МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет радіоелектроніки та телекомунікації
(повне найменування інституту, факультету)

Кафедра інформаційних технологій електронних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти)

на тему «РОЗРОБКА КОНСТРУКЦІЇ, АЛГОРИТМІВ КЕРУВАННЯ ТА ЕЛЕМЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОНОМНОГО РОБОТА ДЛЯ РОБОТИ У СКЛАДІ СИСТЕМ САМОВПОРЯДКУВАННЯ»

Виконав: студент 4 курсу, групи РТ-518сп

Спеціальності 172 Телекомунікації та радіотехніка
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інтелектуальні технології мікросистемної радіоелектронної техніки

Чухрай А.А.
(прізвище та ініціали)

Керівник Фарафонов О.Ю.
(прізвище та ініціали)

Рецензент Неласа Г.В.
(прізвище та ініціали)

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет Інститут інформатики та радіоелектроніки,
Факультет радіоелектроніки та телекомунікацій
Кафедра інформаційних технологій електронних засобів
Ступінь вищої освіти бакалавр
Спеціальність 172 «Телекомунікації та радіотехніка»
(код і найменування)
Освітня програма (спеціалізація) Інтелектуальні технології мікросистемної радіоелектронної техніки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
В.о. зав. Каф. ІТЕЗ Огренич Є.В.,
канд. техн. наук
«31» 05 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Чухрай Артура Андрійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка конструкції, алгоритмів керування та елементів програмного забезпечення автономного робота для роботи у складі систем самовпорядкування
керівник проєкту (роботи) Фарафонов Олексій Юрійович, канд. техн. наук, доцент, доцент кафедри інформаційних технологій електронних засобів
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «26» квітня 2021 року №161
2. Строк подання студентом проєкту (роботи) 7 червня 2021 року
3. Вихідні дані до проєкту (роботи): рекомендована література, технічне завдання

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1 Аналітична частина. 2 Розробка схеми, вибір елементів та алгоритм дії. 3 Розробка програмного забезпечення
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
18 рисунків; 1 таблиця; презентація роботи

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	Прийняв виконане завдання
Розділи 1-3	Фарафонов О.Ю., доц.	05.04	
Нормоконтроль	Поспеєва І.Є., ст. викладач	04.06	

7. Дата видачі завдання «26» квітня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Визначення тематики роботи та постановка задачі	1 тиждень	Виконано
2	Аналітична частина	2 тиждень	Виконано
3	Оформлення першого розділу пояснювальної записки	3 тиждень	Виконано
4	Розробка схеми, вибір елементів та алгоритм дії	4 тиждень	Виконано
5	Оформлення другого розділу пояснювальної записки	5 тиждень	Виконано
6	Розробка програмного забезпечення	6 тиждень	Виконано
7	Оформлення третього розділу пояснювальної записки	7 тиждень	Виконано
8	Оформлення пояснювальної записки	8 тиждень	Виконано
9	Нормоконтроль та рецензування	8 тиждень	Виконано
10	Захист роботи	9 тиждень	Виконано
11			
12			
13			
14			
15			

Студент _____ (підпис) Чухрай А.А. (прізвище та ініціали)

Керівник проєкту (роботи) _____ (підпис) Фарафонов О.Ю. (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 121с., 3 ч., 7 табл., 37 рис., 29 джерел.

Об’єкт дослідження — поведінка автономного робота у складі систем самовпорядкування.

Мета роботи — розробка конструкції, алгоритмів керування та елементів програмного забезпечення автономного робота для роботи у складі систем самовпорядкування.

Метод дослідження — обробка теоретичних відомостей, практична перевірка та створення програми за допомогою інтегрованого середовища розробки Arduino IDE.

Актуальність та проблематика полягає в тому, що автономні роботи все більше використовуються в різних технологічних процесах на виробництві у всьому світі, для звільнення людини від виконання нетворчої, механічної чи небезпечної роботи. Крім того, розробка автономних роботів є перспективним напрямом сучасних науково-технічних досліджень у машинобудуванні, транспорті, медицині, космічній техніці тощо

Запропонована методика дозволяє подивитися як реалізований зв’язок між роботами за допомогою емуляції послідовного порту по повітрю.

РОБОТОТЕХНІКА, АВТОНОМНИЙ РОБОТ, КЕРУВАННЯ, ЕМУЛЯЦІЯ, ВЗАЄМОДІЯ, АЛГОРИТМ, РОЗПІЗНАВАННЯ, СКАНУВАННЯ, ПРОГРАМА.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Скорочення	Пояснення скорочення
ШІ	Штучний інтелект
АР	Автономний робот
ДВ	Давач відстані
ІЧ	Інфрачервоний
МП	Мікропроцесор
МК	Мікроконтролер
ШІМ	Широтно-імпульсна модуляція
UART	Універсальний асинхронний інтерфейс
SPI	Послідовний перефирійний інтерфейс
I2C	Послідовна асиметрична шина
I2S	Цифровий протокол передачі звуку
USB	Універсальна послідовна шина
РК	Ройове керування
АЦП	Аналогово-цифровий перетворювач
ЦАП	Цифро-аналоговий перетворювач

ВСТУП

З розвитком технічного прогресу суспільства, автономні роботи знаходять усе більш широке застосування для виконання різних задач в умовах, коли присутність людини в зоні їх роботи неможлива чи не бажана. Також вони набувають велику популярність в буденному житті людини. Для прикладу всім відомий робот пилосос, газонокосарка і тому подібне. Автономні мобільні роботи в буденному житті як, правило мають набір давачів для аналізу навколишнього середовища, акумулятором і станцією для його заряджання також уособлюють в собі простоту дизайну, будучи складним механізмом, який здатні самостійно приймати рішення у певних поставлених для них задач. Чимале розповсюдження здобули промислові роботи, які стали головною технологічною базою машинобудівної, приладобудівної та електронних галузей світової промисловості. Зараз існує велика кількість працюючих промислових роботів, які виробляють промислові роботи для маніпулювання, зварювання, фарбування, упакування, шліфування, полірування й т.д. з великим спектром застосування й по точності, і по характері виконуваних операцій.

Актуальність автономних роботів полягає в тому, що вони все більше використовуються в різних технологічних процесах на виробництві у всьому світі, для звільнення людини від виконання нетворчої, механічної чи небезпечної роботи. Крім того, розробка автономних роботів є перспективним напрямом сучасних науково-технічних досліджень у машинобудуванні, транспорті, медицині, космічній техніці тощо.

На даний момент немає універсального (АР), який зможе виконувати задачі в різних галузях, як людина.

Однією із важливих проблем робототехніки є створення (АР) який зможе справлятись із завданнями у будь-якому навколишньому середовищі: на землі, під водою; у повітрі, під землею або в космосі.

(АР) повинен дотримуватись три закони робототехніки і повинні бути реалізовані такі процеси як:

- отримувати і обробляти самостійно інформацію про довкілля;
- працювати якнайдовший проміжок часу без людського втручання;
- переміщатися цілком чи переміщати якусь свою частину у просторі без людської допомоги.

В (АР) може також бути реалізовано (ШІ), який зможе навчатися чи здобувати нові уміння, наприклад, удосконалювати алгоритми для виконання своїх задач чи адаптація до змін у довкіллі. Також йому, усе ж, вимагають регулярного технічного обслуговування, як це роблять із іншими машинами.

В своєму розвитку пройшли шлях від простих механізмів, які виконують одну дію по шаблону, до сучасних роботів, які набагато складніші. Вони активно вдосконалюються, що зменшує потребу в постійному втручанню людини при виконання більшості задач. Сучасна електроніка вже давно реагує на зміну умов швидше і точніше, ніж це би зробив оператор.

У даній дипломній роботі розглянуто один із методів реалізації керування автономного робота для роботи у складі систем самовпорядкування.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Групова робототехніка

Це відносно новий підхід до координації систем багатьох роботів, кожен з яких є досить простим пристроєм. На відміну від просто розподілених робототехнічних систем, групова робототехніка підкреслює велику кількість роботів, а також передбачає масштабованість, тобто можливість швидкого переходу системи на роботу з іншою кількістю осіб. Бажана колективна поведінка виникає з взаємодії роботів між собою і з навколишнім середовищем [5]. Це прояв одного із законів системного аналізу - емерджентність, що полягає в тому, що система, що складається із сукупності об'єктів, може мати властивості, не властиві окремим об'єктам. Основні переваги при використанні колективу роботів:

- підвищення надійності (втрата частини членів колективу не впливає на працездатність всієї системи в цілому);
- гнучкість (здатність системи до реконфігурації);
- потенційна можливість розвитку і ускладнення вирішуваних завдань шляхом нарощування потужності колективу.

Сучасні програми використання колективної поведінки роботів досить багатогранні:

- командна робота роботів, що спільно виконують діагностику важкодоступних об'єктів;
- моніторинг навколишнього середовища;
- колективне рішення задач роботами-рятувальниками;
- розвідка і рекогностування;
- військові застосування;

- колективна обробка об'єктів навколишнього середовища;
- охоронні функції, патрулювання і ін.

Дуже важливо колективна взаємодія роботів тоді, коли ми маємо справу з міні- і мікроботами. Будучи досить обмеженими у своїх можливостях, ці роботи здатні вирішити поставлене завдання лише при їх масовому застосуванні. Прикладом такого застосування мікроботів може слугувати так звана "інтелектуальний (розумний) пил", коли з літака скидається "хмара" мікроботів, кожен з яких повинен виконувати деякі найпростіші функції, наприклад, збору інформації про покриття території або ураження об'єктів ворога [3].

Також існують приклади реалізації колективу роботів, що складається з моделей різного типу і призначення, кожна з яких призначена для вирішення свого класу завдань [9].

Існує цілий ряд специфічних проблем, характерних для колективної роботи роботів. Серед них можна відзначити такі, як:

- непередбачувана динаміка зовнішнього середовища аж до свідомої протидії;
- неповнота і суперечливість знань роботів (агентів) про стан зовнішнього середовища і про інших учасників;
- різноманіття варіантів шляхів досягнення мети, структур колективу, розподілу ролей тощо;
- розподілений та динамічний характер планування дій колективу;
- проблеми, пов'язані з тим, що колектив являє собою сукупність фізичних об'єктів, що діють в реальному складному середовищі (проблеми надійної комунікації, розподіл колективу в просторі та ін.);
- інші технічні проблеми (архітектура мережі, протоколи, операційні засоби тощо).

Дослідження в області колективної поведінки роботів можна розбити на наступні напрямки:

1. "Суворе" математичне рішення. Йдеться про дослідження в області теорії систем, створення формальних моделей і механізмів колективної поведінки.

2. Технології багатоагентних систем (БАС).

3. Імітаційне моделювання, тобто реалізація моделей суб'єктів (роботів), що взаємодіють, при цьому за основу беруться біологічні об'єкти. Сюди ж можна віднести і дослідження в області так званого штучного життя.

4. Ройові, бджолині і мурашині алгоритми. Це методи, що досліджують зовнішні, суто з феноменологічного боку поведінки живих організмів. Подібного роду методи і алгоритми лежать в основі ройового інтелекта.

5. Еволюційні методи. Основне завдання - реалізація еволюційним шляхом механізмів колективної взаємодії [6].

1.2 Стратегії керування колективом роботів

Ефективність застосування груп роботів багато в чому залежить від обраної стратегії керування. Розрізняють централізовані і децентралізовані стратегії.

1.2.1 Централізоване керування

При централізованій стратегії керування використовується центральний пристрій, якому доступна інформація про стан усіх роботів групи і навколишнього середовища, що їх оточує. Він оцінює поточну ситуацію і приймає рішення про подальшу поведінку роботів групи. Центральний пристрій може бути

розташований як за межами групи (наприклад, на пульті керування оператора), так і на одному з членів групи (Централізоване управління "з ведучим").

Централізовані стратегії мають хороші показники при керуванні невеликими групами роботів. при збільшенні чисельності, зростає навантаження на канал зв'язку і пристрої керування. Одним із способів вирішення є реалізація ієрархічної моделі, при якій група ділиться на підгрупи, кожна з яких керується своїм лідером. Група лідерів також має свого лідера і так далі, в залежності від кількості рівнів ієрархії. У разі фіксованих лідерів можлива ситуація, коли лідер вийшов з ладу, і група втратила керування.

1.2.2 Децентралізоване керування

До децентралізованих стратегій відносять колективні, зграйні і ройові стратегії керування.

При колективній стратегії керування кожен робот передає в канал зв'язку всю зібрану ним інформацію про навколишнє середовище і про свій стан, а також отримує з нього всю інформацію про стан інших особин колективу. Таким чином, інформаційний обмін в разі колективної стратегії здійснюється за принципом "все з усіма". При цьому кожен робот самостійно оцінює сформовану ситуацію і приймає рішення про свої подальші дії. Колективні стратегії дозволяють групі зберігати працездатність в разі виходу з ладу окремих членів колективу. Однак зростає навантаження на бортові обчислювальні системи роботів, так як їм необхідно оперативно обробляти всю отриману інформацію. Ця стратегія дозволяє будувати групи, що перевершують за розмірами групи з централізованим керуванням, проте, масштабованість залишається на досить високому рівні.

При зграйній стратегії керування відсутній виділений канал зв'язку між членами групи. Кожен робот самостійно збирає інформацію з доступного йому

навколишнього простору і приймає рішення про власну поведінку для досягнення спільної мети. Відсутність комунікації між окремими роботами робить цей підхід ефективним тільки на завданнях, які допускають розпаралелювання на незалежні незв'язні підзадачі. Головна перевага - легка масштабованість [10].

Найбільш перспективним і комплексним підходом є методи на основі (РК).

1.2.3 Ройове керування

(РК) являє собою новий підхід, який вивчає можливості побудови системи із сукупності автономних інтелектуальних агентів (роботів) для досягнення колективних цілей, які не можуть бути досягнуті окремим роботом або для яких колективне виконання поставленого завдання більш ефективно. Основною ідеєю (РК) є "ройовий інтелект" (PI, Swarm Intelligence), широко спостережуваний в природному світі: косяки риб, колонії бджіл, мурах та інші. Термін "ройовий інтелект" був введений в 1989 році в контексті дослідження системи клітинних "роботів". Ройовий інтелект - це система, що реалізує керування колективною поведінкою децентралізованими самоорганізованими однорідними об'єктами (рисунок 1.1). Він розглядається в теорії штучного інтелекту як метод "не до кінця формалізованої оптимізації". В зв'язку з великою зацікавленістю до даного методу було сформовано окремий напрям "ройова робототехніка".

Виділяють два види (РК) (рисунок 1.2). Перший з них можна віднести до самоорганізованих систем, характерні природним утворенням. Ознаки біологічної самоорганізації:

- кожна особина в рої не володіє індивідуальною свідомістю і свободою вибору;
- правила поведінки визначаються генетичним набором, закладеним в кожну особину;

- квазірозумна діяльність рою спрямована на збереження рою як цілого і забезпечення його безпеки;
- рій, як правило, складається з представників одного виду;
- ролі особин можуть відрізнятися (в колоніях мурашок, термітів, бджіл тощо).

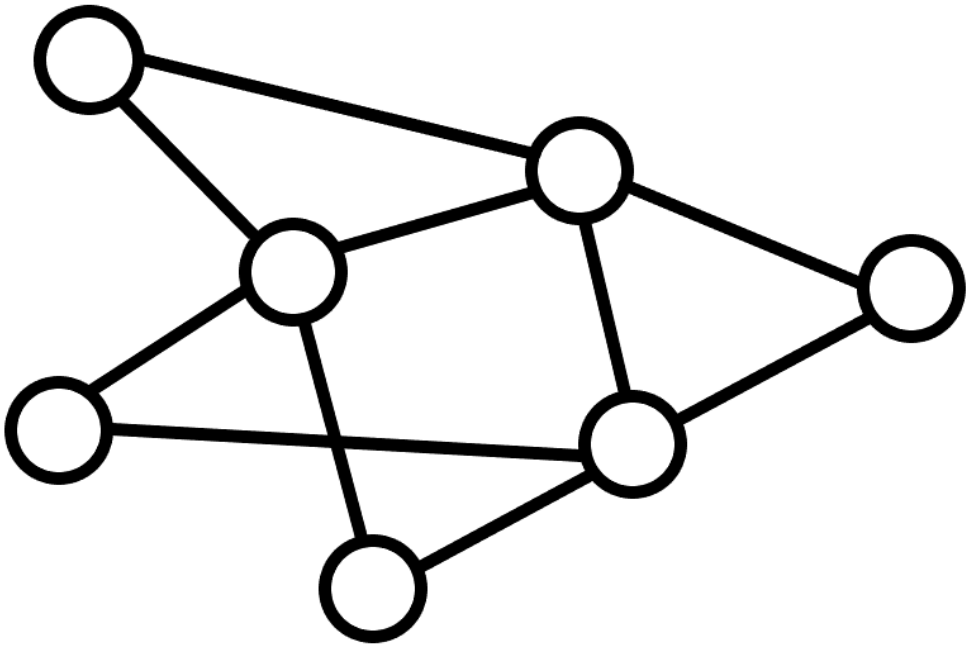


Рисунок 1.1 – Зв'язок між роботами при ройовій стратегії

Для такого рою немає будь-якої централізованої системи управління поведінкою кожної особини. Локальні і, в достатній мірі, випадкові взаємодії призводять до виникнення "квазіінтелектуальної" глобальної поведінки рою, яка може не контролюватися окремими агентами. В цьому випадку ми маємо багатоагентну систему, яка володіє самоорганізованою поведінкою і яка, сумарно, повинна проявляти деяку "розумну" поведінку.

Другий вид РІ передбачає наявність керування, причому керуючі дії можуть надходити як від зовнішнього, по відношенню до рою керуючої системи, так і від "призначеного" або "локального" лідера (агента), що знаходиться всередині рою. У першому випадку "призначений" лідер ("провідний робот")

виконує команди, що надходять від розпорядника майна центру, а решта агентів виконують дії, підкоряючись досить простим правилам. Наприклад, якщо агенти (роботи) переміщаються по площині із завданням утворити певну конфігурацію, то правила можуть бути наступними:

1. Розрахунок відстані від початкової точки;
2. Вибудовування в групу;
3. Освіта заданого градієнта щільності;
4. Локалізація.

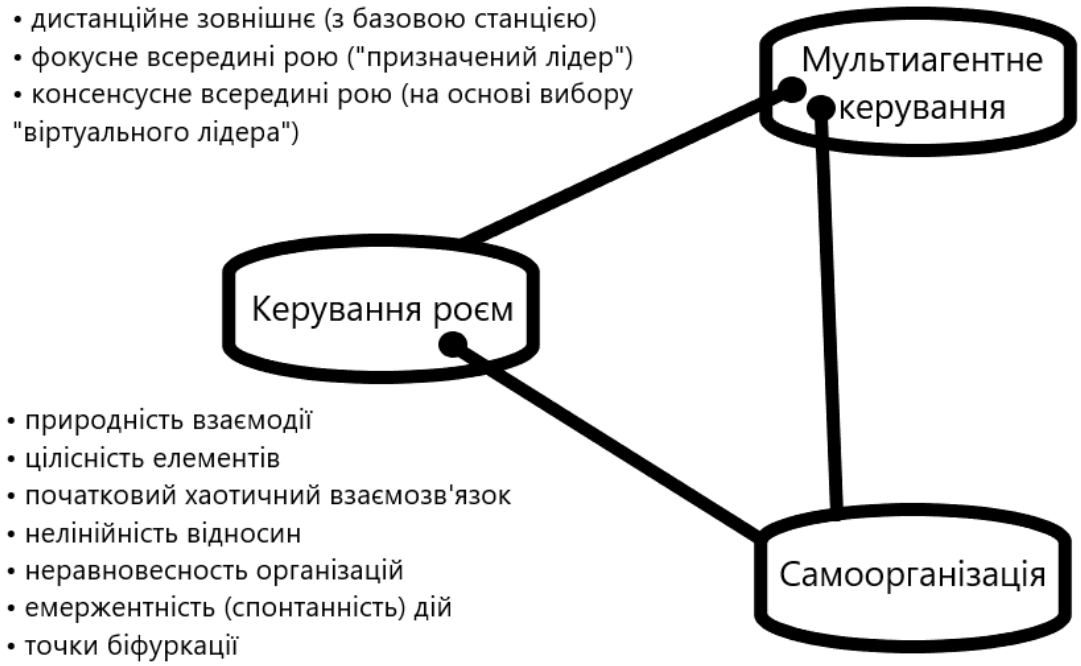


Рисунок 1.2 – Два види ройового керування

Якщо агенти переміщуються в просторі, то вони повинні:

- дотримуватися дистанції між собою і іншими об'єктами, а тому числі іншими особинами рою, а також дотримуватися заданого побудови;

- коригувати курс відповідно до руху сусідніх пристроїв і рою в цілому;
- підтримувати зв'язок з сусідніми апаратами.

Критерії, які є відмінними рисами ройової системи:

- агенти (роботи) є автономними, тобто здатні рухатися і взаємодіяти з навколишнім середовищем без централізованого керування;
- завдання має виконуватися в сукупності великою кількістю об'єктів, це означає, що система повинна бути розроблена з урахуванням масштабованості;
- рій складається з однорідних груп агентів (роботів), акцент робиться на велику кількість однакових об'єктів, ніж на централізовано-керовані гетерогенні об'єкти, де кожному агенту (роботу) призначена "персональна" роль.

Можливі стратегії керування ройовою системою:

- централізована - дистанційне керування з виділеною базовою станцією, лідер рою призначається з центрального вузла;
- децентралізована - лідер рою визначається на основі будь-якого алгоритму і не залежить від центральної керуючої станції;
- змішана - поєднує в собі переваги централізованої і децентралізованої стратегій шляхом виділення лідера рою на основі одного з алгоритмів з передачею прав управління оператору при необхідності [6].

Залежно від характеру інформаційного обміну в рої можливі два сценарії:

1. Робот, який знайшов ціль, повідомляє сусіднім роботам її координати, які передають цю інформацію по ланцюжку своїм сусідам, поки вона не стане відома всім роботам групи; потім вони змінюють траєкторію у напрямку до цілі;

2. Робот, який знайшов ціль, не може повідомити її координати іншим роботам рою; при цьому, він змінює траєкторію руху у напрямку до цілі; інші роботи, пов'язані з ним правилами допустимих дистанцій, йде за ним, тобто він стає "ведучим" роботом.

Незважаючи на всі переваги ройової робототехніки, вона володіє і недоліками:

- об'єктами управління є численні групи невеликих роботів, що вимагає наявності їх щодо недорогого масового виробництва;

- відсутність загальної теорії і підходів у створенні і розробці методів (РК) групами роботів. Кожне нове завдання зараз вирішується практично "з нуля". Значна частина досліджень присвячена використанню природних аналогів методів ройового інтелекту для вирішення технічних завдань (мурахи, бджоли, косяки риб, зграї птахів). Однак відмінності в завданнях і можливостях природних і технічних систем значно ускладнюють пошук і адаптацію природних алгоритмів до вирішення технічних завдань [10].

Массачусетський технологічний інститут США в 2016 році розробив і опублікував роботи по реалізації ройового інтелекту для керування колективом роботів (як наземного, так і повітряного) в умовах мінливого навколишнього середовища і динамічних перешкод [11].

1.3 Бездротові технології обміну даними для взаємодії групи роботів

Безперервний розвиток інформаційних технологій вимагає постійного збільшення ефективності обробки і передачі інформації. Очевидно, що при управлінні колективом мобільних роботів лінії передачі даних повинні мати мінімальний витрата енергії, мати високу пропускною спроможністю і, звичайно, повинні бути бездротовими. Бездротові канали передачі даних можна розділити по використовуваної середовищі поширення сигналу на оптичні і радіочастотні.

1.3.1 Оптичні канали

У багатьох колективах роботи взаємодіють між собою і центральним комп'ютером за допомогою (ІЧ) зв'язку [1-4].

Найпоширенішим стандартом для організації передачі інформації з відкритого (ІЧ) каналу є IrDA standart (Infrared Data Assotiation). Він дозволяє з'єднуватися з периферійним обладнанням на відстані до 1 метра в режимі точка-точка, використовуючи вузький (ІЧ) діапазон (850-900 нм з 880 нм «піком»). IrDA заснований на архітектурі комунікаційного COM-порту ПК, який використовує (UART) і працює зі швидкістю передачі даних 2400-115200 біт / с (бод).

Зв'язок в IrDA напівдуплексний, тому що переданий (ІЧ) промінь неминуче засвічує сусідній PIN-діодний підсилювач приймача. Повітряний проміжок між пристроями дозволяє прийняти (ІЧ) випромінювання тільки від одного джерела в один момент.

Для (ІЧ) випромінювання існує два джерела інтерференції (перешкод), основним з яких є сонячне світло. Але, завдяки переважанню в ньому постійної складової, правильно спроектовані приймачі здатні компенсувати великі постійні струми через PIN-діод.

Інше джерело перешкод - флуоресцентні лампи - часто застосовуються для загального освітлення. Для зниження впливу таких джерел приймачі повинні мати смуговий фільтр. Імовірність помилок зв'язку буде залежати від правильного вибору потужності передавача і чутливості приймача. У IrDA обрані значення, що гарантують, що описані вище перешкоди не будуть впливати на якість зв'язку [5].

Також покладаються великі надії на використання лазерних каналів передачі даних, іменованих як атмосферна оптична лінія зв'язку (АОЛЗ). Технологія базується на передачі даних модульованим випромінюванням в (ІЧ) частині спектра через атмосферу. Передавачем служить потужний напівпровідниковий лазерний діод. Стандартні системи АОЛЗ мають швидкість передачі даних від 6 Мб / с до 1.25 Гб / с. Довжина хвилі в більшості реалізованих систем варіюється в межах 700-950 нм або 1550 нм в залежності від

застосовуваного лазерного діода. Системи АОЛЗ з довжиною хвилі 700-950 нм надійні, економічно доцільні і придатні для більшості додатків, в тому числі для мереж 1 Гб / с Ethernet. Системи атмосферної оптичної лінії зв'язку з довжиною хвилі 1520-1600 нм підходять для передачі даних з більш високою потужністю і на великі відстані.

Якість роботи і саме функціонування систем бездротового зв'язку залежать від погодних умов і екологічних факторів. системи АОЛЗ складніше непомітно зламати, ніж радіочастотні системи, і в них можна вбудовувати контроль за спробою злому.

Коригування пучка є важливим фактором продуктивності системи АОЛЗ. Навіть при належній установці приймач АОЛЗ дуже сприйнятливий до переміщення або зміщення пучка світла. Зсув можуть викликати такі звичайні погодні явища, як вітер, але воно може також бути пов'язано зі зміною температури. Для коригування пучка світла в системах АОЛЗ застосовують два основних способи:

- вузький сфокусований пучок світла з автоматичним коректуванням зсуву;

- широкий пучок світла без коригування.

Системи з автоматичним коректуванням у стані усувати зміщення до того, як небажане зміщення призведе до порушення передачі. Відстань і швидкість передачі є головними факторами при визначенні необхідності автоматичного коректування. Короткі, до 200 метрів, лінії зі швидкістю передачі 10 Мб / с менш уразливі, ніж 500 метрові лінії зі швидкістю передачі 1.25 Гб / с. Широкий пучок збільшує зону прийому. Однак, серйозним недоліком є те, що більш широкий пучок більшою мірою схильний до загасання і тому більш сприйнятливий до погодних умов [6].

Одним з важливих переваг оптичного зв'язку є безпека каналу. Перехопити сигнал - промінь світла - і при цьому не перервати передачу даних дуже важко.

До основних недоліків оптичних технологій можна віднести необхідність забезпечення прямої видимості між оптичними сенсорами приймача і передавача, а також обмежені допуски на кут їх відхилення відносно один одного.

1.3.2 Радіочастотні системи

На відміну від оптичних каналів радіочастотні системи не вимагають забезпечення прямої видимості між приймачем. Найбільш широкого поширення набули три технології бездротової передачі даних: Bluetooth, Wi-Fi і ZigBee [7]. Порівняльні характеристики цих технологій наведені в таблиці 1.1.

Таблиця 1.1 – Порівняльні характеристики технологій Bluetooth, Wi-Fi та ZigBee

Характеристики	Технологія (стандарт)		
	Bluetooth (IEEE 802.15.1)	Wi–Fi (IEEE 802.11b)	ZigBee (IEEE 802.15.4)
Пропускна здатність, кбіт / с	723,1	11000	250
Розмір стека протоколу, кбайт	більше 250	більше 1000	32-64
Розмір стека протоколу, кбайт	1-10	0,5-5	100-1000
Максимальне число вузлів в мережі	7	10	65536
Діапазон дії, м (середні значення)	10-100	20-300	10-100

Технологія Bluetooth підтримує як з'єднання типу «точка - точка», так і «точка - багато точок». Два або більше використовують один і той же канал пристрою утворюють пікомережу (piconet). Один з пристроїв працює як основний (master), а решта - як підлеглі (slave). В одній пікомережі може бути до семи активних підлеглих пристроїв, при цьому інші підлеглі пристрої знаходяться в стані «паркування», залишаючись синхронізованими з основним пристроєм.

Взаємодіючі пікомережі утворюють «розподілену мережу» (scatternet). В кожній пікомережі діє тільки один основний пристрій, проте підлеглі пристрої можуть входити в різні пікомережі. Крім того, основне налаштування однієї пікомережі може бути підлеглим в інший.

В основу Wi-Fi покладена стільникова архітектура, причому мережа може складатися як з однієї, так і з кількох осередків. Кожна сота керується базовою станцією. Точки доступу багатостільникової мережі взаємодіють між собою через розподільну систему, що є еквівалентом магістрального сегменту кабельних локальних обчислювальних мереж.

Стандарт ZigBee визначає три типи пристроїв: координатори, маршрутизатори та кінцеві пристрої. Кожна мережа повинна містити тільки один координатор. Основна задача координатора полягає в тому, щоб поставити параметри для створення мережі і запустити процес настроювання, що передбачає вибір радіочастотного каналу, унікального мережевого ідентифікатора і набору робочих параметрів. Маршрутизатор можуть використовуватися для розширення радіуса дії мережі, оскільки вони здатні виконувати функції і ретрансляторів між пристроями, розташованими занадто далеко один від одного, щоб взаємодіяти безпосередньо. Прикінцеві пристрої не беруть участь в маршрутизації.

Основний недолік даних технологій полягає в тому, що вони мають ієрархічний принцип побудови мережі, тобто дані від одного кінцевого пристрою передаються іншому кінцевого пристрою тільки через маршрутизатор. Уявімо бойового робота на полі бою, де ситуація змінюється щомиті і потрібно оперативно доповісти стан і отримати відповідний наказ. Таке з'єднання крім надійності і безпеки має бути ще й стійким до змін топології, маршрутизація повинна володіти швидкою збіжністю, тобто гарантувати знаходження маршруту заданої якості за розумний час, гарантувати відсутність зациклення, забезпечувати багатоадресну розсилку. При взаємодії великої групи мобільних роботів дана проблема стає ще більш актуальною.

На відміну від розглянутих вище бездротових мереж з виділеними керуючими центрами, MANET (Mobile Ad-hoc NETworks) використовують розподілені принципи управління мережею з можливістю самоорганізації і самоврядування вузлів мережі. Специфіка MANET мереж полягає в тому, що їх топологія постійно змінюється через переміщення вузлів мережі в просторі або зміни умов поширення радіосигналу [8].

Самоорганізовані мережі MANET володіють наступними перевагами над бездротовими мережами традиційної архітектури:

- можливість передачі даних на великі відстані без збільшення потужності передавача;
- стійкість до змін в інфраструктурі мережі;
- можливість швидкої реконфігурації в умовах несприятливої помеховой обстановки;
- простота і висока швидкість розгортання.

Використання MANET ускладнюється тим, що розроблені протоколи маршрутизації не реалізовані фізично, відсутня апаратна реалізація.

Таким чином, оптичні канали здатні забезпечити зв'язок між роботами тільки в зоні прямої видимості в режимі «точка-точка», і не здатні забезпечувати взаємодію групи мобільних роботів. Радіочастотні технології Bluetooth, Wi-Fi, ZigBee не забезпечують потрібну продуктивність внаслідок неефективності ієрархічної структури мережі при використанні даних технологій для взаємодії групи мобільних роботів на відміну від мереж MANET.

1.4 Аналоги автономних роботів-колективістів

Ройова робототехніка розвивається дуже швидко. Якщо зовсім недавно рої роботів могли рухатися тільки по суші, то в даний час вони вміють літати по повітрю, плавати, пересуватися під землею і в космічному просторі. У відповідність з цим розділимо рої роботів на сім класів:

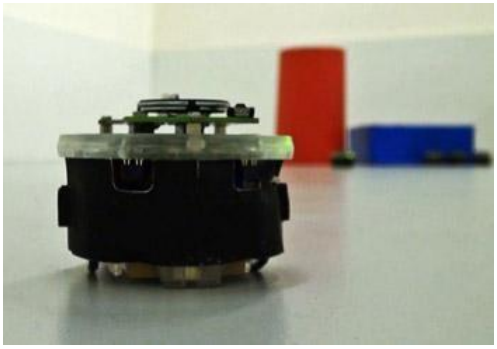
- наземні;
- підземні;
- над- і підводні;
- повітряні;
- космічні;
- комбіновані;
- спеціальні роботи.

Нам невідомі проекти підземних і спеціальних роїв роботів. Тому обмежуємося розглядом роботів інших перерахованих класів.

1.4.1 Наземні роботи

Фахівці з Шеффілдського центру робототехніки (Англія) створили робота (рисунок 1.3, а), який має колеса, батарею для приводів коліс, мікрофон, камеру і вісім сенсорів, які дозволяють роботові орієнтуватися в просторі. Виконано експерименти з управління роєм з 40 роботів. В ході експериментів рій успішно вирішував прості завдання, наприклад, групуючи навколо об'єкта і штовхаючи його в потрібному напрямку.

Вчені з університету Колорадо (США) створили рій мініатюрних роботів, розміри яких порівнянні з розмірами кульок для пінг-понгу (рисунок 1.3, б). Кожен з роботів оснащений (ІЧ) випромінювачем і приймачем, так що роботи в рої можуть спілкуватися між собою по бездротовому (ІЧ) зв'язку. «Інтелект» роботів забезпечують (МП) Atmel XMega 128. Рух роботів забезпечують спеціальні вібраційні двигуни.



а) Колісний робот з Шеффілда

б) Вібраційний робот університету Колорадо

в) Стрибаючий робот Jollbot

г) Робот-павук з Мілану

д) Вібраційний робот кілобот

Рисунок 1.3 – Приклади ройових наземних роботів

Вчені університету Саутгемптона (Англія) запропонували використовувати для дослідження Марса рій з сорока-шістдесяти автономних роботів на ім'я Jollbot - (рисунок 1.3, в). Jollbot долає перешкоди, перестрибуючи їх за методом коника або блохи. Хоча творці робота були натхненні здібностями комах, їх робот не схожий ні на одне з відомих комах. Для мінімізації ваги «скелет» Jollbot утворюють металеві прути, розташовані подібно меридіанах земної кулі. Усередині кулі на його центральній осі прикріплені акумулятори, сервоприводи, давачі, а також приймально-передавач для дистанційного керування і передачі зібраної інформації. Сервоприводи змушують стискати «скелет», а потім відпускають його, змушуючи тим самим підстрибувати кулю на півметра. Для управління роєм роботів Jollbot автори проекту пропонують використовувати принципи функціонування бджолиного рою.

Для дослідження Місяця і, зокрема, доставки на Землю зразків її грунту, команда з Міланського політехнічного університету (Італія) пропонують замість одного великого робота використовувати рій невеликих автономних роботів у вигляді павуків (рисунок 1.3, г). Передбачається, що кожен такий робот буде оснащений власною камерою для зйомки, набором давачів для збору наукової інформації, батареєю і сонячною батареєю для її підзарядки. Роботи за короткий проміжок часу зможуть досліджувати і зібрати проби з великої площі. За рахунок невеликих габаритів цих роботів їх можна компактно розмістити в відсіку, що спускає апарат і заощадити на вазі підйомного пристрою.

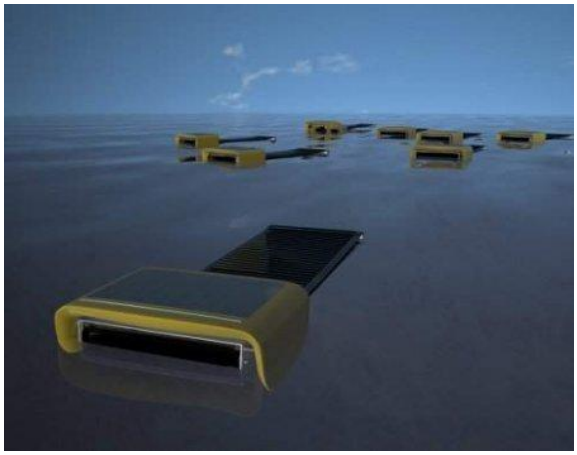
У Гарвардському Університеті (США) створено мініатюрні жукоподібні роботи (рисунок 1.3, д), названі авторами кілоботи (kilibot). Кожен кілобот має діаметр близько трьох сантиметрів і харчується від літієвого акумулятору напругою 3,4 вольта, що забезпечує йому три години безперебійної роботи. Пересувається кілобот вправо, вліво і вперед, вібруючи на трьох жорстких ніжках, оснащених двома двигунами. У нижній частині кожного робота встановлений ширококутний (ІЧ) приймач, який посилає світловий промінь вниз на гладку поверхню, по якій робот рухається. При цьому промінь відбивається, і його

приймають найближчі сусіди, що знаходяться в радіусі 10 сантиметрів. Робот оснащений бортовим (МК). Управління роєм здійснюється за допомогою контрольної станції, розташованої над експериментальною поверхнею з кілоботами. У 2011 році вчені продемонстрували колективні операції рою з 29 кілоботов. В одному з експериментів, наприклад, роботи діяли як мурахи в пошуках їжі. В іншому експерименті кілоботи повторювали дії лідера, утворюючи своєрідний паровозик.

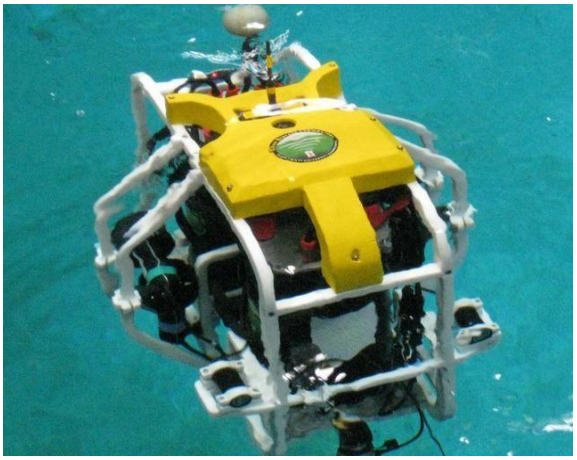
1.4.2 Над-і підводні роботи

Дослідниками з Массачусетського технологічного університету (США) розроблений проект рою роботів Seaswarm, призначеного для боротьби з нафтовими забрудненнями водних поверхонь великої площі (рисунок 1.4, а). Розробники стверджують, що рій Seaswarm з 5-10 тисяч роботів зможе прибрати нафтову пляму з морської поверхні розміром з Мексиканський затоки за один місяць. Дослідний зразок робота Seaswarm має довжину 49 см, завширшки 21 см, вага 16 кг. Енергію для руху робота дають дві сонячні батареї. Робот обладнаний з системами GPS і WiFi, що дозволяє роботам обмінюватися інформацією між собою і управлятися оператором. Нафта поглинається тонким, гідрофобним наноматеріалом стрічкового конвеєра позаду робота.

Коралові рифи - важливий елемент екосистеми, непоправної шкоди якому завдає діяльність людини. Вчені з Університету Хериот-Ватта (Англія) пропонують «лагодити» коралові рифи з допомогою коралоботів (рисунок 1.4, б). Уже побудовано кілька прототипів таких роботів, які обладнані камерою, комп'ютером і гнучкими руками із захопленнями. Планується створити рій коралоботів, які будуть автоматично оглядати пошкоджені коралові рифи і трансплантувати корали в зруйновані місця.



а) Роботи рою Seaswarm



б) Коралобот

в) Роботолодки

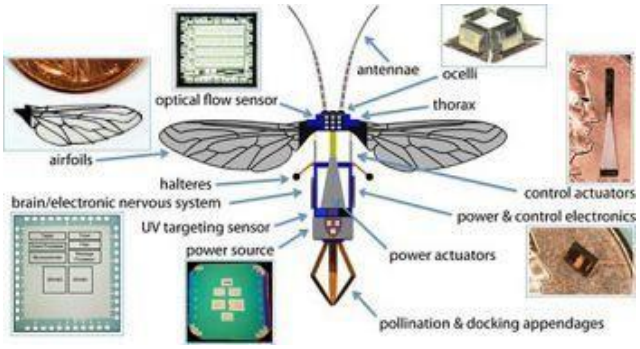
Рисунок 1.4 – Приклади ройових плаваючих роботів

Інженерами з Пенсільванського університету (США) створено, так звані роботолодки (рисунок 1.4, в). Автори намагаються змусити флот роботолодок координовано працювати разом з метою побудови різних складних структур. При достатній кількості таких човнів, вони можуть спорудити мости, злітно-посадочні смуги і навіть острова. В даний час в Пенсильванському університеті є близько ста роботолодок. Кожен човен керується промисловим комп'ютером фірми Gumstix і використовує чотири незалежних мотора, які дозволять їй плисти в будь-якому напрямку і здійснювати повороти з нульовим радіусом. Обговорюється можливість збільшення роботолодок до розміру стандартного вантажного контейнера.

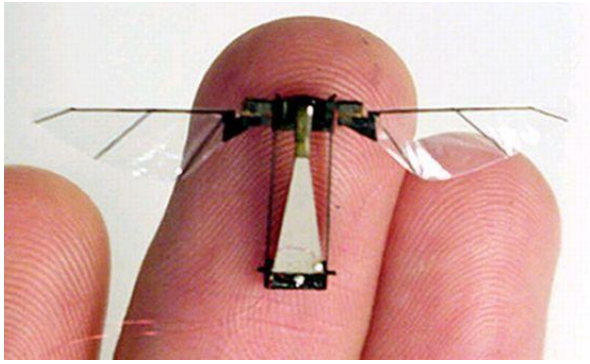
1.4.3 Повітряні роботи

Інженери німецької компанії Festo розробили біонічного робота, якого вони назвали BionicOpter (рисунок 1.5, а). Раніше фахівці цієї ж компанії продемонстрували великого літаючого робота SmartBird, який переміщається аналогічно соколу. Довжина штучної бабки становить 48 сантиметрів, розмах крил - 70 сантиметрів, вага робота - 175 грамів. Як і живий прототип, BionicOpter мають чотири крила, які можуть рухатися незалежно один від одного, забезпечуючи роботу дуже високу маневреність. Так, BionicOpter може висіти в повітрі і літати боком. Автори робота планують розробку системи управління роєм роботів BionicOpter.

а) Робот *BionicOpter*



б) Робот-бджола Гарвардського університету



в) Робот рою *RoboBees*

Рисунок 1.5 – Приклади повітряних ройових роботів

Вчені з Гарвардського університету (США) розробили проект рою літаючих роботів-бджіл (рисунок 1.5, б). Кожен з цих роботів буде в змозі приймати власні рішення, але, в той же час, ці роботи будуть мати можливість «спілкуватися» з іншими роботами рою, з метою спільного вирішення загальних для рою завдань. Проект заснований на розробці цих вчених 2007 року, коли ними була створена унікальна рухова система, повністю копіює рух крил бджоли. Передбачається, що кожен робот-бджола буде обладнаний сучасними «розумними» давачами, що дозволять йому отримувати інформацію про навколишнє середовище. Управління роботом буде здійснюватися за допомогою власного (МП), який реалізує спеціально розроблені інтелектуальні алгоритми, що дозволять імітувати поведінку, як окремої бджоли, так і цілого рою цих комах. Можливі застосування рою ще не визначені.

Несподіване застосування рою мініатюрних роботів демонструють фахівці Пенсільванського університету (США). Роботи рою злітають вгору і направляються до музичних інструментів. Кожен робот самостійно виконує своє завдання: кілька роботів грають на синтезаторі, один - на тарілці, ще один - на барабані і т.д. В цілому, рій роботів може зіграти задану мелодію.

Внаслідок людської діяльності популяції медоносних бджіл у всьому світі знаходяться в занепаді. Зникнення цих комах являє дуже велику небезпеку екосистемі Землі. Проблема настільки гостра, що команда вчених з Гарвардського і Північно-Східного університетів США працюють над проектом створенням бджолиного рою мініатюрних роботів (рисунок 1.5, в), які могли б запилювати квіти і виконувати роботу справжніх бджіл. В рамках проекту створено рій роботів RoboBees, кожен з яких представляє собою літаючий робот розміром з бджолу. Рой роботів RoboBees керується тими ж алгоритмами, яким підпорядковано взаємодія тисячі бджіл в їх природному рої.

1.4.4 Космічні роботи

Інженери з американської компанії SpaceWorks Engineering працюють над проектом захисту Землі від астероїдів за допомогою рою роботів. Кожен з роботів рою є, по суті, космічний корабель, званий MADMEN (Modular Asteroid Deflection Mission Ejector Node) - рисунок 1.6.

MADMEN має вагу в одну тонну і висоту, рівну 11 метрам. У космос робота виводить ракета-носій. Досягнувши астероїда, MADMEN самостійно сідає на його поверхню і починає свердлити скелю. Закріпившись на поверхні астероїда, робот за допомогою своєї ядерної рухової установки починає повільно штовхати астероїд з частотою, що дорівнює приблизно одному імпульсу в хвилину. Оскільки небезпека для Землі представляє тільки великий астероїд, на ньому має працювати відразу кілька роботів, співпрацюючи між собою, щоб координувати свої дії. Необхідність використання рою роботів пояснюється ще й тим, що, як правило, астроїди мають деяку кутову швидкість обертання. Тому висаджувати роботи слід по всій поверхні астероїда, і включати їх коригувальні двигуни так, щоб результуюча їх тяга вела астероїд від Землі.

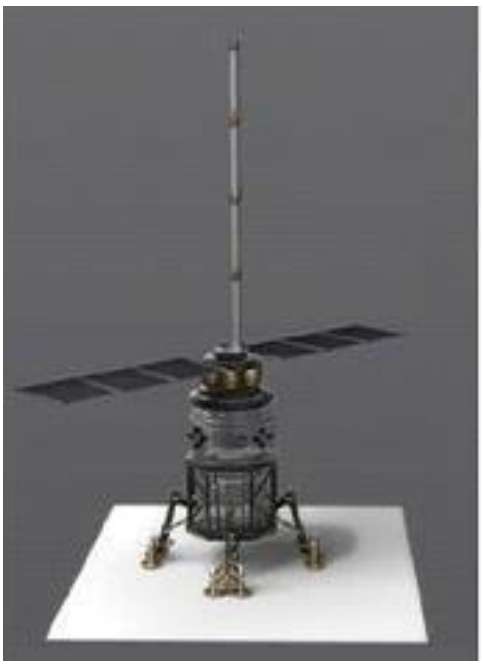


Рисунок 1.6 – Приклад космічного робота MADMEN

Необхідна кількість роботів залежить від маси астероїда, кутової швидкості його обертання, дальності від Землі, починаючи з якої роботи починають свою роботу. Крім того, при визначенні цього числа слід враховувати надійність роботів, так щоб при виході з ладу деякого їх числа залишилися могли вирішити задачу захисту Землі.

1.4.5 Комбіновані роботи

Унікальний проект рою роботів Swarmanoid розробляється в ряді європейських інститутів і лабораторій. Основною метою дослідження є створення розподіленої системи гетерогенних роботів (рисунок 1.7) і розробка алгоритмів управління нею. Передбачається створити рій роботів Swarmanoid з 60 роботів трьох видів: Eye-bot (очі-бот), Hand-bot (рука-бот) і Foot-bot (нога-бот). Автори проекту вважають, що поділ ботів за призначенням підвищує надійність і простоту експлуатації всього рою роботів, дозволяє сформувати більш точну картину місцевості, успішно долати різні перешкоди.

Літаючий робот Eye-bot (найбільший робот на рисунку 1.7) побудований на платформі квадролета, має магнітний гачок і спеціалізується на зондуванні і аналізі навколишнього середовища, завдяки встановленим на ньому телевізійної камері, (ІЧ) сканеру, ультразвуковому локатору і іншим давачам. Цей же робот передає інформацію про місцевість іншим роботам і координує їх дії.

Рухомий робот Foot-bot призначений для транспортування роботів типу Hand-bot та інших об'єктів. Робот має власну телекамеру, вимірювач відстані і здатний об'єднуватися для спільної роботи з іншими ботами.

Робот Hand-bot не може пересуватися по землі, але здатний долати вертикальні перешкоди і маніпулювати об'єктами за допомогою своїх двох маніпуляторів і магнітного гарпуна.

Рисунок 1.7 – Приклад геторогенного рою роботів Swarmanoid

Фахівці федерального інституту технологій в Цюріху (Німеччина) створили автономних роботів-дронів, які можуть об'єднуватися один з одним на землі, а потім підніматися в повітря у вигляді літаючої платформи. Проект носить назву «розподілена літаюча група" (Distributed Flight Array). Обмін інформацією між роботами рою реалізований за допомогою (ІЧ) давачів. Роботи можуть швидко адаптуватися до мінливих умов польоту рою, оптимізуючи свою траєкторію польоту і траєкторію рою в цілому. Такий рій роботів може успішно застосовуватися для стеження. Наприклад, кожен робот може «дивитися» в задану сторону, і при необхідності відокремитися від основної групи і продовжити виконання завдання самостійно.

1.5 Групова взаємодія роботів

Проблема групової взаємодії є дуже важливою в сучасній робототехніці. Великий інтерес для розгляду вона представляє в такій перспективній галузі сучасної мініробототехніки. Основні дослідження в області керування групами роботів ведуться в багатьох індустріально розвинених країнах світу, перш за все в інтересах оборони. Найбільш інтенсивний характер цих робіт з огляду на напрямок військової робототехніки спостерігається в США. Аналіз результатів досліджень показує, що в даний час практично немає будь-якого загального підходу до проблеми групового керування роботами. Кожна дослідницька група намагається розробити свій спосіб вирішення, що стоїть перед нею приватним завданням, яке, як правило, не може бути застосоване при вирішенні інших завдань подібного типу. Таким чином, актуальною стає проблема розробки методів і алгоритмів розподіленого (децентралізованого) управління колективною взаємодією роботів при їх груповому застосуванні в умовах заздалегідь невідомих ситуацій, що динамічно змінюються. Рішення даної проблеми дозволить, по-перше, значно розширити області застосування роботів, по-друге, наблизитися до вирішення проблеми масового застосування мініроботи в складі великих груп, які налічують тисячі і десятки тисяч мініроботів.

Перші наукові дослідження в області застосування груп роботів, які взаємодіють між собою для досягнення мети, проводилися в 80-х роках XX століття, вирішувався ряд вузькоспеціалізованих завдань. Перші спроби систематизації досліджень в області колективного управління роботами при їх груповій взаємодії та побудови теоретичної і методологічної бази для їх подальшого розвитку здійснені в початку 2000-х років.

Еволюція систем групового керування йде в напрямку збільшення децентралізації зі збереженням за центром тільки забезпечення загальносистемних не піддаються декомпозиції функцій груп. Сучасна тенденція до мінітюарізації робототехніки робить більш перспективним їх групове застосування для вирішення практичних завдань [12].

1.5.1 Завдання групового керування

Для досягнення конкретної мети, що стоїть перед групою роботів, в разі детермінованого середовища кожен робот може виконувати заздалегідь визначену послідовність дій [13]. У разі ж недетермінованого динамічного середовища, ця послідовність повинна бути знайдена системою керування групою роботів в процесі досягнення мети. Причому дії роботів групи, повинні бути певним чином скоординовані, узгоджені. Таким чином, виникає задача керування групою роботів. Це завдання полягає або в реалізації системою управління роботами заздалегідь знайденої послідовності дій всіх роботів групи, або в знаходженні такої послідовності і її реалізації в процесі досягнення поставленої мети. Завдання керування групою роботів, дії яких спрямовані на досягнення спільної групової мети, будемо називати завданням групового керування. Суть завдання групового керування полягає в знаходженні і реалізації таких дій кожного окремого робота групи, які приводили б до оптимального, з точки зору деякого групового критерію, досягнення загальної групової мети.

1.5.2 Підходи для вирішення завдань

Серед відомих підходів до вирішення завдання групового керування роботами можна виділити два діаметрально протилежні підходи. У першому випадку це завдання вирішується одним, зосередженим (центральним) пристроєм управління. У другому випадку рішення здійснюється деяким, розподіленим по роботам групи пристроєм управління. Надалі перший підхід називається централізованим груповим керуванням, а другий підхід - децентралізованим груповим керуванням [14].

1.5.3 Види централізованого керування

Централізоване керування роботами може бути двох видів: єдиноначальні та ієрархічні [15].

Єдиноначальне керування (наявність в групі командира або центрального пристрою управління роботів (ЦПУ), на які покладаються завдання планування і керування групою). Перевагою єдиноначального керування є простота її організації та алгоритмізації. До недоліків слід віднести тривалий час прийняття рішення через виконання завдання оптимізації всіх членів групи роботів для досягнення групової мети, а також низьку живучість.

Ієрархічне керування (так само наявність ЦПУ або командира, які керують невеликою кількістю роботів, в підпорядкуванні кожного з них складається своя група об'єктів). Порівняно з єдиноначальним керування, істотно знижується складність завдання, розв'язуваної окремим командиром або ЦПУ, однак ускладнення структури керування може призводити до сильних затримок або збоїв у передачі команд від верхнього до нижнього рівня.

1.5.4 Види децентралізованого керування

Децентралізоване керування роботами може бути двох видів: колективне або зграйне [16].

Колективне керування (в системі немає командира або ЦПУ, усі одиниці рівноцінні і кожен член групи самостійно приймає рішення, намагаючись внести максимально можливий внесок в досягнення групової мети, при цьому члени групи обмінюються інформацією про обрані дії один з одним). За рахунок того, що кожен елемент групи вирішує задачу оптимізації тільки для себе, а не намагається оптимізувати дії всієї групи; задача істотно спрощується, тому її рішення може здійснюватися швидко, в реальному часу. Але це сильно ускладнює алгоритмізацію і пред'являє до елементів великої групи "інтелектуальний рівень",

тому від них вимагається чітко розуміти групову задачу і вміти вибирати такі дії, які призводять до найкращого її рішенням з точки зору всієї групи роботів.

(РК) – це коли в системі немає командира або ЦПУ, усі одиниці рівноцінні і кожен член групи роботів самостійно приймає рішення, намагаючись внести максимально можливий внесок в досягнення групової мети, при цьому обміну інформацією між членами групи немає, і кожен об'єкт "підлаштовує" свої дії на підставі непрямої інформації.

1.6 Проблеми групового керування

Проблеми групового керування роботами, які потребують вирішення завдань децентралізації системи керування, кластеризації сукупності роботів і розподілу цілей, є предметом підвищеного інтересу зарубіжних і вітчизняних вчених, результати досліджень яких знайшли відображення в великій кількості публікацій. Цей інтерес викликаний перспективністю застосування груп роботів в різних областях людської діяльності, а також одночасно складністю вирішуваних завдань [17]:

1. Необхідно вирішити задачу розподілу цілей і завдань між роботами, з урахуванням характеру цілей, функціональних можливостей кожного робота і середовища їх функціонування.
2. Необхідно організувати взаємодію роботів на комунікаційному рівні і узгодити їх поведінку при досягненні цілей.
3. Для забезпечення високої надійності групи, потрібно реалізовувати децентралізоване управління, що призводить до розподіленої структури робототехнічної системи.

Групове застосування роботів як було вже сказано вище, не є тривіальним завданням і не існує її якогось єдиного рішення. При розробці оптимального методу групового керування роботами необхідно керуватися критерієм ефективності. Існує два види критерію ефективності:

- критерій ефективності першого роду - ступінь досягнення мети системи;
- критерій ефективності другого роду - оцінка ефективності в деякому заданому шляху досягнення мети [18].

1.7 Керування групами роботів в різних середовищах

1.7.1 Керування групами роботів в стаціонарних умовах

Прикладом досліджень в області групового керування роботами може служити проект "MARTHA", який виконувався в лабораторії Аналізу системних архітектур Франції. Метою даного проекту була розробка методів організації групової взаємодії роботів (від 10 до 100 шт.), призначених для транспортування вантажів в складських терміналах. В проекті "MARTHA" використовується централізоване управління групою роботів (рисунок 1.8).

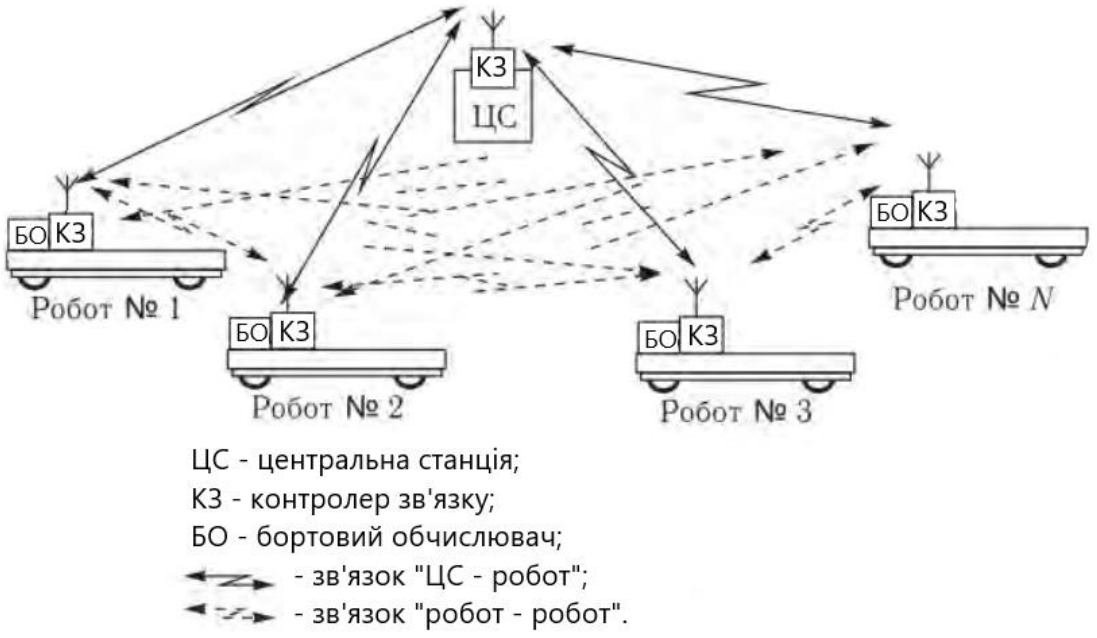


Рисунок 1.8 – Структура системи управління проекту "MARTHA"

Метою проекту "AMADEUS", запропонованого японськими розробниками, є створення групи роботів, які забезпечують підвезення і вивезення виробів для конвеєрних ліній. Використовуються два типи роботів: транспортний робот (ТР), що виконує безпосередню перевезення виробів, і стаціонарний навантажувальний робот (СНР), розташований в безпосередній близькості від конвеєрної лінії і виконує розвантажувально-навантажувальні роботи з ТР на конвеєр і навпаки (рисунок 1.9).

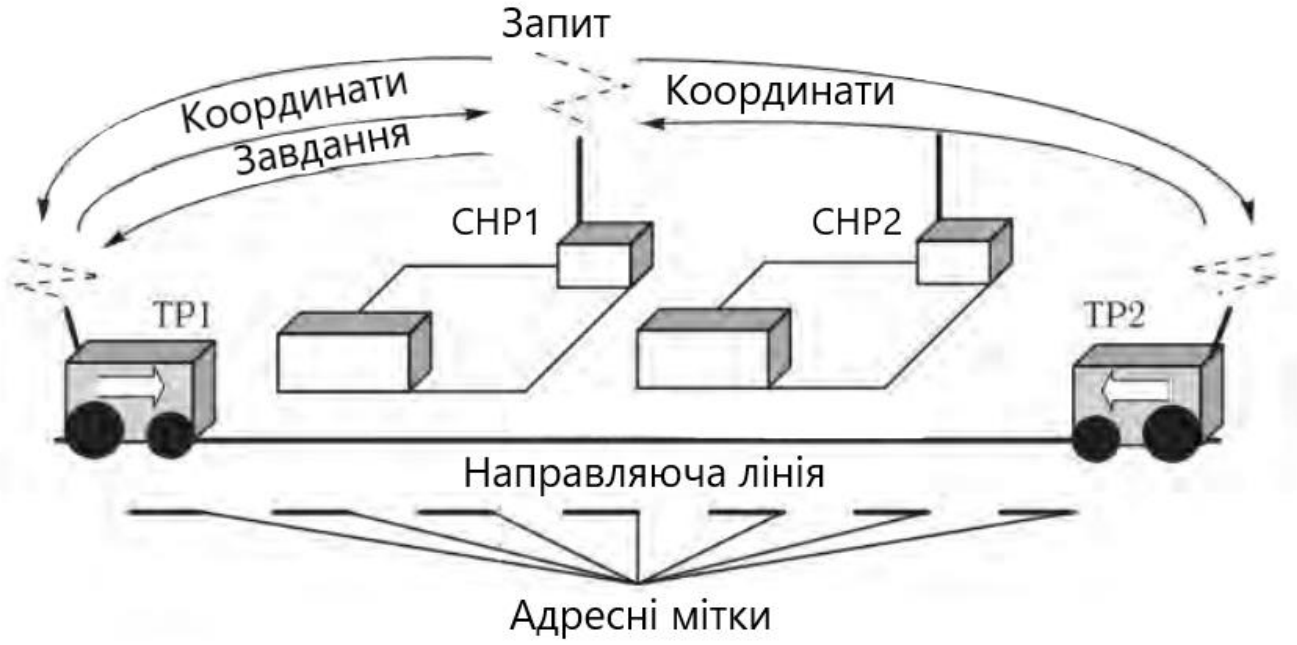


Рисунок 1.9 – Архітектура проекту "AMADEUS"

1.7.2 Керування групами роботів в нестаціонарних умовах

В Японії роботи в області систем групової взаємодії роботів активно ведуться в університеті міста Нагоя. Тут розроблена система DARS (Distributed Autonomous Robotic System), за допомогою якої відпрацьовуються алгоритми і методи планування і керування скоординованими діями групи роботів, що функціонують в природному неорганізованої середовищі, з єдиного мобільного командного центру [19].

Керуванням DARPA Міністерства оборони США також фінансувалася розробка тактичних мобільних мініроботів для групового застосування в міських умовах, що виконувалася в рамках програми "Тактичні мобільні робототехнічні системи" (рисунок 1.10). Дослідження були спрямовані на відпрацювання командної взаємодії групи, що складається з людей і роботів. Система управління дозволяє оператору керувати кількома роботами з єдиного командного центру (переносного). Оператор задає цілі для кожного робота, а роботи переміщуються до цілей автономно [19].

Рисунок 1.10 – Група тактичних мобільних роботів

1.10 Керування групами роботів в умовах протидії

Гарною моделлю для відпрацювання методів і алгоритмів групового керування роботами в умовах протидії може служити гра двох команд роботів в футбол, яка в первісному вигляді призначалася для суто дослідницьких цілей. Однак даний напрямок отримав таку широку популярність серед її розробників,

що в кінцевому підсумку переріс в новий вид інтелектуального спорту, іменований як "гра роботів в футбол" (рисунок 1.11).

У категорії "програмні моделі" гра проходить на програмному рівні, на базі сервера. Сервер створює віртуальне поле, арбітра, моделює переміщення гравців і м'яча по полю, забезпечує зв'язок між гравцями однієї команди і виводить зображення на екран монітора. Роботи-гравці представляють собою клієнтські програми, які можуть планувати дію як одного, так і декількох гравців однієї команди на окремому комп'ютері, підключається до сервера через мережевий IP протокол.

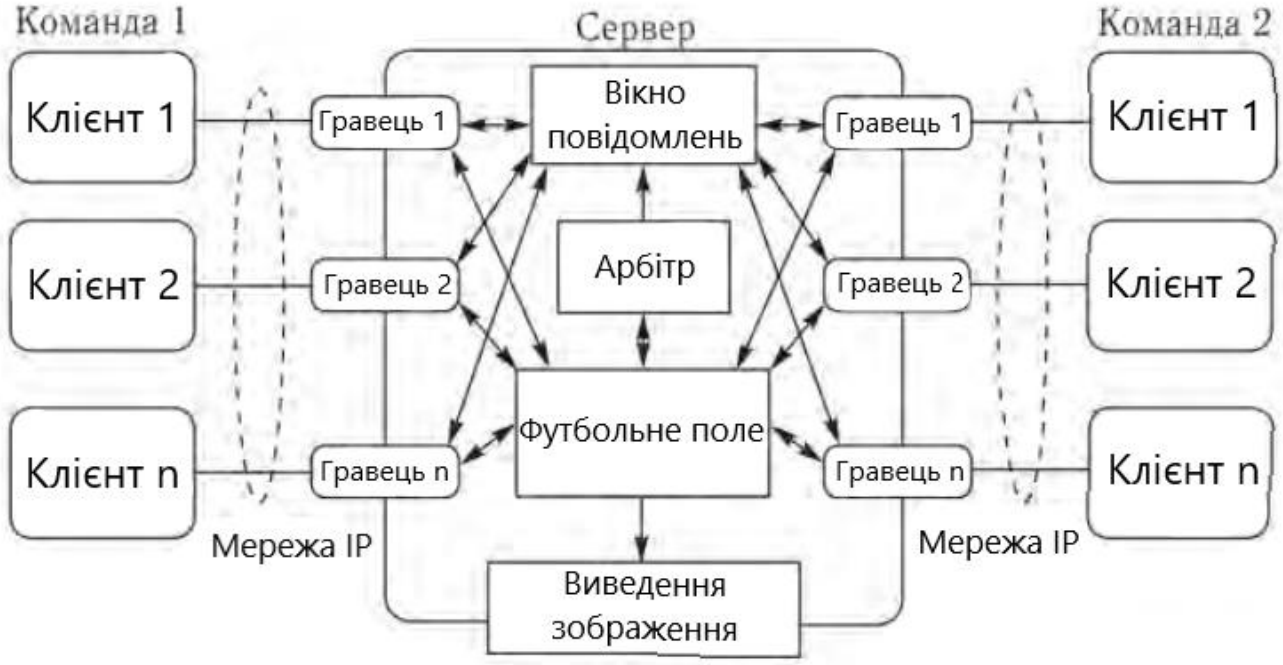


Рисунок 1.11 – Структурна схема організації гри роботів в футбол

При проведенні змагань в лізі апаратних моделей малих роботів для управління ними застосовується система, структура якої показана на рисунку 1.12. структура складається з глобальної відеокамери (ГВК), розташованої над полем, контролера зв'язку і попередньої обробки даних (КЗіПОД) і декількох модулів планування дій (МПД), кожен з яких відповідає одному з гравців команди (рисунок 1.13).

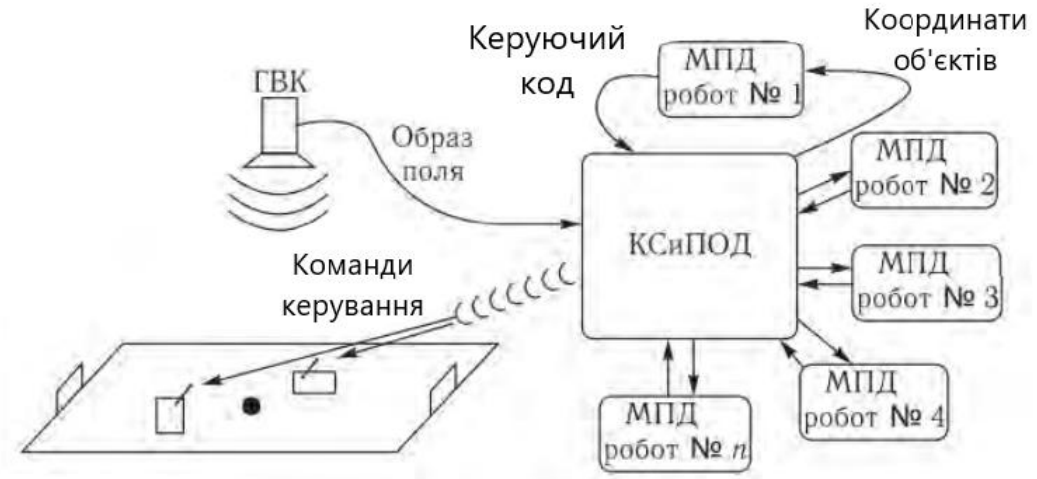


Рисунок 1.12 – Структура організації гри роботів в футбол

Рисунок 1.13 – Зовнішній вигляд футбольного поля і роботів-футболістів

Велика група простих сенсорних роботів розбивається на кілька підгруп, кожна з яких закріплюється за роботом-лідером, керованим з командного центру. Роботи-лідери, в свою чергу, формують цільові завдання для простих сенсорних роботів і взаємодіють з роботами-картографами. Прості сенсорні роботи взаємодіють один з одним для підтримки заданої відстані при русі до мети і передають зібрану інформацію про навколишнє середовище своєму роботові-лідеру. Таким чином, утворюється деяка ієрархічна структура.

2 РОЗРОБКА СХЕМИ, ВИБІР ЕЛЕМЕТІВ ТА АЛГОРИТМ ДІЇ

2.1 Вибір елементної бази

Головний робот складається з таких елементів:

- ESP32;
- драйвер мотора;
- платформа;
- компас;
- лазерний (ДВ);
- інфрачервоний приймач;
- фоторезистор;

Інші роботи зібрані на платформі Arduino Uno та мають наступні елементи:

- інфрачервоний передавач;
- світлодіод;

2.1.1 ESP32

ESP32 – це серія високопродуктивних (МК) типу «система на кристалі», що мають інтегровані контролери Wi-Fi і Bluetooth, низьке енергоспоживання та невисоку ціну.

(МК) ESP32 володіють настільки широкими функціональними можливостями, такою кількістю периферійних інтерфейсів, що зазвичай до них застосовується термінологія "система ESP32" [9].

Є мікросхема, наприклад ESP32-D0WDQ6 (рисунок 2.1). Вона забезпечує більшу частину функціональних можливостей системи, але для застосування вимагає додаткових компонентів.

Рисунок 2.1 – Мікросхема ESP32-D0WDQ6

Додали до нього FLASH-пам'ять, генератор, блокувальні конденсатори та інші зовнішні компоненти. Встановили їх на маленьку друковану плату, прикрили кришкою - вийшов модуль ESP-WROOM-32 (рисунок 2.2).

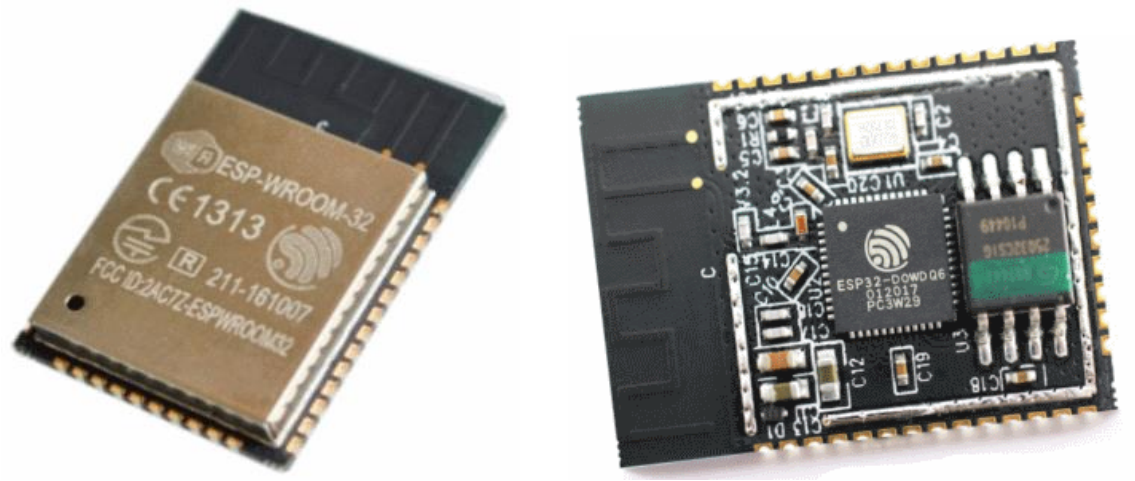


Рисунок 2.2 – Модуль ESP-WROOM-32

На модуль достатньо подати живлення і вже можна використовувати WiFi або Bluetooth. Якщо підключити зовнішні сигнали, то вже буде готова система.

Ще зручніше використовувати плату. В цій дипломній роботі використовується плата ESP DevKit V1 (рисунок 2.3).

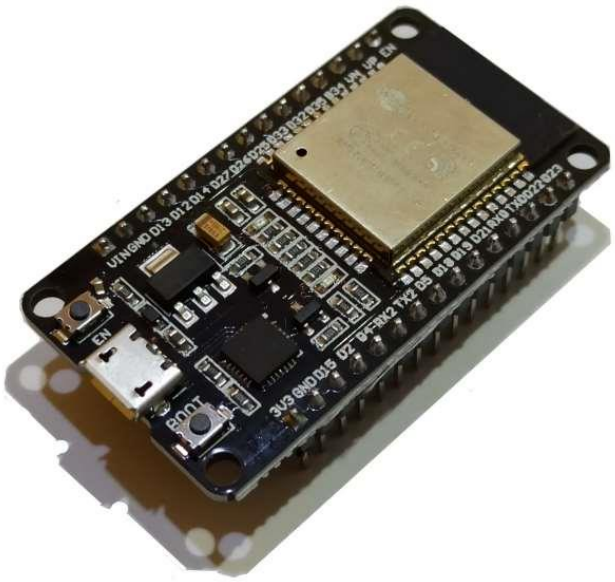


Рисунок 2.3 – Плата ESP DevKit V1

Додалися такі функціональні і конструктивні можливості:

- Конвертер USB/UART CP210X, що дозволяє підключатися до стандартного (USB) комп'ютера;
- Апаратні ланцюги, що забезпечують завантаження FLASH (МК) від комп'ютера в автоматичному режимі;
- Стабілізатор напруги живлення модуля;
- Кнопки скидання і завантаження;
- Світлодіоди;
- Зовнішні сигнали (МК) виведені на краю плати.

Тобто в мінімальному варіанті досить підключити плату ESP32 до (USB) комп'ютера. Через нього плата отримує живлення, завантажується резидентна програма, проводиться налагодження [9].

У платі встановлений модуль ESP-WROOM-32.

У модулі використовується (МК) ESP32-D0WDQ6.

Ці елементи визначають функціональні можливості і параметри плати:

1. Процесорна частина:
    1.1. ядро Tensilica Xtensa LX6;
        1.1.1. два ядра;
        1.1.2. 32 розряду;
        1.1.3. продуктивність до 600 MIPS (мільйонів інструкцій в сек);
    1.2. співпроцесор з ультранизьким енергоспоживанням.
2. Пам'ять:
    2.1. 448 KB ROM для завантажувача і функцій ядра;
    2.2. 520 KB SRAM для даних і команд;
    2.3. 16 KB SRAM в RTC (квазінезалежна пам'ять);
    2.4. 4 MB FLASH - пам'ять програми.
3. Бездротові інтерфейси:
    3.1. Wi-Fi: 802.11 b / g / N;
    3.2. Bluetooth: v4.2 BR / EDR і BLE.
4. Годинники і таймери:
    4.1. внутрішній генератор 8 мГц;
    4.2. внутрішній RC-генератор;
    4.3. зовнішній генератор 40 мГц;
    4.4. зовнішній генератор 32 кГц для RTC;
    4.5. 2 групи таймерів, включаючи 2x64 біт таймери і сторожовий таймер в кожній групі;
    4.6. RTC таймер (годинник реального часу);
    4.7. RTC сторожовий таймер.
5. Периферійні інтерфейси:
    5.1. 34 програмованих універсальних портів введення / виводу;
    5.2. 12ти розрядний (АЦП), до 18 каналів;
    5.3. 2 x 8 бітових (ЦАП);
    5.4. 10 портів для підключення ємнісних давачів;
    5.5. 4 x SPI;

5.6. 2 x (I2S);

5.7. 2 x (I2C);

5.8. 3 x (UART);

5.9. 1 хост контролер SD / eMMC / SDIO;

5.10. 1 слейв контролер SDIO / SPI;

5.11. Ethernet MAC interface з виділеним DMA і IEEE 1588 підтримкою;

5.12. CAN 2.0;

5.13. ІК порт;

5.14. (ШІМ) для двигунів;

5.15. (ШІМ) для світлодіодів до 16ти каналів;

5.16. Давач Холла.

6. Безпека:

6.1. безпечна завантаження;

6.2. шифрування FLASH;

6.3. 1024-бітний ключ, до 768 біт для клієнтів;

6.4. криптографічний апаратне прискорення:

6.4.1. AES;

6.4.2. хешування SHA-2;

6.4.3. RSA;

6.4.4. ECC;

6.4.5. генератор випадкових чисел (RNG).

Функціональна схема (МК) представлена на рисунку 2.4.

(МК) з такими можливостями і такою низькою ціною може бути використаний будь-де. Але виробник - компанія Espressif Systems, перш за все, виділяє призначення ESP32 для створення IoT пристроїв.

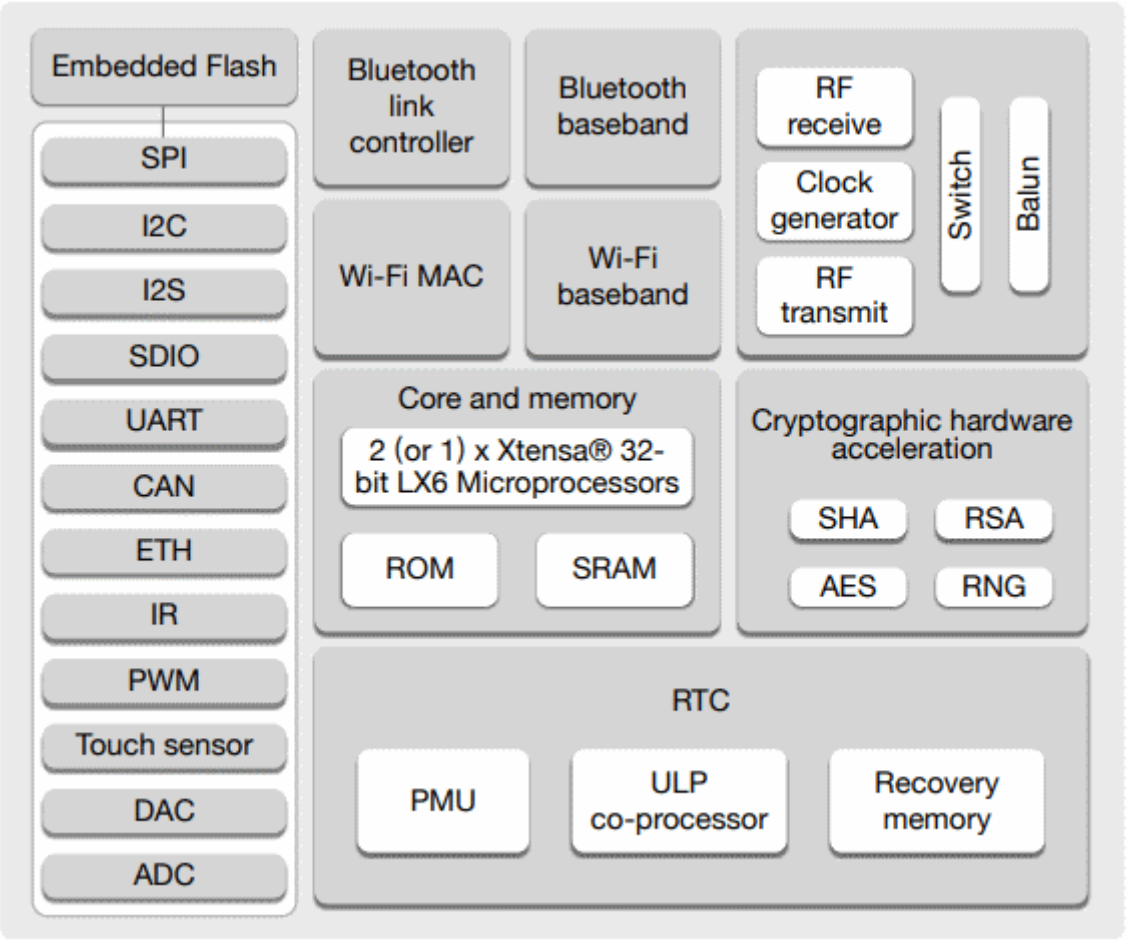


Рисунок 2.4 – Плата ESP DevKit V1

IoT (Internet of Things), в перекладі "Інтернет Речей" - це концепція підключення всього до інтернету, що дозволить управляти цим всім через інтернет. Дійсно, наявність інтегрованих інтерфейсів WiFi і Bluetooth, висока продуктивність і низька ціна роблять ESP32 ідеальними контролерами для IoT технологій.

Висока продуктивність дозволяє використовувати ESP32 в системах обробки зображень і мови в реальному часі. Домашня автоматика, розумний будинок, контроль здоров'я, сільське господарство, промисловість, робототехніка, іграшки тощо [9].

Призначення портів плати показано на рисунку 2.5.

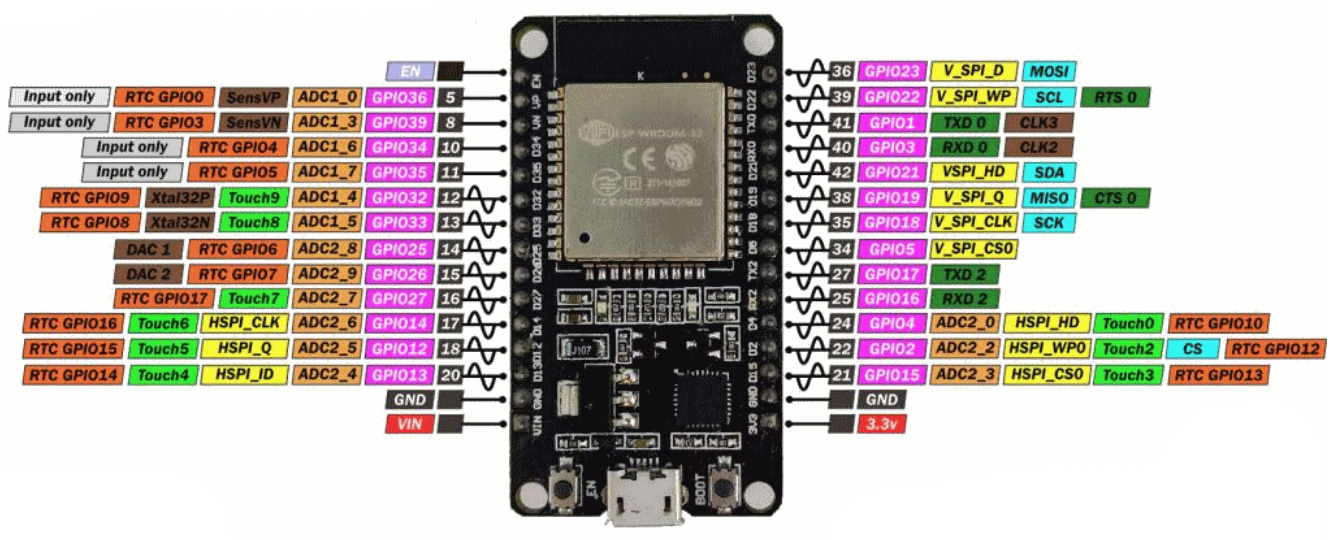


Рисунок 2.5 – Плата ESP DevKit V1

С двох сторін плати розташовані контактні гребінки по 15 пінів з кроком 2,54 мм (модифікація на 30 пінів).

Всі контакти підтримують переривання. Максимальний струм на пінах: 12 мА [8].

Доступні 25 пінів загального призначення:

- Цифрові 21 контакт введення-виведення (GPIO): 1-5, 12-19, 21-23, 25-27, 32 і 33. Контакти загального призначення. Піни можуть бути налаштовані на вхід або на вихід. Логічний рівень одиниці - 3,3 В, нуля - 0 В. Всі порти введення-виведення можуть працювати як (ШІМ), що дозволяє виводити аналогові значення в вигляді ШІМ-сигналу з розрядністю 16 біт. Максимальна кількість каналів 16;
- цифрові 4 контакти введення (GPI): 34, 35, 36 і 39. Можуть бути налаштовані тільки на вхід;
- 15 аналогових входів з (АЦП) (12 біт): 2, 4, 12-15, 25-27, 32-36 і 39. Дозволяє уявити аналогову напругу в цифровому вигляді з розрядністю 12 біт;
- 2 аналогових виходів з (ЦАП) (8 біт): 25 (DAC1) і 26 (DAC2). Аналоговий вихід цифро-аналогового перетворювача, який дозволяє формувати 8-бітові рівні напруги. Порти можуть використовуватися для аудіо-виходу;

- 10 контактів ємнісного сенсора.

Напруга живлення модуля - 3,3 В, але на платі встановлений стабілізатор AMS1117-3.3. В документації на плату DevKit весь час фігурує напруга зовнішнього живлення 5. AMS1117-3.3 допускає живлення до 20 В, але треба ще враховувати потужність розсіювання на стабілізаторі.

Живлення (USB) розв'язане від зовнішнього живлення плати діодом Шотткі. Значить можна живити плату від зовнішнього джерела і одночасно підключати до (USB). Основні параметри входів / виходів наведені у таблиці 2.2.

Середній споживаний струм - 80 мА, але джерело живлення повинен допускати пікові навантаження до 0,5 А.

Температура навколишнього середовища -40 +85 °С. Це для плати. У (МК) -40 +125 °С [9].

Параметри входів та виходів зображені на таблиці 2.1.

Таблиця 2.1 – Параметри входів / виходів.

Позначення	Параметр	Мінімальне значення	Типове значення	Максимальне значення	Од. вимір.
V_{IH}	Вхідна напруга високого рівня	0,75 * V_{DD}	-	V_{DD} + 0,3	В
V_{IL}	Вхідна напруга низького рівня	0,3	-	0,25 * V_{DD}	В
I_{IB}, I_{IH}	Вхідний струм високого і низького рівня	-	-	50	нА
V_{OH}	Вихідна напруга високого рівня	0,8 * V_{DD}	-	-	В
V_{OL}	Вихідна напруга низького рівня	-	-	0,1 * V_{DD}	В

Продовження таблиці 2.1.

I_{OH}	Вихідний струм високого рівня	-	40	-	мА
I_{OL}	Вихідний струм низького рівня	-	28	-	мА
R_{PU}, R_{PD}	Опір підтягуючого резистора	-	48	-	кОм

Піни живлення:

- VIN: Пін для підключення зовнішнього джерела напруги в діапазоні від 5 до 14 вольт (поруч з GND);
- 3V3: Пін від стабілізатора напруги з виходом 3,3 вольта і максимальних струмом 1 А. Регулятор забезпечує живленн модуля ESP32-WROOM (поруч з GND);
- GND: Виводи землі (два контакти, по одному на кожній стороні).

На налагоджувальному модулі розташовані дві тактові кнопки. Кнопка EN призначена для ручного перезапуску плати - аналог кнопки RESET звичайного комп'ютера.

Кнопка Boot слугує для ручного запуску режиму прошивки модуля. Алгоритм наступний: Затисніть кнопку BOOT; Натисніть і відпустіть кнопку EN; Відпустіть кнопку BOOT. Також на платі знаходиться світлодіод живлення і індикаторний світлодіод, підключений до цифрового піну 2 (замість 13 як у стандартних Arduino) [8].

Важливо відмітити, що:

- не можна на входи подавати напругу вище 3,3;
- здатність навантаження виходів (МК) досить висока для подібних пристроїв.

2.1.2 Платформа та драйвер мотору L298N

У якості платформи використовувалася мобільна 3-х колісна робо-платформа (рисунок 2.6). Вона містить два двигуни (3-6 В, струм 100-120 мА), з коефіцієнтом уповільнення 48: 1, відсік для батарейок, три колеса (два провідних підключених до двигунів). До платформи можна кріпити (МК) плати. Розміри дозволяють зібрати повністю всього робота на цій платформі.

Рисунок 2.6 – Мобільна 3-х колісна робо-платформа

Для керування двигуном було вирішено використовувати драйвер L298N.

L298N – драйвер колекторних двигунів на 2 канали, який може також застосовуватися для керування одним кроковим двигуном. Драйвер мостовий, що дозволяє використовувати його без додаткових транзисторів відсутніх плечей. Максимальна напруга живлення моторів - 46 В, струм на канал - 2 А [5].

На рисунку 2.7 показано зовнішній вигляд модулю з коротким описом всіх його складових.

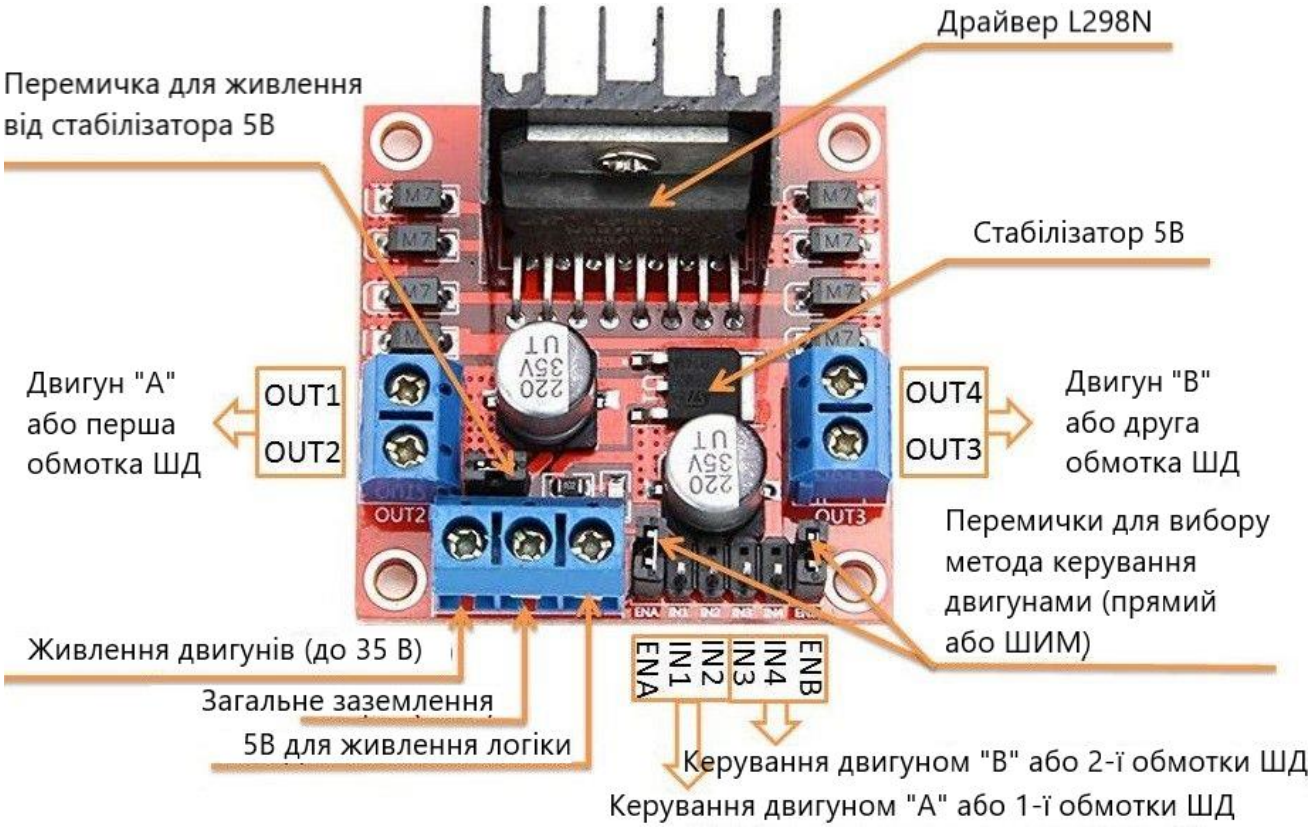


Рисунок 2.7 – Зовнішній вигляд та опис всіх складових модуля L298N

Драйвер дозволяє просто і зрозуміло керувати швидкістю обертання моторів в обох напрямках за допомогою (ШІМ) окремо для кожного мотора.

Драйвер виконаний на мікросхемі L298, особливістю даного драйвера є наявність вимикача, який дає змогу швидко виключити навантаження. Для живлення драйвера потрібно 5V, напрузі від 7 до 20 вольт можна використовувати вбудований понижуючий стабілізатор напруги 5V. Є можливість заживити драйвер від (МК) [4].

Модуль на основі L298N не вимагає зовнішніх компонентів для початку роботи.

Тепер поговоримо більш детально про всі складові цього модуля:

- OUT1 і OUT2 - роз'єми для підключення першого щіткового двигуна або першої обмотки крокового двигуна;

- OUT3 і OUT4 - роз'єми для підключення другого щіткового двигуна або другої обмотки крокового двигуна;
- VSS - вхід для живлення двигунів (максимальний рівень + 35V);
- GND - загальний провід (треба також з'єднати з аналогічним входом плати керування);
- Vs - вхід для живлення логіки + 5V. Через нього безпосередньо живиться сама мікросхема L298N. Є ще другий спосіб живлення, при якому 5V для L298N береться від вбудованого в модуль стабілізатора напруги. В такому випадку на роз'єм подається тільки живлення для двигунів (Vss), контакт Vs залишається не під'єднаним, а на платі встановлюється перемичка харчування від стабілізатора, який обмежить напругу моторів, що живляться, до прийнятних 5V.
- IN1, IN2 - контакти керування першим щітковим двигуном або першою обмоткою крокового двигуна.
- IN3, IN4 - контакти керування другим щітковим двигуном або другою обмоткою крокового двигуна.
- ENA, ENB - контакти для активації / деактивації першого і другого двигунів або відповідних обмоток ШД. Подача логічної одиниці на ці контакти дозволяє обертання двигунів, а логічний нуль - забороняє. Для зміни швидкості обертання щіткових моторів на ці контакти подається ШІМ-сигнал. Для роботи з кроковим двигунів, як правило, на ці контакти ставлять перемички, що забезпечують постійну підтяжку до + 5V.

На рисунку 2.8 показана електрична схема модуля L298N.

Як видно з наведеної схеми, основним елементом модуля є мікросхема L298N, до складу якої входять два повноцінних Н-моста. Кожен Н-міст виконаний у вигляді збірки з чотирьох транзисторних ключів з включеним в центрі навантаженням у вигляді обмотки двигуна. Такий підхід дозволяє змінювати полярність в обмотці і як наслідок напрямок обертання двигуна шляхом чергування пар відкритих і закритих ключів.

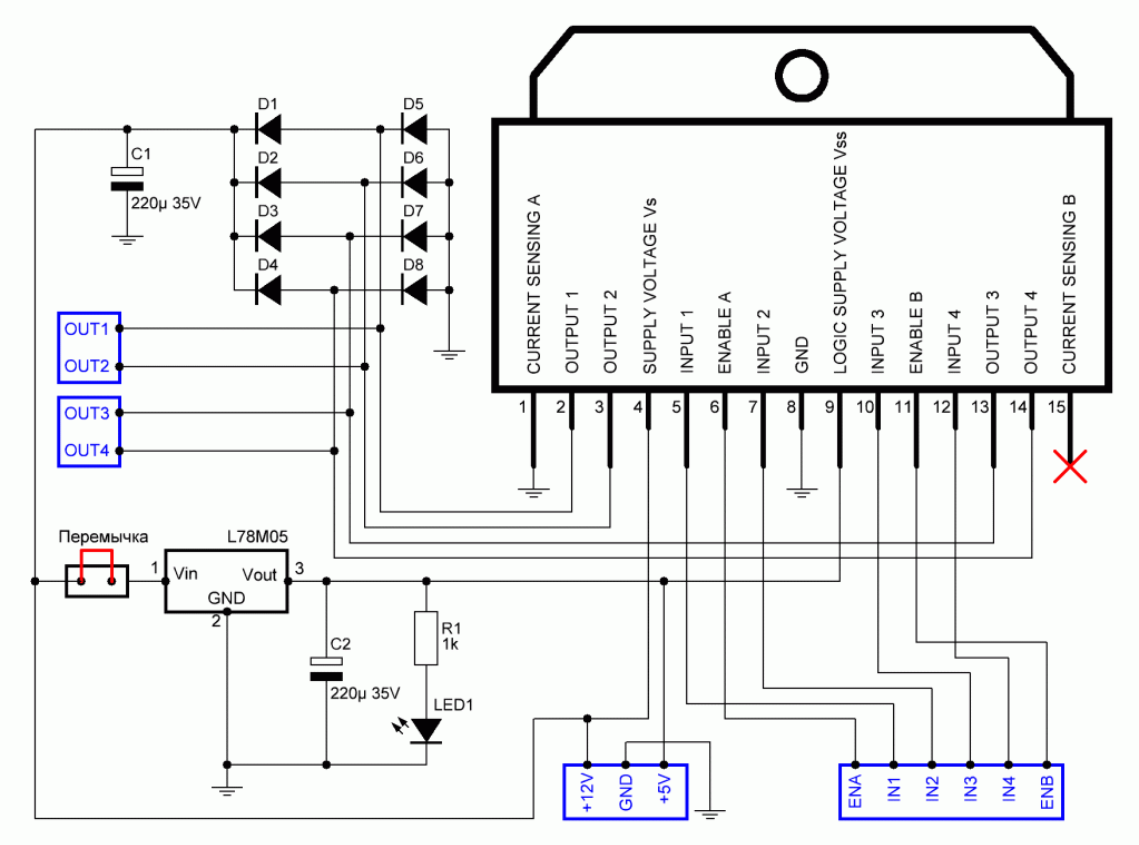


Рисунок 2.8 – Електрична схема модуля L298N

На рисунку 2.9 зображені два транзисторних моста Н-типу.

У першому випадку на вхід IN1 подається логічна одиниця, а на вхід IN2 - логічний нуль. Так як транзистори в схемі моста мають різний тип провідності, то при такому вхідному сигналі транзистори Т1 і Т4 залишаться в закритому стані, в той час, як через транзистори Т2 і Т3 потече струм. З огляду на те, що єдиний шлях протікання струму лежить через обмотку двигуна, то останній виявиться підключений правої клемою до плюса живлення, а лівої до мінуса. Все це призведе до обертання мотора в певному напрямку. Абсолютно протилежна картина показана на нижньому малюнку. Тут IN3 встановлений в логічний нуль, а IN4 в логічну одиницю. Тепер струм тече в зворотному напрямку (ліва клема - плюс, права - мінус), змушуючи другий двигун крутитися в протилежну сторону [4].

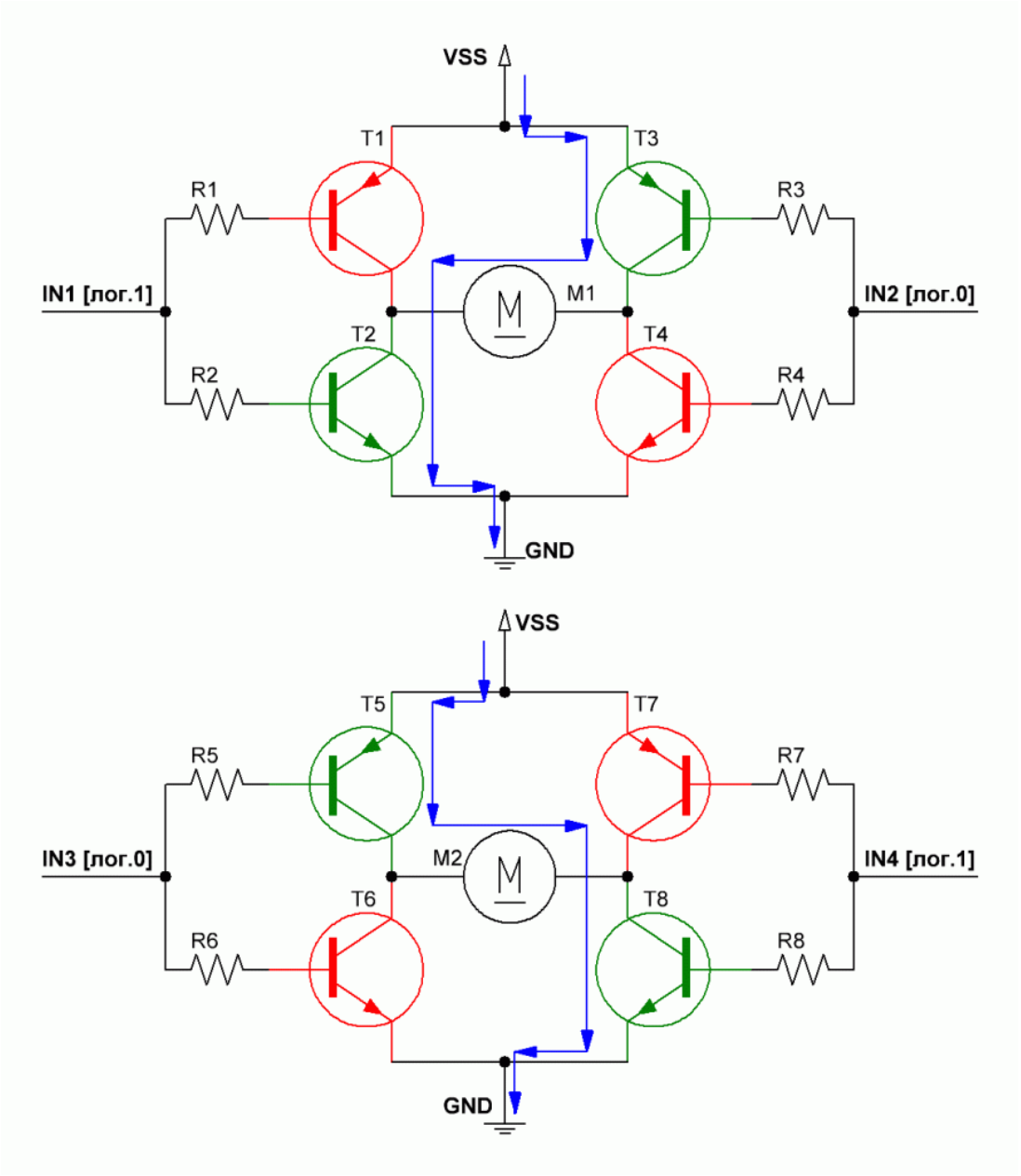


Рисунок 2.9 – Транзисторні мости Н-типу

Технічні характеристики драйвера L298N наведені у таблиці 2.2.

Таблиця 2.2 – Технічні характеристики драйвера L298N

Характеристика	Значення
Напруга живлення драйвера	5 – 7 В
Напруга живлення двигуна	3 – 35 В
Струм керування драйвером	36 мА

Продовження таблиці 2.2.

Максимальний постійний струм двигуна	2 А
Максимальна споживана потужність	20 Вт
Діапазон робочих температур	-25 °С ... +135 °С
Розміри модуля	43.5 х 43.2 х 29.4 мм

2.1.3 Модуль трьохосьового цифрового компаса GY-273 HMC5883L

Модуль GY-273 HMC5883L – це електронний магнітний компас з 3-ма осями (рисунок 2.10). Модуль підтримує інтерфейси (I2C) або (SPI). Для зв'язку з (МК) використовує стандартний (I2C) інтерфейс [26].

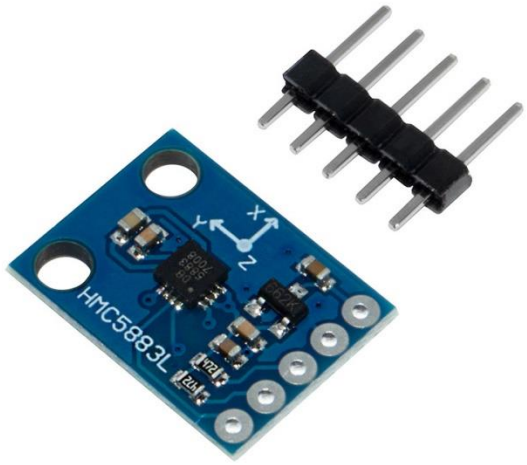


Рисунок 2.10 – Модуль трьохосьового цифрового компаса GY-273

У більшості випадків магнітометр HMC5883L застосовується для навігації робототехніки. Мікросхема HMC5883L дозволяє вимірювати магнітну індукцію по трьох осях за допомогою вбудованого надточного (АЦП) дозволом 12 біт [25].

Точність вимірювання складає від 0.73 до 4.35 мГс, рівень шуму не перевищує 2 мГс, а діапазон виміру простягається від -8 Гс до 8 Гс.

Для забезпечення датчика живленням на платі встановлений стабілізатор XC6206, що видає стабільні 3.3 В. На плату може бути подана напруга від 3.6 В до 6 В, на виході можна отримати 3.3 В для живлення зовнішніх пристроїв (максимальний струм стабілізатора - 250 мА) [25].

Можна використовувати датчик спільно з гіроскопом і акселерометром, з таким комплектом можна домогтися більш точних свідчень.

Активними сенсорами в компасі виступають три магніторезистивних датчика. Процесом вимірювання і обміном даними керує вбудований (МК) [25].

Високоточний тривісний магнітний компас з температурною компенсацією дозволяє отримувати тривимірну картину спрямованості магнітного поля і його величину [26].

Перед застосуванням модуля необхідно провести його калібрування. Вимірювання магнітного поля зазнає спотворень. Існує дві категорії цих спотворень: жорсткі спотворення заліза та спотворення м’якого заліза. Помилки твердого заліза стосуються присутності магнітних полів навколо датчика (магніти, дроти живлення) і пов'язані з похибками зміщення вимірювань, тоді як помилки м'якого заліза стосуються присутності феромагнітних матеріалів навколо датчика, які викривляють щільність магнітного поля Землі локально і пов’язані з масштабуванням помилок зміщення.

Іншими словами, для отримання правильних даних магнітометра потрібно отримати калібровані дані магнітометра. Один із шляхів вирішення цієї проблеми: слід застосувати зміщення до вектора некаліброваних даних магнітометра (координати X, Y, Z), а потім помножити матрицю перетворення на цей результуючий вектор

У цьому випадку калібрування магнітометра - це процес отримання матриці перетворення та зміщення. Щоб отримати ці дані, використовувалася програма MagMaster.

Спочатку був завантажений ескіз для налагодження. Потім запустили MagViewer.exe, виберали послідовний порт плати (швидкість монітіору порту повинна становити 9600 біт / с) і натиснули «Запустити MagViewer». Тепер можемо бачити координати вектора даних магнітометра в тривимірному просторі в реальному часі (рисунок 2.11). Ці дані ще не відкалібровані.

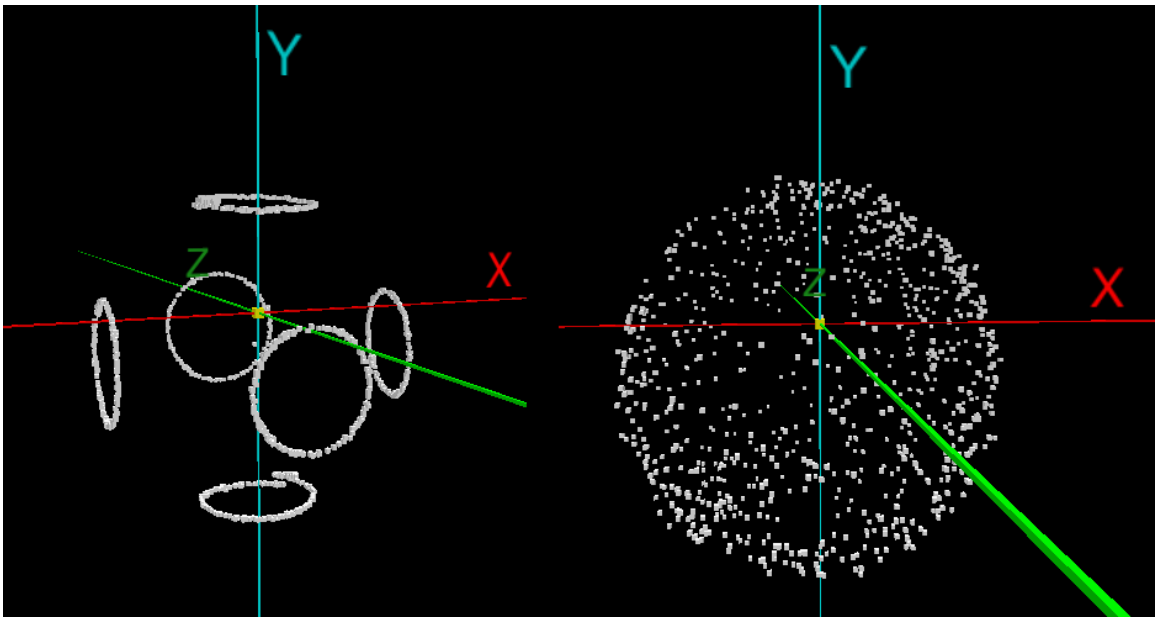


Рисунок 2.11 – Не відкалібровані дані магнітометра в тривимірному просторі

Після цього закрили вікно MagViewer і запустили MagMaster.exe (рисунок 2.12). Вибрали послідовний порт плати. Зелені рядки X, Y, Z вказують координати вектора магнітометра.

Для розміщення модуля можна використовувати дерев'яний брусок або паперову коробку (рисунок 2.13). Якщо не вдається підключити пристрій магнітометра до ПК, можна використовувати будь-який інший спосіб ознайомитися з даними магнітометра. Наприклад, можна ввести ці дані в програму вручну [24].

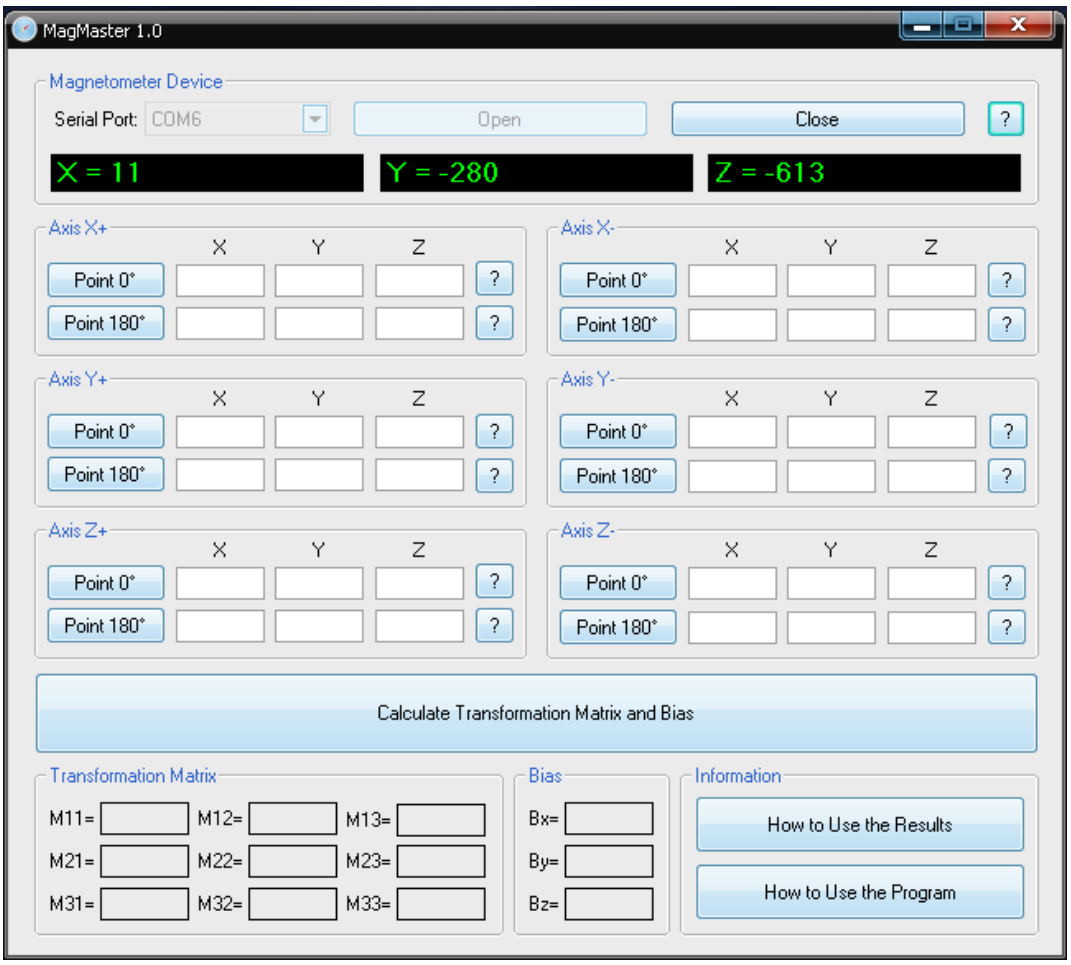


Рисунок 2.12 – Не відкалібровані дані магнітометра у вікні MagMaster

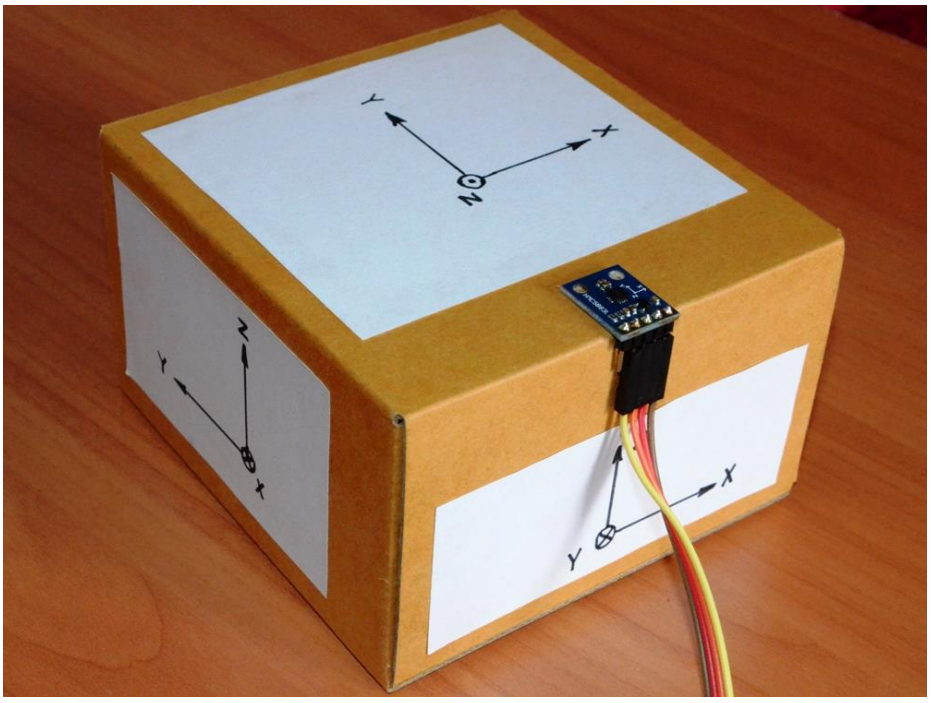


Рисунок 2.13 – Розміщення модулю для точного калібрування

Слід заповнити таблицю. Після цього натиснути «Обчислити матрицю перетворення та зміщення» та отримати необхідну матрицю та зміщення (рисунок 2.14).

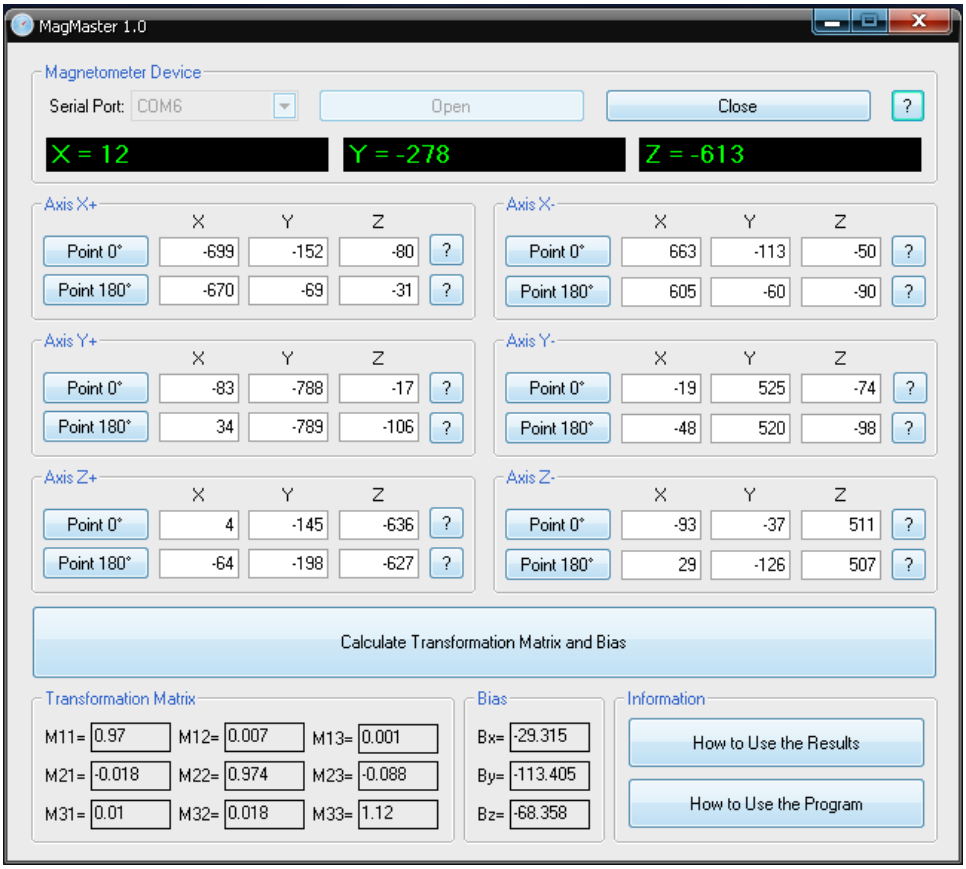


Рисунок 2.14 – Відкалібровані дані магнітометра у вікні MagMaster

Калібрований вектор магнітометру координує в тривимірному просторі зі стабілізацією радіуса (рисунок 2.15).

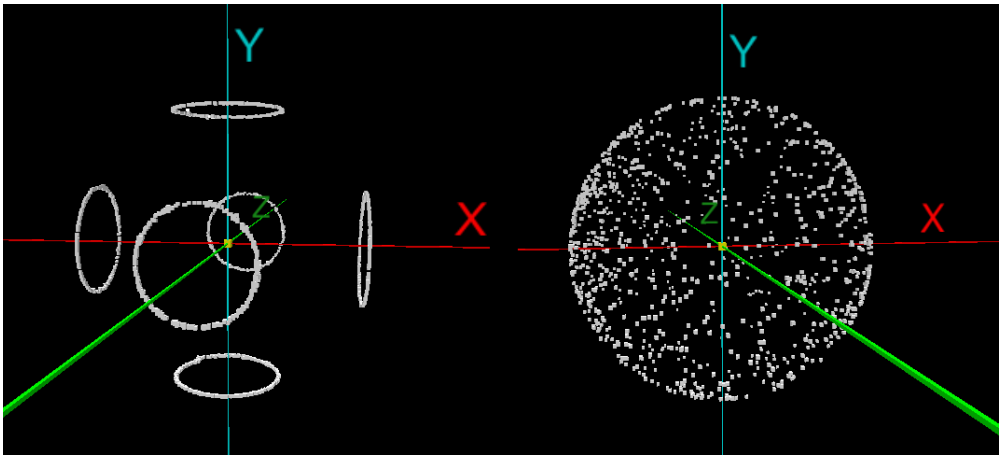


Рисунок 2.15 – Відкалібровані дані магнітометра в тривимірному просторі

Для підключення датчика до (МК) потрібно припаяти штирі до 5 - контактного роз'єму датчика. Після чого завантажити і завантажити робочу бібліотеку для модуля магнітометра в (МК) і з'єднати роз'єми з GY-273 HMC5883L [25].

Основні характеристики магнітометру GY-273 наведені в таблиці 2.3.

Таблиця 2.3 – Технічні характеристики магнітометру GY-273

Характеристика	Значення
Напруга живлення	3 – 5 В
Максимальна частота опитування	220 Гц
Точність вимірювання	1 – 2 °
Інтерфейс	стандартний (I2C) (3.4 МГц) / (SPI) (8 МГц)
Розміри	18 х 14 х 2 мм
Міжосьова відстань отворів кріплення	13 мм

Опис виводів модуля GY-273:

- VDD - позитивний полюс джерела живлення 3 – 5В;
- GND - негативний полюс джерела живлення, земля;
- SCL - вхід тактування шини (I2C);
- SDA - лінія даних інтерфейсу (I2C);
- DRDY - (опціонально) вихід стану готовності, логічна «1» - пристрій готовий до зчитування, може бути підключений до одного порту введення (МК).

Основні переваги:

- низьке споживання (не більше 100 нА);
- висока точність вимірювання;
- можлива робота (I2C) на швидкості до 400 КГц (швидкісний режим).

Використовується як електронний компас. Може бути налагоджена інтеграція з GPS-навігатором для швидкого визначення положення, а також застосовувати датчик повороту флюгера для визначення напрямку вітру

Застосовується в системах позиціонування, туристичному обладнанні і особливо потрібний в авіамоделях, мультикоптерах для орієнтування в просторі.

В дипломній роботі цей модуль використовується для того, щоб головний пристрій орієнтувався у просторі и розумів у якому напрямку знаходиться сторонній робот.

2.1.4 Лазерний давач відстані VL53LOX-V2

VL53L0X V2 – мініатюрний модуль давача відстані та розпізнавання жестів (рисунок 2.16). Швидке та точне вимірювання відстані до 2 м. Модуль підключається через поширений послідовний інтерфейс (I2C) для керування пристроєм і передачі даних. Напруга живлення модуля 5В [27].

Рисунок 2.16 – Зовнішній вигляд лазерного давача відстані VL53LOX-V2

VL53L0X-V2 має передову матрицю на основі високочутливих однофотонних лавинних діодів.

На відміну від звичайних давачів дальності VL53L0X-V2 здатний забезпечити точне вимірювання відстані незалежно від кольору і відбивної здатності об'єкта, забезпечуючи кращий захист від перешкод.

Поверхнево-випромінюючий лазер VCSEL з довжиною хвилі 940 нм, виконує в давачі відстані VL53L0X роль джерела оптичного сигналу. Він оснащений вбудованим (ІЧ) фільтром. Його світіння повністю невидиме для людського ока, і забезпечує більшу дистанцію вимірювання при меншій чутливості до рівня зовнішнього освітлення і більш стійкий до перехресних перешкод, що спричиняються скляними поверхнями [27].

У модулі встановлено стабілізатор напруги і конвертер рівнів сигналу.

Схема електрична принципова представлена на рисунку 2.17.

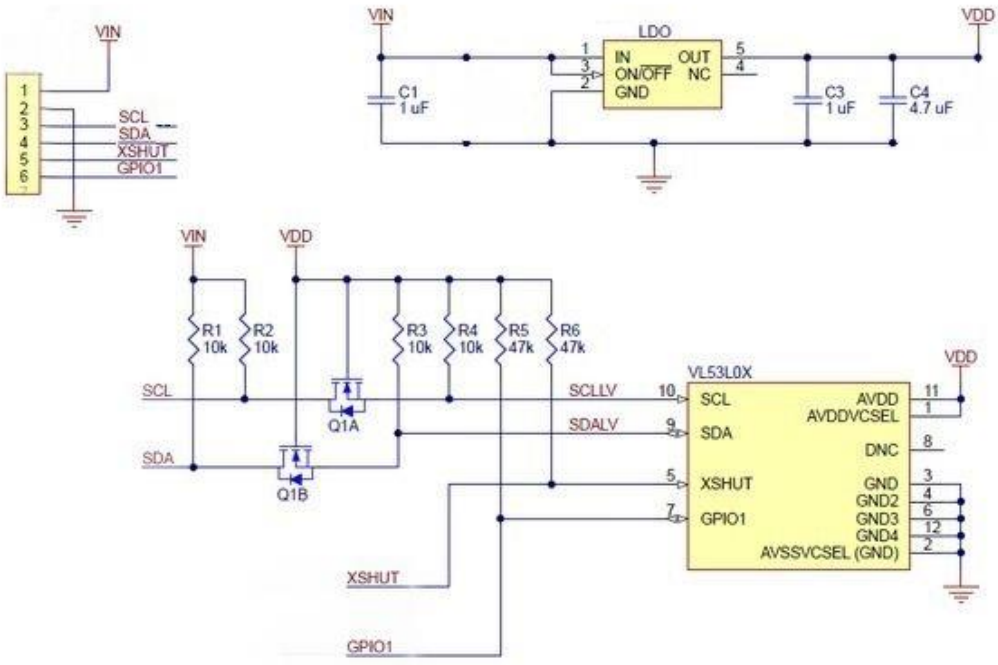


Рисунок 2.17 – Відкалібровані дані магнітометра в тривимірному просторі

Модуль підключається через поширений послідовний інтерфейс (I2C) для управління пристроєм і передачі даних.

Характеристики давача відстані VL53LOX-V2 наведені в таблиці 2.4.

Таблиця 2.4 – Технічні характеристики давача відстані VL53LOX-V2

Характеристика	Значення
Робоча напруга	4.5 – 5.5В
Споживаний струм	30 мА
Діапазон виміру	10 – 80 см
Точність дальності	± 5% (режим високої швидкості) ± 3% (режим високої точності)
Час ранжирування	20 мс (режим високої швидкості) 200 мс (режим високої точності)
Кут огляду	25 °
Довжина хвилі лазера	940 нм
Розміри модуля	44.5 x 18.8 x 13.5 мм
Вага	3.53 г

Призначення контактів:

- VCC і GND - живлення модуля від 2,6 до 5,5 В;
- SDA - лінія даних (Serial Data);
- SCL - лінія тактування (Serial CLock);
- GPIO1 - програмований вихід переривання (логічний рівень живлення);
- XSHUT - вивід є активним-низьким входом відключення, модуль підтягує його до VDD, щоб ввімкнути давач за замовчуванням. Низький рівень цього виводу переводить давач в апаратний режим очікування. Цей вхід не зміщений за рівнем [28].

Давач VL53L0X-V2 використовується для виявлення користувача при включенні і виключенні блокування на стільникових телефонах, комп'ютерів, ноутбуків або планшетах. В цілому даний датчик можна використовувати при проектуванні роботів і інших пристроїв [28].

Область застосування:

- Давачі присутності персональних ПК / ноутбуків / планшетних ПК і пристроїв Інтернету речей (автоматичні вимикачі освітлення)
- Робототехніка (давачі виявлення перешкод)
- Побутова техніка (автоматичні крани, вентилі, дозатори мила і т.д.)
- Одномірні давачі розпізнавання жестів
- Лазерні системи автофокусування. Покращує і прискорює роботу автофокусу камери, особливо в несприятливих умовах (низька освітленість, низька контрастність) або при зйомці на високій швидкості руху

В дипломній роботі цей давач застосовується для того, щоб знайти відстань між об’єктами або іншими роботами.

2.1.5 Модуль давача інфрачервоного випромінювання

Модуль давача (ІЧ) випромінювання призначений для вимірювання та контролю рівня випромінювання (ІЧ) діапазону (рисунок 2.18).

Рисунок 2.18 – Модуль давача інфрачервоного випромінювання

Маленькі розміри, висока чутливість і просте стандартне підключення знайшли застосування в даній дипломній роботі.

Застосовується цей давач для того, щоб ловити (ІЧ) випромінювання від іншого робота та розуміти, що він поряд.

Характеристики:

- використовується чутливий фотодатчик (ІЧ) діапазону випромінювання;
- цифровий вихід з навантажувальною здатністю 15мА;
- потенціометр для регулювання порогу спрацьовування цифрового виходу;
- робоча напруга від 3.3V до 5В;
- аналоговий вихід: відсутній;
- просте і зручне кріплення модуля;
- маленькі розміри: 3.2см х 1.4см;
- використаний поширений компаратор з широким діапазоном напруги живлення LM393 [29].

2.1.6 Аналогово-цифровий давач освітлення

В дипломному проекті використовується модуль давача освітлення з пороговим компаратором.

Застосовується для зчитування сигналу зі світлодіода. У модуля є аналоговий вхід, який передає значення освітлення, а також є цифровий вхід, чутливість якого регулюється за допомогою вбудованого потенціометру (змінного резистора).

Характеристики модуля давача освітлення наведені в таблиці 2.5.

Таблиця 2.5 – Технічні характеристики аналогово-цифрового модуля давача освітлення

Характеристика	Значення
Чутливий елемент	фоторезистор
Вихід компаратора	більш ніж 15 мА
Робоча напруга	3.3 – 5 В
Цифровий вихід компаратора	0, 1
Аналоговий вихід	з датчика освітленості
Розміри модуля	32 х 14 мм
Використаний компаратор	LM393

Призначення виводів:

- VCC - живлення модуля від 23,3 до 5,5 В;
- GND – загальна заземлення;
- D0 – цифровий вихід компаратора;
- A0 – аналоговий вихід;

2.1.7 Інфрачервоний давач перешкод YL-63

Цифровий (ІЧ) давач обходу перешкод YL-63 (або FC-51) застосовується тоді, коли потрібно визначити наявність об'єкта, а точну відстань до об'єкта знати необов'язково. Давач складається з (ІЧ) випромінювача, і фотоприймача (рисунок 2.19) [2].

В цій дипломній роботі давач використовується для того щоб передавати (ІЧ) випромінювання зі сторонніх роботів до основного, який приймає це випромінювання і розуміє що поряд знаходиться інший робот.

Рисунок 2.19 – Інфрачервоний давач перешкод YL-63

Технічні характеристики давача перешкоди YL-63 наведені в таблиці 2.6.

Таблиця 2.6 – Технічні характеристики давача перешкоди YL-63

Характеристика	Значення
Модель	YL-63 (або FC-51)
Напруга живлення	3.3-5 В
Тип давача	дифузний
Компаратор	LM393
Відстань виявлення перешкод	2 - 30 см
Ефективний кут виявлення перешкод	35 °
Розміри	43 x 16 x 7 мм

(ІЧ) джерело випромінює (ІЧ) хвилі, які відбиваються від перешкоди і фіксуються фотоприймачем. Давач виявляє перешкоди в діапазоні відстаней від нуля до встановленої граничної межі. Він побудований на основі компаратора LM393, який видає напругу на вихід за принципом: виявлена перешкода логічний рівень HIGH, не виявлено - логічний рівень LOW, цей стан показує і червоний світлодіод, який знаходиться на давачі. Граничне значення залежить від настройки давача і регулюється за допомогою встановленого на модулі потенціометра. Для індикації живлення на давачі встановлений зелений

світлодіод. Давач застосовується в робототехніці для виявлення перешкод на своєму шляху колісних або гусеничних роботів [2].

Також на цьому модулі присутні: потенціометр для зміни чутливості давача; світлодіод для індикації живлення; світлодіод для індикації спрацьовування.

Модуль має 3 виводи:

- VCC - живлення 3-5 В;
- GND - земля;
- OUT - цифровий вихід.

2.2 Алгоритмічне забезпечення роботів

Відзначимо перш, що за ступенем участі людини в керуванні роботом виділяють біотехнічні і автономні (автоматичні) роботи. До біотехнічних відносять все дистанційно-керовані копіюють роботи, екзоскелетони, роботи, керовані людиною з пульта управління, а також напівавтоматичні роботи (включаючи роботи з супервизорним керуванням, коли оператор втручається в дії робота шляхом, наприклад, цілевказівки). Автономні роботи після їх створення і налаштування можуть, в принципі, функціонувати без участі людини. Такі роботи обов'язково повинні володіти елементами штучного інтелекту. Оскільки в ройовий робототехніці можливе застосування тільки автономних роботів, усюди далі маємо на увазі саме їх.

При побудові систем керування роєм роботів доцільно виходити з так званої композиційної концепції, подібна до фізіологічними моделями керування рухом в живих організмах і їх спільнотах. Відповідно до цієї концепції керування роєм роботів можна розбити на три ієрархічних рівня.

Перший (нижчий) рівень - це процес автоматичного регулювання окремих пристроїв одного робота, наприклад, процес відпрацювання за допомогою відповідного регулятора (елементарного автомата) задає впливу на будь-якої суглоб в андроїдному схопив. Будь-які зв'язки між окремими регуляторами (горизонтальні зв'язки) на цьому рівні відсутні.

Другий рівень керування відповідає за керування всім роботом в цілому. Систему керування цього рівня можна вважати автоматом, що об'єднує всі елементарні автомати першого рівня. При цьому кожен елементарний автомат вирішує своє власне завдання, а, наприклад, процес виконання деякої робочої операції є результатом спільної дії цих автоматів. Важливо, що керування всіма елементарними автоматами здійснюється паралельно.

Третій (верхній) рівень ієрархії керування - це керування усім роєм в цілому. Кінцевою метою керування на цьому рівні є виконання роєм поставленого перед ним завдання.

Методи, алгоритми та програмне забезпечення для синтезу елементарних автоматів в даний час добре розвинені, так що з алгоритмічної точки зору керування на нижчому рівні ієрархії не представляє проблем. Обмежуємося тому двома залишилися рівнями.

В даний час дуже багато зроблено для розробки теорії і методів синтезу автоматів, які керують роботом в цілому. Проблема ця дуже складна, але близькі проблеми доводиться вирішувати при розробці будь-якого складного сучасного виробу (літак, ракета, корабель, підводний човен і т.д.). Специфіка робототехніки в даному випадку полягає в тому, що рушії робота можуть бути кінематично дуже складними. Крім того, роботу часто доводиться працювати в умовах дуже високої природної невизначеності (переміщення по пересіченій місцевості, наприклад) і / або невизначеності, обумовленої діями і протидіями інших роботів, механізмів і людей. Останнє особливо характерно для військових роботів.

Оскільки робот функціонує в мінливому середовищі, він повинен постійно отримувати інформацію про це середовище. Тому в системі керування роботом можна виділити дві підсистеми - сенсорну і керуючу підсистеми.

2.3 Схема принципова

На рисунку 2.20 зображена схема підключення елементів.

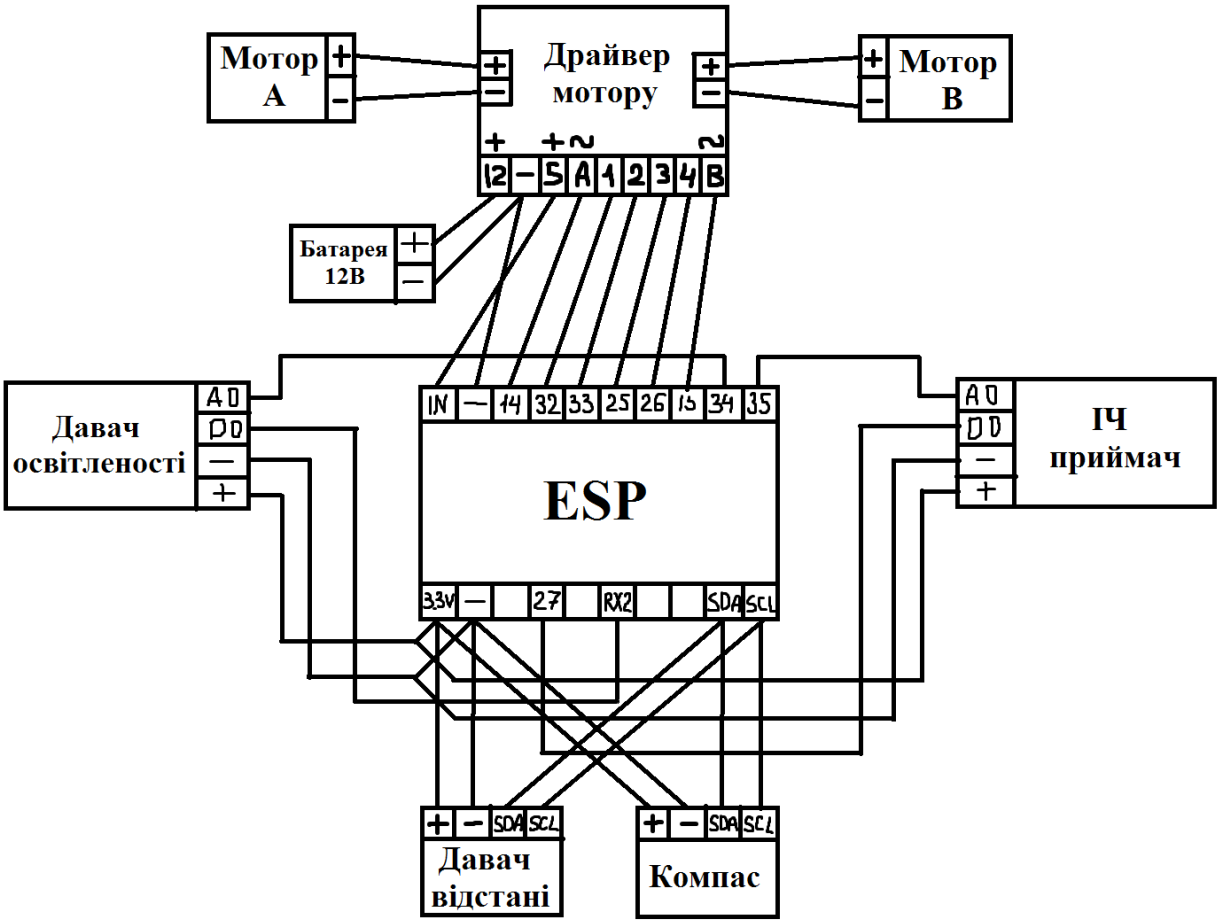


Рисунок 2.20 – Схема підключення елементів до ESP

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В цьому розділі пояснюється програмне забезпечення для окремих функцій роботів та функцій зв’язку між ними.

Робот повинен:

- рухатися в різних напрямках;
- визначати відстань до перешкод або інших роботів;
- орієнтуватися у просторі;
- дізнаватися про присутність інших роботів;
- розпізнавати інших роботів;
- отримувати команди ззовні і віддавати телеметрію.

Програмний код наведено у додатках А та Б.

3.1 Сканування

Першою функцією було реалізовано сканування навколишнього середовища. Це необхідно для того, щоб головний робот зрозумів що його оточує. Це може бути як звичайна перешкода, так і інший робот.

Робот сканує середовище навколо себе за допомогою руху за годинниковою стрілкою.

Робот обертається з кроком в 5 градусів і паралельно вимірює відстань до об’єкта та своє положення у просторі. Відстань та градус записується у двомірний масив, в який потім також будуть записуватися знайдені роботи. Так повторюється поки робот не зробить повний оберт навколо себе, тобто – 72 кроки.

Саме сканування було винесено в окрему функцію для того, щоб її можна було визивати у будь-який момент часу.

```
int calibration[ ][72] = {
        //дистанція
        {},
        //градус
        {},
        //інший робот
        {}
};
void scan() {
        Serial.println("");
        byte count = 0;
        for (int i = 0; i < 360; i = i + 5) {
                //обертаюсь на 5 градусів
                moveRight(100);
                //знаходимо та записуємо значення відстані
                distanceMm();
                calibration[0][count] = int(distance);
                Serial.print("Distance[");
                Serial.print(count);
                Serial.print("]: ");
                Serial.println(calibration[0][count]);
                // знаходимо та записуємо значення компаса
                compas();
                calibration[1][count] = int(degree);
                Serial.print("Degree[");
                Serial.print(count);
                Serial.print("]: ");
                Serial.println(calibration[1][count]);
```

```
            //якщо піймали сигнал від іншого робота
            robotNear();
            Serial.println("----------------------");
            delay(50);
            count++;
        }
    }
```

В програмі застосовуються спеціальні функції, які описані в наступних пунктах, а саме:

- moveRight(100) – функція руху за годинниковою стрілкою, яка описана у пункті 3.1.1;
- distanceMm() – функція визначення відстані до перешкод чи інших роботів, яка описана у пункті 3.1.2;
- compas() – функція орієнтації робота у просторі, яка описана у пункті 3.1.3;
- robotNear() – функція розпізнавання інших роботів за отриманим від них сигналом, яка описана у пункті 3.3.

3.1.1 Рух за годинниковою стрілкою

Рух робота було реалізовано за допомогою платформи та драйвера мотору L298N, які було розглянуто у пункті 2.1.2.

У драйвера мотора є логічні контакти керування напрямку обертів. За перший мотор відповідають IN1 та IN2, а за другий - IN3 та IN4.

Також є контакти для активації/деактивації першого і другого двигунів, це ENA та ENB відповідно. Подача логічної одиниці на ці контакти дозволяє

обертання двигунів, а логічний нуль - забороняє. Для зміни швидкості обертання щіткових моторів на ці контакти подається ШІМ-сигнал

Для того щоб робот рухався необхідно активувати ENA та ENB на необхідну швидкість. Після цього обираємо як треба рухатися. Основне завдання - рухатися за годинниковою стрілкою, тому треба подати високий сигнал на контакти IN1 та IN4.

Більш наглядно це зображено на рисунку 3.1. Де стрілка вгору – це рух колеса вперед, а стрілка вниз – рух колеса назад. Таким чином робот буде обертатися навколо себе.

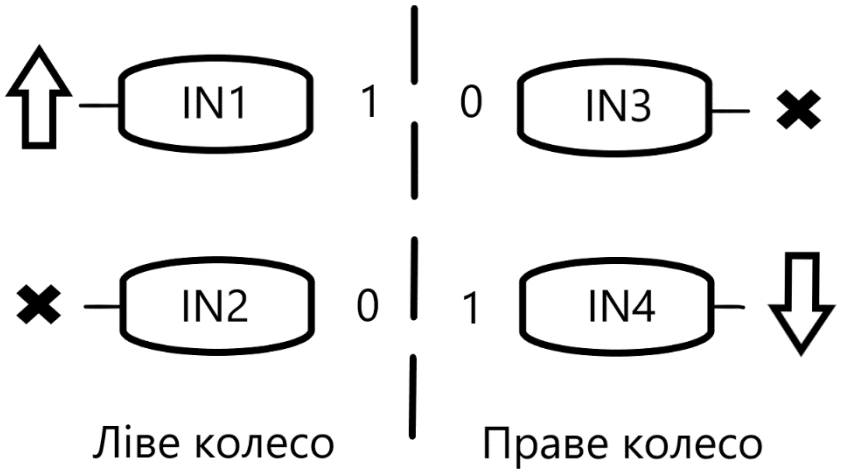


Рисунок 3.1 – Функція руху робота за годинниковою стрілкою

В програмі ця функція описана наступним чином:

```
void moveRight(int t) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(t);
}
```

3.1.2 Визначення відстані до перешкод або інших роботів

Визначення відстані до перешкод або інших роботів виконано за допомогою лазерного давача відстані VL53LOX-V2, який було розглянуто у пункті 2.1.4.

Для того щоб користуватися цим давачем необхідно завантажити бібліотеку:

```
#include <VL53L0X.h>
```

Давач працює по інтерфейсу (I2C), тому треба використовувати ще одну бібліотеку:

```
#include <Wire.h>
```

Далі ініціалізуємо бібліотеку:

```
VL53L0X sensor;
```

Можна розкоментувати рядок нижче, щоб використовувати дальній режим. Це збільшує чутливість датчика і розширює його потенційний діапазон, але збільшує ймовірність отримання неточних показань через відбиття від об'єктів, відмінних від наміченої мети. Найкраще працює в темряві.

```
#define LONG_RANGE
```

Якщо розкоментувати один з двох рядків нижче, то можна отримати більш високу швидкість за рахунок меншої точності або більш високу точність за рахунок меншої швидкості:

```
// #define HIGH_SPEED

// #define HIGH_ACCURACY
```

Оскільки робот буде сканувати місцевість дуже швидко, то в цій функції лазерний (ДВ) працює у режимі високої швидкості, тобто вимірює відстань до перешкоди приблизно за 20 мс.

```
#define HIGH_SPEED
```

Сама функція вимірювання відстані між роботом та перешкодою або іншим роботом виглядає наступним чином:

```
void distanceMm() {
    distance = sensor.readRangeSingleMillimeters();
    if (sensor.timeoutOccurred())  {
        Serial.println("Time out!");
    }
}
```

3.1.3 Орієнтація у просторі

Орієнтація у просторі виконана за допомогою модуля трьохосьового цифрового компаса GY-273 HMC5883L, який було розглянуто у пункті 2.1.3.

Для того щоб користуватися компасом необхідно завантажити бібліотеку:

```
#include < DFRobot_QMC5883.h>
```

Компас також працює по інтерфейсу (I2C), але бібліотека “Wire.h” була підключена раніше у попередньому пункті.

Далі ініціалізуємо бібліотеку:

```
DFRobot_QMC5883 compass;
```

Створюємо змінні для збереження в них значень кожної з осей, які будуть застосовуватися:

```
float xv, yv, zv, heading;
```

Створюємо глобальний масив, куди будуть розміщені калібровані дані [0]=Xc, [1]=Yc, [2]=Zc:

```
float calibrated_values[3];
```

Функція «transformation()» потрібна для корекції даних магнітометра. Як параметр функції передається float uncalibrated_values[3] - масив не каліброваних даних магнітометра.

В цю функцію підставляються відкалібровані значення з програми MagMaster.

```
void transformation(float uncalibrated_values[3]) {
    // calibration_matrix [3] [3] - матриця перетворення
    // в цей масив були підставлені значення з програми MagMaster
    double calibration_matrix[3][3] = {
        { 1.133,  0.048, -0.044},
        { 0.019,  1.141,  0.371},
        {-0.001, -0.071,  0.921}
    };
    //bias[3] - упередження
    // дані зміщення були взяті з програми MagMaster
    double bias[3] = {
        1389.898,
        4981.483,
        -97.721
    };
    // далі функція продовжується розрахунками
}
```

Функція «vector_length_stabilasation()» потрібна для стабілізації довжини вектора магнітометра (стабілізація радіуса сфери). У ній обчислюється нормальна довжина вектора та поточний скаляр.

Функція «getHeading()» потрібна для отримання значень кожної з координатних осей:

Функція «compas()» потрібна для виконання основного алгоритму та застосовування усіх попередніх функцій:

Програмний код наведено у додатку А.

3.2 Чутливість

Наступна але не менш головна – це функція чутливості робота. Під чутливістю розуміється здатність робота розуміти що поруч з ним знаходиться інший робот.

Для того щоб реалізувати почуття близькості робота можна застосовувати різні методи.

Давайте розглянемо вже відому функцію сканування. Наприклад, головний робот перебуває на відстані 1 метра до іншого робота. Роботу спочатку потрібно розвернутися, для цього треба дістати дані з масиву. Потім подається сигнал на колеса та робот під'їжджає. А далі необхідно визначити під'їхав головний робот до іншого робота чи ні. Можна знову просканувати, але це не зручно.

Процес чутливості робота реалізований за допомогою (ІЧ) випромінювання. Інші роботи передають сигнал, а головний робот приймає його.

Якщо повернутися до прикладу вище, то набагато краще отримати сигнал від (ІЧ) давача і зареєструвати що поруч хтось знаходиться. Головний робот реєструє (ІЧ) сигнал і знає без обертання що поруч хтось є.

3.2.1 Інфрачервоний передавач

Передача (ІЧ) світла реалізована за допомогою (ІЧ) давача перешкод YL-63, який було розглянуто у пункті 2.2.2.

(ІЧ) передавач розташований під рамою іншого робота. Це зроблено для того, щоб вийшла (ІЧ) пляма навколо робота з деяким радіусом.

Програмна частина на Arduino Uno дуже проста.

Спочатку вказуємо до якого піна підключений наш модуль:

```
#define IR_TRANSMITTER 3
```

Після цього у функції «setup()» необхідно обрати режим роботи:

```
pinMode(IR_TRANSMITTER, OUTPUT);
```

І врешті решт у функції «loop ()» вказуємо щоб (ІЧ) передавач завжди випромінював сигнал:

```
digitalWrite(IR_TRANSMITTER, HIGH);
```

3.2.2 Інфрачервоний приймач

Прийом (ІЧ) світла реалізований за допомогою модулю давача (ІЧ) випромінювання, який було розглянуто у пункті 2.1.5.

(ІЧ) приймач використовується для визначення сигналу від іншого робота, який знаходиться поруч. Коли роботи близько один до одного, неважливо куди повернуті, головний робот знає що поряд знаходиться інший робот. Він реєструє (ІЧ) сигнал і знає без обертання що поруч хтось є.

Програмна частина виконана на ESP32 (головний робот).

Спочатку створюємо глобальну змінну:

```
boolean near = false;
```

Коли (ІЧ) приймач уловлює сигнал, тоді в монітор порту демонструється інформація, а також в змінну записується поточний стан (true або false). Це необхідно для майбутніх дій з алгоритмами.

Наприклад, якщо поруч є якісь роботи, то вони можуть рухатися в одному напрямку.

Реалізація описана у функції «IR_SENSOR»:

```
void IR_SENSOR() {
    if (!digitalRead(INFRARED_SENSOR_A0)) {
        near = true;
        Serial.println("Robot near");
        delay(200);
    } else {
        near = false;
        Serial.println("Robot not near");
        delay(200);
    }
}
```

3.3 Розпізнавання

Є роботи до яких була знайдена відстань і місцерозташування. Немає централізованого керування. За Wi-Fi роботи отримують загальну початкову задачу (зібратися всім в купу, розбігтися по колу, рулити ними). Вони повинні реалізовувати самі якийсь алгоритм ройової поведінки. Початкове завдання -

визначити хто майстер (ESP), інші повинні під'їхати до нього. Для цього потрібно ідентифікувати хто з них хто.

У нас є резистивний датчик освітлення на стороні ESP і є потужний світлодіод на стороні Arduino. На цей резистивний датчик надівається воронка для того, щоб обмежити кут зору фоторезистора. Таким чином штучно вказуємо напрямок. Далі регулюється рівень спрацьовування. На денне освітлення непрямих сонячних променів воно не спрацьовує. Як тільки робот дивиться на іншого робота, на якому включений потужний світлодіод, повинен спрацювати датчик освітленості і з цього спрацьовування ESP реєструє що головний робот дивиться на іншого робота.

В масив окремо записується відстань та місцезнаходження саме до цього робота.

Але досі невідомо який це робот – A, B або C.

На певній відстані головний робот бачить що блимає світлодіод. Цей сигнал можна реєструвати в моніторі порту.

Фоторезистор може приймати два сигнали з нього: аналоговий і цифровий. Цифровий сигнал можна запустити на монітор порту і визначати що це за робот. Але може бути таке, що цифровий сигнал є слабким і в моніторі порту тільки сміття. За аналоговим сигналом можна визначити що головний робот дивиться на іншого робота, який знаходиться далеко і не зрозуміло який саме це робот. Є два варіанти: можна покрутити колесами щоб налаштуватися точніше або під'їхати ближче. Якщо все одно не вийшло визначити, то можна методом виключення подивитися які роботи вже були знайдені і зрозуміти якого робота не вистачає, значить це буде саме він або один з них.

Тобто, за аналоговим сигналом визначаємо що головний робот дивиться на іншого робота, а по цифровому визначати який саме це робот.

3.3.1 Емуляція послідовного порту по повітрю

Послідовний потік даних складається з бітів синхронізації та бітів даних. Формат послідовних даних містить чотири частини: стартовий біт, біти даних (5-8 бітів), перевірочний і стоповий біт; вся ця конструкція іноді називається символом. На рисунку 3.2 наведений типовий формат послідовних даних.

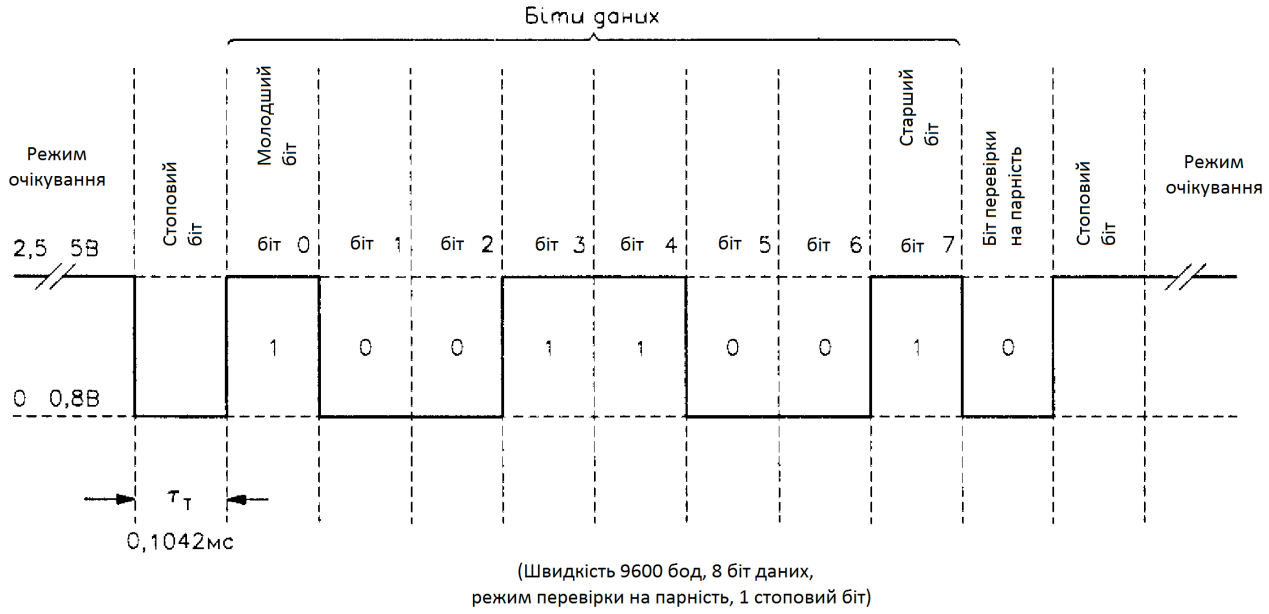


Рисунок 3.2 – Формат послідовних даних, що формовані UART

У початковий момент часу порти мають низький сигнал, як тільки в програмі прописується команда, яка конвертує їх в рівень порту, то сигнал змінюється на високий. А далі порти чекають.

Передача даних реалізована через порт TX, а прийом даних через порт RX.

Для цього може бути використаний дріт для обміну інформацією, але в цьому дипломному проекті розглянута бездротова реалізація.

Обмін інформацією буде виконуватися по повітрю за допомогою світлодіода (TX) на стороні Arduino та датчика освітленості (RX) на стороні ESP.

Схема підключення світлодіода та фоторезистора представлені на рисунку 3.4 та рисунку 3.5 відповідно.

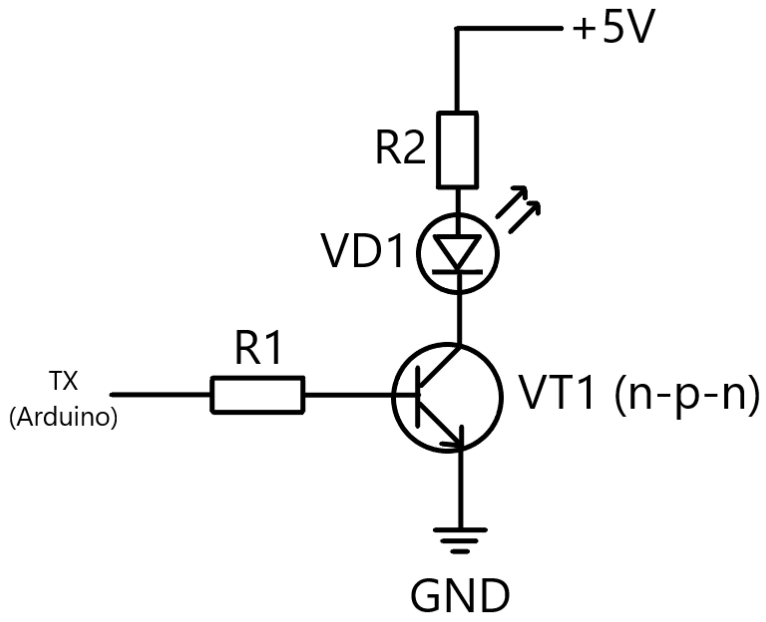


Рисунок 3.3 – Підключення світлодіода з p-n-p транзистором в ключовому режимі

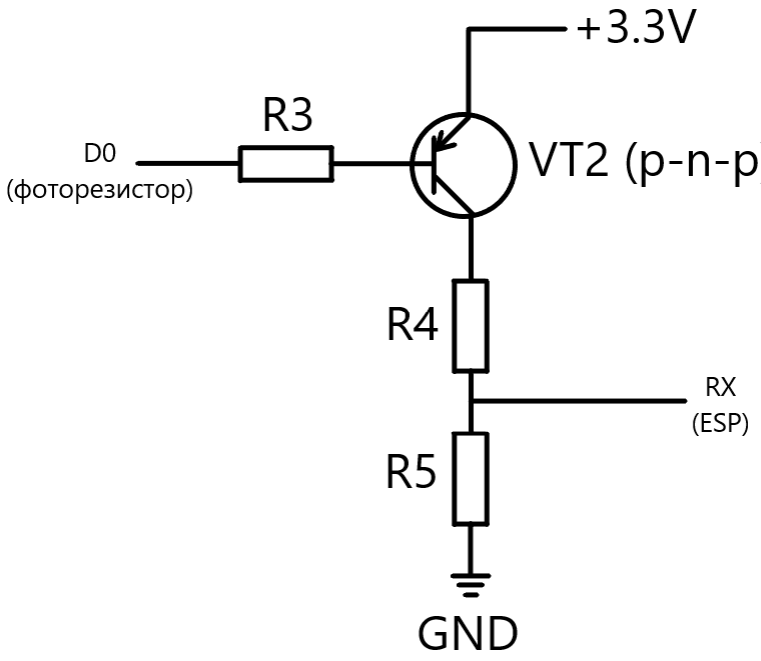


Рисунок 3.4 – Підключення фоторезистора з n-p-n транзистором в ключовому режимі

Транзистор в ключовому режимі для схеми на рисунку 3.3 працює наступним чином. Світлодіод завжди світиться, але коли посилається сигнал на базу, то світлодіод починає блимати. Швидкість порту треба ставити найменшу. В даному випадку використовувалася 1200 бод.

Якщо посилається з порту TX символ '1' на базу, включається транзистор і світлодіод починає блимати передаючи символ одиниці. В той час на порту RX, коли робот повертається і уловлює світлодіод, приходять символи '1'.

Тобто на фоторезистор приходять ті символи, які були передані через блимання світлодіода. У одного робота - це буде символ '1', у іншого - '2' і т.д. Як тільки робот повертається і бачить світлодіод, він його не просто бачить, він подає сигнал і ESP сприймає це як послідовний порт і на ньому з'являються сигнали '1', '2' та інше.

Якщо ж робот повернеться в той момент коли пересилався символ '1', наприклад, з 4 по 8 біт, то робот сприйме цей сигнал як сміття, а наступний сигнал зловить нормально.

Транзистор в ключовому режимі для схеми на рисунку 3.4 працює наступним чином. На цифровому виході фото резистора буде високий рівень коли буде освітлення. Отже, коли отримується сигнал з датчика освітленості , то транзистор постійно замикається на землю.

Таким чином повинен генеруватися сигнал, який потрібно зловити на порту RX. Тобто, те що отримується з другого послідовного порту, який було ініціалізовано для зв’язку між світлодіодом та фото резистором, треба виводити на перший послідовний порт, який ініціалізований для роботи з комп’ютером. Такий принцип називається «відлуння».

Спочатку це перевірялося за допомогою перетворювача USB-UART TTL, що дає змогу створити ще один апаратний порт. А після цього була перевірка вже безпосередньо на схемі.

3.3.2 Реалізація функції розпізнавання

Як було сказано раніше, в ході дипломного проекту було застосовано аналогово-цифровий давач освітленості. На цьому давачі йде інвертований сигнал на цифровому виході, що ускладнює роботу з транзистором і створюється можливість минути цю схему та подати цей сигнал напряму в послідовний порт RX ESP.

В процесі виконання дипломного проекту, при аналізі різних швидкостей виходить так, що в послідовний порт щось приходить. Але ідентифікувати що саме прийшло не виходить, бо приходить лише сміття у вигляді набору випадкових символів. Було виявлено це пов’язано з тим, що у фоторезистора швидкість зчитування менше, ніж швидкість блимання світлодіоду по послідовному порту. Таким чином, виконати цю реалізацію стає неможливим.

Запропоновано ідентифікувати роботів шляхом наступного визначення: яка кількість символів (сміття) прийшла в послідовний порт за певний час.

На стороні Arduino світлодіод посилає сигнал у вигляді блимання. Кількість блимання задається в циклі. У кожного робота кількість блимань своя.

```
#define LED 11
void loop() {
    for(int i = 0; i<6;i++){
        digitalWrite(LED, 1);
        delay(25);
        digitalWrite(LED, 0);
        delay(25);
    }
    delay(200);
}
```

На стороні ESP зчитується кількість символів, що прийшли в порт за час стояння робота.

```
int count;
unsigned long t1;
void setup() {
	Serial.begin(1200);
	Serial2.begin(1200);
}
void loop() {
	if(Serial2.available()){
		count++;
	}
	if(millis() - t1 >= 500) {
		if(count == 2)
			Serial.println("A");
		if(count == 4)
			Serial.println("B");
		if(count == 6)
			Serial.println("C");
		t1 = millis();
		count = 0;
	}
}
```

В прикладі на Arduino цикл повторюється 6 разів, тобто в послідовний порт буде приходити сміття 6 разів. Ця кількість рахується и через 500 мс порівнюється. Якщо прийшло 2 сигнали – це робот А, якщо прийшло 4 сигнали – це робот В і в нашому випадку прийшло 6 сигналів, тому це робот С.

Застосовується функція millis(), яка повертає кількість мілісекунд з моменту початку виконання поточної програми. Повертає unsigned long, від 1 до 4 294 967 295 мілісекунд (приблизно 50 діб), має дозвіл 1 мілісекунда, після переповнення скидається в 0. Працює на системному таймері Timer 0.

Ця функція застосовується для того щоб можна було робити перевірку скільки символів прийшло за проміжок часу. В цьому випадку проміжок часу складає 500 мс.

Для цього необхідно реалізувати свій таймер по наступній схемі:

1. виконати дію;
2. запам'ятати даний час зі старту МК (в окрему змінну);
3. шукати різницю між поточним часом і тим, яке було запам’ятовано;
4. як тільки різниця більше потрібного часу таймера – виконати дію;
5. скинути таймер.

Є два варіанти як скинути таймер, прирівняти змінну таймера до актуального millis(), або збільшувати на розмір періоду. В програмному коді використовувався перший варіант.

3.4 Отримання команд ззовні

Є роботи до яких було знайдено відстань, місце розташування і зрозуміло який саме це робот. Зараз немає централізованого управління. За Wi-fi роботи отримують загальну початкову задачу, наприклад: зібратися в купу, розбігтися по колу, рулити ними тощо.

Роботи повинні реалізовувати якийсь алгоритм ройової поведінки самостійно. Наприклад, потрібно зібратися разом, повернутися на північ і проїхати 3 метри. Початкове завдання – визначити хто майстер (ESP), інші повинні під'їхати до нього. Для цього потрібно ідентифікувати хто з них хто.

Можна реалізувати алгоритм пересування важких предметів. Роботи збираються навколо центрального або най північнішого робота, розвертаються і починають щось штовхати. Треба зрозуміти що штовхати (іншого робота або перешкоду). Роботи можуть за допомогою компасу перебудовуватися. Тобто змінювати своє положення відносно один одного.

Існує можливість зробити повноцінний двосторонній зв'язок. Роботи можуть спілкуватися за допомогою інфрачервоного передавача. Але простіше зробити посилки через роутер за допомогою тієї ж конструкції, яка використовується в цій дипломній роботі. Після ініціалізації, коли всі зареєструвалися, ESP з роутера отримує IP адресу кожного робота. А потім передача команд буде виконуватися по конкретній IP адресі.

Можна не тільки отримувати завдання, а також зробити спілкування між роботами. Зовнішнього керування немає. Використовується Wi-Fi щоб спростити передачу сигналів між роботами.

Але в цей раз, в цьому дипломі така функція не була дороблена, в зв'язку з великим обсягом інформації.

ВИСНОВКИ

Групова робототехніка є однією з ключових областей розвитку робототехнічних систем. Це пов'язано з тим, що для широкого класу практичних завдань застосування групи щодо простих роботів є набагато більш ефективним в порівнянні з використанням одного великого багатоцільового пристрою. Сучасний розвиток обчислювальної техніки і систем зв'язку відкриває широкі можливості для побудови таких систем.

У даній дипломній роботі було здійснено «Розробку конструкції, алгоритмів керування та елементів програмного забезпечення автономного робота для роботи у складі систем самовпорядкування».

За результатами виконання дипломного проєкту можна зробити наступні висновки:

- робот визначає відстань до інших роботів, визначає напрямок за допомогою компаса;
- коли робот знаходиться поруч, то за допомогою (ІЧ) давача також можна визначати який це робот.
- за аналоговим сигналом визначається що наш робот дивиться на іншого робота, а по цифровому визначається який саме це робот.
- реалізовано імітацію послідовного порту за допомогою світлодіода та фото резистора;
- за допомогою Wi-Fi можна передавати загальні команди.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Робототехника [Электронный ресурс]. – URL: https://indicator.ru/tags/robototehnika.

2. ИК датчик препятствий YL-63 [Электронный ресурс]. – URL: https://3d-diy.ru/wiki/arduino-datchiki/infrakrasnyj-datchik-prepyatstvij-yl-63/

3. Википедия. Групповая робототехника [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Групповая_ робототехника.

4. Драйвер двигателя L298N [Электронный ресурс]. – URL: https://3d-diy.ru/wiki/arduino-moduli/drayver-dvigatelya-l298n/

5. Драйвер двигателя L298N [Электронный ресурс]. – URL: https://voltiq.ru/l298n-and-arduino

6. Воротников С.А. Информационные устройства робототехнических систем : учеб. пособие. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2005. – 384 с.

7. Гедзберг Ю.М. Охранное телевидение. – М. : Горячая линия – Телеком, 2005. – 312 с.

8. ESP DevKit V1 [Электронный ресурс]. – URL: http://developer.alexanderklimov.ru/arduino/esp32/

9. Общие сведения о системе ESP32 [Электронный ресурс]. – URL: http://mypractic.ru/urok-1-obshhie-svedeniya-o-sisteme-esp32-plata-devkit-v1.html

10. Информационная надежность, контроль и диагностика навигационных систем / С.П. Дмитриев и др. – СПб. : Изд-во ГНЦ РФ ЦНИИ "Электроприбор", 2004.

11. Методы анализа и компенсации движения в динамических изображениях / Ю.Б. Зубарев, В.П. Дворкович, В.В. Нечепаев и др. // Электросвязь. – 1998. – № 11. – С. 15–21.

12. Изоткина Н.Ю., Осипов Ю.М., Сырямкин В.И. Инновационные технологии управления в мехатронике и робототехнике : учеб. пособие / под общ. ред. Ю.М. Осипова. – Томск : Изд. Дом ТГУ, 2015. – 220 с.

13. Карпова И.П. Псевдоаналоговая коммуникация в группе роботов // Мехатроника. Автоматизация. Управление. – 2016. – № 2. – С. 94–101.

14. Карпенко А.П. Робототехника и системы автоматизированного проектирования : учебное пособие. – М., 2014. – 73 с.

15. Карпов В.Э. Коллективное поведение роботов. Желаемое и действительное // Современная мехатроника. Сборник научных трудов Всероссийской научной школы (г. ОреховоЗуево, 22-23 сентября 2011) – Орехово-Зуево, 2011. – С 35– 51.

16. Каляев И.А., Гайдук А.Р., Капустян С.Г. Модели и алгоритмы коллективного управления в группах роботов. М.: Физматлит, 2009, 280 с.

17. Каляев И.А., Гайдук А.Р. Стайные принципы управления в группе объектов // Мехатроника, автоматизация, управление. 2004. № 12. С. 27—38.

18. Ерофеева В.А, Иванский Ю.В., Кияев В.И. Управление роем динамических объектов на базе мультиагентного подхода // Компьютерные системы в образовании. – СПб., 2015. – № 6. – С. 34–42.

19. Карпенко А.П. Робототехника и системы автоматизированного проектирования : учебное пособие. – М., 2014. – 73 с.

20. Иванов. Д.Я. Использование принципов роевого интеллекта для управления целенаправленным поведением массовоприменяемых микророботов в экстремальных условиях // Известия высших учебных заведений. – 2011. – № 9. – С. 70–78.

21. Arduino Uno [Электронный ресурс]. – URL: http://arduino.ru/Hardware/ArduinoBoardUno

22. АЛГОРИТМЫ

23. Каляев И.А., Гайдук А.Р., Капустян С.Г. Модели и алгоритмы коллективного управления в группах роботов. – М. : Физматлит, 2009. – 280 с

24. Advanced hard and soft iron magnetometer calibration for dummies [Электронный ресурс]. – URL: https://diydrones.com/profiles/blogs/advanced-hard-and-soft-iron-magnetometer-calibration-for-dummies

25. Модуль GY-273 HMC5883L трехосевой цифровой компас [Электронный ресурс]. – URL: https://3v3.com.ua/product_8432.html

26. GY-273 трехосевой цифровой компас магнитометр [Электронный ресурс]. – URL: https://arduinka.biz.ua/ru/Gy-273-trehosevoy-tsifrovoy-kompas-magnitometr-p747c74.html

27. Лазерний датчик відстані VL53L0X-V2 [Электронный ресурс]. – URL: https://arduino.ua/prod2467-lazernii-datchik-rasstoyaniya-vl53l0x-v2

28. Обзор лазерного дальномера VL53L0X [Электронный ресурс]. – URL: https://robotchip.ru/obzor-lazernogo-dalnomera-vl53l0x/

29. Модуль датчика ІЧ-випромінювання [Электронный ресурс]. – URL: https://arduino.ua/prod1208-modyl-datchika-ik-izlycheniya

ДОДАТОК А

(Текст програми для ESP)

```
// підключення необхідних бібліотек
#include <WiFi.h>
#include <DFRobot_QMC5883.h>
#include <VL53L0X.h>
#include <Wire.h>
// підключення електроніки
#define LIGHT_SENSOR_A0 34
#define INFRARED_SENSOR_A0 35
#define INFRARED_SENSOR_D0 27
#define IN1 32
#define IN2 33
#define IN3 25
#define IN4 26

/*------------ Wi-Fi ініціалізація ------------*/

const char* ssid     = "TP-LINK_F9CE";
const char* password = "30741409";
WiFiServer server(80);
```

```
/*------------ ДРАЙВЕР МОТОРА ініціалізація ------------*/

const int driverSpeed = 255;

/*------------ КОМПАС ініціалізація ------------*/

DFRobot_QMC5883 compass;
float xv, yv, zv;
float heading;

//calibrated_values[3] is the global array where the calibrated data will be placed
//calibrated_values[3]: [0]=Xc, [1]=Yc, [2]=Zc
float calibrated_values[3];

/*------------ ДАВАЧ ВІДСТАНІ ініціалізація ------------*/

VL53L0X sensor;

/* Розкоментуйте рядок нижче, щоб використовувати дальній режим. Це збільшує чутливість датчика і розширює його потенційний діапазон, але збільшує ймовірність отримання неточних показань через відбиття від об'єктів, відмінних від наміченого об'єкта. Найкраще працює в темряві.

*/
```

```
#define LONG_RANGE

/*

Розкоментуйте один з двох рядків нижче, щоб отримати:

- більш високу швидкість за рахунок меншої точності АБО

- більш високу точність за рахунок меншої швидкості

*/

//#define HIGH_SPEED

#define HIGH_ACCURACY


/*------------ СКАНУВАННЯ ініціалізація ------------*/


float distance;

float degree;

char robot;


char robotA = 'A';

char robotB = 'B';

char robotC = 'C';


byte newRow = 2;
```

```
int calibration[][72] =
{
  {}, // відстань
  {}, // кут
  {}  // інший робот
};

/*------------ РОЗПІЗНАВАННЯ ініціалізація ------------*/

// змінна для зберігання символа який прийшов зі світлодіода
String msg;
// змінна для зберігання кількості символів, які прийшли в послідовний порт
int count;
// змінна для зберігання значення таймера
unsigned long t1;

/*------------ ДРАЙВЕР МОТОРА функції ------------*/

// рух прямо
void moveForward(int t) {
    digitalWrite(IN1, HIGH);
```

```
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    delay(t);
}

// рух назад
void moveBackward(int t) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(t);
}

// зупинка
void moveStop(int t) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
```

```
    delay(t);
}

// рух ліворуч
void moveLeft(int t) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    delay(t);
}

// рух праворуч
void moveRight(int t) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(t);
}
```

```
/*------------ КОМПАС функції ------------*/


void transformation(float uncalibrated_values[3]) {
    // calibration_matrix [3] [3] - матриця перетворення
    // в цей масив були підставлені значення з програми MagMaster
    double calibration_matrix[3][3] = {
        { 1.133,  0.048, -0.044},
        { 0.019,  1.141,  0.371},
        {-0.001, -0.071,  0.921}
    };
    //bias[3] - упередження
    // дані зміщення були взяті з програми MagMaster
    double bias[3] = {
        1389.898,
        4981.483,
        -97.721
    };
    // розрахунок
    for (int i=0; i<3; ++i)
        uncalibrated_values[i] = uncalibrated_values[i] - bias[i];
    float result[3] = {0, 0, 0};
```

```
        for (int i=0; i<3; ++i)
                for (int j=0; j<3; ++j)
                        result[i] += calibration_matrix[i][j] * uncalibrated_values[j];
        for (int i=0; i<3; ++i)
                calibrated_values[i] = result[i];
}


float scaler;
boolean scaler_flag = false;
float normal_vector_length;


void vector_length_stabilasation(){
        // обчислення нормальної довжини вектора
        if (scaler_flag == false) {
                getHeading();
                normal_vector_length=sqrt(calibrated_values[0]*calibrated_values[0]
                                        + calibrated_values[1]*calibrated_values[1]
                                        + calibrated_values[2]*calibrated_values[2] );
                scaler_flag = true;
        }
        // обчислити поточний скаляр
```

```
        scaler = normal_vector_length / sqrt (
                                        calibrated_values[0]*calibrated_values[0] +
                                        calibrated_values[1]*calibrated_values[1] +
                                        calibrated_values[2]*calibrated_values[2]
                                        );
        // застосувати поточний скаляр до відкаліброваних координат
        calibrated_values[0] = calibrated_values[0]*scaler;
        calibrated_values[1] = calibrated_values[1]*scaler;
        calibrated_values[2] = calibrated_values[2]*scaler;
}

void getHeading() {
        Vector raw = compass.readRaw();
        xv = (float)raw.XAxis;
        yv = (float)raw.YAxis;
        zv = (float)raw.ZAxis;
}

void compas(){
        float values_from_magnetometer[3];
        getHeading();
        values_from_magnetometer[0] = xv;
```

```
        values_from_magnetometer[1] = yv;
        values_from_magnetometer[2] = zv;
        transformation(values_from_magnetometer);
        vector_length_stabilasation();
        heading = atan2(calibrated_values[0], -calibrated_values[1]);
        float declinationAngle = (4.0 + (26.0 / 60.0)) / (180 / PI);
        heading += declinationAngle;
        if (heading < 0) {
            heading += 2 * PI;
        }
        if (heading > 2 * PI) {
            heading -= 2 * PI;
        }
        degree = heading * 180/PI;
}

/*------------ ДАВАЧ ВІДСТАНІ ------------*/

void distanceMm()
{
   distance = sensor.readRangeSingleMillimeters();
```

```
    if (sensor.timeoutOccurred())
    {
      Serial.println("ТАЙМАУТ");
    }
}

/*------------ СКАНУВАННЯ функція ------------*/

void scan()
{
  Serial.println("");
  byte countArr = 0;
  for(int i=0; i<360; i=i+5)
  {
    // обертання на 5 грудсів
    moveRight(100);

    //запис значень відстані у масив
    distanceMm();
    calibration[0][countArr] = int(distance);
    Serial.print("Distance[");
```

```
    Serial.print(countArr);
    Serial.print("]: ");
    Serial.println(calibration[0][countArr]);

    //запис значень градуса у масив
    compas();
    calibration[1][countArr] = int(degree);
    Serial.print("Degree[");
    Serial.print(countArr);
    Serial.print("]: ");
    Serial.println(calibration[1][countArr]);

    //якщо зловили сигнал від іншого робота, то записуємо його дані в масив
    robotNear();

    Serial.println("---------------------");
    delay(50);
    countArr++;

  }
}
```

```
/*------------ ВІДОБРАЖЕННЯ МАСИВУ функція ------------*/


void showArray(){
   for (int j = 0; j < 72; j++) {
      Serial.println("");


      Serial.print("Distance[");
      Serial.print(j);
      Serial.print("]: ");
      Serial.print(calibration[0][j]);


      Serial.print(" ");


      Serial.print("Degree[");
      Serial.print(j);
      Serial.print("]: ");
      Serial.print(calibration[1][j]);


      Serial.println("");
   }
}
```

```
/*---------- ВІДОБРАЖЕННЯ РОБОТІВ функція ----------*/


void showRobots(){
    for (int r = 2; r < sizeof(calibration); r++) {
        Serial.println("");


        Serial.print("Distance: ");
        Serial.print(calibration[0][r]);
        Serial.print(" ");


        Serial.print("Degree: ");
        Serial.print(calibration[1][r]);
        Serial.print(" ");


        Serial.print("Name: ");
        Serial.print(calibration[2][r]);


        Serial.println("");
    }
}
```

```
/*------------ ЗНАХОДЖЕННЯ РОБОТА ПОРУЧ функція ------------*/


void IR_SENSOR(){
   if(!digitalRead(INFRARED_SENSOR_A0))
   {
     Serial.println("Robot near");
     delay(50);
   } else {
     Serial.println("Robot not near");
     delay(50);
   }
}


/*------------ РОЗПІЗНАВАННЯ функції ------------*/


void robotNear()
{
  // якщо прийшли якісь дані то виконується наступна умова
  if(Serial2.available()){
    // можна перевірити що прийшло
    msg = Serial2.read();
```

```
    // коли щось прийшло - додаємо до змінної одиницю
    count++;
   }

  // ця перевірка виконується після того як пройшло 500 мс з початку виконання програми
  if(millis() - t1 >= 500) {

    // якщо за 500 мс прийшло тільки 2 символа, то це робот A
    initialization(2, robotA);
    // якщо за 500 мс прийшло тільки 4 символа, то це робот B
    initialization(4, robotB);
    // якщо за 500 мс прийшло тільки 6 символів, то це робот C
    initialization(6, robotC);

    // скидуємо таймер прирівнюючи змінну таймера до актуального millis()
    t1 = millis();
    // скидуємо кількість символів щоб все працювало коректно
    count = 0;
   }
  }
```

```
void initialization(int symbol, char robots) {
  if(count == symbol){
      // записуємо значення відстані
      distanceMm();
      calibration[newRow][0] = int(distance);
      // записуємо значення градуса
      compas();
      calibration[newRow][1] = int(degree);
      // записуємо ім'я робота
      calibration[newRow][2] = int(robots);


      Serial.print("This is robot ");
      Serial.println(calibration[newRow][2]);
      Serial.print("Distance: ");
      Serial.println(calibration[newRow][0]);
      Serial.print("Degree: ");
      Serial.println(calibration[newRow][1]);
      newRow++;
      delay(50);
    }
}
```

```
void setup()
{
  // запуск послідовного порту для зв'зку з комп'ютером
  Serial.begin(115200);
  // запуск послідовного порту для емуляції світлодіода та фоторезистора
  Serial2.begin(1200);
  pinMode(INFRARED_SENSOR_A0, INPUT);
  pinMode(INFRARED_SENSOR_D0, INPUT);

/*------------ WiFi ------------*/

  //Serial.begin(115200);

  delay(10);

  // Починаємо з підключення до мережі WiFi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
```

```
  WiFi.begin(ssid, password);


  while (WiFi.status() != WL_CONNECTED) {
     delay(500);
     Serial.print(".");
  }


  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());


  server.begin();


/*------------ ДРАЙВЕР МОТОРА ------------*/


  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
```

```
/*------------ КОМПАС ------------*/

  // для роботи з I2C
  Wire.begin();

  while (!compass.begin()){
     Serial.println("Could not find a valid QMC5883 sensor, check wiring!");
     delay(500);
  }
  compass.setRange(QMC5883_RANGE_2GA);
  compass.setMeasurementMode(QMC5883_CONTINOUS);
  compass.setDataRate(QMC5883_DATARATE_50HZ);
  compass.setSamples(QMC5883_SAMPLES_8);

/*------------ ДАВАЧ ВІДСТАНІ ------------*/

  //sensor.init();
  sensor.setTimeout(500);
  //sensor.startContinuous();
  if (!sensor.init())  {
    Serial.println("Не вдалось виявити і ініціалізувати давач!");
```

```
  while (1) {}
 }
 #if defined LONG_RANGE
  // знижує ліміт швидкості обратного сигналу (за умовою 0,25 MCPS (мчип / с))
  sensor.setSignalRateLimit(0.1);
  // збільшити періоди лазерного імпульсу (за умовою 14 і 10 PCLK)
  // * - PCLK - це частота периферії
  sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodPreRange, 18);
  sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodFinalRange, 14);
 #endif

 #if defined HIGH_SPEED
  // зменшити таймінг до 20 мс (за умовою близько 33 мс)
  sensor.setMeasurementTimingBudget(20000);
 #elif defined HIGH_ACCURACY
  // збільшити таймінг 200 мс
  sensor.setMeasurementTimingBudget(200000);
 #endif
 delay(1000);

}
```

```
int value = 0;


void loop()
{
  WiFiClient client = server.available();   // слухати клієнтів, що входять
  if (client) {                             // якщо є клієнт,
    Serial.println("New Client.");          // роздрукувати повідомлення з послідовного порту
    String currentLine = "";                // створити рядок для зберігання вхідних даних від клієнта
    while (client.connected()) {            // цикл, поки клієнт підключений
      if (client.available()) {             // якщо є байти для читання з клієнта,
        char c = client.read();             // прочитайте байт, тоді
        Serial.write(c);                    // роздрукувати його на послідовному моніторі
        if (c == '\n') {                    // якщо байт - символ нового рядка
          // якщо поточний рядок порожній, ви отримали два символи нового рядка поспіль.
          // на цьому клієнтський HTTP-запит закінчується, тому надішліть відповідь:
          if (currentLine.length() == 0) {
            // Заголовки HTTP завжди починаються з коду відповіді (наприклад, HTTP / 1.1 200 OK)
```

```
            // та тип вмісту, щоб клієнт знав, що буде, а потім порожній рядок:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // вміст відповіді HTTP йде за заголовком:
            client.print("Click <a href=\"/S\">here</a> to scan around.<br>");
            client.print("Click <a href=\"/A\">here</a> to show array.<br>");
            client.print("Click <a href=\"/R\">here</a> to show another robots.<br>");
            // Відповідь HTTP закінчується іншим порожнім рядком:
            client.println();
            // вирватися з циклу while:
            break;
          } else {   // якщо у вас новий рядок, то очистіть currentLine:
            currentLine = "";
          }
        } else if (c != '\r') {  // якщо у вас є щось інше, крім символу повернення каретки,
          currentLine += c;      // додаємо його в кінець currentLine
        }
```

```
        // Перевірте, чи був запит клієнта "GET / S" або "GET / A":
        if (currentLine.endsWith("GET /S")) {
          scan();               // вмикаємо функцію сканування
        }
        if (currentLine.endsWith("GET /A")) {
          showArray();          // вмикаємо функцію демонстрації масиву
        }
        if (currentLine.endsWith("GET /R")) {
          showRobots();         // вмикаємо функцію демонстрації інших роботів
        }
      }
    }
    // розірвати з'єднання:
    client.stop();
    Serial.println("Client Disconnected.");
  }

  IR_SENSOR();      // вмикаємо функцію виявлення роботів поруч

}
```

ДОДАТОК Б

(Текст програми для Arduino)

```
// підключеня пінів

#define LED 11

#define IR_TRANSMITTER 3


// змінні роботів

int robotA = 2;

int robotB = 4;

int robotC = 6;


void setup()

{


  // запуск послідовного порту для емуляції світлодіода та фоторезистора

  Serial.begin(1200);

  // передаю ІЧ сигнал

  digitalWrite(IR_TRANSMITTER, 1);


}
```

```
void loop() {
  /* В циклі виконується блимання світлодіодом. Якщо в умову циклу ввести robotC, то світлодіод буде блимати 6 разів. Для інших роботів можно прошивати такий же скетч, але застосовувати змінні robotA або robotC */
  for(int i = 0; i < robotA; i++){
    digitalWrite(LED, 1);
    delay(25);
    digitalWrite(LED, 0);
    delay(25);
  }
  /* Затримка для того щоб таймер на ESP спрацьовував правильно.
  Цикл виконується:
    (25 + 25)* robotA = 100 або
    (25 + 25)* robotB = 200 або
    (25 + 25)* robotC = 100 мс.
  Тому і затримка буде:
    500 - 100 = 400 або
    500 - 200 = 300 або
    500 - 300 = 200 мс.
  */
  delay(400);
}
```