшу популярність здобули чотириколісні платформи, які мають високу стійкість і не потребують складних рішень для балансування. У цьому дослідженні розглядається саме чотириколісна платформа.

Актуальність роботи зумовлена швидким розвитком робототехніки та мехатроніки, а також зростаючою потребою в автоматизованих системах управління чотириколісними платформами в різних галузях. Розробка та створення інтелектуальних роботів має значний практичний потенціал у сучасному світі.

Arduino є зручним набором, що включає електронний блок і програмне забезпечення. Електронний блок – це друкована плата з мікроконтролером і необхідними для роботи елементами. Програмна частина містить середовище розробки і мову програмування (C/C++), що дозволяють створювати алгоритми і системи управління самохідними роботами.

З огляду на викладене, важливим завданням є вдосконалення алгоритмів систем навігації для транспортних засобів і розробка моделей їхнього пересування. Використання платформи Arduino для прототипування та тестування нових рішень сприяє прискоренню цих процесів і кращому розумінню принципів створення роботизованих систем. Це обґрунтовує актуальність розробки системи навігації для самохідної техніки на базі Arduino, яка є предметом і метою цього дослідження.

**Об’єктом дослідження** є система навігації роботизованого транспортного засобу.

**Предмет дослідження** – методи та алгоритми навігації роботизованого транспортного засобу, а також апаратно-програмне забезпечення навігації на базі Arduino.

**Мета дослідження** – створення системи управління мобільного колісного робота з використанням Arduino, розробка мобільного колісного робота.

Досягнення поставленої мети передбачає розв’язання таких задач:

- 1) дослідити історію розвитку робототехніки
- 2) розглянути існуючі системи автоматизованого управління рухомими платформами, обґрунтувати доцільність використання платформи Arduino для створення прототипу роботизованого транспортного засобу;
- 3) виявити проблеми автономності наземних транспортних засобів;
- 4) визначити базовий апаратний та програмний інструментарій створення прототипу роботизованого транспортного засобу на базі Arduino;
- 5) створити програмне забезпечення для реалізації запропонованого алгоритму; здійснити кодування програм, використовуючи Arduino IDE та мову програмування C/C++; розробити схему з’єднання компонентів та створити прототип робота-автомобіля;
- 6) здійснити тестування моделі, її руху та здатності уникати перешкоди; оцінити отримані результати.

Практична цінність результатів дослідження полягає у розробці алгоритму обходу перешкод та створенні прототипу самохідного колісного робота на його основі. Ця розробка може бути використана при створенні роботизованих систем для виконання завдань у замкнених просторах. Особливо перспективним є застосування напрацювань у розробці візків для транспортування поранених, хворих, осіб з інвалідністю та людей похилого віку в установах охорони здоров’я.

# РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1. Історія розвитку робототехніки

Робототехніка є прикладною наукою, що займається розробкою роботів – автоматизованих технічних систем, які виконують певні завдання, часто імітуючи поведінку живих організмів [1]. Сама назва терміну походить від поєднання слів «робот» і «техніка». Відповідно до визначення, поданого у «Великій українській енциклопедії», робот – це «автоматичний пристрій, що імітує поведінку живого організму, або комп'ютерна програма, яка регулярно виконує задану послідовність інструкцій для збору й обробки інформації» [2].

Головна мета робототехніки полягає у створенні робототехнічних систем, які здатні автоматизувати технологічні процеси та виконувати завдання, що є небезпечними, важкими або недосяжними для людини. Застосування робототехніки охоплює широкий спектр сфер людської діяльності, включаючи промисловість, побут, авіацію, будівництво, а також екстремальні умови, як-от космічні місії, підводні дослідження чи військові операції [1].

Термін «робототехніка» вперше ввів письменник-фантаст Айзек Азімов у 1942 році. У своєму відомому оповіданні «Я, робот» він також сформулював так звані «три закони робототехніки», які стали етичним фундаментом для створення автономних систем:

- Робот не може завдати шкоди людині або, своєю бездіяльністю, допустити, щоб людині було завдано шкоди.
- Робот повинен виконувати накази людини, за винятком тих, які суперечать першому закону.
- Робот має піклуватися про власну безпеку, якщо це не суперечить першому та другому законам [1].

Робототехніка, крім вдосконалення технологій, приділяє значну увагу етичним аспектам взаємодії між людьми та роботами, що є особливо важливим у контексті стрімкого розвитку цієї галузі. На сьогодні робототехніка стала невід'ємною частиною численних наукових і виробничих сфер, а інтерес до неї зростає з кожним роком. Завдяки розвитку робототехнічних систем усе більше

рутинних завдань виконується автоматизовано, що значно скорочує час виробництва, сприяє науковому прогресу та дозволяє вирішувати проблеми, які раніше здавалися нездоланими.

Історія робототехніки тісно пов'язана із загальним розвитком науки і техніки. Її початок можна віднести до виникнення ідеї створення штучного аналога живої істоти. Перші згадки про роботизовані механізми датуються часами Стародавньої Греції та Римської імперії [3]. Однак сучасна історія робототехніки бере свій початок з 1942 року, коли Айзек Азімов ввів термін «робототехніка» і сформулював три основні закони роботів. Далі розвиток галузі почав активно набирати обертів, особливо з середини 1940-х років.

У 1947 році було створено перший напівпровідниковий транзистор, що став фундаментом для швидкого прогресу в електроніці та, відповідно, у робототехніці. Вже наступного року Норберт Вінер сформулював основні принципи кібернетики, які заклали основу для сучасних робототехнічних систем [4].

У 1951 році Раймонд Герц представив перші механічні маніпулятори, розроблені для роботи з небезпечними радіоактивними матеріалами. Цей пристрій точно імітував рухи людської руки, що стало важливим кроком у створенні роботів, здатних виконувати складні та небезпечні операції. У цьому ж році було видано наказ про розробку автоматизованих систем управління військовою технікою, що дало поштовх до подальшого вдосконалення робототехнічних рішень [4].

Одним із перших роботів у сучасному розумінні цього терміна було створено наприкінці 1950-х років. У 1959 році (хоча деякі джерела вказують на 1954 рік [5]) винахідник-самоучка Джордж Девол розробив автоматизованого промислового робота Unimation. Цей робот виконував складальні операції на виробничій лінії General Motors. Його маса складала близько двох тонн, а рухи здійснювалися за допомогою гідравлічних приводів. Точність маніпулятора становила 0,254 мм, а програма для його роботи була записана на магнітному



барабані. Вважається, що саме з появи цього робота розпочалося сучасне роботобудування [4].

1960-ті роки стали періодом значного прогресу у технологіях, зокрема й у сфері робототехніки. У цей час було створено мобільного прото-робота «Beast», розробленого в Університеті Джона Гопкінса (США). Цей пристрій демонстрував елементарні ознаки штучного інтелекту та здатність орієнтуватися у просторі. У 1962 році на заводі Ford у Гуанчжоу було встановлено першого промислового циліндричного робота Versatran, розробленого компанією AMF.

Подальший розвиток робототехніки тривав у середині 1960-х років. У 1965 році було створено робота «Walking Truck», який був призначений для допомоги солдатам у пересуванні пересічною місцевістю. У 1968–1969 роках з'явилися роботи-маніпулятори «Стенфордська Рука» та «Щупальця». Особливо важливою подією став 1969 рік, коли вперше створили робота з технічним зором, функціональність якого продемонстрували в Стенфордському дослідницькому інституті.

У цьому ж році Норвезька корпорація Trallfa представила перші у світі промислові фарбувальні роботи, розроблені для власних потреб. Основною причиною їх створення стала нестача робочої сили. Ці розробки стали важливими етапами в еволюції робототехніки та заклали основи для її подальшого розвитку [4].

У 1970-х роках робототехніка досягла значного прогресу, що дозволило створити широкий спектр роботизованих систем для різноманітних галузей. У цей період було розроблено мобільного робота, здатного створювати карти простору та орієнтуватися за ними. Також було впроваджено технологію інтелектуального технічного зору, яка дозволила роботам визначати розташування об'єктів та оцінювати їхні габарити. У 1971 році на заводі Daimler Benz у Європі було запущено першу роботизовану зварювальну лінію. Згодом, у 1973 році, з'явився перший шестиосьовий промисловий робот-маніпулятор з електроприводом на всіх осях. Того ж року було створено роботизовану руку для

складання дрібних виробів, яка використовувала датчики дотику та сенсори зворотного тиску, а її керування здійснювалося за допомогою міні-ЕОМ [4].

Важливими досягненнями стали впровадження першого промислового контролера для управління роботами у 1974 році та використання цифрового мікропроцесора в таких контролерах. Першим роботом, оснащеним мікропроцесорним управлінням, став IRB 6, що працював на базі 8-бітного мікропроцесора Intel із 16 Кб пам'яті. Попит на роботи з великою вантажопідйомністю в автомобільній промисловості привів до створення у 1975 році першого робота, здатного підіймати до 60 кг, який був застосований на заводі Saab у Швеції. У 1976 році робот вперше використовувався у космосі як маніпулятор у зондах Viking 1 і Viking 2 [4].

Наприкінці 1970-х років з'явилися нові складальні роботи, які скоротили кількість рутинних операцій та підвищили продуктивність. У 1979 році заміна гідравлічних приводів на електроредуктори стала переломним моментом у робототехніці. У 1982 році IBM розробила спеціалізовану мову програмування для автоматизованих роботизованих систем, що значно спростило їхнє використання. У 1984 році на ринку з'явилися роботи типу SCARA з електродвигунами, що забезпечило їхню простоту, надійність та високу швидкодію [4].

У 1985 році компанія Fanuc вперше використала роботизовані лінії для виробництва інших роботів, започаткувавши нову еру автоматизації. У 1990-х роках було розроблено нові контролери з підтримкою операційної системи Windows та 6D-мишки, що значно полегшило програмування роботів. У цей період роботи стали невід'ємною частиною досліджень вулканів та космосу. У 1996–1997 роках на Марс був відправлений марсохід, який дистанційно досліджував геологічні зразки та фотографував поверхню планети [4].

Роботи 2000-х років стають дедалі більш досконалими та схожими на людину. Вони не лише отримують оновлений зовнішній вигляд, але й набувають нових можливостей, таких як розпізнавання навколишніх об'єктів, гра на

музичних інструментах, синхронізація один з одним тощо. Їхня еволюція тісно пов'язана із загальним науково-технічним прогресом і триває й досі.

## **1.2. Сфери застосування колісних рухомих платформ**

У цій кваліфікаційній роботі досліджується створення автоматизованої комп'ютерної системи управління рухомою платформою. Такі системи належать до робототехнічних та мехатронних комплексів, розглянутих у попередніх розділах. Рухомі платформи бувають різних типів, серед яких колісні є найбільш поширеними і слугують базою для багатьох робототехнічних систем. Ці платформи широко використовуються у різних сферах людської діяльності. Зважаючи на це, розглянемо детальніше сфери застосування робототехнічних систем на основі колісних рухомих платформ.

Однією з ключових характеристик будь-якої механічної системи, зокрема робототехнічної, є її практична цінність. Залежно від цього виділяють різновиди робіт: промислові, побутові, медичні, дослідницькі, ігрові, будівельні, військові, підводні, космічні тощо. Роботи також класифікуються за типом управління (керовані чи автономні) і мобільністю (мобільні чи стаціонарні). Основними напрямками розвитку робототехніки є: промислова автоматизація, медичні роботи, безпілотники, штучний інтелект та логістика [4].

Колісні рухомі платформи найчастіше зустрічаються у складі промислових, побутових, будівельних, дослідницьких, ігрових та військових робіт. Вони можуть бути як керованими, так і автономними, але найчастіше використовуються керовані варіанти. У контексті основних напрямків розвитку робототехніки, рухомі платформи належать до таких сфер, як промислова автоматизація, логістика та штучний інтелект.

Рухомі платформи також можуть бути компонентами більш складних робототехнічних систем і використовуватися в інших напрямках робототехніки. Знання класифікації робіт дає змогу чіткіше розуміти потенційні сфери застосування автоматизованих комп'ютерних систем управління рухомими платформами.

Часто такі платформи є частиною багатофункціональних роботів, які виконують складніші завдання, ніж просто переміщення у просторі. Це дозволяє використовувати їх у галузях, де переміщення є необхідною умовою для виконання інших функцій, а не лише у сферах, що безпосередньо пов'язані з транспортуванням.

Роботи на базі автоматизованих рухомих платформ мають широкий спектр застосувань і використовуються практично у всіх сферах сучасної діяльності. Наприклад, у медицині такі роботи забезпечують виконання високоточних хірургічних операцій, які є недосяжними для людської руки через фізіологічні обмеження. У виробничій сфері вони беруть на себе виконання небезпечних для людини завдань, зокрема операцій у середовищах із підвищеною температурою чи токсичними речовинами. Значна частина застосувань пов'язана із транспортною галуззю, де безпілотні роботизовані системи ефективно виконують завдання доставки вантажів [3].

Класифікація роботів за різними критеріями дає змогу краще розуміти їхнє призначення та потенціал. Одним із поширених підходів є поділ за сферами застосування, що дозволяє визначити, в яких галузях роботи, зокрема автоматизовані рухомі платформи, можуть бути найкориснішими. Згідно з цією класифікацією, усі робототехнічні системи поділяються на шість основних категорій [3]:

1. Промислові роботи.
2. Побутові роботи.
3. Соціальні роботи.
4. Медичні роботи.
5. Дослідницькі роботи.
6. Бойові роботи.

Промислові роботи є одними з найбільш поширених у світі. Вони призначені для виконання рухових і контрольних функцій у виробничих процесах. Зазвичай такі системи є автономними та поєднують механічний маніпулятор із програмованим пристроєм управління. Промислові роботи

широко використовуються для транспортування об'єктів у просторі та поділяються на кілька підкатегорій, серед яких: ливарні, складальні, фарбувальні, будівельні, транспортні, фасувально-сортувальні, сільськогосподарські та роботи для механічної обробки матеріалів [3].

Автоматизовані рухомі платформи, інтегровані в промислові роботи, найчастіше застосовуються у складальних, транспортних та сільськогосподарських системах. Їхня роль полягає у забезпеченні мобільності та гнучкості процесів, що значно підвищує ефективність і продуктивність. На рис. 1.1 представлено приклад сучасного складального робота, який демонструє основні тенденції розвитку робототехніки в цій сфері.



Рис. 1.1. Складальний робот «BEFARD»

Розвиток сільського господарства тісно пов'язаний з інтеграцією інноваційних технологій, серед яких особливе місце займають роботизовані системи на рухомих платформах. Ці системи покликані автоматизувати трудомісткі та монотонні процеси, підвищуючи ефективність сільськогосподарського виробництва.

Незважаючи на значний потенціал, розробки в цій галузі на сьогодні переважно зосереджені на створенні дослідницьких прототипів. Повсюдного

впровадження агророботів у виробництво поки що не спостерігається. Одним з ключових напрямків досліджень є розробка систем руху, які б мінімізували ущільнення ґрунту та забезпечили ефективне пересування роботів по полю [10].

Прикладом такого роду розробок є робот «AgroBot» (рис. 1.2), який демонструє можливості використання роботизованих систем в сільському господарстві. Однак, для широкого впровадження агророботів необхідне вирішення низки технічних та економічних проблем, таких як:

- Створення модульних платформ, які можуть бути адаптовані під різні типи сільськогосподарських робіт.
- Розробка інтуїтивно зрозумілих інтерфейсів та алгоритмів керування, що дозволять операторам ефективно працювати з роботами.
- Розробка енергоефективних приводів та систем живлення, що дозволять роботам працювати тривалий час без підзарядки.
- Розробка технологій, які дозволять знизити вартість виробництва агророботів та зробити їх доступними для широкого кола користувачів.



Рис. 1.2. Сільськогосподарський робот «AgroBot»

Окрему категорію становлять побутові роботи, призначені для виконання різноманітних завдань у домашньому господарстві. Цей сегмент робототехніки демонструє стрімкий розвиток, оскільки все більше людей прагнуть автоматизувати рутинні побутові процеси. Типовим прикладом побутового

робота є робот-пилосос (рис. 1.3), який, завдяки вбудованій автоматизованій платформі, здатний самостійно орієнтуватися в просторі та виконувати прибирання [3]. Ця технологія базується на принципах автономної навігації та сприйняття навколишнього середовища, що робить її перспективною для розробки нових поколінь побутових роботів.



Рис. 1.3. Робот-пилосос Xiaomi Robot Vacuum E10

Сфера соціальної робототехніки орієнтована на створення машин, здатних до спілкування та взаємодії з людьми на емоційному рівні. Ці роботи можуть використовуватися як компаньйони для літніх людей, допоміжні засоби для навчання або інструменти для проведення терапії.

Медична робототехніка спрямована на розробку автоматизованих систем, які можуть виконувати різноманітні медичні процедури, від хірургічних операцій до доставки ліків. Роботи-фармацевти та роботи-аптечки, обладнані рухомими платформами, є яскравими прикладами застосування робототехніки в медицині.

Дослідницькі роботи використовуються для збору даних в умовах, які є недоступними або небезпечними для людини. Вони оснащуються різноманітними датчиками та інструментами, що дозволяють їм збирати інформацію про навколишнє середовище. Отримані дані передаються оператору для подальшого аналізу. Типовим прикладом є марсоходи, які досліджують поверхню Марса (рис. 1.4).



Рис. 1.4. Марсохід

Бойові роботи представляють собою нове покоління військової техніки, здатне виконувати широкий спектр завдань, від розвідки до безпосередньої участі в бойових діях. Завдяки своїй автономності або напівавтономності, вони здатні діяти в умовах підвищеної небезпеки, зменшуючи ризики для життя військовослужбовців та цивільного населення [3].

Російсько-українська війна 2022 року стала потужним стимулом для розвитку роботизованих систем в Україні. Нагальна потреба в ефективних засобах ведення бойових дій прискорила розробки нових рішень у сферах розвідки, розмінування, транспортування, протиповітряної оборони та інших. Особливу актуальність набуло питання забезпечення стійкості роботизованих систем в умовах радіоелектронної боротьби. Саме тому пріоритетним напрямком досліджень стала розробка надійних автономних платформ, здатних виконувати завдання навіть за відсутності зв'язку з оператором.

Одним з перспективних напрямків застосування роботизованих систем є розмінування. Колісні роботи, оснащені маніпуляторами (рис. 1.5), можуть ефективно виявляти та знешкоджувати вибухонебезпечні предмети, значно знижуючи ризик для саперів.





Рис. 1.5. Робот-сапер

Універсальні бойові роботи (рис. 1.6) представляють собою багатофункціональні платформи, здатні виконувати широкий спектр завдань на полі бою. Вони можуть використовуватися для транспортування вантажів, включаючи евакуацію поранених, а також для проведення розвідки та участі в бойових діях. Залежно від конкретної моделі, такі роботи можуть працювати як в автономному режимі, так і під управлінням оператора.



Рис. 1.6. Універсальний бойовий робот

### **1.3. Огляд існуючих автоматизованих систем управління рухомими платформами**

Автоматизовані рухомі платформи набули широкого застосування в різних сферах людської діяльності, від промисловості до медицини. Інтенсивний розвиток цієї галузі викликає необхідність детального дослідження алгоритмів та методів керування такими системами.

Серед різноманіття рухомих платформ особливе місце займають колісні платформи. Вони відрізняються конструктивною простотою та високою

маневреністю. Однак, кожна конфігурація колісної платформи (одноколісна, двоколісна, багатоколісна) має свої особливості керування, що вимагає розробки специфічних алгоритмів, особливо для підтримки стійкості одно- та двоколісних платформ.

Сучасні наукові дослідження в галузі автоматизованого керування рухомими платформами охоплюють широкий спектр тем. Дослідники працюють над оптимізацією алгоритмів керування, розробкою систем сприйняття навколишнього середовища, підвищенням точності позиціонування, а також розробкою спеціального програмного забезпечення для управління платформами.

Більшість наукових робіт розглядають рухомі платформи як складові частини більш складних робототехнічних систем. Для керування такими системами застосовується широкий спектр методів, включаючи штучний інтелект, нейронні мережі та нечітку логіку.

З метою систематизації існуючих досліджень, рухомі платформи можна класифікувати за типом рухомого органу. Найбільш поширені колісні платформи можуть бути додатково класифіковані за кількістю коліс: одноколісні, двоколісні та багатоколісні. Такий підхід дозволяє виділити специфічні особливості керування кожною з цих груп платформ.

Розглянемо спочатку неколісні рухомих платформи, які відіграють важливу роль у багатьох галузях.

Роботи-гексаподи є одним з найбільш вивчених типів неколісних роботів. Їхнє шестиноге конструкція надає їм підвищену прохідність по пересіченій місцевості, що робить їх перспективними для використання в складних умовах, таких як шахти або зони катастроф [11]. Автори статті [11] підкреслюють важливість розвитку роботів-гексаподів, оскільки вони можуть виконувати завдання в місцях, недоступних для колісних та гусеничних роботів.

Гусеничні платформи також широко застосовуються в робототехніці. Їхні переваги полягають у високій прохідності та здатності долати перешкоди. В статті [12] розглядається застосування гусеничних платформ для створення

роботів, призначених для вимірювання теплових характеристик об'єктів. Автори відзначають, що такі роботи можуть бути використані для розвідки в небезпечних зонах, зменшуючи ризик для людей.

Найбільший науковий інтерес з точки зору теорії автоматичного керування викликають одно- та двоколісні рухомі платформи. Це пов'язано з тим, що їхня стійкість є неінтуїтивною і вимагає розробки спеціальних алгоритмів керування. Основна задача при керуванні такими платформами полягає в забезпеченні їхньої динамічної стійкості, тобто утриманні платформи у вертикальному положенні під час руху.

Більшість досліджень в цій галузі зосереджені саме на розробці ефективних алгоритмів балансування, використовуючи різноманітні методи регулювання. При цьому питання руху платформи часто розглядається як вторинне або взагалі не розглядається.

На відміну від одно- та двоколісних платформ, багатколісні платформи, як правило, є більш стійкими і не вимагають складних алгоритмів балансування. Основна увага при їх розробці приділяється задачам маневрування, навігації та виконання конкретних завдань, таких як транспортування вантажів, розвідка або розмінування.

Переваги багатколісних платформ:

- Вища стійкість: завдяки більшій кількості точок опори багатколісні платформи більш стійкі до перекидання.
- Більша вантажопідйомність: за рахунок більшої кількості двигунів і редукторів багатколісні платформи можуть переміщувати значні вантажі.
- Краща прохідність: завдяки своїй конструкції багатколісні платформи здатні долати різноманітні перешкоди.

Одноколісні роботи є одними з найскладніших для керування робототехнічних систем. Їхня нелінійна динаміка, багато ступенів свободи та необхідність постійного балансування роблять їх цікавим об'єктом для наукових

досліджень. Незважаючи на це, практичне застосування таких роботів обмежене через високу складність розробки ефективних систем керування.

У статті [13] українські дослідники розглядають можливість застосування одноколісних роботів в якості керованих коліс для багатоколісних платформ. Автори відзначають, що такі роботи можуть бути використані для автоматизації руху агрегатів по кривих траєкторіях в польових умовах. Проте, вони також підкреслюють складність керування одноколісними роботами, особливо при врахуванні динамічних характеристик системи.

Основні проблеми при керуванні одноколісними роботами:

Багато ступенів свободи: одноколісні роботи мають більше ступенів свободи, ніж багатоколісні, що ускладнює розробку моделей і алгоритмів керування.

Нелінійна динаміка: динаміка одноколісних роботів є сильно нелінійною, що ускладнює застосування традиційних методів керування.

Нестійкість: одноколісні роботи є нестійкими системами, що вимагає розробки спеціальних алгоритмів для підтримки балансу.

Серед рухомих платформ особливий інтерес представляють двоколісні самобалансуючі роботи. Їхня конструкція, що забезпечує відносно високу стійкість, дозволила знайти їм ширше застосування порівняно з одноколісними аналогами. Простота конструювання та доступність інформації сприяли тому, що багато ентузіастів збирають таких роботів самостійно на базі популярних мікроконтролерних платформ.

Яскравим прикладом двоколісного робота є розробка компанії Boston Dynamics (рис. 1.7), призначена для роботи на виробництві. Оснащений маніпулятором, цей робот здатний виконувати різноманітні завдання, пов'язані з переміщенням і обробкою предметів.



Рис. 1.7. Двоколісний робот Boston Dynamics

Двоколісні самобалансуючі платформи стають все більш популярними завдяки своїй маневреності та можливості виконання складних маневрів. Дослідження [14] демонструє, що за допомогою сучасних технологій можна створювати ефективні та доступні системи управління для таких платформ. Використання ПД-регулятора та бездротового зв'язку відкриває нові перспективи для розвитку цієї технології. В майбутньому можна очікувати появи більш складних і функціональних платформ, здатних виконувати різноманітні завдання, від доставки товарів до проведення наукових досліджень.

Розвиток технологій комп'ютерного моделювання відкриває нові можливості для дослідження та оптимізації систем управління двоколісними самобалансуючими платформами. Роботи [15-16] демонструють ефективність використання таких моделей для аналізу динаміки платформи та синтезу оптимальних алгоритмів керування.

Автори досліджень [15-16] обґрунтовують вибір двоколісних платформ як об'єкта дослідження їхньою нелінійністю та нестійкістю, що робить такі системи цікавим викликом для розробників алгоритмів керування. Для забезпечення стійкості платформи пропонується використовувати класичні ПД-регулятори або більш складні лінійно-квадратичні регулятори. Оптимальні параметри

регуляторів можуть бути розраховані за допомогою програмних пакетів, таких як Matlab/Simulink.

Важливим етапом дослідження є створення математичної моделі платформи. Автори [15-16] пропонують моделювати двоколісну платформу як інвертований маятник, що дозволяє отримати достатньо точне описування її динаміки. На основі отриманої моделі можна розробити імітаційну 3D-модель платформи в середовищі розробки ігор, наприклад, Unity. Це дозволяє візуалізувати поведінку платформи та проводити різноманітні симуляційні експерименти.

Двоколісні самобалансуючі платформи представляють собою складну та цікаву систему для дослідження в галузі автоматичного керування. Їхня нестабільність у природному стані вимагає розробки ефективних алгоритмів для підтримки балансу. Крім того, важливим аспектом є розробка зручного інтерфейсу для взаємодії оператора з системою. Комп'ютерне моделювання відіграє ключову роль у розробці та оптимізації таких систем, дозволяючи дослідникам експериментувати з різними алгоритмами та параметрами без необхідності створення фізичних прототипів.

На відміну від двоколісних платформ, багатоколісні системи, такі як чотириколісні, зазвичай більш стійкі та не вимагають складних алгоритмів для підтримки балансу. Основні дослідження в цій області зосереджені на розробці алгоритмів руху, навігації та маневрування. Однак, потенціал багатоколісних платформ не обмежується лише переміщенням. Сучасні дослідження спрямовані на розширення функціональності таких систем шляхом оснащення їх маніпуляторами, датчиками, системами відеоспостереження та іншими пристроями. Це дозволяє використовувати багатоколісні платформи в різних сферах, від промислової автоматизації до служб доставки.

Сучасні дослідження в галузі багатоколісних платформ виходять за рамки традиційних задач управління рухом. Все частіше роботизовані системи оснащуються додатковими функціями, такими як комп'ютерний зір та штучний

інтелект, що дозволяє їм взаємодіяти з навколишнім середовищем більш гнучко та автономно.

Робота [17] є яскравим прикладом такого підходу. Автори розробили мобільну платформу, здатну не лише рухатися і оминати перешкоди, але й виконувати завдання, що вимагають високої точності позиціонування. Для цього платформа оснащена системою стабілізації, яка підтримує робочий орган у горизонтальному положенні, навіть при русі по нерівній поверхні.

Важливою особливістю розробки є використання технології комп'ютерного зору. Камера, встановлена на стабілізованій платформі, дозволяє роботу розпізнавати об'єкти в навколишньому середовищі та орієнтуватися в просторі. Це відкриває широкі можливості для застосування таких платформ у різних галузях, від промислової автоматизації до служб доставки.

Автори роботи [17] також звертають увагу на важливість мініатюризації та енергоефективності таких систем. Використання мікроконтролера Arduino та оптимізація алгоритмів дозволяють створювати компактні та енергоефективні роботизовані платформи.

Дослідження, представлене в роботі [18], присвячене актуальній проблемі автоматизації керування багатоколісними платформами. Автори акцентують увагу на необхідності враховувати механічну взаємодію платформи з навколишнім середовищем для забезпечення ефективного та безпечного руху.

Традиційні підходи до керування такими платформами часто не враховують динамічні зміни в середовищі, що може призводити до неточностей у русі та навіть аварійним ситуаціям. Для вирішення цієї проблеми автори пропонують використовувати системи автоматичного регулювання, які постійно відстежують фактичний стан платформи та порівнюють його з бажаним. Завдяки цьому можна забезпечити більш точне та адаптивне керування.

Одним з ключових аспектів роботи є розгляд багатоколісної платформи з шістьма колесами. Така конфігурація забезпечує підвищену маневреність та прохідність, але також ускладнює процес управління. Автори пропонують

використовувати автоматичний регулятор для компенсації зовнішніх впливів та забезпечення стабільності руху платформи.

Сучасна європейська нормативна база [4] оперує наступними категоріями транспортних засобів:

Автоматизований транспортний засіб (Automated vehicle): транспортний засіб, обладнаний системами підтримки водія, які частково делегують виконання водійських маневрів автоматизованим системам.

Автономний транспортний засіб (Autonomous vehicle): транспортний засіб, повністю здатний виконувати всі функції керування без необхідності прямої участі людини. Згідно з визначенням Національної Адміністрації Безпеки Дорожнього Руху США [5], такий транспортний засіб здійснює всі водійські маневри, включаючи керування, прискорення та гальмування, без втручання водія.

Автор дослідження [6, с. 150] зазначає, що термін "автономний транспортний засіб" є неофіційним і використовується для опису транспортних засобів, оснащених розвиненими системами автоматизованого керування (ADS), які здатні функціонувати в автономному режимі.

За даними SAE International [7], існує неоднозначність у тлумаченні поняття "автономність водіння". Деякі дослідники ототожнюють її виключно з повністю автоматизованим керуванням (рівень 5), тоді як інші розширюють це поняття на всі рівні автоматизації. Варто зазначити, що законодавство деяких штатів США пов'язує автономність з транспортними засобами, оснащеними системами автоматизованого керування (ADS) щонайменше рівня 3.

Українські вчені Поліщук М.М. та Ткач М.М. [8, с. 31] пропонують класифікацію роботизованих транспортних засобів, виділяючи дві основні групи: транспортні роботи з фіксованим маршрутом та робочари з безконтактним керуванням. Системи керування таких роботів можуть бути як циклічними, так і позиційними з автоматичним адресуванням. Вибір системи керування залежить від функціонального призначення робота. Проблематика роботизованих транспортних засобів також розглядається в роботах [9-11].



Оскільки такі автомобілі поєднують фізичні компоненти з віртуальними системами керування, їх часто класифікують як "кіберфізичні системи" [7]. Водночас, згідно з опитуванням SAE, термін "самокерований автомобіль" (self-driving car) є найбільш поширеним серед фахівців галузі. Цей термін, популяризований компанією Waymo, точно відображає здатність таких транспортних засобів рухатися без прямої участі людини.

Оскільки цей термін точно відображає здатність таких транспортних засобів до автономного руху, ми будемо використовувати його як основний у цьому дослідженні. Термін "мобільний робот" також може бути використаний як синонім.

Архітектурна модель SDC може бути розглянута з різних точок зору, включаючи фізичні компоненти, етапи розробки, логічні функції та технологічні блоки (рис.1.8).

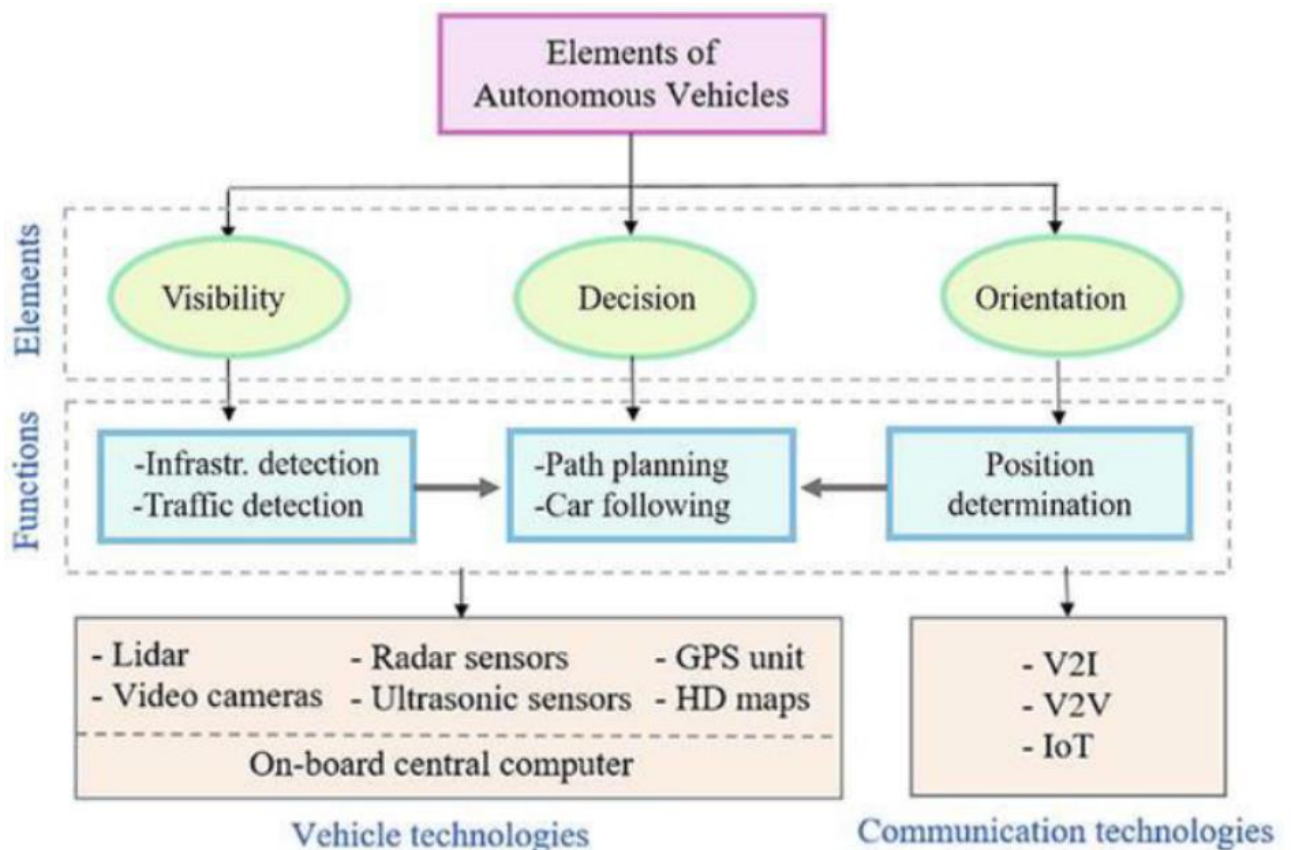


Рис. 1.8. Елементи SDC, їх функції та базові технології [12]

Для досягнення поставленої мети в цій роботі було проведено комплексне дослідження самокерованих автомобілів (SDC), зосередившись на їхніх технічних та функціональних характеристиках.

Технічний аспект дослідження зосередився на аналізі апаратного та програмного забезпечення SDC. Ці два компоненти утворюють основу архітектури SDC, кожен з яких складається з підсистем, що відповідають за різні функції [11-12] (рис. 1.9).

Для ефективної обробки великих обсягів даних, необхідних для керування SDC, використовуються потужні обчислювальні платформи, оснащені багатоядерними процесорами та графічними процесорами (GPU).

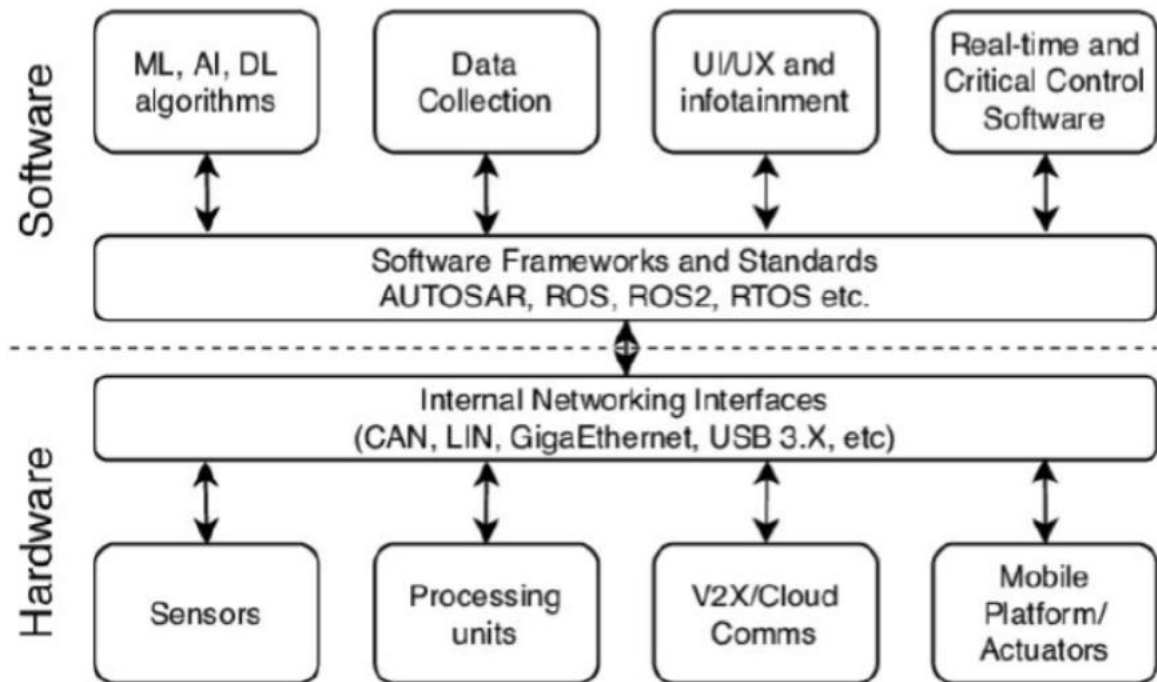


Рис. 1.9. Технічна архітектура системи SDC [11]

Крім даних, які збирає сам автомобіль, системи SDC використовують інформацію з зовнішніх джерел, таких як Інтернет, інші транспортні засоби та інфраструктура (технологія V2X). Апаратна частина SDC включає не лише сам автомобіль як мобільну платформу, але й різноманітні датчики та пристрої, які можуть варіюватися залежно від призначення та умов експлуатації.

Здатність SDC обробляти величезні обсяги даних у реальному часі дозволяє порівняти їх з потужними суперкомп'ютерами. Це стало можливим завдяки розвитку як апаратного, так і програмного забезпечення. Одним з найпоширеніших стандартів для програмного забезпечення автомобільної електроніки є AUTOSAR, розроблений консорціумом провідних

автовиробників, як BMW, Toyota, GM, Chrysler, Continental, Bosch, Daimler, Volkswagen та багато інших. AUTOSAR заснований на мові C [13].

З функціональної точки зору, SDC можна розглядати як систему, що складається з чотирьох основних блоків, які зустрічаються в більшості архітектурних рішень: блок сприйняття, блок планування та прийняття рішень, блок управління рухом і транспортним засобом та блок системного нагляду. Ці блоки, представлені на рисунку 1.10, забезпечують взаємодію SDC з навколишнім середовищем та виконання поставлених завдань.

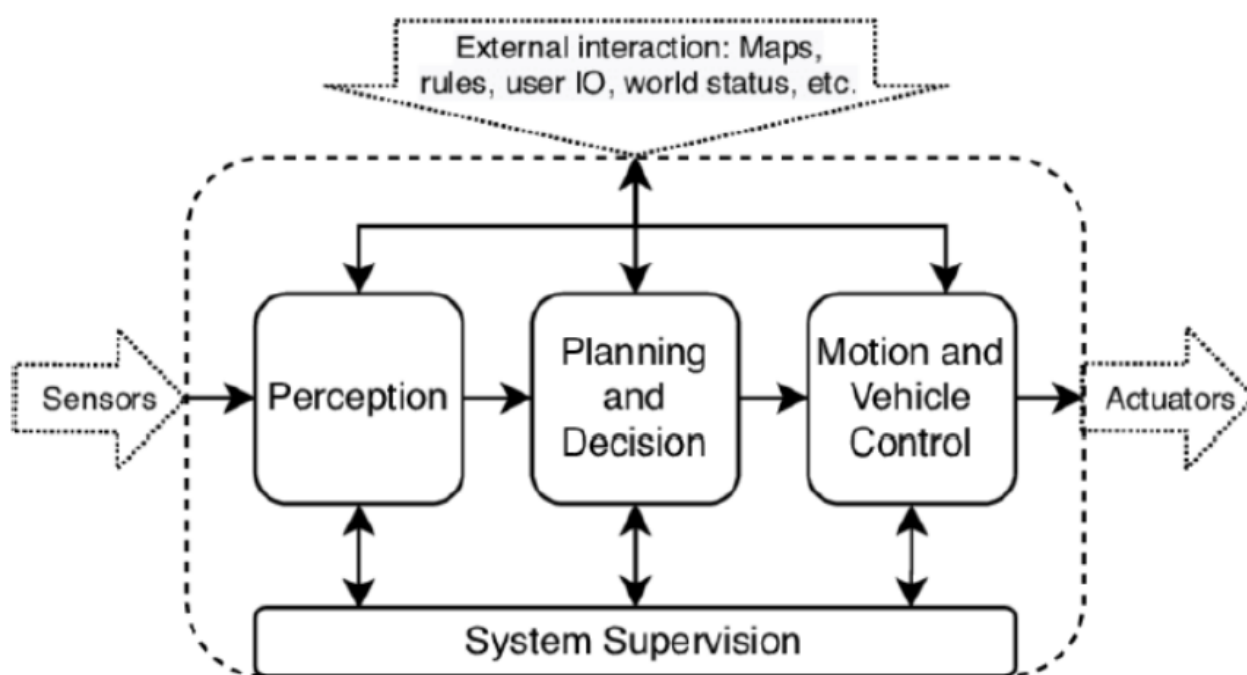


Рис. 1.10. Функціональна архітектура системи автономного водіння [11]

#### 1.4. Платформа Arduino як інструмент для швидкого прототипування

Платформа Arduino, як зазначено в роботах [35-37], є широко використовуваним інструментом для швидкого створення прототипів електронних пристроїв. Її популярність обумовлена простотою використання як апаратного, так і програмного забезпечення. Завдяки своїй гнучкості, Arduino дозволяє зчитувати різноманітні дані з датчиків та активувати різноманітні виконавчі механізми, що робить її універсальним інструментом для широкого спектру застосувань.

Процес прототипування на базі Arduino, описаний в роботі [36], включає кілька послідовних етапів:

1. Формулювання проблеми та генерація ідеї: на початковому етапі визначається проблема, яку необхідно вирішити за допомогою електронного пристрою, та формулюється відповідна ідея.
2. Концептуалізація: на цьому етапі відбувається деталізація ідеї та розроблення концептуальної моделі майбутнього пристрою. Визначаються основні функціональні блоки, алгоритми роботи та взаємодії між ними.
3. Прототипування: на етапі прототипування концептуальна модель втілюється в реальний фізичний пристрій. За допомогою платформи Arduino та відповідного програмного забезпечення створюється перша робоча версія пристрою, яка підлягає подальшому тестуванню та доопрацюванню.

Цикл розробки прототипу на базі Arduino можна узагальнити наступним чином: ідея → концепція → прототип (альфа-версія) → прототип (бета-версія). Кожна наступна версія прототипу містить все більше функціональних можливостей і наближає пристрій до кінцевого продукту.

В робототехніці Arduino широко використовується для керування різноманітними механізмами, такими як двигуни, сервоприводи та датчики. Це дозволяє створювати прототипи роботів, автономних транспортних засобів та інших інтелектуальних систем.

Платформа Arduino, як зазначено в дослідженнях [35-37], стала де-факто стандартом для швидкого прототипування електронних пристроїв, зокрема, у сфері робототехніки. Її популярність зумовлена низкою переваг:

1. Доступність: низька вартість платформи та компонентів робить Arduino привабливим варіантом для як для професіоналів, так і для любителів.
2. Кросплатформенність: середовище розробки Arduino IDE підтримує більшість операційних систем, що забезпечує гнучкість у роботі.
3. Енергоефективність: низьке енергоспоживання дозволяє використовувати Arduino в автономних системах та пристроях з обмеженим живленням.

4. Швидкість прототипування: проста конфігурація та інтуїтивно зрозуміле середовище розробки значно прискорюють процес створення прототипів.

5. Просте програмування: мова програмування Arduino, заснована на C++, є відносно легкою для освоєння, що робить платформу доступною для широкого кола користувачів, включаючи початківців.

6. Відкритість: відкритий вихідний код дозволяє розширювати функціональність платформи та створювати нові бібліотеки.

7. Інноваційність: постійна розробка та впровадження нових технологій забезпечують актуальність платформи.

8. Гнучкість: широкий вибір датчиків, модулів і щитів дозволяє створювати різноманітні робототехнічні системи.

9. Спільнота: велика спільнота користувачів забезпечує постійну підтримку, обмін досвідом та розробку нових рішень.

Простота підключення та модульна структура роблять Arduino ідеальною платформою для швидкого прототипування робототехнічних систем. Наявність стандартних інтерфейсів і широкий вибір готових модулів дозволяють навіть новачкам створювати складні системи без глибоких знань в електроніці. Масштабованість платформи дозволяє розширювати функціональність проєктів у міру необхідності.

Незважаючи на значну кількість переваг, платформа Arduino має ряд обмежень, які слід враховувати при розробці складних систем [36].

1. Обмежена обчислювальна потужність: мікроконтролери, що лежать в основі плат Arduino, мають обмежені обчислювальні ресурси порівняно з більш потужними процесорами. Це може створювати труднощі при виконанні складних алгоритмів, особливо в реальному часі.

2. Обмежена пам'ять: обсяг оперативної та постійної пам'яті плат Arduino є обмеженим. Це може бути критичним для проєктів, що вимагають великих обсягів даних або складних програмних алгоритмів.

3. Вартість додаткових компонентів: хоча самі плати Arduino відносно недорогі, загальна вартість проєкту може значно зрости через необхідність придбання додаткових датчиків, актуаторів та інших компонентів.

4. Обмеження швидкості: тактова частота мікроконтролера обмежує швидкість виконання програм. Це може бути критичним для додатків, що вимагають високої швидкості обробки даних, наприклад, для систем керування двигунами.

Незважаючи на обмеження, пов'язані з обчислювальною потужністю, обсягом пам'яті та швидкістю, платформи Arduino продовжують широко використовуватися в освітніх цілях, хобі-проєктах та для розробки простих прототипів. Для більш вимогливих застосувань, таких як промислова автоматизація або розробка складних роботів, можуть знадобитися більш потужні платформи. Однак, Arduino залишається відмінною стартовою точкою для вивчення основ електроніки та програмування мікроконтролерів.

Аналіз літератури [36] засвідчив, що платформа Arduino знайшла широке застосування у розробці прототипів у різноманітних галузях, включаючи промислове проєктування, побутову автоматизацію, сільське господарство, охорону здоров'я, енергетику та освіту. Зокрема, Arduino виявилася особливо ефективною у створенні прототипів автономних транспортних засобів (табл. 1.1).

З метою подальшого вдосконалення інструментарію для розробки мобільних роботів, компанія Arduino презентувала у 2022 році низку інновацій. Серед них особливої уваги заслуговує оновлена інтегрована середовище розробки Arduino IDE 2.0. Ця версія, що перейшла зі статусу бета-тестування до стабільного релізу, пропонує сучасний інтерфейс користувача та тісно інтегрується з хмарною платформою Arduino Cloud. Це дозволяє розробникам продовжувати роботу над проєктами з будь-якого пристрою, що має веб-браузер.

Крім того, Arduino представила новий інтерфейс командного рядка, що спрощує взаємодію з платформою та її хмарними сервісами. Для управління пристроями Інтернету речей, компанія розробила інструмент Arduino Cloud CLI,

який дозволяє автоматизувати процеси ініціалізації, оновлення та передачі даних.

Таблиця 1.1

### Приклади моделей роботизованих автомобілів на базі Arduino

Продукт	Дослідження
	Мініатюрна модель автономного автомобіля з використанням Arduino UNO та Open CV
	Розробка роботизованої системи уникнення перешкод на основі Arduino для безпілотного автомобіля
	Розробка автономного автомобіля на основі рукавичок Arduino
	Роботизований автомобіль із голосовим керуванням на базі Arduino
	Платформа Robotic-agent для вбудовування програмних агентів з використанням плат Raspberry Pi та Arduino

Філіп Торрон у 2011 році передбачив широке розповсюдження платформи Arduino, обґрунтувавши це такими факторами, як універсальність програмного забезпечення, багатий набір бібліотек, надійність драйверів, доступність та відкритість коду [36]. Ці аргументи й досі залишаються актуальними і пояснюють тривалу популярність Arduino серед розробників.

### 1.5. Постановка задачі

На основі аналізу існуючих досліджень в галузі автоматизованих систем управління рухомими платформами, ми ставимо за мету розробку багатофункціональної системи управління багатоколісною платформою, яка б

забезпечувала ефективне переміщення в просторі та взаємодію з навколишнім середовищем.

Вибір багатоколісної платформи як об'єкта дослідження обумовлений кількома факторами. По-перше, багатоколісні платформи демонструють більшу стійкість та маневреність порівняно з одно- та двоколісними аналогами. По-друге, їхня конструкція дозволяє реалізувати різноманітні алгоритми керування та забезпечити виконання широкого спектру завдань.

Створення повністю автономних роботів, здатних самостійно орієнтуватися в просторі та виконувати різноманітні завдання, є однією з найперспективніших галузей сучасної робототехніки. Однак, розробка таких систем є складним і ресурсомістким процесом.

Для прискорення досліджень в цій галузі пропонується розробити прототип самохідного робота (SDC), який би демонстрував основні функції автономної навігації. Такий прототип стане універсальною платформою для експериментів з різними алгоритмами, датчиками та конструктивними рішеннями.

Розроблений робот може знайти широке застосування не лише в домашньому господарстві, а й у промисловості, наприклад, для інспекції складних або небезпечних ділянок. Також, його можна використовувати як платформу для розробки більш складних роботів, здатних виконувати різноманітні завдання, від доставки товарів до проведення наукових досліджень.

Використання платформи Arduino для створення прототипу дозволить швидко та економічно перевірити працездатність розроблених алгоритмів і конструктивних рішень. Це, в свою чергу, створить міцну основу для подальших розробок і вдосконалень, таких як додавання нових датчиків, розширення функціональності та підвищення автономності робота.

### **Висновки до розділу**

У першому розділі роботи було проведено детальний аналіз сучасного стану робототехніки, зокрема, було досліджено розвиток систем навігації для мобільних роботів. З'ясовано, що хоча сучасні роботи здатні виконувати



широкий спектр завдань, питання автономної навігації залишається одним з ключових викликів у цій галузі.

Аналіз архітектури SDC (самокерованих автомобілів) показав, що їхня техніко-технологічна основа складається з великих складних систем, обладнаних численними датчиками для внутрішнього і зовнішнього моніторингу. Ці системи здатні генерувати величезні обсяги даних за короткі проміжки часу. Функціональні блоки побудовані на принципах обробки інформації, починаючи зі збору даних (зондування) і завершуючи управлінням транспортним засобом.

Технологічні та функціональні компоненти SDC утворюють основу системи навігації, яка забезпечує здатність автомобіля планувати маршрут і виконувати його без участі людини. Система вирішує дві ключові задачі: визначення оптимального маршруту до мети та уникнення перешкод на шляху. Для цього використовуються відповідні апаратні засоби, програмне забезпечення та навігаційний алгоритм.

## РОЗДІЛ 2. АНАЛІЗ СУЧАСНИХ МЕТОДІВ АВТОНОМНОЇ НАВІГАЦІЇ МОБІЛЬНИХ РОБОТІВ З АКЦЕНТОМ НА АЛГОРИТМАХ УХИЛЕННЯ ВІД ПЕРЕШКОД

### 2.1. Автономна навігація

Автономна навігація самокерованих транспортних засобів (SDC) та мобільних роботів є складним процесом, який вимагає інтеграції різноманітних апаратних та програмних компонентів. Здатність транспортного засобу самостійно планувати та виконувати маршрут без участі людини є ключовою для його автономності.

За визначенням українських вчених [41], навігація – це процес визначення оптимального маршруту руху з урахуванням статичних та динамічних перешкод. У робототехніці існують три основні типи навігаційних систем [9]: глобальна, локальна та персональна. Глобальна навігація використовується для визначення абсолютного положення об'єкта в просторі, тоді як локальна – для визначення відносного положення щодо заданої точки. Персональна навігація, в свою чергу, стосується позиціонування частин роботів та їх взаємодії з оточенням.

На рисунку 2.1 зображено структурну схему локальної навігаційної системи, розроблену Рудиком А. В., яка детально демонструє її компоненти та взаємозв'язки.

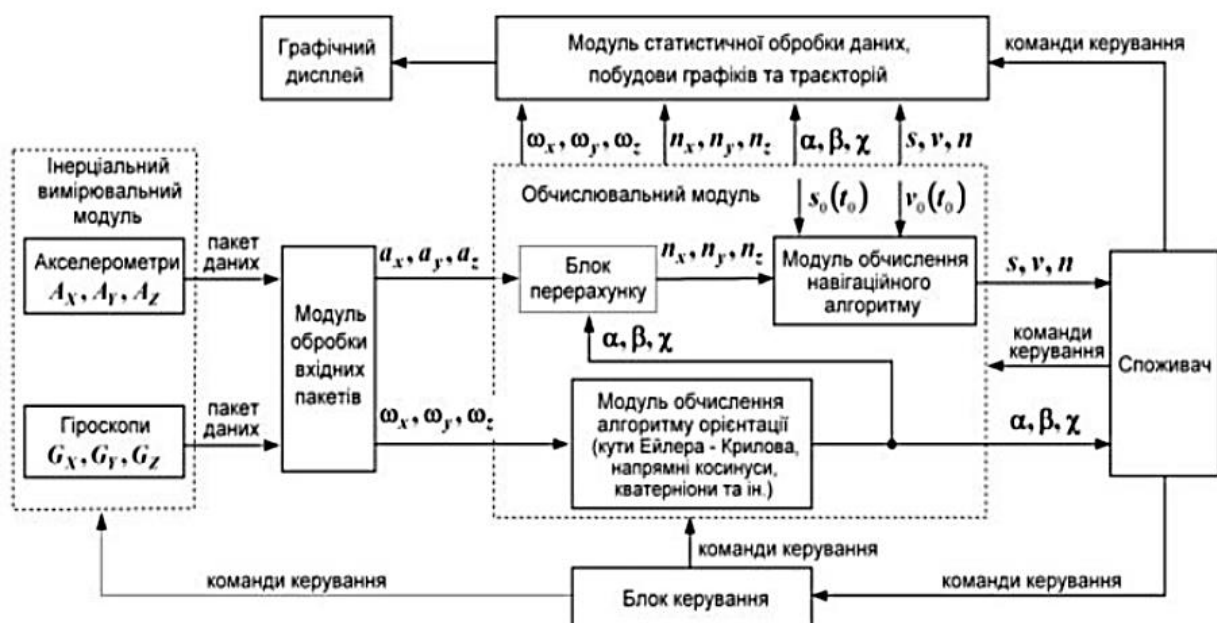


Рис. 2.1. Структурна схема локальної навігаційної системи [9, с. 341]

Системи навігації поділяються на пасивні та активні. Пасивні системи отримують інформацію про своє місцезнаходження від зовнішніх джерел, таких як супутники GPS. Активні системи, навпаки, самостійно визначають своє положення, наприклад, шляхом вимірювання відстані до відомих об'єктів або за допомогою лазерного сканування. Глобальні системи навігації, як правило, є пасивними, тоді як локальні можуть бути як пасивними, так і активними. Персональні системи навігації, які використовуються для позиціонування частин робота, зазвичай є активними.

Здатність автономно рухатися та орієнтуватися в просторі є основною характеристикою мобільних роботів. Як зазначається у роботі [42, с. 77], навігація дозволяє роботу орієнтуватися в навколишньому середовищі за допомогою даних від датчиків.

Автономна навігація включає в себе такі ключові компетенції:

- Побудова карти: створення цифрової моделі навколишнього середовища.
- Самолокалізація: визначення точного положення робота на створеній карті.
- Інтерпретація карти: розуміння інформації, що міститься в карті, для прийняття рішень.
- Планування шляху: розробка оптимального маршруту до мети з урахуванням перешкод та інших обмежень.

Аби ефективно виконувати поставлені завдання, автономний робот повинен вміти створювати карту навколишнього середовища, визначати своє місцезнаходження на цій карті та використовувати отриману інформацію для планування маршруту. Ці три основні компетенції – побудова карти, самолокалізація та інтерпретація карти – є невід'ємною частиною будь-якої системи автономної навігації. Для планування маршруту найчастіше використовуються методи, засновані на обчисленні шляху та навігації за орієнтирами.

Модульний підхід до сприйняття та обробки інформації є ключовим для забезпечення автономної навігації, як зазначається в дослідженні [43, с. 351]. Цей підхід дозволяє створювати гнучкі та масштабовані системи, здатні адаптуватися до змінних умов навколишнього середовища.

Автономна навігаційна система, що дозволяє транспортному засобу самостійно планувати та виконувати маршрут, є складною інженерною системою. Вона об'єднує в собі різноманітні технології, такі як інерційні навігаційні системи, супутникові системи позиціонування, радары, камери та ультразвукові датчики. Для забезпечення безпечної та ефективної навігації використовуються спеціальні алгоритми, які дозволяють обробляти дані від датчиків та приймати оптимальні рішення в реальному часі.

Як зазначено в роботі [44], система навігації робота може бути представлена у вигляді трирівневої ієрархічної структури, де кожен рівень відповідає за вирішення певних завдань. Нижній рівень відповідає за безпосереднє керування рухом робота, середній рівень займається плануванням траєкторії, а верхній рівень відповідає за загальну стратегію поведінки робота.

Одним з найважливіших завдань системи автономної навігації є ухилення від перешкод. Як зазначається в роботі [45, с. 89], системи контролю ухилення від перешкод є одними з найактуальніших напрямів дослідження в галузі автономної навігації.

Аналізуючи існуючі рішення в галузі автономної навігації, можна виділити три основні рівні: стратегічний, тактичний та виконавчий (див. [49]). Кожен рівень має свої специфічні завдання та використовує різні алгоритми. Одним з основних викликів при розробці систем автономної навігації є необхідність прийняття рішень в реальному часі в умовах невизначеності та обмежених обчислювальних ресурсів.

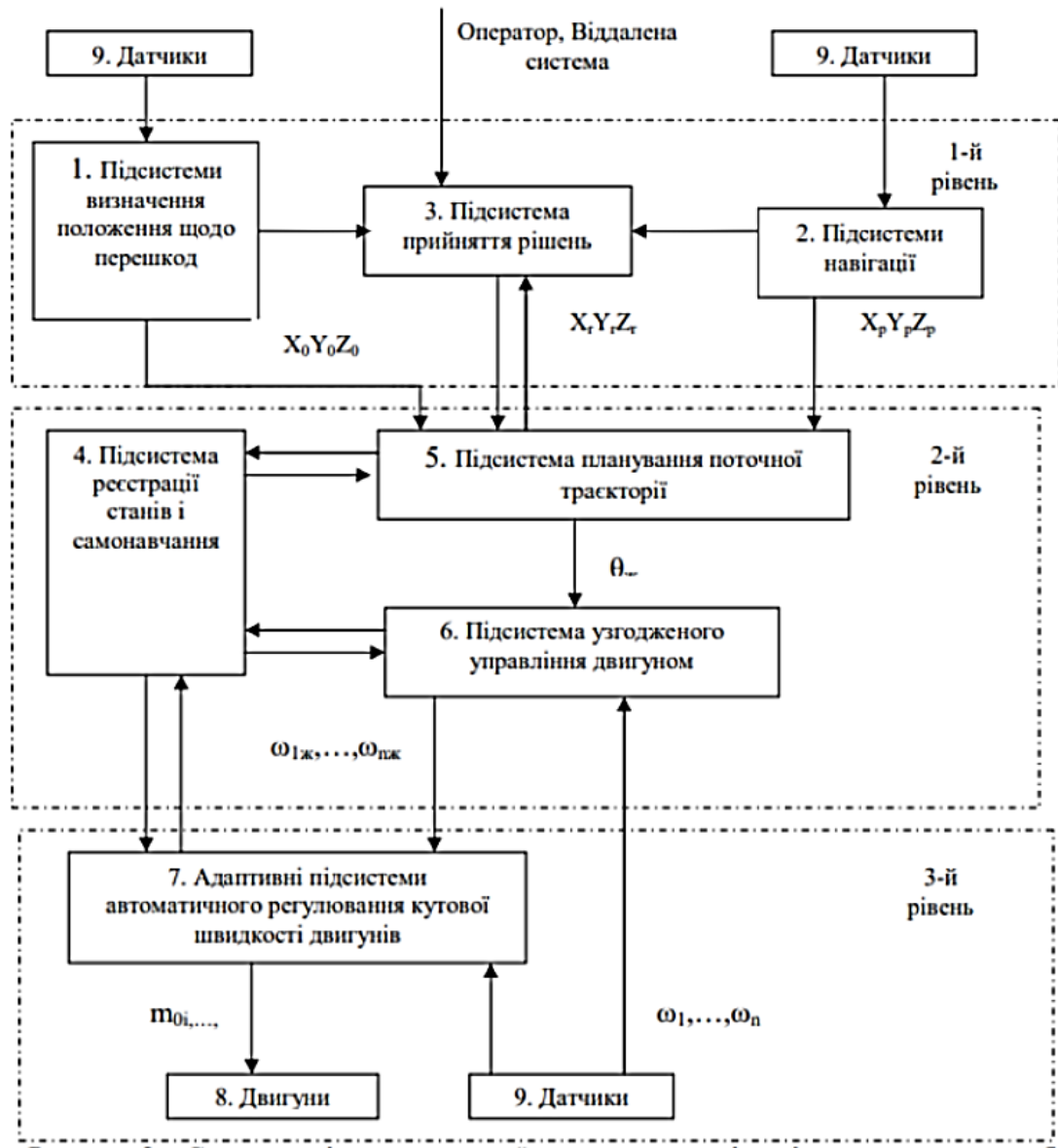


Рис. 2.2. Інтелектуальна система управління мобільним роботом [44]

Типовою архітектурою систем навігації для самокерованих транспортних засобів (SDC) є трирівнева модель, яка включає в себе такі основні функціональні блоки: сприйняття, планування та виконання. Такий підхід дозволяє структурувати систему та розподілити відповідальність між окремими компонентами.

Трирівнева архітектура систем навігації для мобільних роботів, описана в роботі [50], базується на трьох основних методах керування:

- Реактивному керуванні: це простий і швидкий метод, при якому дії робота визначаються безпосередньо результатами зчитування даних з датчиків. Тобто, робот реагує на зміни в навколишньому середовищі миттєво, без складної обробки інформації.

- Методах виконання шаблонних дій: цей метод передбачає виконання роботом заздалегідь запрограмованих послідовностей дій, які можуть бути змінені лише в межах обмежених параметрів. Такий підхід ефективний для виконання простих завдань у статичному середовищі.
- Складних обчислювальних методах: цей метод передбачає використання складних алгоритмів для планування оптимальних дій робота в динамічному середовищі. Такі методи дозволяють роботу приймати складні рішення, враховуючи велику кількість факторів, але вимагають значних обчислювальних ресурсів.

Однак, традиційні трирівневі архітектури мають ряд обмежень, пов'язаних з обробкою сенсорної інформації, навчанням та побудовою карт середовища [51]. Зокрема, вони часто орієнтовані на статичні та добре структуровані середовища, що обмежує їх застосування в реальних умовах.

О.П. Адамів пропонує більш досконалу архітектуру системи навігації, яка враховує динаміку навколишнього середовища та особливості задач, що вирішуються роботом [50]. Детальний опис цієї архітектури наведено на рисунку 2.3.

Японські вчені [52] виділяють такі основні етапи процесу навігації SDC: зчитування інформації з датчиків, розпізнавання об'єктів у навколишньому середовищі, прийняття рішень щодо подальших дій та виконання відповідних маневрів (рис. 2.4).

Процес навігації автономного транспортного засобу ґрунтується на зборі даних з різних датчиків, таких як лідари, радары та камери. Зібрана інформація обробляється та об'єднується в єдину цифрову карту навколишнього світу. Ця карта постійно оновлюється і включає в себе не лише статичні об'єкти, такі як будівлі та дороги, а й динамічні об'єкти, наприклад, інші транспортні засоби та пішоходів. За допомогою алгоритмів машинного навчання транспортний засіб може розпізнавати різні об'єкти та прогнозувати їхню поведінку. На основі цієї інформації транспортний засіб приймає рішення про свої дії, такі як вибір

маршруту, прискорення, гальмування або зміна смуги руху. Таким чином, автономний транспортний засіб постійно взаємодіє зі своїм оточенням, адаптуючись до змінних умов і забезпечуючи безпечний та ефективний рух.

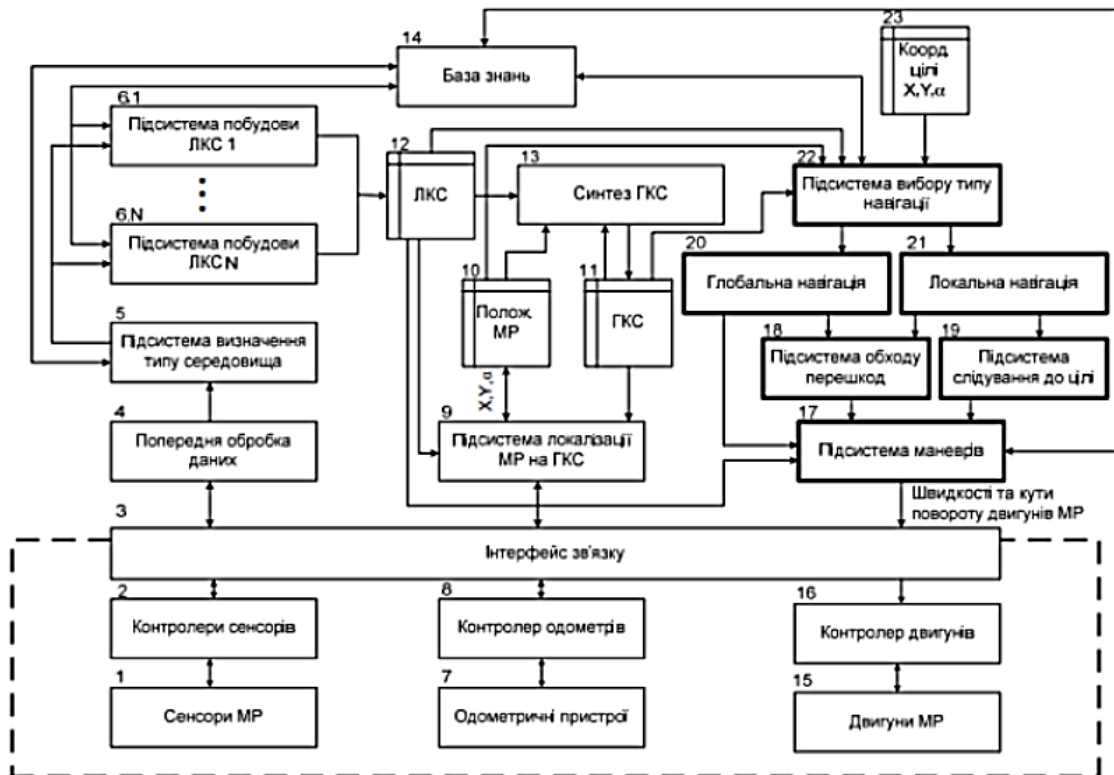


Рис. 2.3. Архітектура системи навігації у середовищі з перешкодами [50]

Процес керування автономним транспортним засобом (АТЗ) є багаторівневою системою, що включає в себе планування траєкторій, оцінку їхньої безпеки та виконання. Планування руху АТЗ є централізованим процесом, який координує всі аспекти автономного керування. Моделювання можливих траєкторій здійснюється з урахуванням динамічних характеристик транспортного засобу та оцінки потенційних ризиків [50]. Оптимальна траєкторія обирається на основі комплексного аналізу різних критеріїв, таких як безпека, ефективність та комфорт. Реалізація обраної траєкторії здійснюється шляхом генерації команд для виконавчих механізмів (актуаторів).

Традиційні навігаційні системи, що використовуються в АТЗ, складаються з модулів, призначених для виконання таких завдань як локалізація, картографування та планування руху. Ці системи відіграють ключову роль у забезпеченні безпеки та ефективності руху АТЗ [50]. Сенсорні системи, такі як

лідари, радари та камери, забезпечують необхідну інформацію про навколишнє середовище для виконання цих завдань.

Ключові функції навігаційних систем:

- Локалізація: визначення точного географічного положення АТЗ в реальному часі.
- Картографування: створення детальних карт навколишнього середовища для навігації.
- Планування маршруту: обчислення оптимального маршруту до заданої точки з урахуванням дорожніх умов та обмежень.
- Виявлення об'єктів: розпізнавання та відстеження інших транспортних засобів, пішоходів та статичних об'єктів.
- Уникнення перешкод: планування маневрів для безпечного об'їзду перешкод.

Ефективне функціонування автономного транспортного засобу безпосередньо залежить від надійності та точності його навігаційної системи. Завдяки постійному розвитку технологій в області сенсорики, обробки даних та алгоритмів планування, автономні транспортні засоби стають все більш досконалими і безпечними.

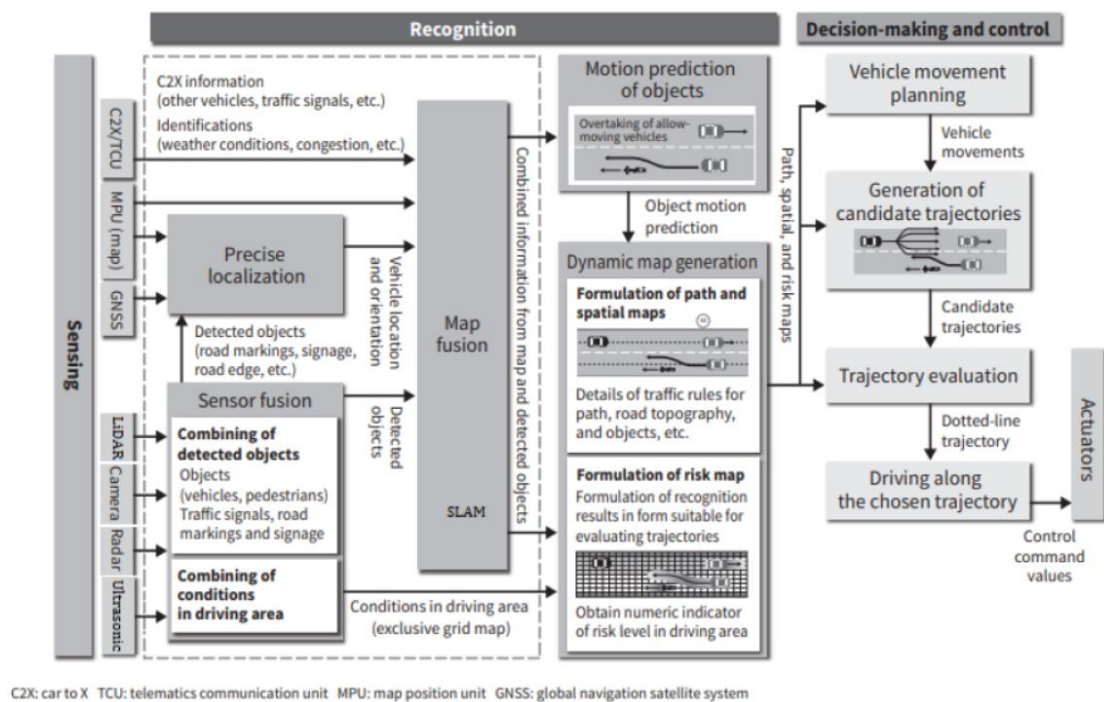


Рис. 2.4. Блок-схема системи навігації SDC [52]



## 2.2 Алгоритми навігації

Проблема уникнення перешкод в мобільній робототехніці є однією з найважливіших і активно досліджуваних тем. Для її вирішення розроблено широкий спектр алгоритмів, які можна класифікувати за принципом їх роботи [53]. Серед найпоширеніших підходів виділяють:

- детерміновані алгоритми. Ці алгоритми базуються на чітко визначених правилах і процедурах, які дозволяють однозначно визначити дії робота в кожній конкретній ситуації.
- недетерміновані (стохастичні) алгоритми. На відміну від детермінованих, ці алгоритми передбачають елемент випадковості. Вони часто використовуються для моделювання невизначеностей в середовищі та для пошуку оптимальних рішень в складних ситуаціях.
- еволюційні алгоритми. Ці алгоритми натхненні природною еволюцією і використовують такі механізми, як мутація і природний відбір для пошуку оптимальних рішень. Вони часто є гібридами детермінованих і стохастичних алгоритмів.

Рисунок 2.5 наочно ілюструє цю класифікацію, демонструючи різноманітність алгоритмів, які використовуються в сучасних системах навігації мобільних роботів. Вибір того чи іншого алгоритму уникнення перешкод залежить від конкретних вимог до системи, таких як точність, швидкість, адаптивність та обчислювальні ресурси. Подальші дослідження в цій галузі спрямовані на розробку більш ефективних і універсальних алгоритмів, здатних забезпечити надійну навігацію мобільних роботів в складних і динамічних середовищах.

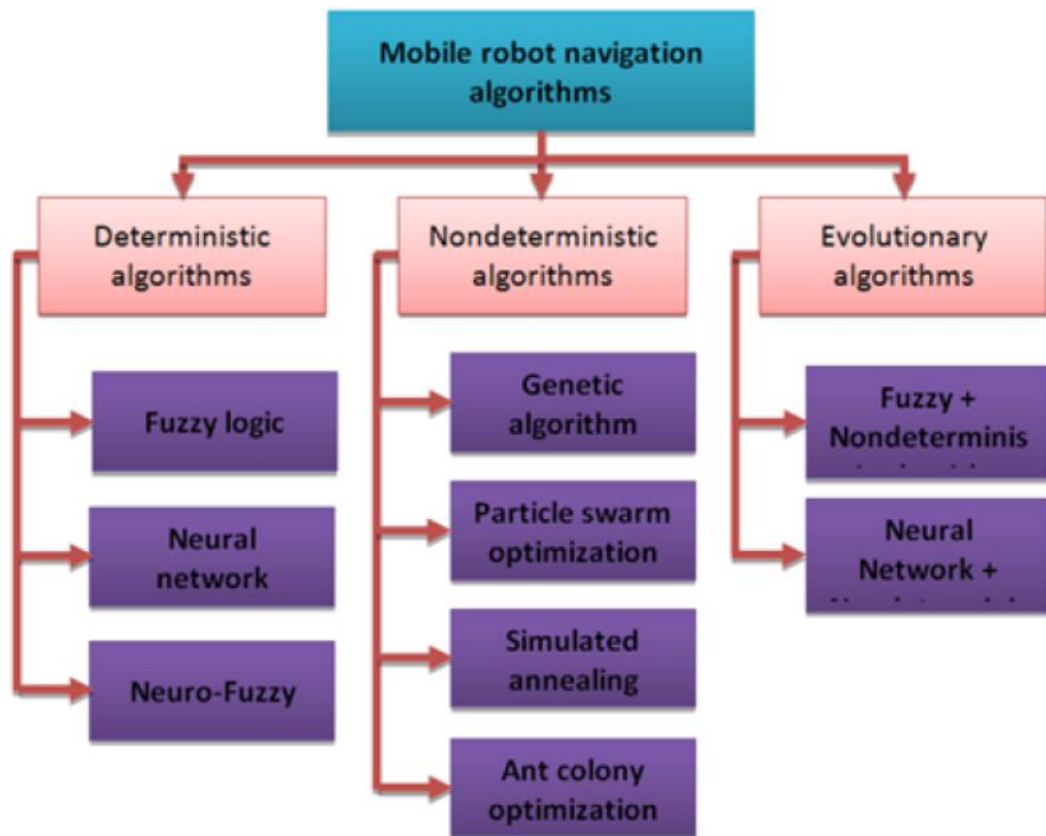


Рис. 2.5. Алгоритми навігації мобільних роботів [53]

Проблема навігації мобільних роботів у складному середовищі є актуальною темою сучасних досліджень у галузі робототехніки. Одним із перспективних підходів до її вирішення є використання нечіткої логіки, запропонованої Заде.

За останні роки було розроблено численні алгоритми на основі нечіткої логіки для забезпечення ефективної навігації мобільних роботів у статичних та динамічних середовищах, що ускладнені різноманітними перешкодами. Дослідники успішно застосовували нечіткі контролери для вирішення задач позиціонування та орієнтації роботів у просторі.

Існуючі наукові роботи демонструють широкі можливості використання нечіткої логіки в робототехніці. Зокрема:

- Розроблено інтелектуальні нечіткі контролери, які дозволяють мобільним роботам ефективно орієнтуватися в невідомих та змінних середовищах.

- Запропоновано оптимальні нечіткі контролери Такагі-Сугено, налаштовані за допомогою градієнтних методів, для точного управління рухом роботів та уникнення зіткнень.

- Створено нечіткі архітектури на основі поведінки, які дозволяють роботам реагувати на різноманітні ситуації, що виникають під час руху, такі як досягнення мети, уникнення перешкод, відстеження об'єктів тощо.

Деякі дослідники [54] поєднали нечітку логіку з генетичними алгоритмами для вирішення задач планування маршруту та управління рухом автономних мобільних роботів. У своїй роботі автори:

- розробили детальну модель мобільного робота, включаючи кінематичні рівняння та конфігурацію датчиків;
- запропонували алгоритми для виявлення перешкод та прийняття рішень щодо обходу;
- створили блок-схему навігації робота в середовищі з множинними перешкодами.

Таким чином, нечітка логіка є потужним інструментом для розробки інтелектуальних систем управління мобільними роботами, які здатні ефективно працювати в складних та непередбачуваних умовах.

### **2.3 Алгоритми планування маршруту**

Проблема планування маршруту для автономних транспортних засобів (АТЗ) є однією з ключових задач сучасної робототехніки. Вона полягає в розробці алгоритмів, які дозволяють АТЗ ефективно рухатися від початкової точки до заданої мети в динамічному середовищі, уникаючи при цьому зіткнень з перешкодами.

Існуючі алгоритми планування маршруту для АТЗ можна класифікувати за різними критеріями, такими як складність обчислень, необхідність датчикової інформації, точність та швидкість реагування. Класифікацію методів планування шляху МР зображено на рис. 2.6.

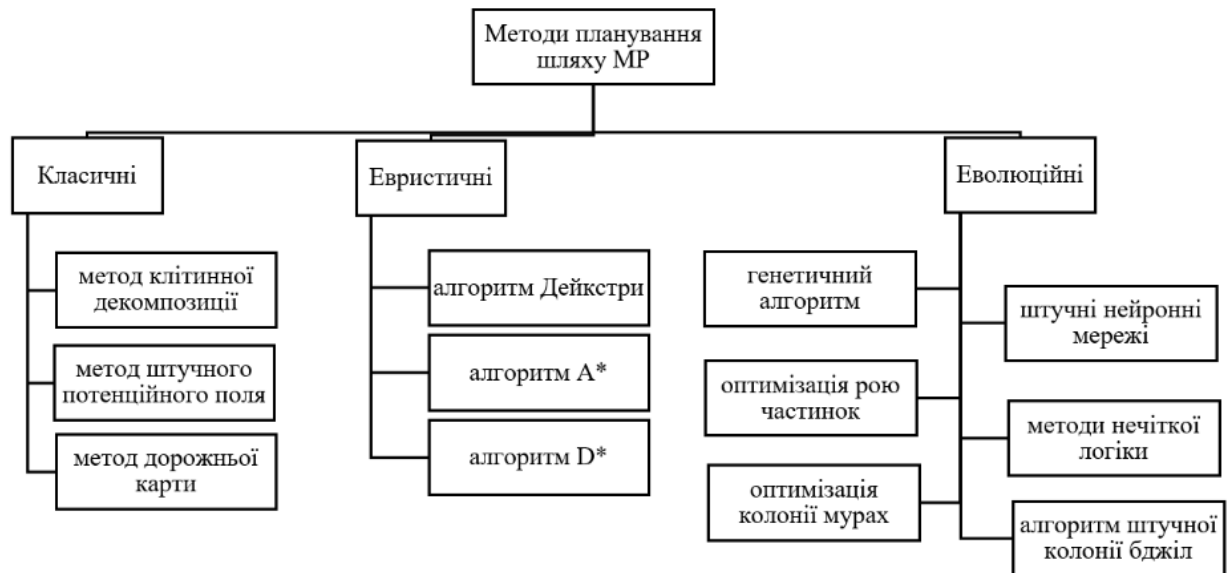


Рис. 2.6. Класифікація методів планування шляху МР

Серед них можна виділити:

Прості алгоритми уникнення перешкод: засновані на реактивних діях, таких як зупинка або зміна напрямку руху при виявленні перешкоди.

Алгоритми, що використовують датчики відстані: дозволяють оцінити відстань до перешкод та прийняти рішення про зміну траєкторії.

Потенціальні польові алгоритми: створюють потенціальне поле, яке направляє рух АТЗ до мети і відштовхує від перешкод.

Алгоритми на основі графів: представляють простір конфігурацій АТЗ у вигляді графу та шукають оптимальний шлях за допомогою відомих алгоритмів пошуку (Дейкстри, Флойда-Воршала).

Евристичні алгоритми: використовують евристичні функції для прискорення пошуку оптимального шляху (наприклад, A\*).

Гібридні алгоритми: поєднують в собі елементи різних підходів для досягнення кращих результатів.

Окрему увагу заслуговують алгоритми сімейства BUG, розроблені для вирішення задачі планування маршруту в динамічному середовищі з невідомими перешкодами. Ці алгоритми характеризуються простотою реалізації та невисокими обчислювальними витратами.

Класичні алгоритми BUG1 та BUG2 передбачають наступну послідовність дій:

- прямий рух до мети: АТЗ рухається безпосередньо до мети до зустрічі з перешкодою.
- обхід перешкоди: при зіткненні з перешкодою АТЗ починає рухатися вздовж її контуру до найближчої точки, з якої можна продовжити рух до мети.
- повторення: після обходу перешкоди АТЗ продовжує рух до мети, повторюючи описані вище кроки при зустрічі з новими перешкодами.

Незважаючи на свою простоту, алгоритми BUG гарантують досягнення мети за умови її доступності.

Робота [55] систематизує ці алгоритми, поділяючи їх на кілька категорій за принципом дії. Дослідник звертає увагу на різноманітність підходів, від простих реактивних алгоритмів до складних евристичних і гібридних методів, кожен з яких має свої переваги та недоліки.

Особливе місце серед розглянутих алгоритмів займають алгоритми сімейства BUG, детально описані в роботі [56]. Ці алгоритми призначені для вирішення задачі навігації в динамічних середовищах з невідомими перешкодами. Їх популярність зумовлена простотою реалізації та відносно низькими вимогами до обчислювальних ресурсів. Багато модифікацій алгоритмів BUG, таких як Alg1, Al2, DistBug та інші, дозволяють адаптувати їх до різних типів середовищ і завдань.

Класичний алгоритм BUG1 передбачає послідовне виконання наступних кроків: прямий рух до мети до зустрічі з перешкодою, обхід перешкоди вздовж її контуру до найближчої точки, з якої можна продовжити рух до мети, і повторення цих кроків до досягнення мети. Незважаючи на свою простоту, алгоритм гарантує досягнення мети за умови її доступності. Однак, ефективність алгоритму BUG1 може бути низькою в складних середовищах з великою кількістю перешкод.

Таким чином, алгоритми сімейства BUG представляють собою ефективний інструмент для вирішення задач навігації в динамічних середовищах. Їх простота і надійність роблять їх привабливими для використання в системах керування автономними роботами. Однак, для більш складних задач, таких як планування

маршруту в міських умовах, можуть знадобитися більш складні і ефективні алгоритми.

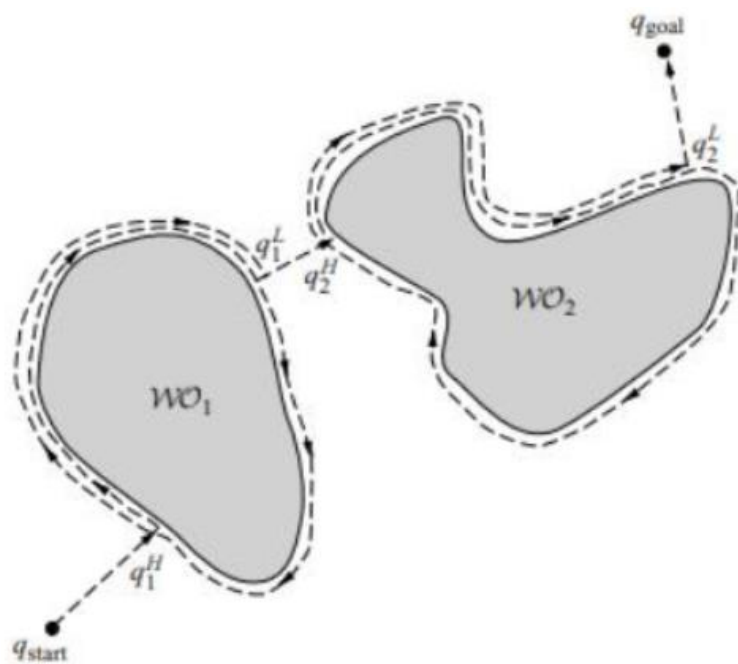


Рис. 2.7. Алгоритм Bug1 успішно знаходить мету

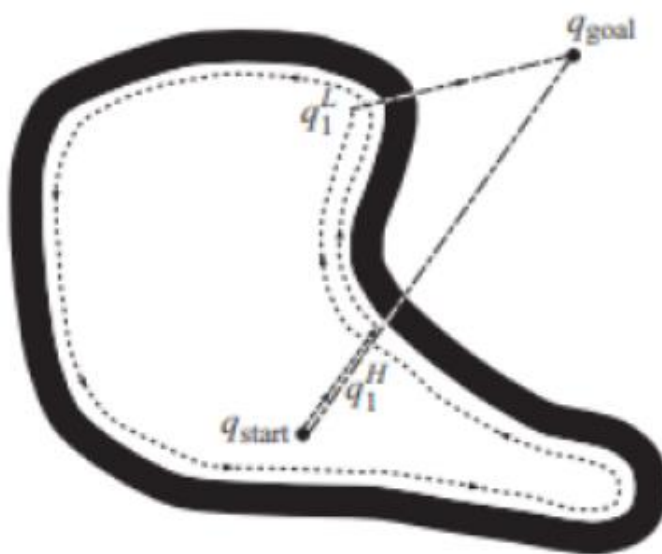


Рис. 2.8. Алгоритм Bug1 повідомляє, що мета недосяжна

**Algorithm 1 Bug1 Algorithm****Input:** A point robot with a tactile sensor**Output:** A path to the  $q_{\text{goal}}$  or a conclusion no such path exists

```
1: while Forever do
2:   repeat
3:     From  $q_{i-1}^L$ , move toward  $q_{\text{goal}}$ .
4:   until  $q_{\text{goal}}$  is reached or an obstacle is encountered at  $q_i^H$ .
5:   if Goal is reached then
6:     Exit.
7:   end if
8:   repeat
9:     Follow the obstacle boundary.
10:  until  $q_{\text{goal}}$  is reached or  $q_i^H$  is re-encountered.
11:  Determine the point  $q_i^L$  on the perimeter that has the shortest distance to the goal.
12:  Go to  $q_i^L$ .
13:  if the robot were to move toward the goal then
14:    Conclude  $q_{\text{goal}}$  is not reachable and exit.
15:  end if
16: end while
```

Рис. 2.9. Алгоритм Bug1

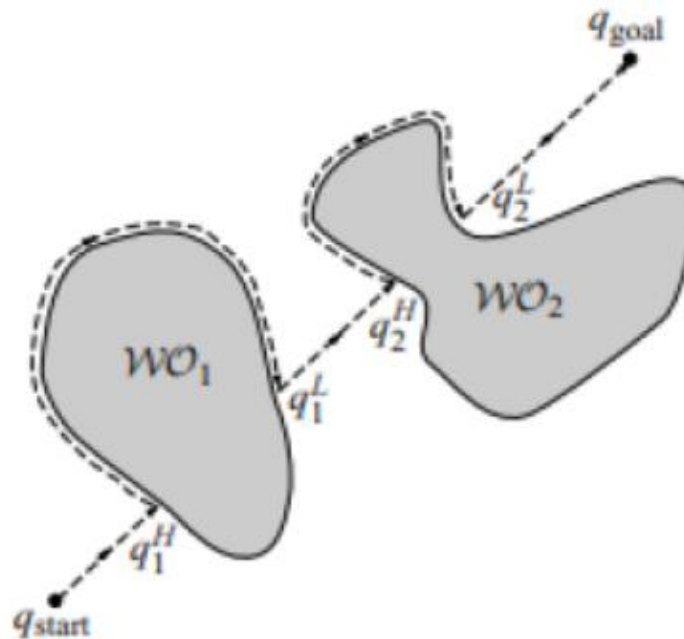


Рис. 2.10. Алгоритм Bug2 успішно знаходить мету

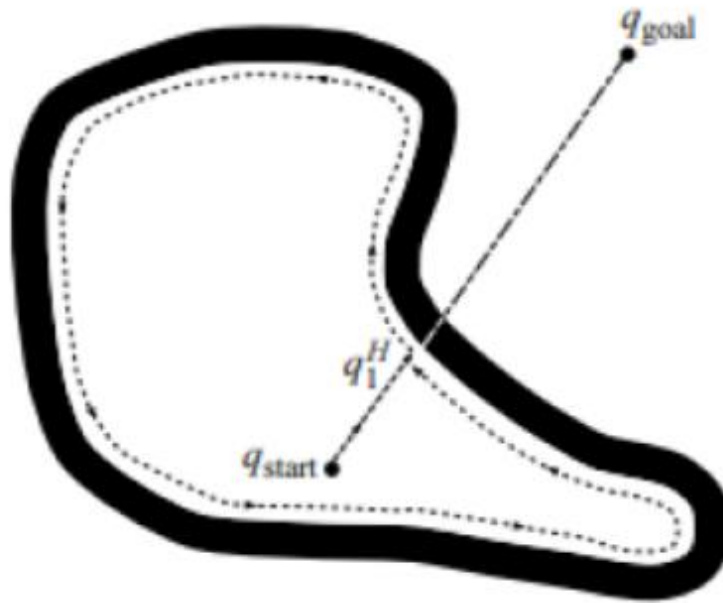


Рис. 2.11. Алгоритм Bug 2 повідомляє, що мета недосяжна

---

**Algorithm 2** Bug2 Algorithm

---

**Input:** A point robot with a tactile sensor

**Output:** A path to  $q_{\text{goal}}$  or a conclusion no such path exists

---

```

1: while True do
2:   repeat
3:     From  $q_{i-1}^L$ , move toward  $q_{\text{goal}}$  along  $m$ -line.
4:   until
      $q_{\text{goal}}$  is reached or
     an obstacle is encountered at hit point  $q_i^H$ .
5:   Turn left (or right).
6:   repeat
7:     Follow boundary
8:   until
      $q_{\text{goal}}$  is reached or
      $q_i^H$  is re-encountered or
      $m$ -line is re-encountered at a point  $m$  such that
9:      $m \neq q_i^H$  (robot did not reach the hit point),
10:     $d(m, q_{\text{goal}}) < d(m, q_i^H)$  (robot is closer), and
11:    If robot moves toward goal, it would not hit the obstacle
12:    Let  $q_{i+1}^L = m$ 
13:    Increment  $i$ 
14:  end while

```

Рис. 2.12. Алгоритм Bug 2



Алгоритми сімейства BUG, незважаючи на свою простоту, демонструють різноманітні підходи до вирішення задачі навігації в невідомих середовищах. Кожен з алгоритмів цього сімейства має свої особливості та компроміси між ефективністю та обчислювальною складністю.

Алгоритми BUG1 та BUG2 відрізняються стратегією обходу перешкод. BUG1 гарантує відсутність зациклювання, але вимагає повного обходу перешкоди для знаходження точки виходу. BUG2, хоча й може потрапити в локальні мінімуми, демонструє більш інтуїтивну поведінку. Алгоритм Dist-Bug вдосконалює BUG1, використовуючи інформацію про відстань до мети для вибору оптимальної точки виходу з перешкоди.

Алгоритм DH-Bug пропонує гібридний підхід, поєднуючи попереднє планування на основі неповних даних з реактивним управлінням. Такий підхід дозволяє скоротити довжину шляху та обходити рухомі перешкоди. Алгоритм TangentBug використовує додаткову інформацію від далекоміра для побудови графу локальних дотичних, що дозволяє приймати більш обґрунтовані рішення про напрямки руху.

Одним з найбільш ефективних алгоритмів сімейства BUG є CBUG. Він враховує розміри перешкод і дозволяє оптимізувати довжину шляху. Цей алгоритм демонструє високу ефективність у складних середовищах.

Таким чином, алгоритми сімейства BUG представляють собою універсальний інструмент для вирішення задач навігації в невідомих середовищах. Вибір конкретного алгоритму залежить від вимог до точності, швидкості та обчислювальних ресурсів.

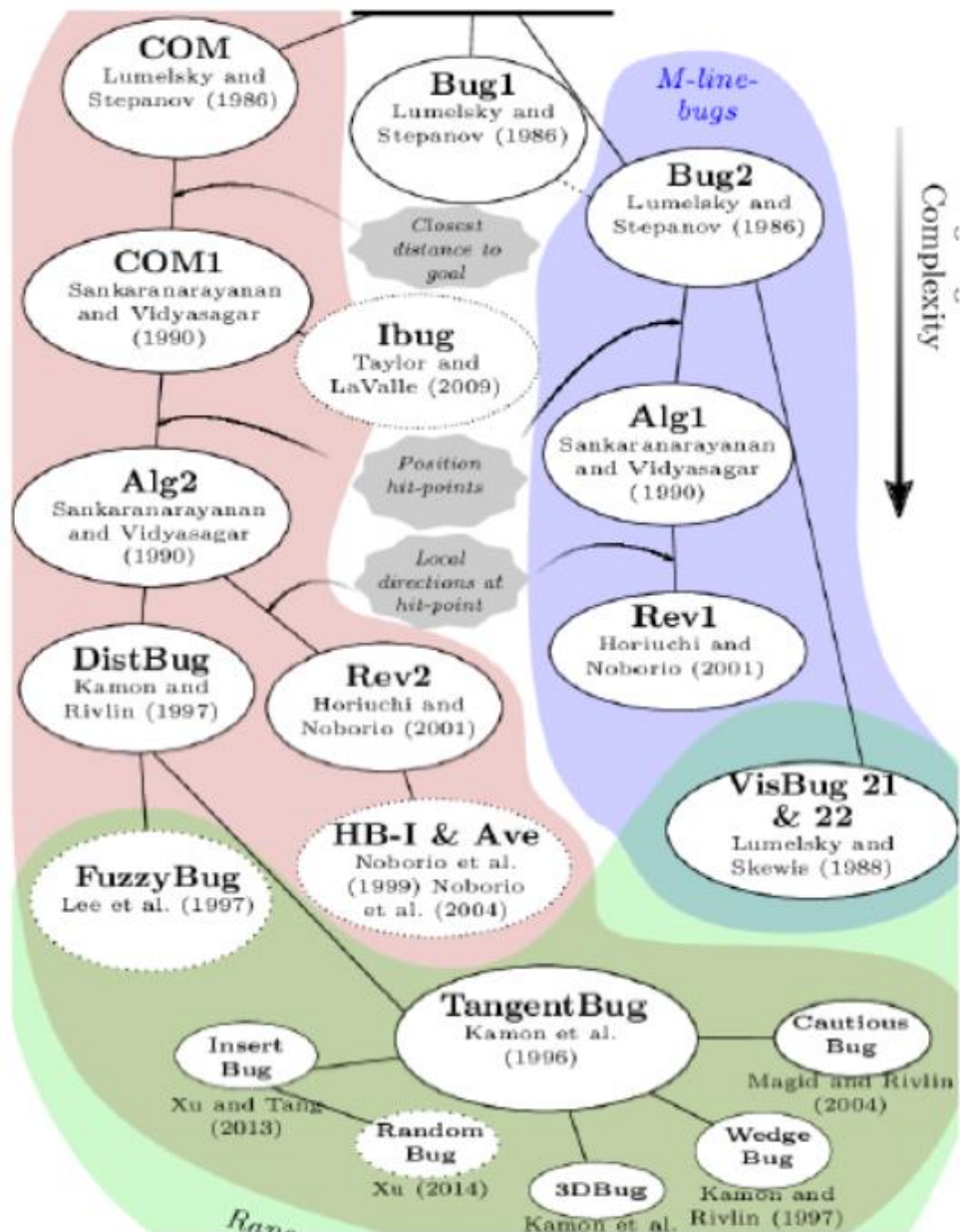


Рис. 2.13. Всі Bug-алгоритми [56]

З метою вибору оптимального алгоритму планування маршруту в роботі було проведено всебічне дослідження різних підходів, включаючи алгоритми сімейства Bug та інші методи. Результати порівняльного аналізу наведено в таблиці 2.1.

**Порівняння алгоритмів планування маршруту мобільних роботів, здатних  
оминати перешкоди**

Алгоритм		Метод на основі датчика зору	"Follow the Gape" метод	Алгоритм гібридної навігації з прохідними стежками	Новий алгоритм гібридної навігації (NHNA)
Реалізація	необхідне обладнання	камера, гідроакустичний датчик(сонар), процесор, beagle board, ноутбук	ультразвукові, лідарні та датчики, оптичних швидкостей камери, pirxi811108rt процесор rxi7954r fpga	лазерні та гідроакустичні датчики, роботи pioneer 3-at	лазерний датчик, мікропроцесор
	необхідні параметри	положення перешкод, кут і відстань	відстань і кут перешкоди	попередня інформація	попередня інформація
Продуктивність	ефективність	висока, розрахунки реальні і точні (залежить від обладнання)	висока, простий в налаштуванні, завжди вибирає найкоротший шлях, здатний уникнути симетричних перешкод	середня, генерує найкоротший шлях, але не обмежує відхилення від шляху	середній і більш ефективний, ніж hybrid, використовує dh-bug алгоритм
	конвергенція	може чи ні (залежить від характеру алгоритму)	немає (у тупиковому випадку, u-образних перешкод)	є, але в деяких сценаріях робот може зупинитися перед перешкодою	в основному сходиться, за винятком деяких сценаріїв, таких як тупик
	часова складність	залежить від роздільної здатності камери та додатка; займає більше часу для розрахунків	менше часу, оскільки рішення приймаються на основі поточного сприйняття	потребує більше часу для створення прикладу шляху (потрібен пошук a*) хвилини або секунди	витрата більше часу, оскільки пошук «a*» необхідний для створення шляху
Зауваження		не кращий варіант мініавто з мікроконтролером, вимагає ноутбука або процесора і конкретного застосування, як matlab	не оминає u-образної перешкоди	вимагає складних розрахунків вимагає більше часу	складний розрахунок, витрачається більше часу

Алгоритм		Алгоритм Bug	Метод потенційного поля (VFF)	Гістограма векторного поля (VFH)
Реалізація	необхідне обладнання	датчики відстані (іч, гідролокатор), мікроконтролер	датчики відстані (іч, гідролокатор), мікроконтролер	сонарний датчик, процесор, великий обсяг пам'яті
	необхідні параметри	поточне та цільове положення	відстань цілі та перешкоди	відстань перешкоди
Продуктивність	ефективність	не дуже висока, може віддаляти робота від місця призначення	низька, розрахунок не точний, обмеження не враховуються	низька, розрахунок може бути точним, але споживає більше ресурсів, таких як пам'ять, процесор і енергія
	конвергенція	є, але потрібно більше часу	немає (у випадку u-образних і симетричних перешкод)	немає (у випадку u-образних і симетричних перешкод)
	часова складність	завжди рухається в одному напрямку, аби уникнути перешкоди, що збільшує часову складність	витрачається менше часу, завдяки вибору коротшого шляху	потрібно більше часу для створення 2d сітки і перетворення з 2d в 1d полярної гістограми
Зауваження		локальних мінімумів не відбувається; обертається найдовший шлях	локальний мінімум можливий	навантажує мікроконтролер, оскільки потрібні складні обчислення

## Висновки розділу

Існуючі алгоритми навігації для самокерованих транспортних засобів (SDC) є результатом еволюції технологій та відповідають певним наборам вимог та обмежень. Однак, постійний розвиток технологій та виникнення нових завдань для SDC стимулюють пошук більш ефективних та універсальних методів навігації.

Дослідження алгоритмів сімейства Bug, а також інших методів обходу перешкод, дозволяє виявити їхні сильні та слабкі сторони. Наприклад, алгоритм

Dist-Bug демонструє кращу ефективність порівняно з методом потенційних полів, але може відводити робота від мети в певних ситуаціях. З іншого боку, метод потенційних полів легкий в реалізації, але схильний до локальних мінімумів.

Алгоритми Bug1 та Bug2, незважаючи на свою простоту, мають ряд обмежень, пов'язаних з геометрією середовища та можливостями датчиків. Зокрема, вони передбачають, що перешкоди мають просту геометричну форму, датчики надають точну інформацію про положення робота, а робот сам по собі є точковою масою. Ці обмеження можуть бути суттєвими для реальних застосувань.

Багато модифікацій алгоритмів Bug спрямовані на подолання цих обмежень. Так, алгоритм TangentBug використовує додаткові датчики для отримання більш детальної інформації про середовище, а алгоритм Dist-Bug оптимізує вибір напрямку руху. Однак, кожна модифікація має свої переваги та недоліки.

Аналіз різних алгоритмів обходу перешкод дозволяє зробити висновок про те, що вибір оптимального алгоритму залежить від конкретних умов задачі, таких як геометрія середовища, тип датчиків, вимоги до точності та швидкості навігації. Результати цього дослідження можуть бути використані для розробки нових алгоритмів, які будуть більш ефективними та універсальними.



## РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА СТВОРЕННЯ ПРОГРАМОВАНОГО РОБОТА НА БАЗІ ARDUINO

### 3.1. Апаратна база мобільного робота на основі Arduino Uno

Ключовим елементом розробленого мобільного робота є мікроконтролер Arduino Uno, який виконує функції центрального процесора. Завдяки своїм характеристикам, він забезпечує ефективне керування рухом робота, збір даних з датчиків та реалізацію алгоритмів управління (рис. 3.1 – 3.2).



Рис. 3.1. Плата Arduino Uno

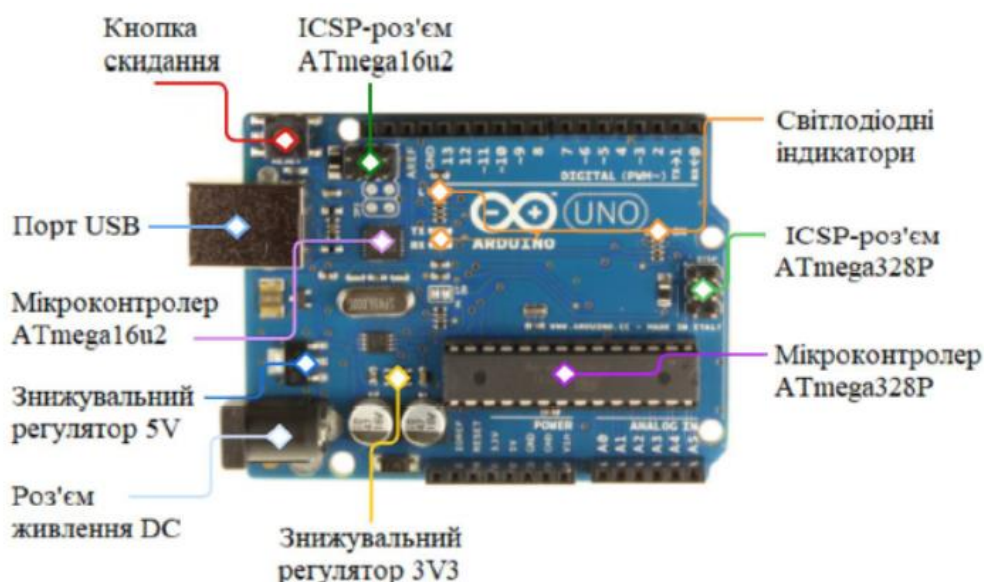


Рис. 3.2. Складові плати Arduino Uno

Плата Arduino Uno є однією з найпопулярніших платформ для швидкого прототипування електронних пристроїв завдяки своїй доступності та простоті

використання. Її технічні характеристики роблять її універсальним інструментом для широкого кола завдань.

Вхідно-вихідні можливості:

- Arduino Uno надає 14 цифрових пінів, які можуть виконувати функції як входів, так і виходів. Шість з цих пінів підтримують режим ШІМ, що дозволяє регулювати ширину імпульсу і, відповідно, яскравість підсвічування світлодіодів, швидкість обертання моторів тощо.
- Плата обладнана 6 аналоговими входами, які дозволяють зчитувати аналогові сигнали з датчиків, таких як потенціометри, фоторезистори, датчики температури тощо. Це дозволяє створювати більш складні та інтерактивні системи.

Пам'ять:

- Для зберігання програмного коду використовується 32 кБ флеш-пам'яті. Цього обсягу достатньо для більшості проєктів.
- Оперативна пам'ять (SRAM): 2 кБ оперативної пам'яті використовуються для зберігання змінних під час виконання програми.
- 1 кБ енергонезалежної пам'яті EEPROM дозволяє зберігати дані, які повинні зберігатися навіть після відключення живлення.

Інтерфейси:

- За допомогою USB-порту плату підключають до комп'ютера для програмування та обміну даними.
- Роз'єм ICSP призначений для програмування мікроконтролера в обхід USB.

Живлення:

- Arduino Uno може працювати від зовнішнього джерела живлення з напругою від 7 до 12 вольт або від USB-порту комп'ютера.
- Вбудований регулятор напруги забезпечує стабільну напругу 5 вольт для живлення всіх компонентів плати.

Фізичні характеристики:

- Компактні розміри плати (68.6 мм x 53.4 мм) дозволяють легко інтегрувати її в різноманітні конструкції.

Індикація:

- Плата оснащена світлодіодами, які індикують живлення, процес завантаження програми та стан цифрового виходу 13.

Основою нашого робота є мікроконтролерна плата Arduino Uno, яка служить "мозком" системи. Цей потужний мікроконтролер забезпечує обчислювальні ресурси та набір вводів-виводів, необхідних для управління різноманітними компонентами робота. Завдяки своїй гнучкості, Arduino Uno дозволяє програмувати цифрові та аналогові входи/виходи для керування двигунами, сервоприводами, зчитування даних з ультразвукових сенсорів та виконання інших завдань. Для розширення можливостей керування двигунами ми використовуємо щит Arduino Motor Shield L293D, який дозволяє підключати та керувати декількома двигунами одночасно.

Компонент – Arduino Motor Shield L293D (рис. 3.3) має наступні характеристики.

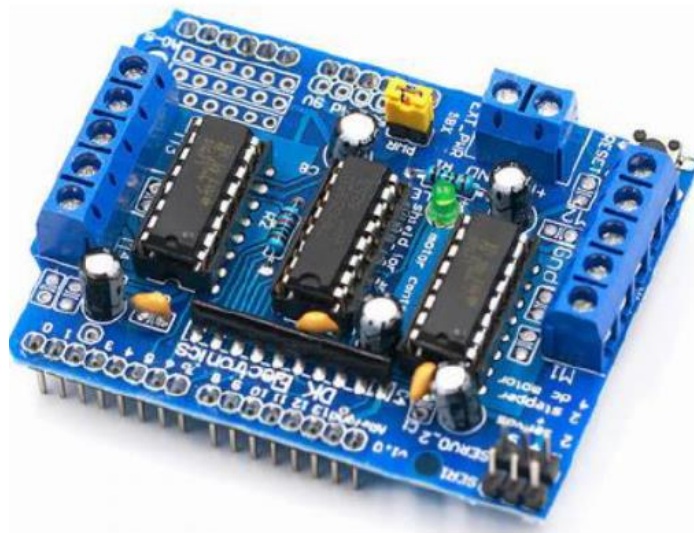


Рис. 3.3. Шилд драйверів двигунів на L293D

L293D — це інтегрована мікросхема, яка виступає двоканальним драйвером моторів, здатним забезпечити управління двигунами постійного струму зі струмом до 600 мА на кожному каналі. Вона одночасно реалізує підтримку чотирьох двигунів постійного струму, що робить її придатною для



широкого спектру програм у робототехніці та автоматизації. Робоча напруга пристрою варіюється в межах 4,5–36 В, що дозволяє використовувати її з вільними джерелами живлення та забезпечує гнучкість у проєкті.

Ключовими характеристиками L293D є наявність двох роз'ємів для підключення сервоприводів із напругою живлення 5 В, які синхронізуються з високороздільним таймером платформи Arduino. Додатково мікросхема забезпечена світлодіодними індикаторами для візуального відображення стану каналів, що значно покращує діагностику та контроль роботи [2]. Важливою функцією є вбудований захист від перевантажень, який запобігає пошкодженню мікросхем та підключених моторів у разі надзвичайного навантаження. Також драйвер підтримує швидкість та напрямок обертання двигунів за рахунок використання широтно-імпульсної модуляції (ШИМ), що дозволяє досягти високої точності.

Шилд на базі L293D для Arduino забезпечує простоту використання за допомогою інтуїтивно зрозумілого інтерфейсу. Це дозволяє швидко інтегрувати його в проєкти без складних налаштувань, що є важливою перевагою для новачків і професійних розробників. Крім того, висока сумісність із платформою Arduino дозволяє використовувати шилд у різних проєктах без потреби в додаткових модифікаціях.

Таким чином, L293D Based Arduino Motor Shield вирізняється простотою у використанні, високою сумісністю з платформою Arduino, здатністю одночасно керувати кількома двигунами, підтримкою регулювання швидкості та можливістю розширення функціональності. Ці характеристики роблять його оптимальним вибором для реалізації проєктів, пов'язаних із керуванням двигунами постійного струму на основі Arduino.

Основою роботи цього шилда є два драйвери двигунів L293D та регістр зсуву 74HC595. Драйвер L293D є двоканальним H-Bridge контролером, який дозволяє керувати двома двигунами постійного струму або одним кроковим двигуном. Завдяки наявності двох таких драйверів у конструкції шилда, можливо

одночасно підключати та контролювати до чотирьох двигунів постійного струму або два крокові двигуни.

Регістр зсуву 74HC595 виконує функцію розширення цифрових контактів Arduino. Зокрема, він дозволяє збільшити кількість керуючих контактів з чотирьох до восьми, що забезпечує можливість керування напрямком обертання двох мікросхем L293D. Завдяки цьому підходу до оптимізації ресурсів Arduino, L293D Based Arduino Motor Shield є ефективним інструментом для створення складних систем керування двигунами.

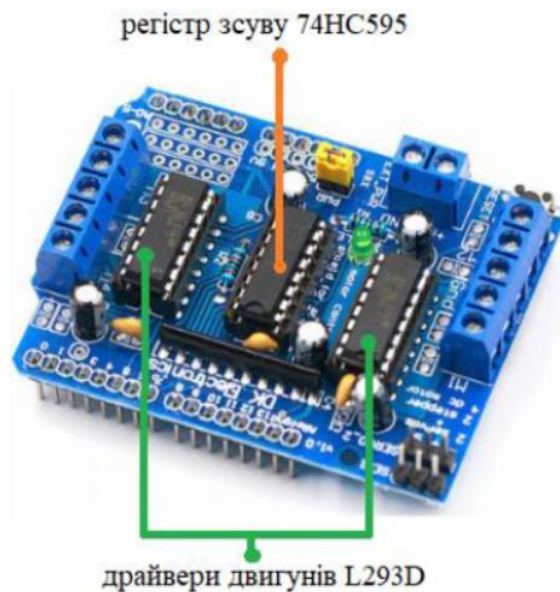


Рис. 3.4. Розміщення L293D драйверів двигунів і 74HC595 регістра зсуву

Для живлення двигунів шилд підтримує широкий діапазон напруги від 4,5 до 36 вольт. Це дозволяє використовувати різноманітні типи двигунів. Існує можливість підключення живлення до шилда та Arduino як від одного, так і від різних джерел. Вибір режиму живлення здійснюється за допомогою перемички PWR, розташованої біля роз'єму живлення. (рис. 3.5).

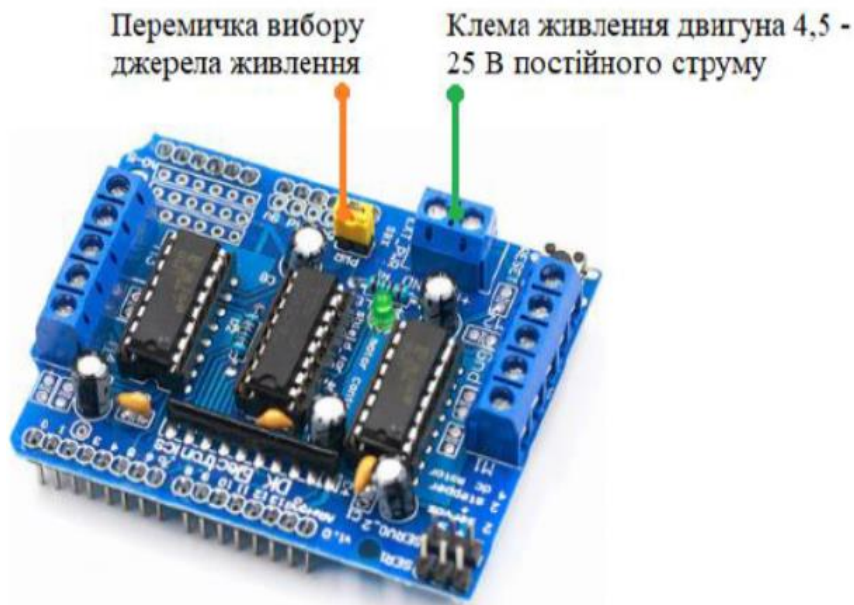


Рис. 3.5. Розміщення системи живлення шилда

Спосіб живлення шилда залежить від положення перемички PWR. Якщо перемичка встановлена, живлення двигунів здійснюється через роз'єм живлення Arduino, тобто двигуни та Arduino мають спільне джерело живлення. Це спрощує підключення, але обмежує максимальну напругу живлення двигунів 12 В. Коли перемичка видалена, двигуни живляться від окремого джерела, підключеного до роз'єму EXT\_PWR, забезпечуючи повну електричну ізоляцію від Arduino. Це дозволяє використовувати двигуни з більшою напругою живлення, але вимагає додаткового джерела живлення.

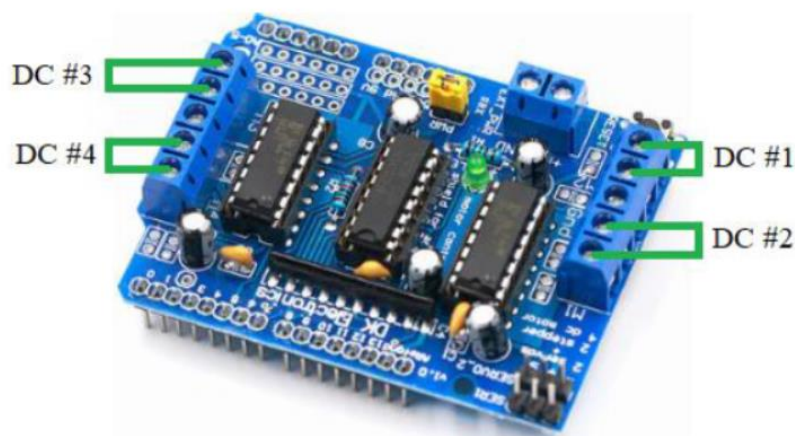


Рис. 3.6. Гвинтові клеми для підключення двигунів

Кожен канал модуля здатний забезпечити електричним струмом до 600 міліампер (з піковим значенням 1,2 ампера) для живлення одного двигуна постійного струму. Однак, фактична сила струму, що надходить до двигуна,

безпосередньо залежить від потужності зовнішнього джерела живлення. Детальна схема підключення сервоприводу наведена на рисунку 3.7.

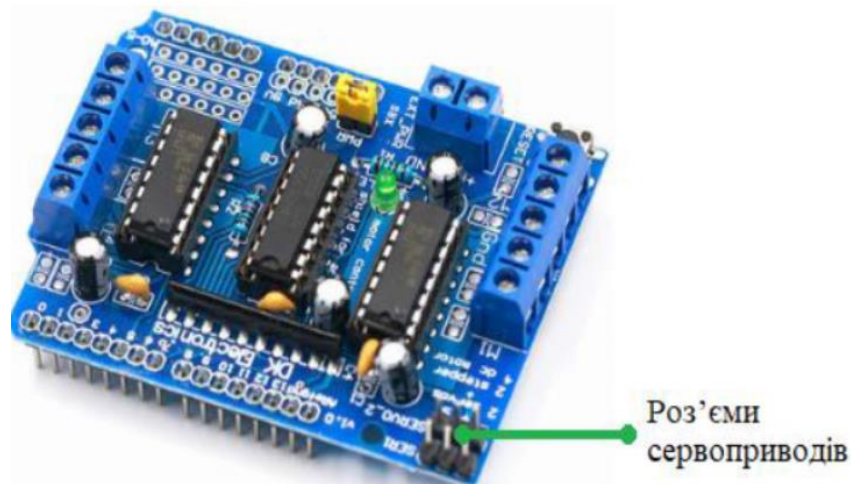


Рис. 3.7. Гвинтові клеми для підключення двигунів

Безпосереднє живлення сервоприводів від 5-вольтового виходу Arduino може призвести до перегріву стабілізатора напруги мікроконтролера та підвищення рівня шуму в системі живлення. Наявність 100 мкФ конденсатора частково компенсує негативний вплив такого підключення. Проте, для забезпечення стабільної роботи системи рекомендується використовувати цей шилд лише з малопотужними сервоприводами, такими як SG90.

Загальна схема шилда подана на рис. 3.8.





Рис. 3.9. Ультразвуковий сенсор HC-SR04

Датчик HC-SR04 забезпечує високу точність вимірювань відстані з похибкою близько  $\pm 3$  мм. Працюючи на частоті 40 кГц, він випромінює ультразвукові хвилі, які відбиваються від об'єктів і повертаються до датчика. За часом проходження цих хвиль датчик обчислює відстань до об'єкта.

Датчик має чотири виводи: живлення (Vcc), тригер (Trigger), ехо-сигнал (Echo) та загальний провід (GND). Для управління датчиком достатньо використовувати цифрові входи та виходи мікроконтролера.

В нашому проєкті ультразвуковий датчик HC-SR04 (рис. 3.10) відповідає за виявлення перешкод. Шляхом вимірювання відстані до об'єктів робот може уникнути зіткнень. Завдяки цьому датчику ми можемо реалізувати функції автоматичного керування та побудови карт оточення.

Датчик HC-SR04 є ключовим компонентом нашої системи навігації. Він дозволяє роботу вимірювати відстань до навколишніх об'єктів, що забезпечує безпечне пересування та можливість створення карт оточення.



Рис. 3.10. Ультразвуковий датчик, закріплений на тримачі для подальшого закріплення на серводвигуні



Принцип роботи ультразвукового датчика HC-SR04 (рис. 3.11-3.12) полягає в наступному: коли на тригерний вивід подається логічна одиниця тривалістю 10 мкс, датчик генерує серію з восьми ультразвукових імпульсів частотою 40 кГц. Така модуляція сигналу дозволяє відрізнити власні імпульси датчика від зовнішніх перешкод. Після випромінювання імпульсів, ехо-вихід переходить у високий стан, очікуючи відбитого сигналу. Якщо протягом 38 мс відбитий сигнал не прийнятий, ехо-вихід повертається в низький стан, що свідчить про відсутність об'єктів в зоні дії датчика.



Рис. 3.11. Принцип роботи сигналу за відсутності перешкод

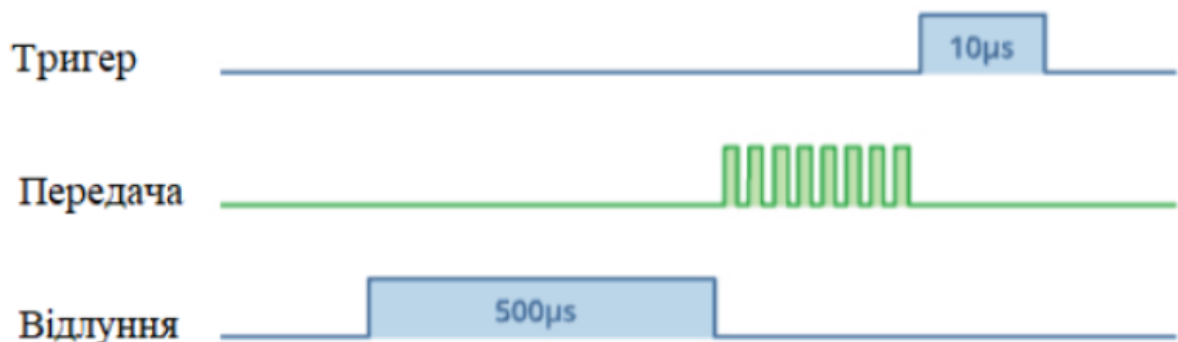


Рис. 3.12. Принцип роботи сигналу за наявності перешкод

Якщо ультразвукові хвилі зустрічають перешкоду, вони відбиваються назад до датчика. При прийомі відбитого сигналу, ехо-вихід змінює свій стан з високого на низький, генеруючи імпульс. Тривалість цього імпульсу напряму пов'язана з відстанню до об'єкта: чим далі об'єкт, тим довше триває імпульс, в діапазоні від 150 мікросекунд до 25 мілісекунд.

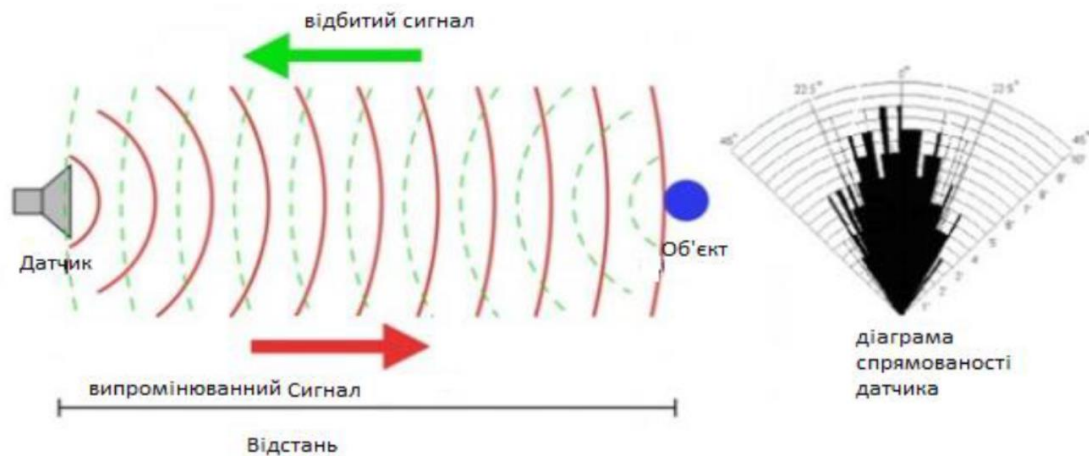


Рис. 3.13. Принцип роботи ультразвукового далекоміра HC– SR04

Для визначення відстані до об'єкта використовується тривалість імпульсу, який приймається датчиком після відбиття від об'єкта. Цей час, разом зі швидкістю поширення звуку в середовищі, дозволяє розрахувати відстань за відомою формулою, що пов'язує відстань, швидкість і час. Для наочного представлення цієї залежності часто використовують трикутник (рис. 3.14), в вершинах якого розташовані відповідні величини.

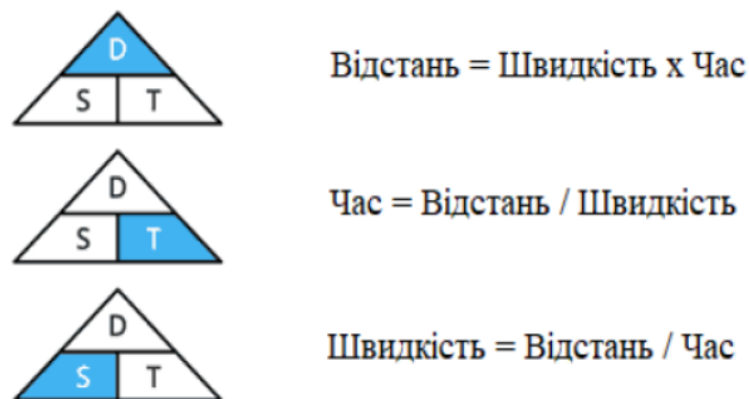


Рис. 3.14. Графічне зображення розрахунку відстані

Щоб визначити відстань до об'єкта, скористаємося отриманим часом відлуння – 500 мкс. З огляду на швидкість звуку в повітрі, яка становить 340 м/с, або 0,034 см/мкс, ми можемо розрахувати пройдений звук за цей час. Однак, оскільки звуковий сигнал спочатку рухається до об'єкта, а потім повертається назад, отримане значення необхідно поділити на два. Таким чином, відстань до об'єкта складе  $(0,034 \text{ см/мкс} * 500 \text{ мкс}) / 2 = 8,5 \text{ см}$ . Сервопривід SG90 (рис. 3.14), який є компактним і легким пристроєм, дозволяє керувати рухом різних механізмів.



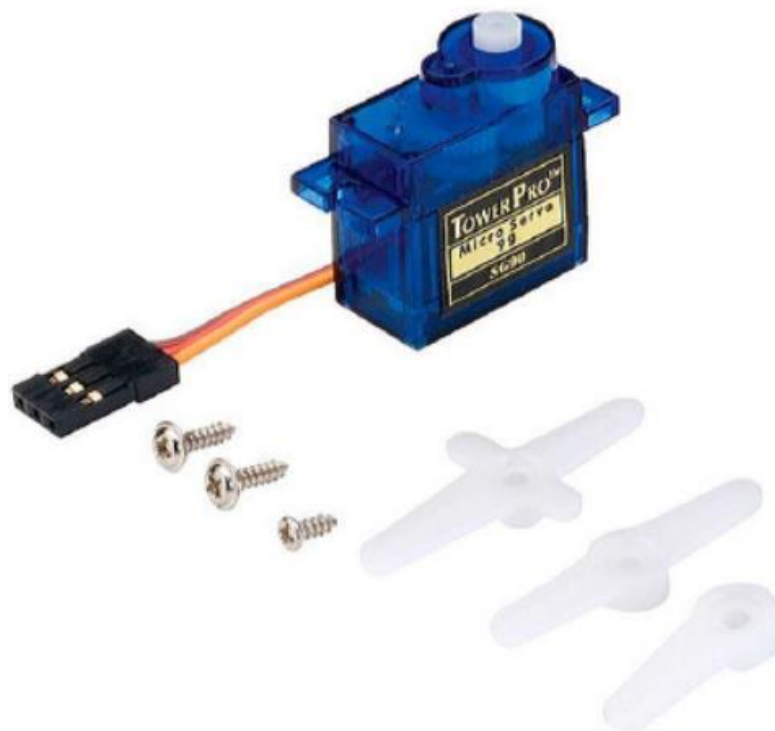


Рис. 3.15. Зовнішній вигляд сервоприводу SG90

Сервопривід SG90 є компактним пристроєм, здатним обертатися приблизно на 180 градусів. Він має середній крутний момент, достатній для керування легкими та середніми навантаженнями. Для роботи сервоприводу необхідна напруга живлення в діапазоні 4.8-6 В. Керування кутом повороту здійснюється за допомогою ширини модульованих імпульсів (ШИМ). Вбудований потенціометр забезпечує зворотний зв'язок, дозволяючи точно позиціонувати вал сервоприводу.



Рис. 3.16. Складові сервоприводу

Коли виникає розбіжність між бажаним і фактичним положенням валу сервоприводу, вбудована система управління коригує роботу двигуна, доки не буде досягнуто потрібного положення. Такий принцип дії називається сервомеханізмом. Це замкнена система управління, в якій поточний стан постійно порівнюється з бажаним, а відхилення автоматично усувається. Сервопривід має три основні виводи: загальний провід (GND), живлення (5V) та керуючий вхід (Control).



Рис. 3.17. Положення сервоприводу відносно довжини імпульсу

Керування сервоприводом здійснюється за допомогою послідовності імпульсів (рис. 3.18). Оптимальна частота цих імпульсів становить 50 Гц, тобто один імпульс подається кожні 20 мілісекунд. Саме тривалість кожного імпульсу визначає, на який кут повернеться вал сервоприводу.

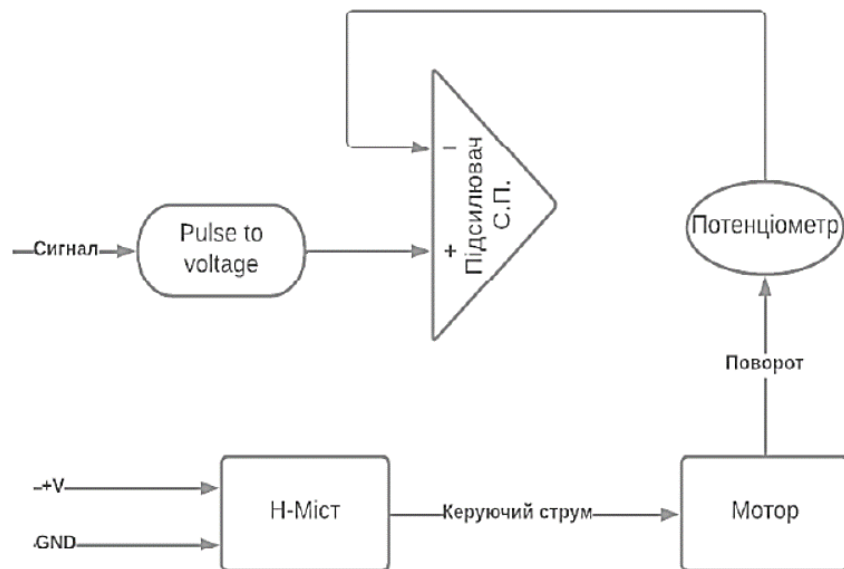


Рис. 3.18. Схема роботи сервоприводу

Положення сервоприводу безпосередньо залежить від тривалості керуючого імпульсу. Короткі імпульси (менше 1 мс) встановлюють сервопривід у крайнє ліве положення (0 градусів), а довгі імпульси (близько 2 мс) - у крайнє праве (180 градусів). Імпульси середньої тривалості (близько 1,5 мс) відповідають нейтральному положенню (90 градусів). Таким чином, змінюючи тривалість імпульсу в діапазоні від 1 до 2 мс, можна плавно переміщати сервопривід у будь-яке проміжне положення (рис. 3.19).

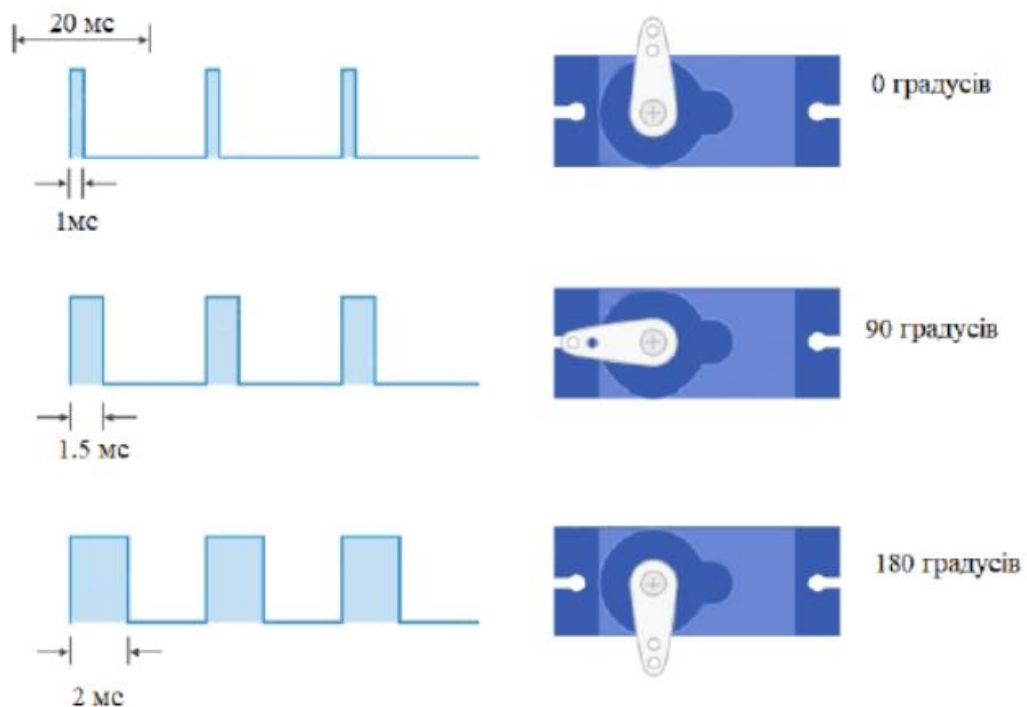


Рис 3.19. Положення сервоприводу відносно довжини імпульсу

Двигуни постійного струму (рис. 3.20) відіграють ключову роль в мобільності робота. Вони надають роботу необхідну силу та швидкість для виконання завдань, таких як переміщення по поверхні, подолання перешкод та маневрування в обмеженому просторі.



Рис. 3.20. Зовнішній вигляд двигуна постійного струму

Рухи нашого робота координуються за допомогою мікроконтролера Arduino Uno, який керує двома основними компонентами: двигунами та сервоприводом. Двигуни, керовані драйвером L293D, забезпечують рух робота вперед, назад і повороти. Їхня швидкість та напрямок обертання регулюються електронними сигналами від Arduino. Сервопривід SG90, встановлений на спеціальній платформі, дозволяє роботу обертати голову в горизонтальній площині. Arduino контролює кут повороту сервоприводу, надаючи роботу можливість "дивитися" в різні напрямки. Завдяки злагодженій роботі цих компонентів, робот може ефективно орієнтуватися в просторі та виконувати різноманітні завдання.

#### Основні характеристики двигунів постійного струму

1. Напруга живлення (Supply Voltage). Більшість DC Gear Motor функціонують у діапазоні напруги від 3 до 12 вольт, хоча цей параметр може змінюватися залежно від моделі.

2. Швидкість обертання (Rotation Speed). Швидкість двигуна визначається у кількості обертів за хвилину (RPM) і може варіюватися від кількох десятків до сотень обертів за хвилину залежно від конкретної конструкції.

3. Момент обертання (Torque). Ця характеристика, яка вимірюється у Ньютонах-метрах (Н·м) або кілограмах-силах на сантиметр (кгс·см), демонструє силу, з якою вал двигуна обертається. Чим вищий момент обертання, тим більшу вагу або опір двигун здатний долати.

4. Передавальне відношення (Gear Ratio). Наявність редуктора в конструкції двигуна дозволяє змінювати швидкість обертання і збільшувати момент обертання. Передавальне відношення описує співвідношення між швидкістю обертання вала двигуна і вихідного вала з шестернею.

5. Потужність (Power). Потужність двигуна визначається у ватах (Вт) або міліватах (мВт) і відображає кількість електричної енергії, яку двигун споживає під час роботи.

6. Ефективність (Efficiency). Цей параметр показує здатність двигуна перетворювати електричну енергію на механічну з мінімальними втратами. Ефективність вимірюється у відсотках, і чим вище це значення, тим продуктивніший двигун.

Двигун постійного струму (DC Motor) — це електромеханічний пристрій, який перетворює електричну енергію постійного струму на механічну енергію (рис. 3.21). Основний елемент двигуна — індуктор (катушка), який створює магнітне поле під час подачі напруги на клеми.

В конструкції двигуна передбачений залізний вал, обмотаний провідником, що розташований між двома постійними магнітами — північним і південним. Взаємодія магнітного поля індуктора з полями магнітів створює сили притягання і відштовхування, що приводять вал у рух. Таким чином, генерується крутний момент, який забезпечує обертання валу.

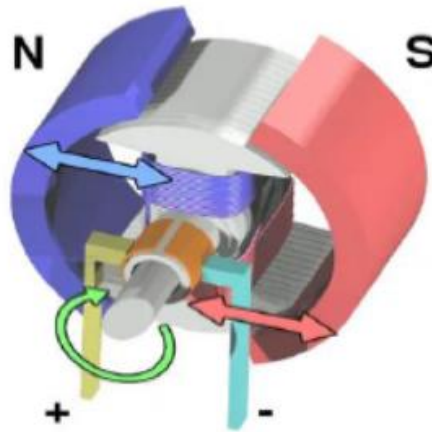


Рис. 3.21. Внутрішня частина DC мотору

### Характеристика двигунів використаних в даній роботі:

Швидкість	230 $\pm$ 10% обертів на хвилину (без навантаження); 140 $\pm$ 10% обертів на хвилину (з навантаженням)
Колір	Жовтий
Матеріал	ABS-пластик, силікон, метал
Крутний момент	0.7 кг/см
Напруга	допустимі 3-6 В (робочі 6 В)
Розмір	19 x 22.5 x 64.5 мм
Струм	180-500 мА
Діаметр валу	5.5 мм
Довжина кабелю	13 см

Мотор-редуктор - це компактний пристрій, який складається з електродвигуна та механічної передачі (редуктора). Редуктор дозволяє зменшити швидкість обертання валу, одночасно збільшуючи силу, яку може розвивати двигун. Головними характеристиками мотор-редуктора є швидкість обертання, крутний момент і ефективність перетворення енергії.

### **3.2. Побудова мобільного робота на базі Arduino**

Для створення мобільного робота було використано платформу Arduino Uno. Робот обладнаний ультразвуковим датчиком HC-SR04 для виявлення перешкод. Для руху використовуються чотири двигуни постійного струму з редукторами, керовані драйвером L293D. Для зміни напрямку огляду встановлено сервопривод SG90. Програмне забезпечення, розроблене на платформі Arduino, забезпечує автономний рух робота та уникнення перешкод.

Компоненти апаратного забезпечення (рис. 3.22):

- 1x Arduino UNO R3
- 1x 4-колісне шасі автомобіля
- 4x Двигун постійного струму з редуктором
- 1x Ультразвуковий датчик HC-SR04
- 1x Тримач ультразвукового датчика HC-SR04
- 1x Шилд драйвера двигунів L293D
- 1x SG90 Сервопривод
- 2x Батареї 9В
- 1x Кріплення для акумулятора
- 1x Вимикач живлення
- З'єднувальні дроти

#### **Процес збирання моделі мобільного робота.**

Крок 1. До двигунів припаюються товсті дроти червоного та чорного кольорів, відповідно до позитивних і негативних клем. Після цього двигуни закріплюються на шасі за допомогою гвинтів, а дроти фіксуються шляхом опресування (рис. 3.23).

Крок 2. На шасі монтуються бокси для розміщення плати Arduino Uno та системи живлення. Для цього використовується клей. Сервопривод також фіксується на передній частині шасі за допомогою клею, після чого на нього встановлюється тримач із ультразвуковим сенсором HC-SR04 (рис. 3.24).

Крок 3. Перемикач припаюється до кріплення для акумуляторів і фіксується на шасі за допомогою клею. Дроти, підключені до перемикача,



опресовуються. У зафіксовані бокси встановлюються система живлення та плата Arduino Uno. На плату Arduino Uno монтується драйвер двигунів L293D. До клем драйвера приєднуються дроти, що ведуть від двигунів і акумуляторів (рис. 3.25–3.26).

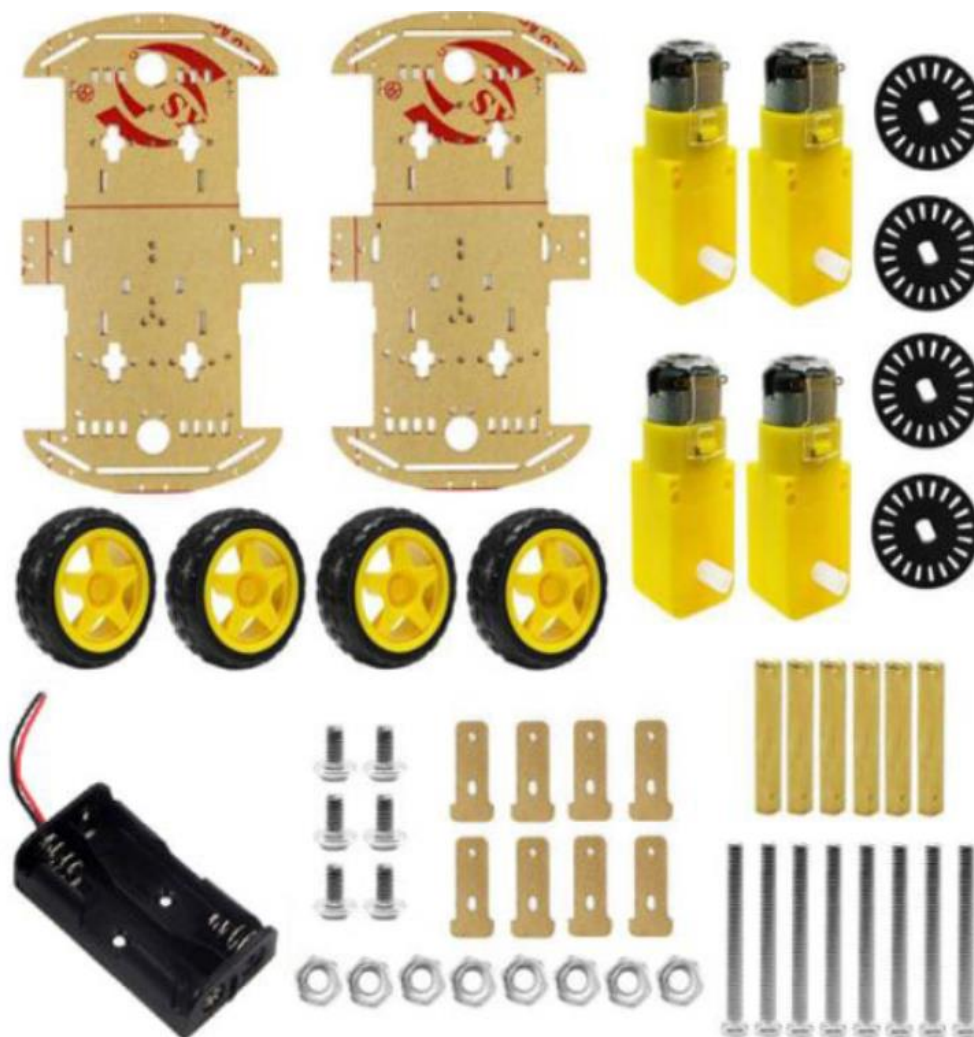


Рис. 3.22. Компоненти шасі робота

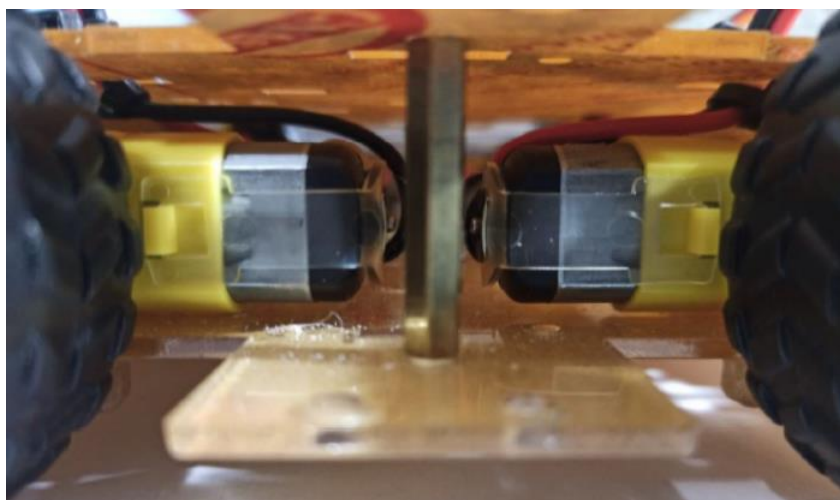


Рис. 3.23. Розміщення двигунів в середині шасі



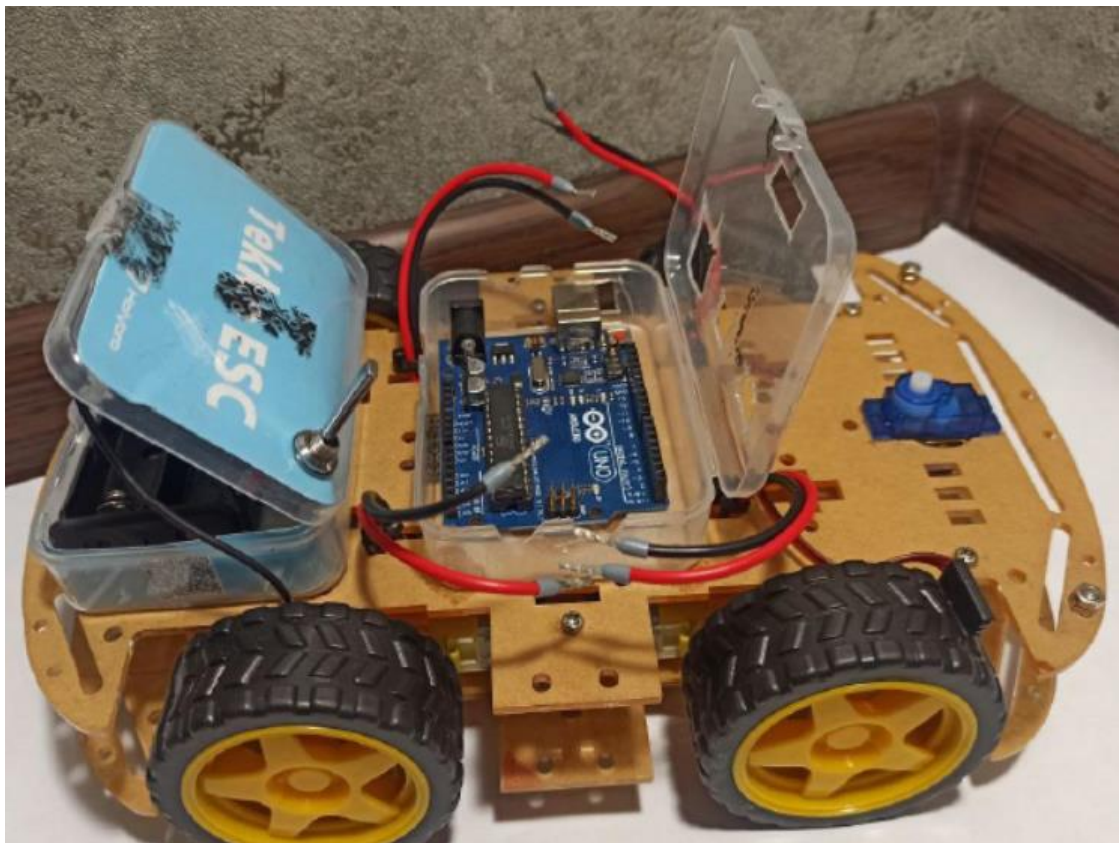


Рис. 3.24. Шасі з встановленими боксами для деталей робота та закріпленим сервоприводом

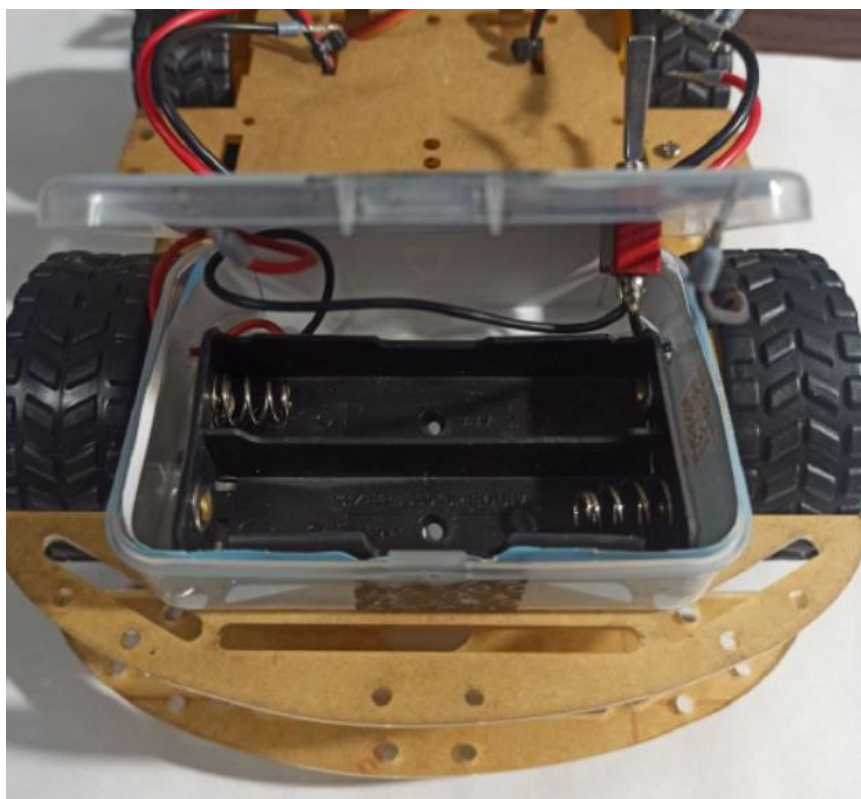


Рис. 3.25. З'єднання системи живлення



Рис. 3.26. З'єднання шилда з двигунами та системою живлення

Крок 4. До шилда підключаються дроти від сервопривода та ультразвукового сенсора (рис. 3.27–3.28).

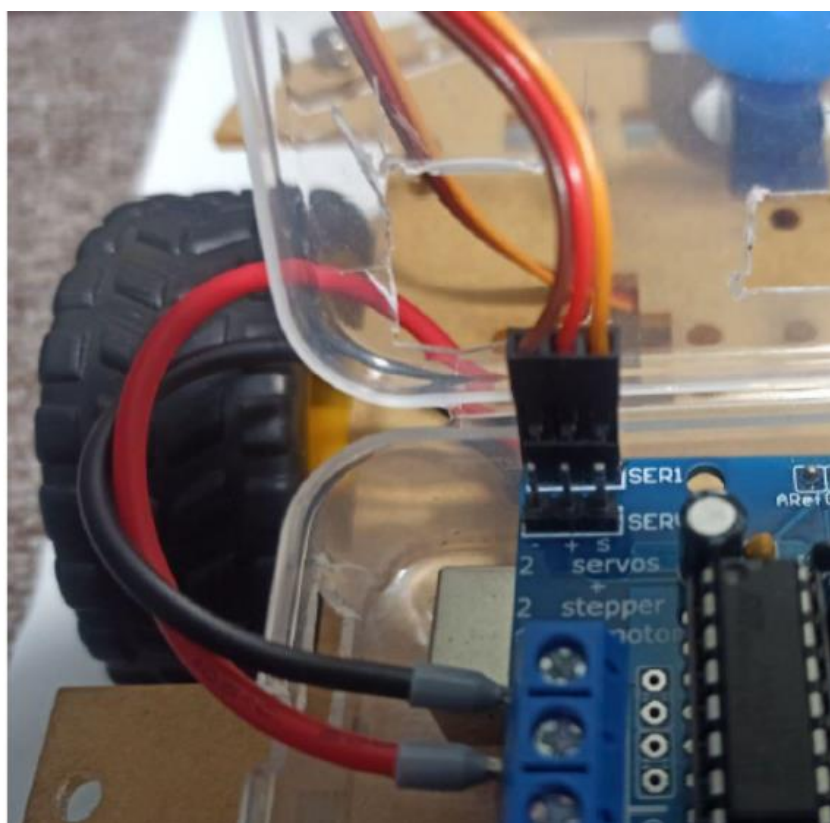


Рис. 3.27. Підключення сервоприводу до шилда



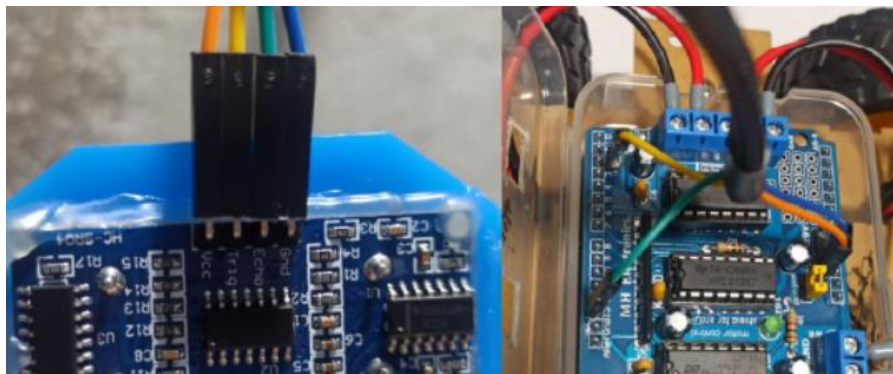


Рис. 3.28. Підключення ультразвукового датчика до шилда

На рис. 3.29 подано загальну схему з'єднання компонентів робота - автомобіля.

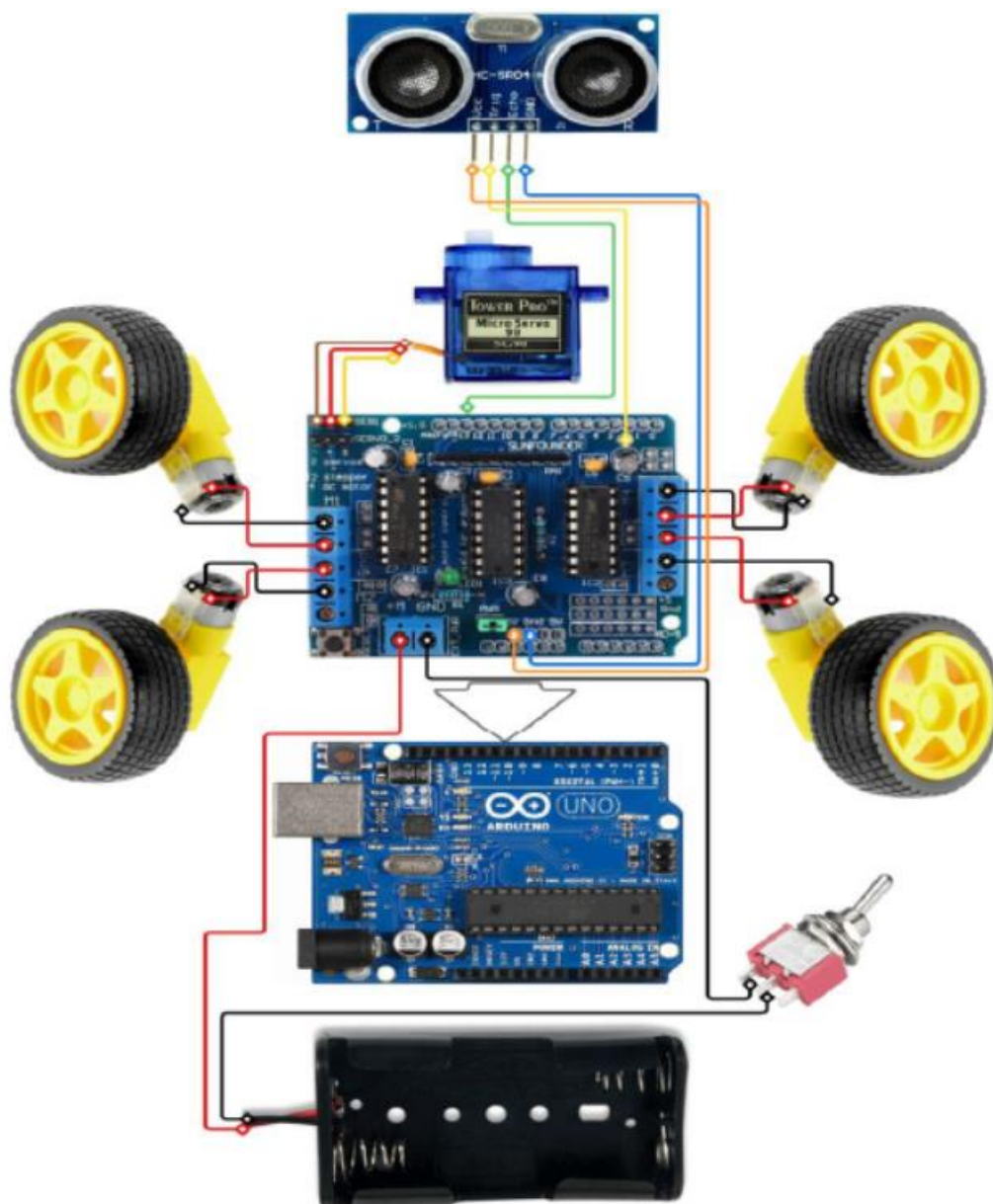


Рис. 3.29. Схема з'єднання компонентів робота-автомобіля

### 3.3 Програмні засоби проєктування мобільного робота

Arduino IDE - це інтегроване середовище розробки, призначене для створення програмного забезпечення для платформи Arduino. Воно надає зручний інтерфейс для написання коду, його компіляції та завантаження в мікроконтролер. Середовище підтримує широкий спектр платформ і містить бібліотеки функцій для роботи з різноманітними датчиками та виконавчими пристроями.

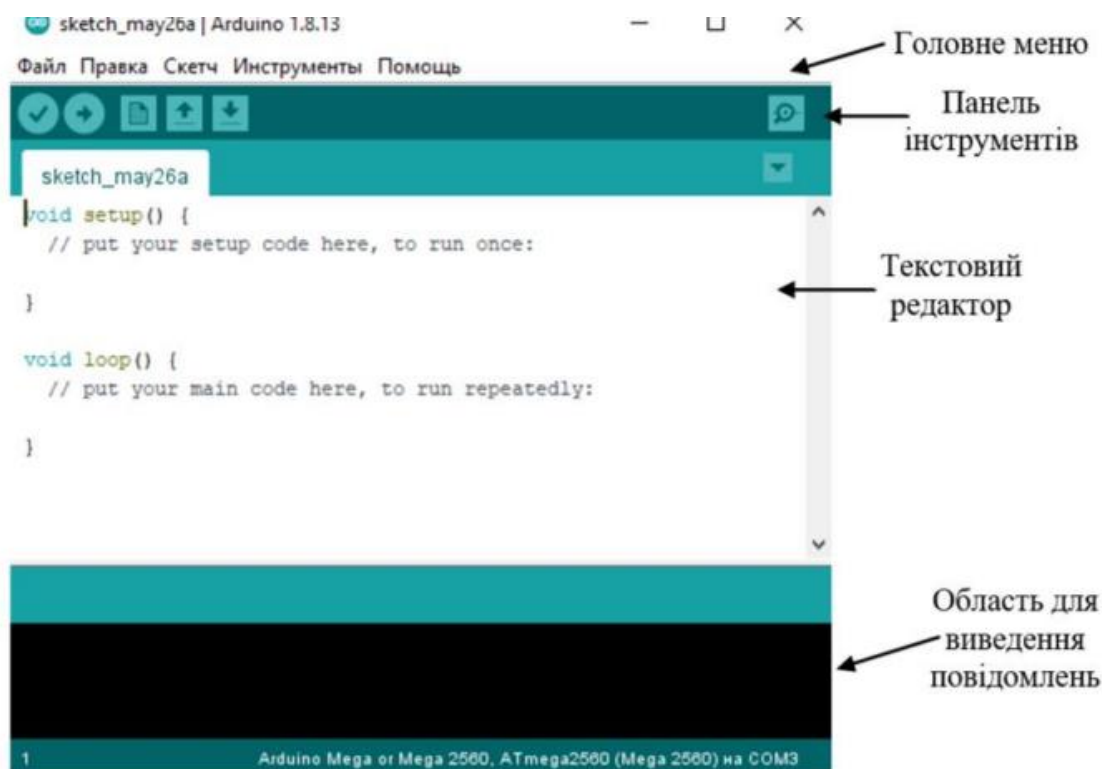


Рис. 3.30. Інтерфейс Arduino IDE

Скетчі, написані в середовищі розробки Arduino IDE, зберігаються у файлах з розширенням .ino. Редактор коду в IDE забезпечує стандартний набір функцій для роботи з текстом: копіювання, вставка, пошук і заміна. Інформація про хід виконання програми та можливі помилки відображається в області повідомлень. Вибір моделі плати Arduino та послідовного порту здійснюється в нижній частині вікна програми. Панель інструментів надає швидкий доступ до основних функцій середовища розробки. На панелі інструментів знаходиться 6 кнопок (рис. 3.31):



Рис. 3.31. Панель інструментів

### Основні функції Arduino IDE

- **Перевірити.** Дана опція дозволяє перевірити синтаксис написаного коду. У разі виявлення помилок вони відображаються в області повідомлень.

- **Завантажити.** Використовується для компіляції програми та її завантаження у мікроконтролер Arduino.

- **Створити.** Дозволяє створити новий файл для написання скетчу.

- **Відкрити.** Призначена для відкриття існуючого файлу зі скетчем.

- **Зберегти.** Дозволяє зберегти поточний скетч у файл.

- **Монітор послідовного порту.** Використовується для відкриття програми Serial Monitor, яка відображає дані, що надходять від Arduino на комп'ютер через послідовний інтерфейс. Монітор підтримує роботу як із USB-варіантами плати, так і зі звичайними версіями Arduino. Для передачі даних на зовнішній пристрій у вікні монітора вводиться текст, після чого натискається кнопка "Відправити" або клавіша Enter. Швидкість передачі даних потрібно налаштувати відповідно до параметрів, вказаних у функції `Serial.begin()` у вашому коді. Функція `Serial.print()` дозволяє виводити текст у вікно монітора.

Бібліотеки значно розширюють функціональність програм, надаючи додаткові можливості, такі як робота з апаратними засобами або обробка даних. Для додавання бібліотеки слід перейти до меню Sketch, вибрати опцію Include Library і обрати необхідну бібліотеку. Після цього в програму додається оператор `#include`, а бібліотека компілюється разом зі скетчем. Важливо зазначити, що кожна підключена бібліотека використовує частину пам'яті мікроконтролера.

Більшість бібліотек уже попередньо встановлені разом із програмним забезпеченням Arduino, однак додаткові бібліотеки можна завантажити з зовнішніх джерел.

Для завантаження скетчу необхідно вибрати відповідну плату та порт (див. рис.3.32), які використовуються для вашої операційної системи. Це здійснюється через меню Tools, де у розділі Board вибирається модель плати, а в розділі портів — відповідний COM-порт (наприклад, COM1, COM2, COM4, COM5, COM7 тощо) або USB-з'єднання.

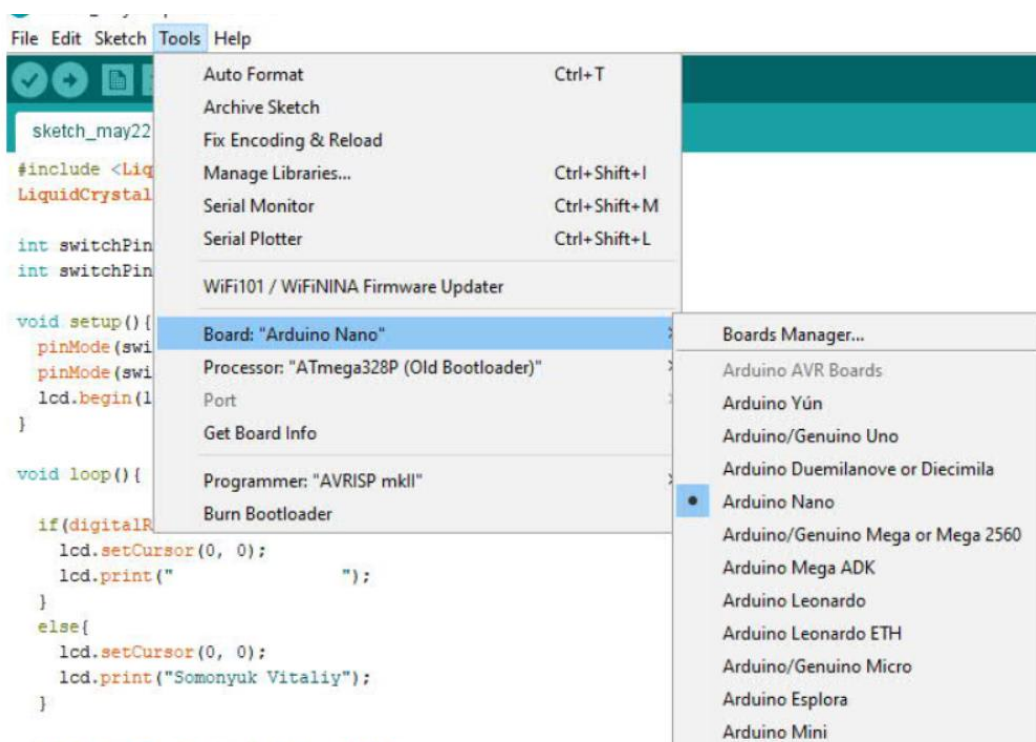


Рис. 3.32. Зображення вибору плати

Для завантаження програми в плату Arduino необхідно встановити з'єднання між комп'ютером і платою за допомогою USB-кабелю. В середовищі розробки Arduino IDE слід вибрати відповідну модель плати та послідовний порт, до якого підключено Arduino. Ця інформація необхідна для того, щоб комп'ютер міг правильно спілкуватися з платою.

Після вибору потрібних параметрів можна приступати безпосередньо до процесу прошивки. Для цього необхідно скопіювати написаний код. Компілятор перетворить ваш код, написаний на спрощеній версії C++, в машинний код, який розуміє мікроконтролер Arduino. Скопійований код

передається в буфер завантажувача - спеціальної програми, яка зашита в пам'ять мікроконтролера. Завантажувач, у свою чергу, переписує отриманий код в основну пам'ять мікроконтролера і запускає його на виконання.

Процес прошивки супроводжується скиданням мікроконтролера та миганням світлодіодів, що індикують активність передачі даних. Для забезпечення зручності користувача середовище розробки Arduino IDE надає інтуїтивно зрозумілий інтерфейс та автоматизує більшість рутинних операцій.

Функції `setup()` та `loop()` є основою будь-якого скетчу для Arduino.

- `setup()` виконується один раз на початку програми і використовується для ініціалізації портів введення-виведення, встановлення початкових значень змінних та підключення бібліотек.

- `loop()` виконується циклічно і містить основний алгоритм програми. Все, що має повторюватися, розміщується саме тут.

Основні функції для роботи з портами введення-виведення:

- Цифрові порти: `pinMode()`, `digitalWrite()`, `digitalRead()`. Ці функції дозволяють керувати цифровими виходами (наприклад, світлодіодами) та зчитувати значення з цифрових входів (наприклад, кнопок).

- Аналогові порти: `analogWrite()`, `analogRead()`, `analogReference()`. Ці функції використовуються для роботи з аналоговими сигналами, такими як напруга.

- Спеціальні функції: `tone()`, `noTone()` для генерації звукових сигналів, `millis()` для вимірювання часу.

**Розробка програмного забезпечення для платформи Arduino передбачає кілька послідовних етапів:**

### ***1. Ініціалізація проєкту:***

- Створення нового проєкту в середовищі Arduino IDE.
- Вибір відповідної моделі плати (Uno, Mega тощо) та налаштування порту, до якого підключена плата.

### ***2. Написання коду:***

- Кожна програма для Arduino складається з двох основних функцій:

- `setup()`: виконується один раз на початку програми. Тут ініціалізуються порти введення-виведення, встановлюються початкові значення змінних та підключаються необхідні бібліотеки.

- `loop()`: виконується циклічно після завершення `setup()`. Містить основний алгоритм програми, тобто дії, які будуть повторюватися постійно.

- Для взаємодії з різними компонентами (датчиками, актуаторами тощо) використовуються спеціальні функції:

- `pinMode()`, `digitalWrite()`, `digitalRead()`: для роботи з цифровими портами.

- `analogWrite()`, `analogRead()`: для роботи з аналоговими портами.

- `tone()`, `noTone()`: для генерації звукових сигналів.

- `millis()`: для вимірювання часу.

Для розширення функціональності програми можна підключати готові бібліотеки, наприклад, для керування сервоприводами, LCD-дисплеями тощо.

### ***3. Компіляція та завантаження:***

- Перетворення коду, написаного людиною, в машинний код, зрозумілий мікроконтролеру. Цей процес здійснюється автоматично при натисканні кнопки "Завантажити" в Arduino IDE.

- Передача скомпільованого коду в пам'ять мікроконтролера Arduino через USB-з'єднання.

### ***4. Тестування та налагодження:***

- Перевірка коректності роботи програми та усунення виявлених помилок.

- Спостереження за поведінкою підключених пристроїв та порівняння результатів з очікуваними.

Рисунок 3.33 візуалізує алгоритм навігації мобільного робота. Конкретна реалізація цього алгоритму мовою програмування представлена в коді, наведеному в Додатку А.



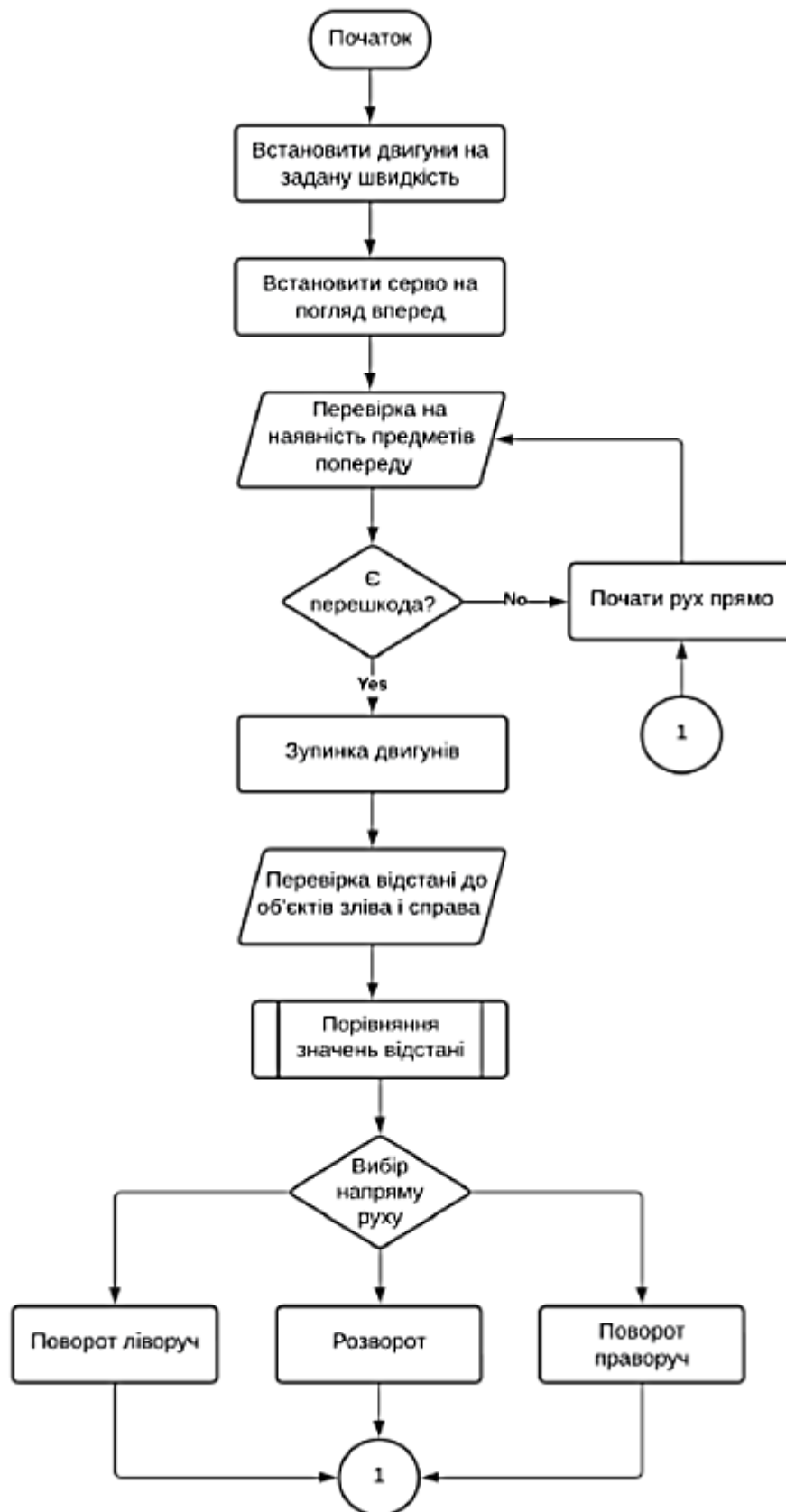


Рис. 3.33. Блок-схема алгоритму системи навігації мобільного робота

### 3.4. Тестування мобільного робота

Результатом проєкту є мобільний робот на базі Arduino, який здатен пересуватися, виявляючи перешкоди на своєму шляху, та уникати їх (рис. 3.34).

У процесі роботи робот використовує сонар для надсилання ультразвукових хвиль у трьох напрямках: прямо ( $90^\circ$ ), ліворуч ( $180^\circ$ ) та праворуч ( $0^\circ$ ). При зіткненні звукової хвилі з перешкодою вона відбивається, і мікроконтролер реєструє відстані для кожного з напрямків. Отримані дані аналізуються за допомогою алгоритму, після чого робот обирає оптимальний напрямок для розвороту.

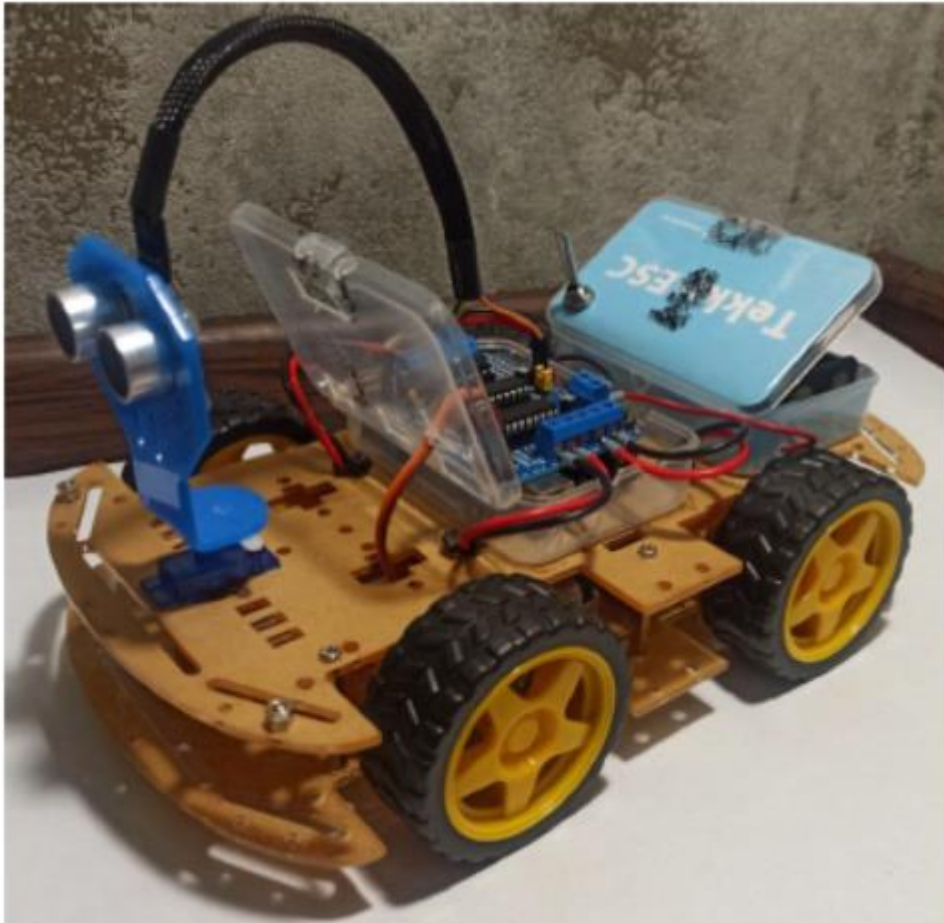


Рис. 3.34. Апаратна модель мобільного робота

Проведені тести на створеному апаратному забезпеченні виявили обмеження алгоритму розпізнавання перешкод. Проблема полягала в тому, що деякі об'єкти не виявлялися через обмежений діапазон датчика. Якщо перешкода знаходиться за межами зони «видимості» датчика, робот не здатен її ідентифікувати, що може призвести до зіткнення. Для уникнення таких ситуацій випробування проводилися в замкнутому просторі, де єдиною перешкодою були стіни, що дозволяло роботу вільно пересуватися без зіткнень. Щоб створити робота, здатного ефективно розпізнавати численні перешкоди, слід використовувати більше різних датчиків для розширення зони виявлення.

## Результати тестування

### 1. Ефективність датчика

Ультразвуковий датчик HC-SR04 є ключовим компонентом системи. Він працює за рахунок випромінювання високочастотної звукової хвилі (40 кГц) та приймання відбитих імпульсів через п'єзоелектричні перетворювачі. Проте датчик не міг виявити перешкоди, розташовані поза кутом 30°, або об'єкти, які знаходилися нижче його рівня. У таких випадках можливе зіткнення. В інших ситуаціях датчик надійно виявляв об'єкти в межах свого діапазону, що забезпечувало уникнення перешкод.

### 2. Напруга батареї

Компоненти робота, особливо двигуни, створюють значне навантаження на батарею. Через низький ККД двигуни потребують багато енергії.

- Arduino живиться від батареї на 9 В, а щит драйвера двигуна — від двох послідовно з'єднаних батарей по 9 В (загалом 18 В).
- Двигуни, серводвигун і датчик вимагають напруги в межах 3–6 В.

Така конфігурація дозволяє забезпечувати живлення робота протягом приблизно 15 хвилин активної роботи.

### 3. Вплив температури на щит драйвера двигуна

Ефективність щита драйвера двигуна залежить від його здатності розсіювати тепло. При збільшенні температури мікросхеми подача струму зменшується, що знижує потужність двигунів і їх здатність обертатися.

- Щит протестовано за кімнатної температури без додаткового охолодження, і він зміг забезпечити стабільну роботу двигунів для чотирьох каналів до моменту спрацьовування теплового захисту.
- Для покращення продуктивності доцільно використовувати радіатори або примусове охолодження.

Таким чином, створений робот ефективно виконує базові завдання, але потребує вдосконалення, зокрема розширення зони виявлення перешкод, оптимізації системи живлення та покращення тепловідведення драйвера двигунів.

## Висновки розділу

Проблема виявлення та уникнення перешкод у процесі руху є ключовою при розробці мобільних роботів. Її вирішення досягається завдяки оснащенню роботів сенсорами, які забезпечують навігацію в невідомому середовищі. Проведені експерименти й отримані результати довели, що створення мобільного робота на базі Arduino UNO та Adafruit Motor Shield, із кодом, написаним у середовищі Arduino IDE, є цілком реальним.

Створений робот, будучи повністю автономним після завантаження відповідного коду, не потребує участі оператора для керування під час руху. Експериментальні випробування підтвердили його здатність ефективно маневрувати в невідомому середовищі, уникаючи зіткнень. Використана методологія, яка включає апаратні та програмні засоби навігації, дозволила досягти поставлених завдань.

Робот володіє наступними можливостями:

- Орієнтація у відкритому просторі: здатність виявляти перешкоди на основі заданої граничної відстані.
- Маневрування та уникнення зіткнень: ухвалення рішень у режимі реального часу для зміни курсу.
- Автономність у невідомому середовищі: виконання функцій без зовнішнього втручання.

Використане апаратне забезпечення є доступним та недорогим, що робить прототип легко відтворюваним. Базові принципи, реалізовані у цій роботі, можуть слугувати основою для подальших досліджень у галузі навігаційних систем. Розширення функціональних можливостей робота дозволить адаптувати його до нових завдань у майбутніх проєктах.

## ВИСНОВКИ

Протягом останніх двох десятиліть мобільна робототехніка як технологічна галузь і сфера наукових досліджень зазнала значних трансформацій, зумовлених розвитком інформаційно-комунікаційних технологій. Це створило умови для прискореної автоматизації транспортних засобів та появи SDC (самокерованих автомобілів) і мобільних роботів різного призначення, здатних діяти без участі людини-оператора. Водночас, реалізація проєктів такого масштабу вимагає значних фінансових, трудових ресурсів і спеціалізованих інструментів для втілення інженерних ідей і дослідницьких розробок. Часто розробникам доводиться витрачати час на створення експериментальних зразків, фактично заново розв'язуючи інженерні задачі.

Платформа Arduino із її функціональними можливостями спрощує створення прототипів мобільних роботів, зокрема завдяки доступності, простоті налаштування та недорогим рішенням. Це сприяє пришвидшенню дослідницької роботи та полегшує тестування нових розробок. Arduino стимулює інновації у робототехніці, відкриваючи можливості для створення нових проєктів, таких як "Система навігації роботизованого транспортного засобу на базі Arduino". У цьому проєкті використовується ультразвуковий датчик для виявлення статичних перешкод, а Motor Driver Shield керує двигунами постійного струму, що забезпечує рух мобільного робота під управлінням мікроконтролера Arduino.

Ефективність навігації робота залежить від умов тестування, кількості перешкод і якості використовуваних датчиків. Ультразвукові датчики мають певні обмеження: вони менш ефективні на великих відстанях (понад 3 метри), при малих кутах або якщо об'єкт має погану відбивну здатність. Для підвищення автономності робот повинен бути здатним адаптуватися до змін середовища, працювати тривалий час без втручання людини, уникати небезпечних ситуацій і мати додаткові датчики для поліпшення точності.

Перспективи для вдосконалення включають:

- використання камер для обробки зображень і розширення функціональності за межі прямої видимості;

- впровадження бездротових технологій для дистанційного управління;
- застосування потужніших приводів для підвищення швидкості та маневреності.

Ці заходи забезпечать підвищення ефективності мобільних роботів і розширяють їх функціональні можливості.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sabry F. Self Driving Car: Solving Full Self-driving Need Solving Real-world Artificial Intelligence. One Billion Knowledgeable. 2022 . 293 p.
2. Shapiro D. G. Three Anecdotes from the DARPA Autonomous Land Vehicle Project. AI Magazine, 2008. 29(2), 40. <https://doi.org/10.1609/aimag.v29i2.2108>
3. Unmanned ground vehicle. URL: [https://en.wikipedia.org/wiki/Unmanned\\_ground\\_vehicle](https://en.wikipedia.org/wiki/Unmanned_ground_vehicle).
4. Pillath S. Automated vehicles in the EU. URL: [https://www.europarl.europa.eu/RegData/etudes/BRIE/2016/573902/EPRS\\_BR I\(2016\)573902\\_EN.pdf](https://www.europarl.europa.eu/RegData/etudes/BRIE/2016/573902/EPRS_BR I(2016)573902_EN.pdf)
5. U.S. Department of Transportation Releases Policy on Automated Vehicle Development. URL: <https://www.transportation.gov/briefing-room/usdepartment-transportation-releases-policy-automated-vehicle-development>.
6. Young R. Critical Analysis of Prototype Autonomous Vehicle Crash Rates Six Scientific Studies from 2015-2018 SAE International. 2021. 252 p
7. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016\_202104. URL: [https://www.sae.org/standards/content/j3016\\_202104/](https://www.sae.org/standards/content/j3016_202104/)
8. Поліщук М. М., Ткач М.М. Робототехнічні системи: проектування і моделювання: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2021. 112 с.
9. Рудик А. В. Наукові основи та принципи побудови приладової системи вимірювання прискорення мобільного робота: дис. ... д-ра техн. наук: 05.11.01 – Прилади та методи вимірювання механічних величин. Київ, 2018. – 460 с.
10. Мигаль В. Д. Інтелектуальні системи в технічній експлуатації автомобілів: монографія. Х.: Майдан, 2018. 262 с.
11. Velasco-Hernandez, G. et al. Autonomous Driving Architectures, Perception and Data Fusion: A Review. In Proceedings of the 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP

2020), Cluj-Napoca, Romania, 3–5 Sept. 2020.

DOI:[10.1109/ICCP51029.2020.9266268](https://doi.org/10.1109/ICCP51029.2020.9266268)

- 12.Said E. et al. Visibility-Based Technologies and Methodologies for Autonomous Driving. In book: "Self-driving Vehicles and Enabling Technologies" edited by Prof. Marian Gaiceanu. 2020. pp.1-25.
- 13.Staron M. Automotive Software Architectures: An Introduction. Second Edition Springer International Publishing A&G. 2022. 274 p
- 14.Raouf, I. et al. Sensor-Based Prognostic Health Management of Advanced Driver Assistance System for Autonomous Vehicles: A Recent Survey. Mathematics 2022, 10, 3233. <https://doi.org/10.3390/math10183233>
- 15.Vargas J. et al. An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions. Sensors. 2021; 21(16):5397. <https://doi.org/10.3390/s21165397>
- 16.Wevolver 2020 Autonomous Vehicle Technology Report. URL: <https://www.coursehero.com/file/62601304/Wevolver2020AutonomousVehicleTechnologyReportpdf/>
- 17.Yeong, D.J. et al. Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. Sensors 2021, 21, 2140. <https://doi.org/10.3390/s21062140>
- 18.Background Information on CCD and CMOS Technology. URL: [https://www.tedpella.com/cameras\\_html/ccd\\_cmos.aspx](https://www.tedpella.com/cameras_html/ccd_cmos.aspx)
- 19.A Look Inside ADAS Modules. URL: <https://amkor.com/semiconductor-story/a-look-inside-adas-modules/>
- 20.AV and ADAS Sensors. URL: <https://community.sw.siemens.com/s/article/AV-and-ADAS-Sensors>
- 21.A Look Inside ADAS Modules..URL: <https://amkor.com/semiconductor-story/a-look-inside-adas-modules/>
- 22.Drive and stay cool changing the lane with Blind Spot Detection. URL: <https://www.continental-automotive.com/en-gl/Passenger-Cars/Autonomous-mobility/Functions/Cruising-Driving/Blind-Spot-Detection>
- 23.Гуржій А. М. Основи автоматики та робототехніки: Навчальний посібник/



- A. М. Гуржій, А. Т. Нельга, В. М. Співак, О. С. Ітякін:—Дніпро: «Гарант СБ», 2021.- 243с.
- 24.Lim, B.S. et al. Autonomous Vehicle Ultrasonic Sensor Vulnerability and Impact Assessment. In Proceedings of the IEEE World Forum on Internet of Things (WF- IoT), Singapore, 5–8 February 2018; pp. 231–236.
- 25.Xu W. et al. Analyzing and Enhancing the Security of Ultrasonic Sensors for Autonomous Vehicles, in IEEE Internet of Things Journal, vol. 5, no. 6, pp. 5015- 5029, Dec. 2018. DOI: 10.1109/JIOT.2018.2867917
- 26.Ultrasonic Sensors in Self-Driving Cars Babak Shahian Jahromi Babak Shahian Jahromi. URL: <https://medium.com/@BabakShah/ultrasonic-sensors-in-self-driving-cars-d28b63be676f>
- 27.LIDAR Sensor in Autonomous Vehicles: Why it is Important for Self-Driving Cars? URL: <https://www.cogitotech.com/blog/lidar-sensor-in-autonomous-vehicles-why-it-is-important-for-self-driving-cars/>
- 28.Agnihotri N. What are the sensors used in self-driving cars? URL: <https://www.engineersgarage.com/what-are-the-sensors-used-in-self-driving-cars/>
- 29.A Guide to Lidar Wavelengths for Autonomous Vehicles and Driver Assistance. URL: <https://velodynelidar.com/blog/guide-to-lidar-wavelengths/>
- 30.SAE Levels of Driving Automation™ Refined for Clarity and International Audience. URL: <https://www.sae.org/blog/sae-j3016-update>
- 31.GM Super Cruise. URL: <https://gmauthority.com/blog/gm/general-motors-technology/general-motors-autonomous-technology/gm-super-cruise/>
- 32.Padgett M. Here's how the 2019 Audi A8 will become the first Level 3 self-driving car\* URL: [https://www.motorauthority.com/news/1111528\\_heres-how-the-2019-audi-a8-will-become-the-first-level-3-self-driving-car#](https://www.motorauthority.com/news/1111528_heres-how-the-2019-audi-a8-will-become-the-first-level-3-self-driving-car#)
- 33.Hyundai Motor Group's Roboride Experience: A Peek into the Autonomous Driving Era. URL: [https://www.hyundaimotorgroup.com/story/CONT0000000000082639\\_0](https://www.hyundaimotorgroup.com/story/CONT0000000000082639_0)
- 34.Mullen D. Watch waymo self-driving car navigating san francisco in heavy rain.

URL: <https://www.driving.co.uk/news/technology/watch-waymo-self-driving-car-navigating-san-francisco-in-heavy-rain/>

- 35.Q&A: Arduino founder Massimo Banzi. URL: <https://www.helsinkitimes.fi/business/22390-q-a-arduino-founder-massimo-banzi.html>
- 36.Саліхов М. М. Arduino – перспективний інструмент протитопування у робототехніці // Концептуальні шляхи розвитку науки та освіти: матеріали VIII Міжнародної науково-практичної конференції м. Львів, 9-10 червня 2023 року. – Львів: Львівський науковий форум, 2023. – с. 73-79.
- 37.Bräunl T. Embedded Robotics: From Mobile Robots to Autonomous Vehicles with RaspberryPi and Arduino Springer Nature, 2022. 519 p.
- 38.Navigation URL: <https://www.britannica.com/technology/navigation-technology>
- 39.Навігаційні системи [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / С.Л. Лакоза; КПІ ім. Ігоря Сікорського, 2021. — 80 с.
- 40.Imad M. et al. Navigation system for autonomous vehicle: A survey. JCSTS., vol. 2, no. 2, pp. 20–35, 2020.
- 41.Ковалець І. В. та ін. Технологія планування траєкторій руху мобільних об'єктів з урахуванням перешкод на складній місцевості. Енергетика і автоматика. - 2017. - № 1. - С. 110-122.
- 42.Aguilo I. et al. Artificial Intelligence Research and Development Hardcover by IOS Press. 2003. 500 p.
- 43.Rathin Chandra Shit Precise localization for achieving next-generation autonomous navigation: State-of-the-art, taxonomy and future prospects. Computer Communications Volume 160, 1 July 2020, Pages 351-374. <https://doi.org/10.1016/j.comcom.2020.06.007>
- 44.Гребенюк Б. А. Розробка підсистеми управління інтелектуальним роботом / Б. А. Гребенюк // «Automation and Development of Electronic Devices» ADED-2023: Collection of Students' Scientific Paper. – Kharkiv : Kind of

Kharkiv National University of Radio Electronics [electronic edition], 2023. – Part 1. –336p. P. 263- 269

- 45.Erick J. Rodríguez-Seda, Dušan M. Stipanović Chapter 4 - Guaranteed Collision Avoidance with Discrete Observations and Limited Actuation // Advances in Intelligent Vehicles. Academic Press, 2014, p. 89-110.  
<https://doi.org/10.1016/B978-0-12-397199-9.00004-5>
- 46.Becker M. et al Obstacle avoidance procedure for mobile robots. ABCM Symposium series in Mechatronics. Vol. 2, 2006. P. 250-257.
- 47.Nowakowski M., Kurylo, J. Usability of Perception Sensors to Determine the Obstacles of Unmanned Ground Vehicles Operating in Off-Road Environments. Appl. Sci. 2023, 13, 4892. <https://doi.org/10.3390/app13084892>
- 48.Yasin, J.N., Mohamed, S.A.S., Haghbayan, MH. et al. Low-cost ultrasonic based object detection and collision avoidance method for autonomous robots. Int. j. inf. tecnol. 13, 97–107 (2021). <https://doi.org/10.1007/s41870-020-00513-w>
- 49.Звенігородський О.С. Інтелектуальна система планування тактики руху автономного робота в квазістаціонарному середовищі: Дис. канд. техн. наук: 05.13.23. — Д., 2002. — 127с.
- 50.Адамів, О.П. Моделі та інтелектуальні засоби адаптивного керування автономним мобільним роботом. Дис. канд.техн. наук. Одеса, 2007. 124 с.
- 51.Koval V., Adamiv O., Proc. of the Third IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2005). – Sofia (Bulgaria). – 2005. – P. 120- 124.
- 52.Kojima T. et al. Intelligent Technology for More Advanced Autonomous Driving. 2018. Hitachi Review Vol. 67, No. 1. p. 58–63
- 53.Pandey A, Pandey S, Parhi DR. Mobile robot navigation and obstacle avoidance techniques: A review. Int Rob Auto J. 2017;2(3):96-105. DOI: 10.15406/iratj.2017.02.00023
- 54.Liu Q, Lu YG, Xie C. Optimal Genetic Fuzzy Obstacle Avoidance Controller of Autonomous Mobile Robot Based on Ultrasonic Sensors. IEEE International Conference on Robotics and Biomimetics. 2006. p. 125–129.

- 55.Mamchur, D.; Peksa, J.; Kolodinskis, A.; Zigunovs, M. The Use of Terrestrial and Maritime Autonomous Vehicles in Nonintrusive Object Inspection. *Sensors* 2022, 22, 7914. <https://doi.org/10.3390/s22207914>
- 56.McGuire K. et al. A Comparative Study of Bug Algorithms for Robot Navigation. *Robotics and Autonomous Systems*. Vol. 121, 2019. <https://doi.org/10.1016/j.robot.2019.103261>
- 57.Саліхов М. М. Від автоматизованих до автономних транспортних засобів // Наукові досягнення та інновації: шлях до успіху. X Всеукраїнська мультидисциплінарна науково-практична Інтернет-конференція, 31 травня 2023, Україна, Київ : зб. матеріалів — Електрон. дан. — Київ : Ярочé нко Я. В., 2023. — с. 136-142.
- 58.Саліхов М. М. Технологічні бар'єри розвитку безпілотних транспортних засобів // зб.наук.пр. Міжнародної науково-практичної конференції «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії», 01- 03 червня 2023 р., м. Київ. С.101-102.
- 59.Саліхов М.М. Самокеровані автомобілі та системи їх навігації// Практичні та теоретичні питання розвитку науки та освіти: матеріали VIII Міжнародної науково-практичної конференції: м. Львів, 19-20 червня 2023 року. – Львів: Львівський науковий форум, 2023. – С.53-59.

## ДОДАТОК А

### Опис основних функціональних блоків

У програмі для робота використовуються дві бібліотеки: AFMotor і Servo. Розглянемо їх як окремі функціональні блоки:

Бібліотека AFMotor - надає функціональність для керування DC-моторами та кроковими двигунами через моторний щиток на базі чіпа L293D.

Ця бібліотека надає змогу контролювати швидкість, напрямок та зупинку моторів нашого робота. Нижче наведений приклад коду для ініціалізації об'єкту AFMotor і керування двигуном:

```
#include <AFMotor.h> //Імпорт бібліотеки для керування моторним шилдом
```

```
AF_DCMotor motor(1); // Створення об'єкту для першого мотора
```

```
void setup() {
```

```
// Ініціалізація моторного щитка
```

```
motor.setSpeed(255); // Встановлення швидкості мотора
```

```
}
```

```
void loop() {
```

```
motor.run(FORWARD); // Рух уперед
```

```
delay(1000);
```

```
motor.run(RELEASE); // Зупинка мотора
```

```
delay(1000);
```

```
}
```

У цьому прикладі використано об'єкт AF\_DCMotor для управління першим мотором (підключеного до піна M1 на моторному щитку). За допомогою методів setSpeed та run ми встановлюємо швидкість та напрямок руху мотора.

Бібліотека Servo має низку особливостей. Бібліотека Servo дозволяє керувати сервоприводами, які використовуються для точного кутового позиціонування. За допомогою цієї бібліотеки ми можемо встановлювати кут повороту сервопривода та виконувати плавні рухи. Ось приклад коду для керування сервоприводом:

```

#include <Servo.h>

Servo servoMotor; // Створення об'єкту для сервоприводу

void setup() {
  // Підключення сервоприводу до піна 9
  servoMotor.attach(9);
}

void loop() {
  servoMotor.write(90); // Встановлення кута повороту 90 градусів
  delay(1000);
  servoMotor.write(0); // Встановлення кута повороту 0 градусів
  delay(1000);
}

```

В цьому прикладі ми використовуємо об'єкт Servo для керування сервоприводом, підключеного до піна 9 на Arduino. За допомогою методу write ми встановлюємо кут повороту сервопривода.

Обидві бібліотеки дуже корисні для керування рухом та позиціонування різних компонентів робота, і їх використання в програмі Arduino дозволяє досягти потрібної функціональності та керованості роботом.

Нижче подано методологію написання програми.

I. Імпорт бібліотек використаних в роботі.

Крок 1. По-перше, код має включати всі бібліотеки, необхідні для роботи.

Код програми:

```

#include <AFMotor.h> //Імпорт бібліотеки для керування моторним
шилдом

```

```

#include <Servo.h> //Імпорт бібліотеки для керування сервоприводом

```

Крок 2: Створення об'єктів для керування окремими двигунами та сервоприводом. У цьому блоку коду створюються об'єкти для керування кожним окремим двигуном та сервоприводом. Кожен об'єкт відповідає конкретному мотору або сервоприводу із підключених компонентів до Arduino.

Код програми:

AF\_DCMotor RBMotor(1); //Створюються об'єкти для керування кожним окремим двигуном

AF\_DCMotor RFMotor(2);

AF\_DCMotor LFMotor(3);

AF\_DCMotor LBMotor(4);

Servo servoWatch; //Створюється об'єкт для керування сервоприводом

Крок 3: Встановлення параметрів системи. Цей блок коду визначає параметри, які використовуються для налаштування системи руху робота.

Параметри ультразвукового датчика:

- byte TRIG\_PIN = 2; - призначення пін для вихідного сигналу (тригер) ультразвукового датчика.

- byte ECHO\_PIN = 13; - призначення пін для вхідного сигналу (ехо) ультразвукового датчика.

- byte distWatch = 150; - встановлення максимальної відстані, на яку датчик реагує. Об'єкти, що знаходяться далі, ігноруються.

- byte rangeStop = 50; - встановлення мінімальної відстані до об'єкту, при якій рух зупиняється (у сантиметрах).

- float timeEcho = 2\*(distWatch+10)/100/340\*1000000; - обчислення максимального часу очікування зворотного сигналу. Цей час використовується для визначення, коли датчик вважає, що немає перешкоди.

Параметри двигуна наступні:

- byte MAX\_SPEED = 100; - встановлення максимальної швидкості двигуна.

- int OFFSET\_SPEED = 10; - коефіцієнт, який враховує різницю у потужності між двигунами по різних сторонах.

Параметри повороту:

- TURN\_SPEED = 50; - сума, яка додається до швидкості двигуна під час повороту. Це дозволяє роботу повертатись.

Код програми:

byte TRIG\_PIN = 2; //Призначення пінів ультразвукового датчика

```
byte ECHO_PIN = 13;  
byte distWatch = 150; //Максимальна відстань, на яку датчик реагує(об'єкти  
що знаходяться далі ігноруються)  
byte rangeStop = 50; //Мінімальна відстань до об'єкту, при якій рух  
зупиняється (в сантиметрах)  
float timeEcho = 2*(distWatch+10)/100/340*1000000; //Максимальний час  
очікування зворотного сигналу byte MAX_SPEED = 100; //Максимальна  
швидкість двигуна int OFFSET_SPEED = 10; //Коефіцієнт для врахування того,  
що одна сторона є більш потужною int TURN_SPEED = 50; //Сума, яка додається  
до швидкості двигуна під час повороту
```

Крок 4: Налаштування і початкова ініціалізація.

У цьому блоку коду виконуються деякі початкові налаштування і ініціалізація компонентів системи. Цей код виконується один раз при запуску програми (у функції setup()).

Налаштування швидкостей двигунів: rightBack.setSpeed(motorSpeed); - встановлює швидкість для правого заднього двигуна.

rightFront.setSpeed(motorSpeed); - встановлює швидкість для правого переднього двигуна.

leftFront.setSpeed(motorSpeed+motorOffset); - встановлює швидкість для лівого переднього двигуна з додаванням коефіцієнта motorOffset.

leftBack.setSpeed(motorSpeed+motorOffset); - встановлює швидкість для лівого заднього двигуна з додаванням коефіцієнта motorOffset.

Зупинка всіх моторів:

RBMotor.run(RELEASE); - забезпечує зупинку правого заднього мотора.

RFMotor.run(RELEASE); - забезпечує зупинку правого переднього мотора.

LFMotor.run(RELEASE); - забезпечує зупинку лівого переднього мотора.

LBMotor.run(RELEASE); - забезпечує зупинку лівого заднього мотора.

Ініціалізація сервоприводу:

servoWatch.write(90); - призначає пін 10 для сервоприводу, що дозволяє керувати його положенням.



Налаштування пінів ультразвукового датчика:

`pinMode(TRIG_PIN,OUTPUT);` - встановлює режим вихідного піна `trig` у режим "OUTPUT" (вихідний).

`pinMode(ECHO_PIN,INPUT);` - встановлює режим вхідного піна `echo` у режим "INPUT" (вхідний).

Крок 5: Основний цикл програми.

У цьому кроці я описую функцію `loop()`, яка є основним циклом програми.

Цей код виконується безперервно після ініціалізації (у функції `setup()`).

У циклі `loop()` зазвичай розміщується основний алгоритм керування роботом.

Залежно від контексту програми, цей алгоритм може включати рух, взаємодію з датчиками, прийом команд зовнішнього контролера тощо.

В цьому циклі можуть бути використані різні команди, функції та умовні оператори для керування роботом залежно від поставленої задачі. Цей код повторюється безкінечно, доки програма не завершиться або не буде вручну зупинений.

Крок 6: Виконання основного алгоритму керування в циклі `loop()`.

У цьому кроці виконується основна логіка керування роботом на основі отриманих відстаней та виконання необхідних дій.

Блок коду 1: Перевірка передньої відстані

`servoWatch.write(90);` //Встановлення сервоприводу на погляд прямо вперед

`delay(750);`

`int distance = getRange();` //Перевірка, що попереду немає предметів

`if(distance >= rangeStop)` //Якщо в межах шляху немає предметів -

рух вперед

{

`movementForward();`

}

Блок коду 2: Зупинка руху

while(distance >= rangeStop) //Продовження перевірки відстані до об'єкту,  
поки вона не стане менше мінімальної відстані зупинки

```
{  
distance = getRange();  
delay(250);  
}  
stopMove(); //Зупинка двигунів
```

Блок коду 3: Визначення напрямку повороту

```
int directionTurn = checkDir(); //Перевірка відстані до об'єктів зліва і справа  
та отримання інструкції щодо повороту
```

```
Serial.print(directionTurn);  
switch (directionTurn) //Поворот ліворуч, праворуч або здійснити розворот  
залежно від інструкції
```

```
{  
case 0: //Поворот ліворуч  
turnLeft (400);  
break;  
case 1: //Розворот  
turnLeft (700);  
break;  
case 2: //Поворот праворуч  
turnRight (400);  
break;  
}
```

- Функція accelerate(): Прискорення двигунів від 0 до повної швидкості.
- Функція decelerate(): Уповільнення двигунів від повної швидкості до нуля.
- Функція movementForward(): Встановлення всіх двигунів на рух вперед.
- Функція stopMove(): Встановлення всіх двигунів на зупинку.

- Функція turnLeft(): Встановлення двигунів на поворот ліворуч на вказаний час і зупинка їх після повороту.
- Функція turnRight(): Встановлення двигунів на поворот праворуч на вказаний час і зупинка їх після повороту.
- Функція getRange(): Вимірювання відстані до перешкоди за допомогою ультразвукового датчика.
- Функція checkDir(): Перевірка відстаней до об'єктів зліва і справа, прийняття рішення щодо напрямку повороту.

Блок коду 4: Функція accelerate()

void accelerate() //Функція для прискорення двигунів від 0 до повної швидкості

```
{
for (int i=0; i<MAX_SPEED; i++) //Цикл від 0 до повної швидкості
{
RBMotor.setSpeed(i); //Встановлення двигунів на поточну швидкість циклу
RFMotor.setSpeed(i);
LFMotor.setSpeed(i+OFFSET_SPEED);
LBMotor.setSpeed(i+OFFSET_SPEED);
delay(10);
}
}
```

Блок коду 5: Функція decelerate()

void decelerate() //Функція уповільнення двигунів від повної швидкості до нуля

```
{
for (int i=MAX_SPEED; i!=0; i--) //Цикл від повної швидкості до 0 {
RBMotor.setSpeed(i); //Встановлення двигунів на поточну швидкість циклу
RFMotor.setSpeed(i);
LFMotor.setSpeed(i+OFFSET_SPEED);
LBMotor.setSpeed(i+OFFSET_SPEED);
}
```

```
delay(10);
```

```
}
```

```
}
```

Блок коду 6: Функція movementForward()

```
void movementForward() //Встановлення всіх двигунів на рух вперед
```

```
{
```

```
RBMotor.run(FORWARD);
```

```
RFMotor.run(FORWARD);
```

```
LFMotor.run(FORWARD);
```

```
LBMotor.run(FORWARD);
```

```
}
```

Блок коду 7: Функція stopMove()

```
void stopMove() //Встановлення всіх двигунів на зупинку
```

```
{
```

```
RBMotor.run(RELEASE);
```

```
RFMotor.run(RELEASE);
```

```
LFMotor.run(RELEASE);
```

```
LBMotor.run(RELEASE);
```

```
}
```

Блок коду 8: Функція turnLeft()

```
void turnLeft(int duration) //Встановлення двигунів на поворот ліворуч на  
вказаний час і зупинити їх
```

```
{
```

```
RBMotor.setSpeed(MAX_SPEED+TURN_SPEED); //Встановлення всіх  
двигунів на задану швидкість
```

```
RFMotor.setSpeed(MAX_SPEED+TURN_SPEED);
```

```
LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
```

```
LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
```

```
RBMotor.run(FORWARD);
```

```
RFMotor.run(FORWARD);
```

```

LFMotor.run(BACKWARD);
LBMotor.run(BACKWARD);
delay(duration);
RBMotor.setSpeed(MAX_SPEED); //Встановлення всіх двигунів на задану
швидкість
RFMotor.setSpeed(MAX_SPEED);
LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED);
LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED);
RBMotor.run(RELEASE);
RFMotor.run(RELEASE);
LFMotor.run(RELEASE);
LBMotor.run(RELEASE);
}

```

Блок коду 9: Функція turnRight()

```

void turnRight(int duration) //Встановлення двигунів на поворот праворуч на
вказаний час і зупинити їх
{
    RBMotor.setSpeed(MAX_SPEED+TURN_SPEED); //Встановлення всіх
двигунів на задану швидкість
    RFMotor.setSpeed(MAX_SPEED+TURN_SPEED);
    LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
    LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
    RBMotor.run(BACKWARD);
    RFMotor.run(BACKWARD);
    LFMotor.run(FORWARD);
    LBMotor.run(FORWARD);
    delay(duration);
    RBMotor.setSpeed(MAX_SPEED); //Встановлення всіх двигунів на задану
швидкість
    RFMotor.setSpeed(MAX_SPEED);

```

```

LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED);
LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED);
RBMotor.run(RELEASE);
RFMotor.run(RELEASE);
LFMotor.run(RELEASE);
LBMotor.run(RELEASE);
}

```

Блок коду 10: Функція getRange()

Вимірює відстань до об'єкта за допомогою ультразвукового сенсора.

```

int getRange() //Вимір відстані до предмета
{
    unsigned long pulseDuration; //Створення змінної для зберігання часу шляху
імпульсу
    int distance; //Створення змінної для зберігання обчисленої відстані
    digitalWrite(TRIG_PIN, HIGH); //Генерація імпульсу тривалістю 10
мікросекунд
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    pulseDuration = pulseIn(ECHO_PIN, HIGH, timeEcho); //Вимір часу
повернення імпульсу
    distance = (float)pulseDuration * 340 / 2 / 10000; //Обчислення відстані до
об'єкта на основі часу імпульсу
    return distance;
}

```

У цій функції спочатку генерується короткий імпульс на пін trig для запуску ультразвукового сенсора. Потім вимірюється час pulseTime, який потрібен для повернення ехо-сигналу від об'єкта до сенсора. На основі цього часу обчислюється відстань до об'єкта з урахуванням швидкості звуку. Отримане значення відстані повертається як результат функції.

Ця функція допомагає роботу визначати відстань до перешкод і приймати рішення про подальші дії на основі цих вимірів.

Блок коду 11: Функція checkDir ()

Перевіряє відстані в лівому та правому напрямках, щоб визначити, у якому напрямку повернути робота.

```
int checkDir() //Перевірка напрямку ліворуч і праворуч, прийняти рішення,
у який бік повернути
{
    int rangeArray [2] = {0,0}; //Відстань ліворуч і праворуч
    int directionTurn = 1; //Напрямок повороту, 0 ліворуч, 1 розворот, 2 праворуч
    servoWatch.write(180); //Поворот серво, щоб подивитися ліворуч
    delay(500);
    rangeArray [0] = getRange(); //Отримати відстань до об'єкта зліва
    servoWatch.write(0); //Поворот серво, щоб подивитися праворуч
    delay(1000);
    rangeArray [1] = getRange(); //Отримати відстань до об'єкта справа
    if (rangeArray[0]>=200 && rangeArray[1]>=200) //Якщо обидва напрямки
    вільні - поворот ліворуч
        directionTurn = 0;
    else if (rangeArray[0]<=rangeStop && rangeArray[1]<=rangeStop) //Якщо
    обидва напрямки заблоковані, розворот
        directionTurn = 1;
    else if (rangeArray[0]>rangeArray[1]) //Якщо ліворуч більше місця,
    поворот ліворуч
        directionTurn = 0;
    else if (rangeArray[0]<rangeArray[1]) //Якщо праворуч більше місця,
    поворот праворуч directionTurn = 2;
    return directionTurn;
}
```

У цій функції спочатку встановлюється сервопривод на позицію, щоб подивитися в лівий напрямок. Потім вимірюється відстань до об'єкта в лівому напрямку за допомогою функції `getDistance()` і зберігається в `rangeArray[0]`.

Потім сервопривод повертається в позицію праворуч, щоб подивитися в правий напрямок. Після цього вимірюється відстань до об'єкта в правому напрямку за допомогою функції `getRange()` і зберігається в `rangeArray[1]`.

Після цього виконується порівняння отриманих відстаней. Якщо обидва напрямки мають відстань більше або рівну 200, то поворот здійснюється ліворуч (`directionTurn = 0`). Якщо обидва напрямки мають відстань менше або рівну `rangeStop`, то здійснюється розворот (`directionTurn = 1`). Якщо відстань зліва більше, ніж відстань справа, то поворот виконується ліворуч (`directionTurn = 0`).

Якщо відстань зліва менше, ніж відстань справа, то поворот виконується праворуч (`turnDir = 2`).

На основі цих перевірок визначається напрямок повороту `directionTurn`, який повертається як результат функції.



## ДОДАТОК Б

```
* Arduino Robot Car with Motor Shield, UltraSonic distance sensor, Bluetooth remote control
* Використовуються бібліотеки:
* Sweep by BARRAGAN <http://barraganstudio.com> modified 8 Nov 2013 by Scott
Fitzgerald http://arduino.cc/en/Tutorial/Sweep
* Adafruit Motor shield library copyright Adafruit Industries LLC, 2009
*/
#include <AFMotor.h>
#define VERSION "RoboCar4W ver.2015.02.12"
/*
* Рівень налагодження.
* Чим більше, тим докладніше.
* поки що використовуються рівні:
* Більше 1 - видача налагоджувальних повідомлень
* Більше 5 - реально мотори не включаються і дистанція не заміряється, а генерується
рандомно
*/
// визначаємо мотори
AF_DCMotor motorFrontLeft(2); // Передній лівий
AF_DCMotor motorFrontRight(1); // передній правий
AF_DCMotor motorRearLeft(3); // задній лівий
AF_DCMotor motorRearRight(4); // задній правий
const byte SPEED_MIN = 100; // мінімальна швидкість моторів, якщо менше – мотори
зупиняться.
const byte SPEED_MAX = 255; // максимальна швидкість моторів
byte SPEED_CURRENT = 0; // поточна швидкість моторів
/*
* Види поворотів
*/
const byte MOTOR_ROTATE_RIGHT = 0; // Праворуч різкий розворот на місці (всі ліві
коління крутяться вперед, всі праві - назад)
const byte MOTOR_TURN_RIGHT = 1; // праворуч плавний поворот
const byte MOTOR_ROTATE_LEFT = 2; // Вліво різкий розворот на місці
const byte MOTOR_TURN_LEFT = 3; // Ліворуч плавний поворот
const byte MOTOR_TURN_BACK_RIGHT = 4; // Поворот праворуч заднім ходом
const byte MOTOR_TURN_BACK_LEFT = 5;

// У сантиметрах (distance threshold) Пороги відстаней до перешкоди
// Якщо ближче, то зупинка
const int DST_STOP = 20;
boolean SAFE_DISTANCE = true; // Вимір дистанції та безпечний рух. Якщо попереду
близька перешкода, то зупинка
int distance = 0;

/* Піни для підключення HC-SR04 Ultrasonic Module Distance Measuring
* 13, 2 цифрові піни
* 14, 15 аналогові піни A0 та A1 відповідно
*/
#define SONIC_PIN_TRIG 14 //13
#define SONIC_PIN_ECHO 15 //2
// Detection distance: 2cm--450cm
```

```

const int SONIC_DISTANCE_MAX = 450;
const int SONIC_DISTANCE_MIN = 2;

// для управління з блютуз
char btCommand = 'S';
// лічильники визначення втрати зв'язку з блютуз
unsigned long btTimer0 = 2000; //Stores the time (in millis since execution started)
unsigned long btTimer1 = 0;    //Stores the time when the last command was received from
the phone

/*****
Main program
*****/
void setup() {
  Serial.begin(9600);
  pinMode(SONIC_PIN_TRIG, OUTPUT);
  pinMode(SONIC_PIN_ECHO, INPUT);
  motorInit();
}
void loop() {
  if (Serial.available() > 0) {
    btTimer1 = millis();
    btCommand = Serial.read();
    //Serial.println(btCommand);
    switch (btCommand){
      case 'F':
        motorRunForward();
        break;
      case 'B':
        motorRunBack();
        break;
      case 'L':
        motorRotateLeft();
        break;
      case 'R':
        motorRotateRight();
        break;
      case 'S':
        motorStop();
        break;
      case 'I': //FR
        motorTurnRight();
        break;
      case 'G': //FL
        motorTurnLeft();
        break;
      case 'J': //BR
        motorTurnBackRight();
        break;
      case 'H': //BL
        motorTurnBackLeft();
        break;
      case 'W':

```

```

    roboCarDemo();
    break;
/*    case 'w': //Lights OFF
        break;
    case 'U': //Back ON
        break;
    case 'u': //Back OFF
        break;
*/    case 'D': //Everything OFF
        motorStop();
        break;
    case 'V':
        SAFE_DISTANCE = true;
        break;
    case 'v':
        SAFE_DISTANCE = false;
        break;
/*    case 'X':
        break;
    case 'x':
        break;
*/    default: //Get SPEED_CURRENT
        if ( btCommand == 'q' ){
            motorSetSpeed(SPEED_MAX); //Full SPEED
        } else{
            //Chars '0' - '9' have an integer equivalence of 48 - 57, accordingly.
            if ( (btCommand >= 48) && (btCommand <= 57) ) {
                // Subtracting 48 changes the range from 48-57 to 0-9.
                // Multiplying by 25 changes the range from 0-9 to 0-225.
                motorSetSpeed( SPEED_MIN + (btCommand - 48) * 15 );
            }
        } // else
    } // switch
} // if (Serial.available() > 0)
else{
    btTimer0 = millis(); //Get the current time (millis since execution started).
    //Check if it has been 500ms since we received last btCommand.
    if ((btTimer0 - btTimer1) > 500) {
        //More tan 500ms have passed since last btCommand received, car is out of range.
        //Therefore stop the car and turn lights off.
        motorStop();
    }
}
}

/*****
Functions
*****/
// ініціалізація моторів
void motorInit() {
    // turn on motor
    motorSetSpeed(100); // швидкість мотора 0-255, реально менше 100 не працює. Див
SPEED_MIN

```

```

    motorStop();
}

// чи безпечно їхати вперед?
boolean isSafeDistance() {
    if ( SAFE_DISTANCE ) {
        // замір відстані
        distance = measureDistance();
        // перешкода так близько що треба їхати назад?
        if ( distance < DST_STOP ) {
            return false;
        }
    } else {
        return true;
    }
}

// рух уперед по прямій
// якщо виявлено перешкоду, машина зупиняється і повертається FALSE, інакше -
TRUE
boolean motorRunForward() {
    if ( !isSafeDistance() ) {
        motorStop();
        return false;
    }
    motorFrontLeft.run(FORWARD);
    motorFrontRight.run(FORWARD);
    motorRearLeft.run(FORWARD);
    motorRearRight.run(FORWARD);
    return true;
}

// рух назад по прямій
void motorRunBack() {
    motorFrontLeft.run(BACKWARD);
    motorFrontRight.run(BACKWARD);
    motorRearLeft.run(BACKWARD);
    motorRearRight.run(BACKWARD);
}

// правий розворот на місці
void motorRotateRight() {
    motorFrontLeft.run(FORWARD);
    motorFrontRight.run(BACKWARD);
    motorRearLeft.run(FORWARD);
    motorRearRight.run(BACKWARD);
}

// правий плавний поворот (під час руху вперед)
// якщо виявлено перешкоду, машина зупиняється і повертається FALSE, інакше -
TRUE
boolean motorTurnRight() {
    if ( !isSafeDistance() ) {
        motorStop();
    }
}

```

```

        return false;
    }
    motorFrontLeft.run(FORWARD);
    motorFrontRight.run(RELEASE);
    motorRearLeft.run(FORWARD);
    motorRearRight.run(RELEASE);
    return true;
}

// правий плавний поворот (під час руху назад)
void motorTurnBackRight() {
    motorFrontLeft.run(BACKWARD);
    motorFrontRight.run(RELEASE);
    motorRearLeft.run(BACKWARD);
    motorRearRight.run(RELEASE);
}

// лівий розворот на місці
void motorRotateLeft() {
    motorFrontLeft.run(BACKWARD);
    motorFrontRight.run(FORWARD);
    motorRearLeft.run(BACKWARD);
    motorRearRight.run(FORWARD);
}

// лівий плавний поворот (під час руху вперед)
// якщо виявлено перешкоду, машина зупиняється і повертається FALSE, інакше -
TRUE
boolean motorTurnLeft() {
    if ( !isSafeDistance() ) {
        motorStop();
        return false;
    }
    motorFrontLeft.run(RELEASE);
    motorFrontRight.run(FORWARD);
    motorRearLeft.run(RELEASE);
    motorRearRight.run(FORWARD);
    return true;
}

// лівий плавний поворот (під час руху назад)
void motorTurnBackLeft() {
    motorFrontLeft.run(RELEASE);
    motorFrontRight.run(BACKWARD);
    motorRearLeft.run(RELEASE);
    motorRearRight.run(BACKWARD);
}

// зупинка різко
void motorStop() {
    motorFrontLeft.run(RELEASE);
    motorFrontRight.run(RELEASE);
    motorRearLeft.run(RELEASE);
    motorRearRight.run(RELEASE);
}

```

```

// встановити швидкість 0-255
// см SPEED_MIN
void motorSetSpeed(int speed) {
    // швидкість мотору 0--255
    if (speed > SPEED_MAX)
        speed = SPEED_MAX;
    if (speed < SPEED_MIN)
        speed = SPEED_MIN;
    motorFrontLeft.setSpeed(speed);
    motorFrontRight.setSpeed(speed);
    motorRearLeft.setSpeed(speed);
    motorRearRight.setSpeed(speed);
    // запам'ятовуємо поточну швидкість
    SPEED_CURRENT = speed;
}

// Повертає відстань до перешкод у сантиметрах
int measureDistance() {
    long duration;
    int distance;
    /* Для запуску передавача потрібно подати на Trig сигнал тривалістю 10мкс.
    * Передавач який надсилає 8 коротких імпульсів з частотою 40kHz.
    * Приймач отримує відбитий сигнал і на вході Echo генерується сигнал,
    * Тривалість якого дорівнює часу проходження звукового сигналу.
    */
    digitalWrite(SONIC_PIN_TRIG, LOW); // ініціалізація перед виміром
    delayMicroseconds(5); // 3
    digitalWrite(SONIC_PIN_TRIG, HIGH);
    delayMicroseconds(12); // 10
    digitalWrite(SONIC_PIN_TRIG, LOW);

    duration = pulseIn(SONIC_PIN_ECHO, HIGH);
    // Швидкість звуку 340 м/с чи 29 мікросекунд на сантиметр.
    // Звук йде вперед і повертається назад, таким чином час треба ділити на два
    distance = duration/58; // = microseconds / 29 / 2

    if (distance < SONIC_DISTANCE_MIN ) // out of range
        return SONIC_DISTANCE_MIN;
    if (distance > SONIC_DISTANCE_MAX ) // out of range
        return SONIC_DISTANCE_MAX;

    return distance;
}

// правий або лівий дуже плавний поворот (при русі вперед)
// Поворот за рахунок різних швидкостей коліс
// SpeedL - швидкість лівих коліс
// SpeedR - швидкість правих коліс
// якщо (speedL > speedR) то поворот буде праворуч і навпаки
// якщо виявлено перешкоду, машина зупиняється і повертається FALSE, інакше -
TRUE
boolean motorSmoothTurn(byte speedL, byte speedR) {

```

```

if ( !isSafeDistance() ) {
motorStop();
return false;
}
// Пізні швидкості
// ліві мотори
motorFrontLeft.setSpeed(speedL);
motorRearLeft.setSpeed(speedL);
// Праві мотори
motorFrontRight.setSpeed(speedR);
motorRearRight.setSpeed(speedR);
// їхати вперед
motorFrontLeft.run(FORWARD);
motorFrontRight.run(FORWARD);
motorRearLeft.run(FORWARD);
motorRearRight.run(FORWARD);
return true;
}

// Демо програма в автоматичному (некерованому) режимі.
// Випишує фігуру, схожу на 8-ку
// У разі виявлення перешкоди машина зупиняється.
void roboCarDemo() {
SAFE_DISTANCE = true; // безпечний рух - зупинка для виявлення перешкоди

// Вперед
motorSetSpeed(200);
if (!motorRunForward()) // якщо виявлено перешкоду, то зупинка
return;
delay(500); // тимчасова затримка підбирається експериментально

// Розворот вліво
if ( !motorTurnLeft() ) // якщо виявлено перешкоду, то зупинка
return;
delay (2000);

// Вперед
motorSetSpeed(130);
if (!motorRunForward()) // якщо виявлено перешкоду, то зупинка
return;
delay(500);

// Розворот праворуч
if ( !motorTurnRight() ) // якщо виявлено перешкоду, то зупинка
return;
delay (2000);

// Вперед на максимум
motorSetSpeed(SPEED_MAX);
if (!motorRunForward()) // якщо виявлено перешкоду, то зупинка
return;
delay(500);

```

```
// зупинка  
motorStop();  
}
```