Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМПЛЕКСУ РОБОТ-САПЕР

Виконав студент 4 курсу, групи ЗКН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Дацюк А. І.
(прізвище та ініціали)
Керівник: к.т.н., доцент каф. КН
Барабан С. В.
(прізвище та ініціали)
«07» 06 2024 р.
Рецензент:
Юхимчук М. С.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри Яровий А. А.
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

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

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«07» _____03_____ 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Дацюку Андрію Івановичу

1. Тема роботи: розробка програмного забезпечення для комплексу робот-сапер.
Керівник роботи: к.т.н., доцент Барабан С. В.
Затверджені наказом вищого навчального закладу "11" 03 2024 року № 80 .
2. Термін подання студентом роботи «27» _____05_____ 2024 р.
3. Вихідні дані до роботи:
Мова програмування C++ та Python;
Середовище розробки Visual Studio Code;
Середовища роботи програм Arduino Uno та Windows комп'ютер;
Типи передачі команд по проводу і bluetooth;
Кількість керованих механізмів 2-3.
4. Зміст текстової частини (перелік питань, які потрібно розробити):
   - вступ;
   - аналіз предметної області;
   - розробка апаратної частини;
   - розробка програмної частини;
   - перевірка комплексу;
   - висновок;
   - список використаних джерел;
   - додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
   - схема розведення електричної проводки в роботі;
   - діаграми класів програмного забезпечення;
   - загальний вигляд інтерфейсу користувача;
   - приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	к.т.н., доцент Барабан С. В.		

7. Дата видачі завдання «07» 03 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва і зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	06.05.2024	11.05.2024	
2	Розробка апаратної частини	12.05.2024	20.05.2024	
3	Розробка програмної частини	21.05.2024	26.05.2024	
4	Тестування комплексу	27.05.2024	29.05.2024	
5	Розробка інструкції користувача	30.05.2024	03.06.2024	
6	Оформлення матеріалів БДР	04.06.2024	06.06.2024	

Студент _____ Дацюк А

Керівник роботи _____ Барабан

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 68 сторінок формату А4, на яких є 11 рисунків, список використаних джерел містить 24 найменувань.

У даній бакалаврській дипломній роботі досліджено процес розробки програмного забезпечення для програмно-апаратного комплексу робот-сапер та розробка цього комплексу. В роботі проаналізовано види дистанційного керування, види роботизованих апаратів та способи їх взаємодії в комплексі. Наведено переваги та недоліки подібних рішень. Запропоновано вирішення проблеми на основі розробки програмного модуля та комплексу керування та виконання. Наведено діаграми станів та діаграми компонентів, розроблено алгоритм роботи комплексу. Проведено тестування комплексу. Результатом дослідження є застосунок з необхідним функціоналом, що реалізований на мові програмування Python а також модуль керування робота з використанням мови C++ та сам робот. Розроблений комплекс призначений для допомоги саперам в знешкодженні деяких типів мін.

Ключові слова: робот, мікроконтролер, дистанційне керування, застосунок, Python, C++, Arduino, Windows.

ABSTRACT

Bachelor thesis consists of 68 pages of A4 format, which have 11 pictures, the list of used sources has 24 denominations.

In this bachelor thesis, the process of software development for the robot-sapper hardware and software complex and the development of this complex are investigated. The work analyzed types of remote control, types of robotic devices and methods of their interaction in a complex. The advantages and disadvantages of such solutions are given. A solution to the problem is proposed based on the development of a software module and a control and execution complex. State diagrams and component diagrams are presented, the algorithm of the complex's operation is developed. The complex was tested. The result of the research is an application with the necessary functionality implemented in the Python programming language, as well as a robot control module using the C++ language and the robot itself. The developed complex is designed to help sappers in decontamination of some types of mines.

Keywords: robot, microcontroller, remote control, application, Python, C++, Arduino, Windows.

ЗМІСТ

ВСТУП

Актуальність теми. В наш час значного поширення набули роботизовані комплекси різного ступеня автоматизації та різного призначення. Їх використовують для вирішення значного кола різноманітних завдань.

Значну роль роботизовані комплекси відіграють і у військовій справі, в тому числі і в саперній. Потрібна роботизована платформа на яку можна буде встановити різноманітні модулі розширення, різні для різних ситуацій та сценаріїв використання.

Дана бакалаврська робота присвячена створенню роботизованого комплексу та системи дистанційного управління ним. Робот буде являти з себе багатофункціональну платформу, а за допомогою додатку можна буде керувати ним дистанційно з безпечного місця.

При створенні комплексу буде використано сучасні технології керування та передачі сигналу.

Очікується що даний комплекс дозволить розширити можливості саперних військ та дозволить зменшити можливі втрати та розширить можливості.

Об’єктом досліджень є процес керування комплексом робот-сапер.

Предметом досліджень є програмні засоби керування комплексом робот-сапер.

Метою роботи є розширення функціональних можливостей програмного забезпечення керування роботизованим комплексом робот-сапер.

Для реалізації мети бакалаврської роботи необхідно вирішити наступні задачі:

- проаналізувати аналоги та розробити відповідний концепт
- розробити структурну схему робота
- розробити програмний модуль робота
- розробити програне забезпечення для керування роботом
- провести тестування комплексу

Публікації: за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз проблем розробки комплексів управління роботами

Розробка комплексів управління роботами є важливим аспектом сучасної робототехніки, оскільки ці системи забезпечують автономність та ефективність роботів у різних сферах діяльності. Проте, цей процес супроводжується низкою проблем, які вимагають детального аналізу та вирішення.

Однією з ключових проблем є створення ефективного інтерфейсу людина-машина. Він має бути інтуїтивно зрозумілим для користувачів з різним рівнем технічної підготовки, а також забезпечувати можливість швидкого навчання та адаптації до змін у функціональності роботів.

Кілька можливих рішень:

- використання природних мов: інтеграція систем розпізнавання мови та голосових команд може значно спростити взаємодію з роботом.
- графічні інтерфейси: розробка зручних графічних інтерфейсів, які дозволяють користувачам легко управляти роботом через сенсорні екрани або інші візуальні елементи.

Безпека є критично важливим аспектом при розробці таких комплектів, особливо тому що роботи взаємодіють з людьми та працюють у небезпечних умовах. Необхідно забезпечити захист як користувачів, так і самого обладнання від потенційних аварій або неправильних дій роботів та інших загроз. Для цього використовується дистанційне керування. У випадку ж неможливості використання радіоканалу можливо використовувати керування по дротах. Це більш затратне керування і менш зручне проте більш надійне та просте у використанні, особливо у складних умовах. Також будуть застосовані спеціальні елементи захисту робота в разі необхідності.

Також важливими факторами є ціна робота та його простота та малий час виробництва. Саме тому робот побудований з доступних комплектуючих, має доволі просту конструкцію та відносно низьку ціну.

1.2 Огляд існуючих комплексів управління роботами

На сьогоднішній день існує багато різноманітних роботизованих комплексів та засобів керування ними.

Одним із таких комплексів є «TALON»[21] (рис. 1.1) розроблений компанією Foster-Miller (нині QinetiQ North America), є одним з найбільш поширених військових роботів-саперів. Він використовується для розвідки, знешкодження вибухових пристроїв і виконання бойових завдань.

Робот TALON - це американський робот-сапер. Він був створений для розмінування територій, виявлення та знешкодження вибухонебезпечних предметів, що залишилися після бойових дій.

Робот TALON має наступні характеристики та можливості:

- Видимість: Робот може працювати в різних умовах, включаючи темряву, дощ та сніг.
- Дистанційне керування: TALON може бути керований з відстані, що дозволяє операторам залишатися в безпеці під час розмінування.
- Виявлення вибухонебезпечних предметів: Робот оснащений сенсорами, які можуть виявляти вибухонебезпечні предмети, такі як міни, гранати та інші боєприпаси.
- Знешкодження: TALON може знешкоджувати виявлені вибухонебезпечні предмети, використовуючи різні методи, такі як демонтаж або дистанційне знищення.
- Мобільність: Робот може рухатися по різноманітних поверхнях, включаючи ґрунтові, асфальтові та бетонні.
- Збереження життя: TALON дозволяє операторам залишатися в безпеці під час розмінування, що зменшує ризик втрат серед військовослужбовців та цивільних осіб.

У 2014 році компанія QinetiQ North America передала Україні кілька партій роботів TALON для допомоги у розмінуванні територій, які були зайняті під час російсько-української війни. Ці роботи допомогли українським військовослужбовцям

та саперам виявляти та знешкоджувати вибухонебезпечні предмети, що залишилися після бойових дій.

Компанія QinetiQ North America також розробила бойові версії роботів TALON, які можуть оснащуватися вогнепальною зброєю, такою як гвинтівка M16 або кулемет M249. Ці версії називаються SWORDS TALON, і вони можуть бути використані для знищення цілей на полі бою. Однак немає інформації, що саме ці бойові версії передавалися Україні.

Вартість одного робота TALON становить близько 1,5 мільйона доларів США. Компанія QinetiQ North America отримала кілька замовлень на роботу TALON від різних країн, включаючи США, Великої Британії та України.

Робот TALON є важливим інструментом для розмінування територій та знищення вибухонебезпечних предметів. Його можливості та характеристики роблять його корисним для різних країн та організацій, які працюють над безпекою та стабільністю.

Рисунок 1.1 – Робот «TALON»

Плюси:

- універсальність: даний робот може бути оснащений різними модулями, включаючи сенсори, камери, маніпулятори та зброю.
- надійність: висока ефективність та доведена надійність у бойових умовах.
- простота управління: інтуїтивно зрозумілий інтерфейс для операторів.
- мобільність: може пересуватися по складній місцевості, включаючи воду, пісок і сніг.

Мінуси:

- вартість: висока вартість як самого робота, так і його обслуговування.
- вага: досить важкий, що може ускладнювати його транспортування в деяких ситуаціях.

Ще одним схожим комплексом являється робот «PackBot» [2] (рис. 1.2), створений компанією iRobot (нині Endeavor Robotics), є компактним і мобільним рішенням для знешкодження вибухових пристроїв.

PackBot - це серія військових роботів, розроблених компанією Endeavor Robotics (раніше - iRobot), міжнародною компанією з робототехніки, заснованою у 2016 році на базі iRobot, яка виробляла військових роботів з 1990 року. Більше 2000 таких роботів було використано в Іраку та Афганістані. Вони також допомагали у пошуках по завалах Всесвітнього торгового центру після подій 11 вересня 2001 року. Технологія PackBot також використовувалася для оцінки пошкодженого ядерного реактора в Фукусімі після землетрусу та цунамі 2011 року. У листопаді 2014 року армія США відновлює 224 роботи iRobot 510. Технологія PackBot також використовується у співпраці з NASA для їхніх роверів та зондів.

PackBot був розроблений як пошуково-рятувальний та робот для розмінування компанією iRobot. Більше 4500 одиниць було використано американськими військовими в Іраку та Афганістані для пошуку та знешкодження саморобних вибухових пристроїв. Пізніше робот був комерціалізований компанією Endeavor Robotics, яку у 2019 році придбала Teledyne FLIR.

PackBot має можливість піднімати вантаж вагою до 20 кг, сходити по сходах та нахилам до 45 градусів, обладнаний кількома камерами та аудіосистемою. Він також

має різні сенсори, включаючи камери, GPS, компас, акселерометри, інклінометр, а також може бути оснащений додатковими сенсорами для виявлення газів, хімічних боєприпасів, гамма-випромінювання та теплової камери. PackBot працює на літій-іонній батареї з можливістю видалення, що забезпечує 4 години роботи.

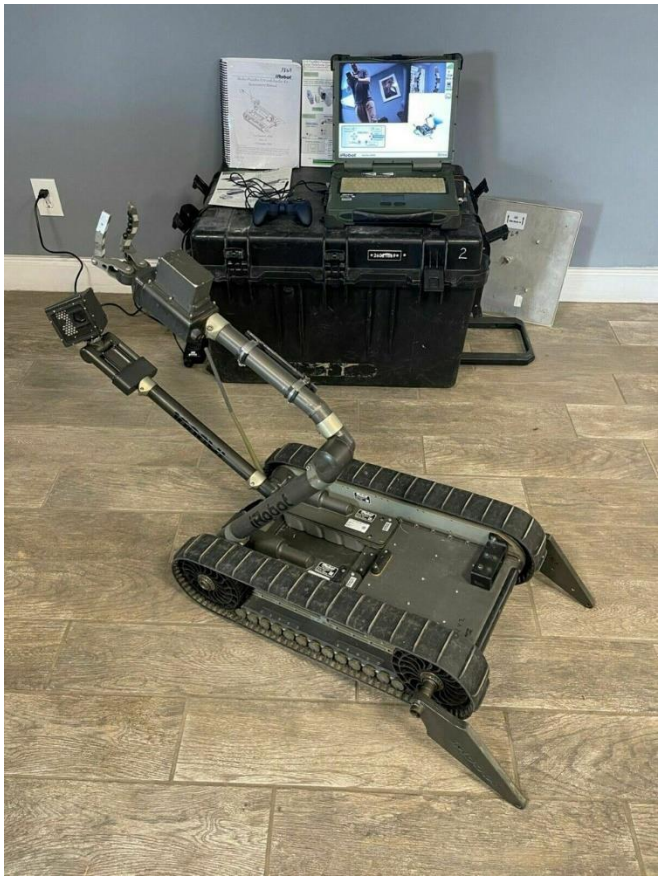

Рисунок 1.2 – Робот «PackBot»

Плюси:

- компактність: легкий і портативний, що дозволяє швидко розгортати його в різних умовах.
- модульність: може оснащуватися різними датчиками та маніпуляторами для виконання широкого спектра завдань.
- швидкість: висока швидкість пересування, що дозволяє оперативно реагувати на загрози.

Мінуси:

- обмежена вантажопідйомність: не може піднімати важкі об'єкти, що може обмежувати його використання у деяких ситуаціях.

- чутливість до пошкоджень: менша стійкість до фізичних пошкоджень порівняно з більш масивними роботами.

Також існує такий роботизований комплекс як «ANDROS F6A» [3] (рис. 1.3), вироблений компанією Northrop Grumman, є одним з найпотужніших роботів-саперів, що використовується військовими та правоохоронними органами по всьому світу.

ANDROS F6A - це військовий робот, розроблений компанією Remotec, що є дочірньою компанією Northrop Grumman. Робот був створений для дистанційного розмінування та пошуку вибухонебезпечних предметів.

ANDROS F6A має наступні характеристики та можливості:

- Дистанційне керування: Робот може бути керований оператором з відстані, що дозволяє операторам залишатися в безпеці під час розмінування.
- Виявлення вибухонебезпечних предметів: ANDROS F6A оснащений сенсорами, які можуть виявляти вибухонебезпечні предмети, такі як міни, гранати та інші боєприпаси.
- Знешкодження: Робот може знешкоджувати виявлені вибухонебезпечні предмети, використовуючи різні методи, такі як демонтаж або дистанційне знищення.
- Мобільність: ANDROS F6A може рухатися по різноманітних поверхнях, включаючи ґрунтові, асфальтові та бетонні.
- Збереження життя: Робот дозволяє операторам залишатися в безпеці під час розмінування, що зменшує ризик втрат серед військовослужбовців та цивільних осіб.

ANDROS F6A використовувався в різних країнах, включаючи США та Ізраїль, для розмінування територій та пошуку вибухонебезпечних предметів. Робот був успішно використаний у різних операціях, включаючи пошуки по завалах Всесвітнього торгового центру після подій 11 вересня 2001 року.

ANDROS F6A був розроблений компанією Remotec, що є дочірньою компанією Northrop Grumman. Компанія Remotec була заснована в 2005 році і спеціалізується на розробці дистанційно керованих військових роботів. ANDROS F6A є частиною серії

військових роботів, розроблених компанією Remotec, яка включає також інші моделі, такі як H1, Wolverine V2 та II mini.

Вартість одного робота ANDROS F6A становить близько 1,5 мільйона доларів США. Компанія Northrop Grumman отримала кілька замовлень на роботу ANDROS F6A від різних країн, включаючи США та Ізраїль.

ANDROS F6A є важливим інструментом для розмінування територій та пошуку вибухонебезпечних предметів. Його можливості та характеристики роблять його корисним для різних країн та організацій, які працюють над безпекою та стабільністю.

Рисунок 1.3 – Робот «ANDROS F6A»

Плюси:

- висока вантажопідйомність: може маніпулювати великими та важкими об'єктами.
- міцність: стійкий до пошкоджень, що робить його ефективним у складних умовах.
- далекобійність: можливість дистанційного управління на великі відстані.

Мінуси:

- розмір: досить великий, що може обмежувати його використання у вузьких просторах.
- складність управління: потребує спеціалізованого навчання для ефективного управління роботом.

Іншим же комплексом такого типу є «Dragon Runner» [4] (рис. 1.4), компактний робот, розроблений компанією QinetiQ для проведення розвідки та знешкодження вибухових пристроїв.

Dragon Runner - це серія військових роботів, розроблених компанією QinetiQ, міжнародною оборонною компанією з базою в США. Роботи випускаються в двох версіях: військовій та компактній цивільній.

Військова версія Dragon Runner була замовлена Міністерством оборони Великої Британії в 2010 році. Компанія QinetiQ отримала замовлення на близько 100 одиниць Dragon Runner на суму, що не була опублікована. Ці роботи були розроблені для розмінування та пошуку вибухонебезпечних предметів на полі бою.

Компактна цивільна версія Dragon Runner була розроблена для використання в різних цивільних ситуаціях, таких як пошук та рятування після стихійних лих. Вона має менші розміри та меншу вагу, ніж військова версія, що робить її більш придатною для використання в обмежених просторах.

Dragon Runner 10 та Dragon Runner 20 - це дві основні моделі, які були розроблені компанією QinetiQ. Вони мають наступні характеристики:

- Дистанційне керування: Роботи можуть бути керовані оператором з відстані, що дозволяє операторам залишатися в безпеці під час розмінування.
- Виявлення вибухонебезпечних предметів: Dragon Runner оснащений сенсорами, які можуть виявляти вибухонебезпечні предмети, такі як міни, гранати та інші боєприпаси.
- Знешкодження: Роботи можуть знешкоджувати виявлені вибухонебезпечні предмети, використовуючи різні методи, такі як демонтаж або дистанційне знищення.

- Мобільність: Dragon Runner може рухатися по різноманітних поверхнях, включаючи ґрунтові, асфальтові та бетонні.
- Збереження життя: Роботи дозволяють операторам залишатися в безпеці під час розмінування, що зменшує ризик втрат серед військовослужбовців та цивільних осіб.

Вартість одного робота Dragon Runner становить близько 1,5 мільйона доларів США. Компанія QinetiQ отримала кілька замовлень на роботу Dragon Runner від різних країн, включаючи Велику Британію та США.

Dragon Runner є важливим інструментом для розмінування та пошуку вибухонебезпечних предметів. Його можливості та характеристики роблять його корисним для різних країн та організацій, які працюють над безпекою та стабільністю.

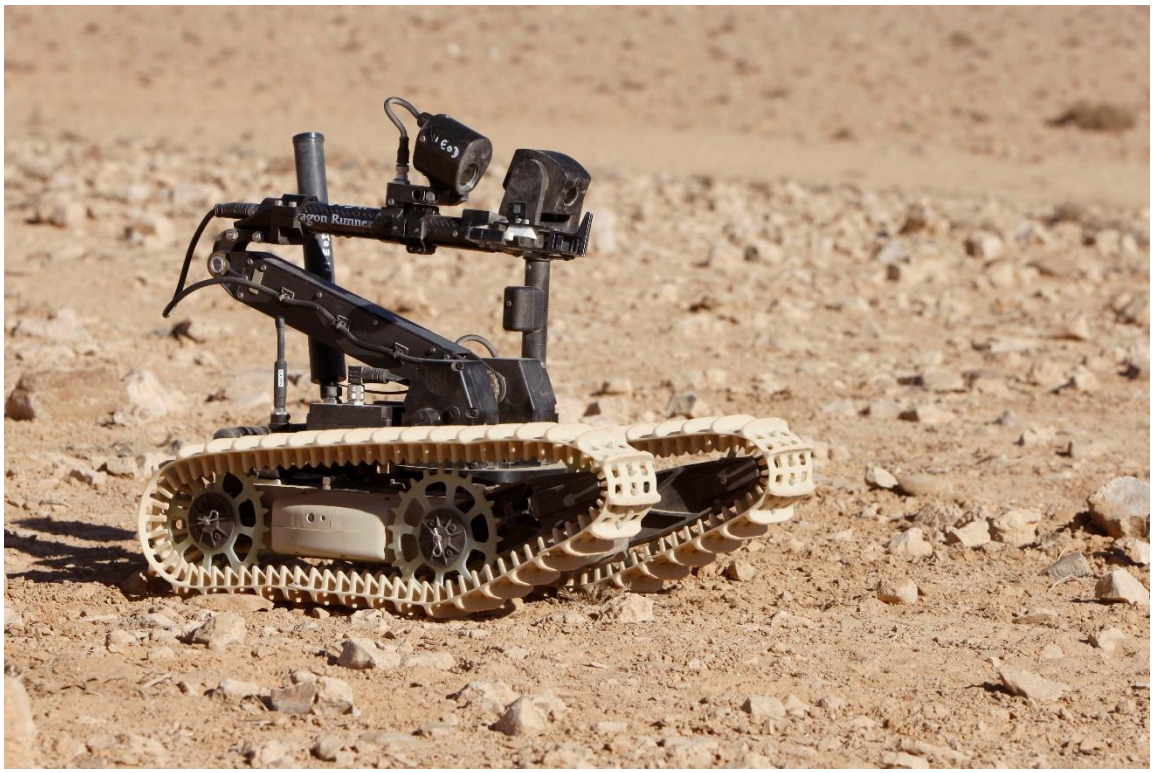

Рисунок 1.4 – Робот «Dragon Runner»

Плюси:

- маленький розмір: легко транспортується і може працювати у вузьких просторах.

- гнучкість: може виконувати різні завдання, від розвідки до знешкодження вибухових пристроїв.
- швидкість розгортання: швидко готовий до використання у будь-яких умовах.

Мінуси:

- обмежена функціональність: менші можливості порівняно з більшими роботами.
- вантажопідйомність: обмежена здатність піднімати важкі предмети.

1.3 Висновки до розділу 1

В ході аналізу предметної області роботизованих комплексів, а саме наземних роботизованих комплексів роботів-саперів було розглянуто особливості такого типу комплексів та наведено декілька прикладів таких комплексів.

Результати аналізу показують що такі комплекси є затребуваними, а можлива модульність робить їх більш багатофункціональними.

Проте основною завадою до їх масового використання є їх висока вартість та складність виробництва.

На підставі цього аналізу можна зробити висновок про те що потрібно здешевлювати та дещо спрощувати такі комплекси для того щоб вони могли стати масовими та доступними.

2 СИСТЕМНИЙ ДИЗАЙН РОБОТА

2.1 Проектування робота

Розробка та проектування робота є важливим етапом в розробці комплексу. Основною метою розробки та проектування є отримання принципової моделі а потім і конструкторської моделі робота.

Робот повинен бути простим по конструкції, простим у експлуатації та дешевим у виробництві. Для цього будуть використані поширені та відносно недорогі комплектуючі та припустимо спрощена конструкція.

Робот складатиметься з таких наступних вузлів:

- рама,
- 2х електричні мотор-колеса,
- 2х колеса,
- 2х контролера безколекторних двигунів,
- головний контролер,
- акумулятор.

Рама буде мати форму прямокутника 700х500мм та буде зварена з стального кута 50х3мм. Рама буде виконувати функції основи, на неї будуть навішуватися інші вузли та елементи конструкції.

Електричні мотор колеса будуть використані такого типу як на гіроскутерах[5] та іншій подібній техніці. Двигуни які в них встановлені будуть безколекторними електричними двигунами [6] на 36 вольт та 150 ват кожний.

Мотор-колесо [7] — вузол, який об’єднує в собі колесо і має вбудовані в нього тяговий електродвигун, силову передачу, а в деяких випадках і гальмівну систему (таким чином, кожне мотор-колесо має індивідуальний привід). Встановлюється, як правило, в кронштейн, підвішений до рами (якщо колесо некероване) або в підшипник, встановлений у шворні (якщо колесо і ведене, і кероване). Живиться від двигуна внутрішнього згоряння через електромеханічну передачу (в основному на автомобільній техніці, переважно важкої), від контактної мережі (на тролейбусах і

тролейбусах) або від акумулятора (на електромобілях і електровелосипедах, або як додаткове джерело енергії, на автомобільному двигуні внутрішнього згоряння, наприклад на гібридних автомобілях або тролейбусах). Існує два режими роботи мотор-колеса - тяговий і генераторний. У режимі тяги обертання передається від вала якоря електродвигуна до ведучого колеса; в генераторному режимі, який використовується для електричного гальмування, електродвигун перетворюється на генераторний режим роботи, а електроенергія перетворюється в тепло на гальмівному реостаті (реостатне гальмування) або повертається в електричну мережу або використовується для зарядки акумуляторів (регенеративне гальмування).

Вентильний електродвигун [8] – різновид синхронної машини, реалізованої в замкнутій системі за допомогою датчика положення ротора, системи керування (перетворювача координат) і силового напівпровідникового перетворювача. Часто їх також називають безконтактними (безколекторними) двигунами постійного струму або інвертованими машинами постійного струму.

Цей тип двигуна був створений для поліпшення властивостей двигунів постійного струму.

У вентильному двигуні (ВД) індуктор розташований на роторі (у вигляді постійних магнітів), обмотка якоря розташована на статорі. Напруга живлення обмоток двигуна формується в залежності від положення ротора. Якщо в двигунах постійного струму для цього використовувався колектор, то в вентильному двигуні його функцію виконує напівпровідниковий комутатор.

Основною відмінністю ВД від синхронного двигуна є його самосинхронізація за допомогою ДПР, внаслідок чого у ВД, навпаки, частота обертання поля пропорційна частоті обертання ротора, а частота обертання ротора залежить від напруги живлення.

Статор має традиційну конструкцію і схожий на статор асинхронного двигуна. Складається з корпусу, сердечника з електротехнічної сталі і мідної обмотки, прокладеної в пазах по периметру сердечника. Кількість обмоток визначає кількість фаз двигуна. Зазвичай це трифазні, рідше чотирифазні двигуни.

За способом укладання витків в обмотках статора розрізняють двигуни з трапецієподібною і синусоїдальною зворотною електрорушійною силою. За способом живлення фазний електричний струм у відповідних типах двигунів також змінюється трапецієподібно або синусоїдально.

Ротор виготовлений з використанням постійних магнітів і зазвичай має від двох до восьми пар полюсів з чергуванням північних і південних полюсів.

Спочатку для виготовлення ротора використовувалися феритові магніти. Вони поширені і дешеві, але мають недолік у вигляді низького рівня магнітної індукції. В даний час набирають популярність рідкоземельні магніти, оскільки вони дозволяють отримати високий рівень магнітної індукції і зменшити розмір ротора.

У двигунах великої потужності замість постійного магніту на роторі використовується електромагніт. Напруга живлення до нього подається через контактні кільця, встановлені на роторі.

Датчик положення ротора (ДПР) забезпечує зворотний зв'язок про положення ротора, виконує ту ж функцію, що і колектор в двигуні постійного струму. Його робота може бути заснована на різних принципах — фотоелектричному, індукційному, на ефекті Холла та ін. Найбільшої популярності набули датчики Холла і фотоелектричні, оскільки вони практично безінерційні і дозволяють позбутися затримки в каналі зворотного зв'язку за положенням ротор.

Фотоелектричний датчик у своїй класичній формі містить три фіксованих фоторецептори, які по черзі закриваються шторкою, що обертається синхронно з ротором. Двійковий код, отриманий від ДПР, фіксує шість різних положень ротора. Сигнали датчиків перетворюються керуючим пристроєм в комбінацію керуючих напруг, які керують силовими ключами, так що в кожному такті (фазі) роботи двигуна вмикаються два ключі і дві з трьох обмоток якоря підключаються до мережі. в серії. Обмотки якоря U, V, W розташовані на статорі зі зміщенням 120° і їх початок і кінці з'єднані так, що при перемиканні ключів створюється обертове магнітне поле.

Система керування містить силові ключі, часто тиристори або силові транзистори з ізольованим затвором. З них збирається інверсія напруги або інверсія струму. Система управління ключами зазвичай реалізується на основі використання

мікроконтролера. Наявність мікропроцесора вимагає виконання великої кількості обчислювальних операцій для керування двигуном.

Принцип дії ВД заснований на використанні датчика положення ротора (ДПР), перетворювача координат і силового напівпровідникового перетворювача. Вони спільно формують фазні напруги на обмотках статора машини таким чином, що результуючий вектор напруги завжди зміщений на кут 90° і нерухомий відносно осі магнітного поля ротора.

Комутація здійснюється так, щоб потік збудження ротора — Ф0 залишався постійним відносно потоку якоря. В результаті взаємодії потоку якоря і збудження створюється крутний момент М, який прагне повернути ротор так, щоб потоки якоря і збудження збігалися, але при повороті ротора під дією ДПР обмотки закручуються. перемикається, і потік якоря повертається до наступного кроку.

У цьому випадку результуючий вектор струму буде зміщений і нерухомий щодо потоку ротора, який створює момент на валу двигуна.

У режимі руху МРС статора передує МРС ротора на кут 90°, який спирається на ДПР. У режимі гальмування МРС статора відстає від МРС ротора, кут 90° також підтримується за допомогою ДПР.

Швидкість ВД регулюється величиною напруги, що подається на статор. Необхідно послідовно подавати напругу на обмотки статора.

Регулювання напруги, що подається на двигун, здійснюється за допомогою силових вимикачів. Крім того, силові перемикачі регулюють напругу, що подається на двигун, за рахунок чого змінюється частота обертання вала двигуна.

На відміну від щіткового двигуна постійного струму, комутація в ВД здійснюється та контролюється за допомогою електроніки, напр. електронний регулятор швидкості.

Загальні системи керування, що реалізують алгоритми широтно-імпульсного регулювання та широтно-імпульсної модуляції при управлінні ВД.

Система, що забезпечує найширший діапазон регулювання обертів — у двигунах з векторним керуванням. За допомогою частотного перетворювача регулюються оберти двигуна і підтримується муфта потоку в машині на заданому

рівні. Особливість регулювання електроприводу з векторним керуванням — контрольовані координати, зміряні в нерухомій системі координат перетворяться до системи, що обертається, з них виділяється постійне значення пропорційне складовим векторів контрольованих параметрів, по яких здійснюється формування керувальних дій, далі зворотний перехід.

Останнім часом цей тип двигуна стрімко набирає популярність, проникаючи в багато галузей. Застосовується в різних сферах використання: від побутової техніки до залізничного транспорту.

Двигуни постійного струму з електронною системою управління часто поєднують в собі найкращі якості безконтактних двигунів і двигунів постійного струму.

Переваги:

- висока швидкість і динаміка, точність позиціонування
- широкий діапазон зміни частоти обертання
- безконтактність і відсутність вузлів, які потребують обслуговування — безколекторна машина
- можливість використання у вибухонебезпечному та агресивному середовищі
- висока моментна перевантажувальна здатність
- висока енергоефективність (ККД більше 90% і cosφ більше 0,95)
- тривалий термін служби, висока надійність і підвищений термін служби за рахунок відсутності ковзних електричних контактів
- малий перегрів електродвигуна, при роботі в режимах з можливими перевантаженнями

Недоліки:

- відносно складна система керування двигуном
- висока вартість двигуна через використання в конструкції ротора дорогих постійних магнітів

Також є два простих колеса на роботі. Вони можуть витримувати значну вагу, наприклад від ваги самого робота та від ваги додаткових модулів.

З моторколесами на безколекторних двигунах працюють контролери безколекторних двигунів [9]. Їх завдання генерувати фази для двигуна, задавати його швидкість обертання та відслідковувати та утримувати її. Також вони використовуються для рекуперації та для зміни напрямку обертання колеса.

Контролер безколекторного двигуна - це електронний пристрій, який використовується для керування роботою електричного двигуна або двигунів у електричних транспортних засобах, таких як електробайки, електроскутери, електричні велосипеди і т.д. Основною функцією контролера є керування потужністю, що постачається до двигуна, для забезпечення потрібної швидкості та динаміки руху.

Контролери безколекторних двигунів широко використовуються у різних галузях, включаючи автомобільну промисловість, промислові роботи, моделювання та робототехніку. Вони є важливим компонентом у сучасних електричних системах, де потрібне точне та ефективне керування рухом безколекторних двигунів.

Контролери для безколекторних двигунів доступні в різних магазинах та інтернет-магазинах, які спеціалізуються на електроніці та компонентах для робототехніки. Широкий вибір моделей та виробників дозволяє знайти оптимальний контролер для конкретних потреб та проектів.

Основні характеристики універсального контролера:

- напруга: контролер підтримує діапазон напруги від 36V до 48V. Це означає, що його можна використовувати з різними типами літій-іонних або свинцево-кислотних батарей, які зазвичай працюють в цьому діапазоні напруги.
- струм: максимальний допустимий струм в контролері становить 17А. Це означає, що контролер може забезпечити потужність до 17 ампер для електричного двигуна.
- потужність: максимальна потужність контролера становить 350 Вт. Ця величина обмежує максимальну вихідну потужність двигуна, яка може бути досягнута за умов використання контролера.

- універсальність: контролер є універсальним, що означає, що його можна використовувати з різними типами електротранспорту, які відповідають підтримуваному діапазону напруги і струму.
- функціональність: в залежності від моделі контролера, він може мати додаткові функції, такі як регулювання швидкості, режими їзди, системи безпеки та інші.

Роботу усієї системи забезпечує головний мікроконтролер. Він складається з комутаційної частини де з’єднані всі необхідні дроти, конструкції прийому команд яка складається з bluetooth модуля та з модуля головного керування який являє собою мікроконтролер Ардуіно [10].

Модуль Bluetooth HC-06 [11] – це недорогий та популярний модуль Bluetooth, який забезпечує бездротове з'єднання між пристроями. HC-06 є частиною сімейства модулів Bluetooth.

Основні характеристики модуля HC-06:

- тип Bluetooth: HC-06 підтримує стандарт Bluetooth 2.0, що робить його дещо застарілим, тому що на даний момент існують новіші версії стандарту Bluetooth.
- режими роботи: Модуль HC-06 може працювати в режимі Slave (допоміжний пристрій), що робить його ідеальним для підключення до головного пристрою, такого як смартфон, комп'ютер або мікроконтролер.
- підтримка інтерфейсів: HC-06 зазвичай підтримує інтерфейс UART (Universal Asynchronous Receiver/Transmitter), що робить його легким у використанні з різними мікроконтролерами та іншими пристроями, які мають UART.
- швидкість передачі: Модуль HC-06 зазвичай має обмежені швидкості передачі даних у порівнянні з новими версіями Bluetooth. Однак для багатьох програм це достатньо.
- конфігурація: HC-06 можна конфігурувати за допомогою команд AT (Attention), що дозволяє налаштувати різні параметри модуля, такі як ім'я пристрою, пароль та швидкість передачі даних.

- зазвичай модуль працює від напруги 3.3 В, що робить його сумісним з більшістю мікроконтролерів та інших пристроїв.
- з кнопкою "Reseach" (ON/OFF/WAKE, зовнішній вихід MCU "High level" може керувати модулем для повторного пошуку)

Для використання модуля потрібно виконати наступні кроки:

Підготовка модуля

- Встановлення батарей чи живлення: Модуль HC-06 працює від батарей, тому потрібно встановити батареї, які підходять до модуля.
- Встановлення програмного забезпечення: Модуль HC-06 працює з програмним забезпеченням, яке потрібно встановити на пристрій, до якого підключається модуль.

Підключення модуля

- Підключення до Arduino: Модуль HC-06 підключається до Arduino через інтерфейс RS-232.
- Конфігурація порту: Порт, через який підключається модуль, повинен бути конфігований для роботи з Bluetooth.
- Встановлення драйвера: Драйвер для модуля HC-06 повинен бути встановлений на пристрій, до якого підключається модуль.

Підключення до пристрою

- Виявлення модуля: Модуль HC-06 повинен бути виявлений пристроєм, до якого підключається модуль.
- Підключення до пристрою: Після виявлення модуля, потрібно підключити його до пристрою, до якого підключається модуль.
- Встановлення пароля: Пароль, який потрібно встановити для підключення модуля до пристрою, повинен бути встановлений на модулі.

Підключення до програми

- Встановлення програми: Програма, яка буде використовуватися для підключення модуля до пристрою, повинна бути встановлена на пристрій.
- Конфігурація програми: Програма повинна бути конфігована для роботи з модулем HC-06.

- Виконання операцій: Після встановлення та конфігурації програми, модуль HC-06 може бути використаний для виконання операцій, таких як передача даних між пристроями.

Arduino Uno [12] – це пристрій на основі мікроконтролера ATmega328. У нього входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (6 з них можна використовувати як ШІМ-виходи), 6 аналогових входів, кристалічний резонатор 16 МГц, роз'єм USB, роз'єм живлення, In- гніздо програмування схеми (ICSP) і кнопка скидання. Щоб почати роботу з пристроєм, достатньо просто подати живлення від адаптера змінного/постійного струму або акумулятора, або підключити його до комп'ютера за допомогою кабелю USB.

Особливості:

- Мікроконтролер ATmega328
- Робоча напруга 5В
- Напруга живлення (рекомендована) 7-12В
- Напруга живлення (гранична) 6-20В
- 14 цифрових входів/виходів (6 з них можна використовувати як ШІМ-виходи)
- Аналогові входи 6
- Максимальний струм одного виходу 40 мА
- Максимальний вихідний струм становить 3,3 В 50 мА
- Флеш-пам'ять 32 КБ (ATmega328), з яких 0,5 КБ використовуються завантажувачем
- SRAM 2 Кб (ATmega328)
- EEPROM 1 КБ (ATmega328)
- Тактова частота 16 МГц

Arduino Uno можна вбудувати від USB або від зовнішнього джерела живлення - тип джерела вибирається автоматично.

Адаптер змінного/постійного струму або батарею/акумулятор можна використовувати як зовнішнє джерело живлення (не USB). Штекер адаптера (діаметр - 2,1 мм, центральний контакт - плюс) необхідно вставити у відповідний роз'єм

живлення на платі. У разі живлення від батареї, її дроти повинні бути підключені до клем Gnd і Vin роз'єму POWER.

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

Нижче наведено висновки живлення, розташовані на платі:

- VIN. Напруга, що подається на Arduino безпосередньо від зовнішнього джерела живлення (не підключеного до 5 В від USB або іншої стабілізованої напруги). Зовнішнє живлення може подаватися через цей штифт, а також струм може використовуватися, коли пристрій запитується від зовнішнього адаптера.
- 5В. На вихід надходить напруга 5В від стабілізатора напруги на платі, незалежно від того, як живиться пристрій: від адаптера (7 - 12В), від USB (5В) або через вихід VIN (7 - 12В). Живлення пристрою через вихід 5В або 3В не рекомендується, оскільки в цьому випадку не використовується стабілізатор напруги, що може призвести до виходу з ладу.
- 3,3В надходить від стабілізатора напруги на платі. Максимальний струм, що споживається з цього висновку, становить 50 мА.
- GND. Вилучення землі.
- IOREF. Цей контакт надає платам розширення інформацію про робочу напругу мікроконтролера Arduino. Залежно від напруги, яка зчитується з контакту IOREF, плата розширення може перемикатися на відповідне джерело живлення або залучати перетворювачі рівня, що дозволяє їй працювати як з пристроями 5 В, так і з 3,3 В.

Використовуючи функції pinMode(), digitalWrite() і digitalRead(), кожен із 14 цифрових контактів може діяти як вхід або вихід. Рівень напруги на виходах обмежений 5В. Максимальний струм, який може видавати або споживати один

висновок, становить 40 мА. Всі висновки підключаються до внутрішніх резисторів (за замовчуванням відключені) номіналом 20-50 кОм. Крім того, деякі виводи Arduino можуть виконувати додаткові функції:

- Послідовний інтерфейс: контакти 0 (RX) і 1 (TX). Вони використовуються для отримання (RX) і передачі (TX) даних через послідовний інтерфейс. Ці контакти з'єднані з відповідними контактами чіпа ATmega8U2, який діє як перетворювач USB-UART.
- Зовнішні переривання: контакти 2 і 3. Можуть бути джерелами переривань, які виникають на фронті, падінні або низькому рівні сигналу на цих контактах. Для отримання додаткової інформації.
- ШІМ: контакти 3, 5, 6, 9, 10 і 11. 8-бітні аналогові значення можна виводити як ШІМ-сигнал за допомогою функції analogWrite().
- Інтерфейс SPI: контакти 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Використовуючи бібліотеку SPI, ці контакти можуть обмінюватися даними з інтерфейсом SPI.
- Світлодіод: 13. Вбудований світлодіод, підключений до контакту 13. При надсиланні ВИСОКОГО значення світлодіод вмикається, при надсиланні НИЗЬКОГО значення він вимикається.

Arduino Uno має 6 аналогових входів (A0 - A5), кожен з яких може представляти аналогову напругу як 10-бітове число (1024 різних значення). За замовчуванням вимірювання напруги виконується в діапазоні від 0 до 5 В. Однак верхню межу цього діапазону можна змінити за допомогою виводу AREF і функції analogReference(). Крім того, деякі з аналогових входів мають додаткові функції:

- TWI: контакт A4 або SDA і контакт A5 або SCL. Використовуючи бібліотеку Wire, ці контакти можуть обмінюватися даними через інтерфейс TWI (послідовна шина даних для зв'язку з інтегральною схемою, проста внутрішня шина зв'язку для створення керуючої електроніки. Використовується для підключення низькошвидкісних периферійних пристроїв до материнської плати, вбудованих систем і мобільних телефонів.) .

- AREF. Опорна напруга для аналогових входів. Може використовуватися функцією analogReference().

Датчик Холла - це пристрій, який використовується для вимірювання частоти обертання валів, коліс та інших обертових деталей. Він працює на основі ефекту Холла, коли електрони в провіднику відхиляються в магнітному полі, створюючи різницю потенціалів.

Ефект Холла - це явище, при якому виникає поперечна різниця потенціалів, коли провідник постійного струму поміщається в магнітне поле. Це явище було відкрито Едвіном Холом у 1879 році на тонких золотих пластинах.

Ефект Холла виникає тому, що магнітне поле змінює рух електронів у провіднику, що призводить до зміни електричного поля. Це явище використовується для вимірювання магнітних полів і для вивчення поведінки електронів у магнітних полях.

Принцип роботи датчика Холла для вимірювання частоти обертання:

- Датчик розміщується поруч з обертовим валом, колесом або іншою деталлю, на якій закріплені магнітні мітки.
- При обертанні деталі магнітні мітки проходять поруч з датчиком Холла, створюючи змінне магнітне поле.
- Датчик Холла реагує на зміну магнітного поля, генеруючи імпульси напруги.
- Частота проходження імпульсів відповідає частоті обертання деталі. Підрахувавши кількість імпульсів за одиницю часу, можна визначити частоту обертання в оборотах за хвилину (об/хв).

Датчики Холла для вимірювання частоти обертання бувають аналоговими та цифровими. Аналогові видають напругу, пропорційну магнітному полю, а цифрові - імпульси при перетині порогового значення магнітного поля.

Такі датчики широко використовуються в автомобілях для вимірювання частоти обертання колінчастого та розподільчого валів двигуна, частоти обертання коліс при ABS, в промисловому обладнанні для контролю швидкості обертання валів та інших застосуваннях, де потрібно виміряти частоту обертання.

2.2 Реалізація робота

Як уже відомо робот складатиметься з кількох вузлів які пов’язані між собою та змонтованих на рамі.

До кожного моторколеса буде під’єднано по одному контролеру безколекторного двигуна. В свою чергу обидва контролера будуть під’єднані до головного контролера. Всім буде курувати Ардуіно, яка буде в свою чергу приймати команди від оператора робота через bluetooth. Схват являється одним з можливих навісних модулів.

Нижче наведена схема електричної проводки робота (рис. 2.1).

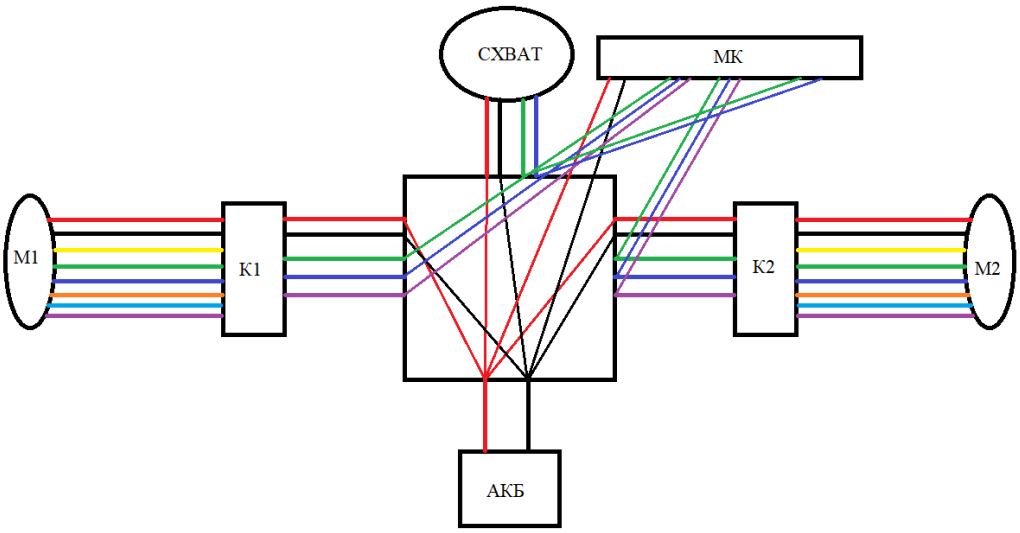

Рисунок 2.1 – Схема електричної проводки робота

На рисунку 2.1 елементи мають наступні позначення:

- М1 та М2 це моторколеса з безколекторними двигунами,
- К1 та К2 це контролери безколекторних двигунів,
- АКБ це акумуляторна батарея,
- МК це мікроконтролер з модулем bluetooth,
- СХВАТ це один з варіантів можливого навісного модуля.

На кожен контролер подається плюс живлення, земля, та керуючі сигнали через три проводи: швидкість, напрямок та гальма.

На кожне моторколесо з контролера подається 8 проводів: 3 фази та 5 проводів для датчиків Холла: 2 живлення і 3 від датчиків.

Хоча навіть при відсутності датчиків Холла контролер може визначати приблизну частоту обертання мотора використовуючи сплески на фазних провідниках.

До Ардуіно під’єднаний модуль bluetooth який забезпечує прийом команд.

2.3 Висновки до розділу 2

У цьому розділі були детально розглянуті ключові аспекти проектування робота та основи його конструкції.

Розглянуто ключові елементи та компоненти які використовуються в конструкції робота такі як мотор-колеса з безколекторними двигунами, контролери безколекторних двигунів, мікроконтролер Arduino Uno, модуль Bluetooth.

Розглянуто основні принципи керування мотор-колесами, розглянуто які сигнали відповідають за керування ними. Також розглянули принцип роботи контролера, розглянули його ключові характеристики та особливості застосування.

Було описано Arduino Uno та модуль Bluetooth, було розглянуто та описано їх ключові характеристики, можливості та принцип застосування. Розглянуто їх місце в керуванні комплексом.

Створено та розглянуто електричну схему та структурну модель робота.

3 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ

3.1 Розробка програмного модуля робота

Програмне забезпечення яке буде знаходитись в мікроконтролерній платформі Ардуіно безпосередньо на самому роботі є дуже важливим, адже воно буде забезпечувати прийом команд, їх обробку та видачу відповідних команд на контролери двигунів та на навісні додаткові модулі.

Програма буде написана на мові Wiring, яка являє собою незначну модифікацію мови C++ [13][22] для Ардуіно.

Як мікроконтролерну платформу з великого сімейства Ардуіно було обрано Ардуіно Уно як доволі універсальну, дешеву та просту.

Ардуіно — апаратна обчислювальна платформа для аматорського будівництва, основними компонентами якої є плата мікроконтролера з елементами вводу-виводу та середовище розробки Processing/Wiring на мові програмування, що є спрощеною підмножиною C/C++. Arduino можна використовувати як для створення автономних інтерактивних об'єктів, так і для підключення до програмного забезпечення, яке працює на комп'ютері. Інформація про плату (зображення друкованої плати, характеристики елементів, програмне забезпечення) знаходиться у відкритому доступі і може бути використана тими, хто вважає за краще створювати плати своїми руками.

Інтегроване середовище розробки Ардуіно — це багатоплатформна програма Java, яка містить редактор коду, компілятор і модуль для передачі мікропрограми на плату. Середовище розробки базується на мові програмування Processing і призначене для програмування початківцями, які не знайомі з розробкою програмного забезпечення. Мова програмування схожа на мову Wiring. Загалом, це C++, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою AVR-GCC. Програми Ардуіно пишуться на мові програмування C або C++. Середовище розробки Ардуіно поставляється разом із бібліотекою програм «Wiring».

Програма повинна забезпечувати отримання команд отриманих bluetooth модулем по радіоканалу, повинна забезпечувати інтерпретацію цих команд, обробку даних та видачу команд на сигнальні виводи.

Розроблено діаграму класів програмного модуля робота (рис. 3.1).

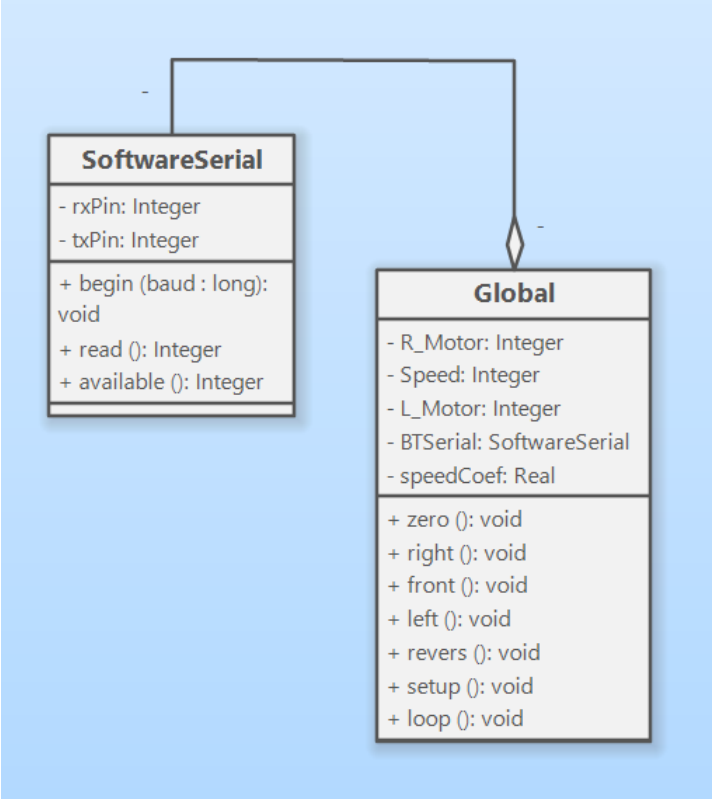

Рисунок 3.1 – Діаграма класів програмного модуля робота

Для початку ми підключаємо необхідні бібліотеки, в нашому випадку бібліотеку для роботи з серіал портом :

```
#include <SoftwareSerial.h>
```

Також для серіал порта необхідно оголосити налаштування, щоб бібліотека знала на яких виводах мікроконтролера емулювати серіал порт :

```
SoftwareSerial BTSerial(10, 11); // RX, TX
BTSerial.begin(9600);
```

Нам також потрібно оголосити деякі константи та задати початкові значення деяких змінних :

```
#define R_Motor 2
#define Speed   3
```

```
#define L_Motor 4
float speedCoef = 0.3;
```

У функції setup(){} потрібно оголосити налаштування портів для правильної роботи з ними :

```
void setup() {
  pinMode(R_Motor, OUTPUT);
  pinMode(Speed, OUTPUT);
  pinMode(L_Motor, OUTPUT);
}
```

У функції loop(){} буде знаходитись основна логіка роботи програми, основна логіка обробки команд що приходять від оператора та власне прийом цих команд.

```
void loop() {
  if (BTSerial.available()) {
    char c = BTSerial.read();
    Serial.println(c);
    switch(c){
      case '0':
        zero();
        break;
      case 'R':
        right();
        break;
      case 'F':
        front();
        break;
      case 'L':
        left();
        break;
      case 'B':
        revers();
```

```
        break;
      case '1':
        speedCoef = 0.3;
        break;
      case '2':
        speedCoef = 0.5;
        break;
      case '3':
        speedCoef = 1;
        break;
    }
  }
}
```

Як видно зчитування команд відбувається як тільки вони наявні, а вибір реакції на команду у вигляді запуску функції команди виконується завдяки конструкції типу switch(){}. Тут же відбувається і зміна модифікатора швидкості їзди робота.

Далі буде розглянуто функції які відповідають за керування роботом, функції в які інтерпретуються команди від оператора.

Функція zero(){} відповідає за зупинку робота, в ній відбувається виставка лівого і правого мотора на хід вперед та виставка швидкості нуль.

```
void zero(){
  digitalWrite(R_Motor, HIGH);
  digitalWrite(Speed, LOW);
  digitalWrite(L_Motor, HIGH);
}
```

Далі розглянемо функцію front(){}, яка відповідає рух. Через специфічний алгоритм видачі команд оператором ця функція складається лише з одного рядка, просто іде видача ШІМ сигналу регулювання швидкості на контролери. Перед нею для руху саме вперед обов’язково має бути команда на виконання функції zero(){}.

```
void front() {
```

```
  analogWrite(Speed, 255 * speedCoef);
}
```

Функція revers(){} відповідає за задній хід, точніше за переключення робота в режим заднього ходу. Сам же рух відбувається з використанням функції front(){}. Затримка необхідна для того щоб контролери встигли переключитися в потрібний режим.

```
void revers(){
  digitalWrite(R_Motor, LOW);
  digitalWrite(L_Motor, LOW);
  delay(500);
}
```

Далі розглянемо функцію right(){}, яка відповідає за поворот вправо. Поворот відбувається шляхом реверсу правого мотора.

```
void right() {
  digitalWrite(R_Motor, LOW);
  delay(500);
  analogWrite(Speed, 255 * speedCoef);
}
```

Функція left(){}відповідає за поворот вліво. Поворот відбувається шляхом реверсу лівого мотора.

```
void left() {
  digitalWrite(L_Motor, LOW);
  delay(500);
  analogWrite(Speed, 255 * speedCoef);
}
```

Програма на мікроконтролері виконується постійно та циклічно. Затримки типу delay(500) необхідні для того щоб контролери встигли перелаштуватися на новий режим роботи. Без цих затримок контролери будуть працювати неправильно, а відповідно й мотори теж.

3.2 Розробка програмного забезпечення пристрою дистанційного керування

Як пристрій оператора для дистанційного керування роботом буде використовуватися комп’ютер з ОС Windows 10 [14] або Ubuntu. Головна вимога наявність інтерпретатора Python [15]. Програма написана на мові Python з використанням бібліотеки tkinter [16]. Ця мова програмування була обрана через свою простоту та ефективність.

Windows 10 [24] - це операційна система, розроблена компанією Microsoft. Вона була випущена в 2015 році як наступник операційної системи Windows 8.1. Windows 10 є частиною сімейства операційних систем Windows, які були розроблені для роботи на персональних комп'ютерах та інших пристроях.

Windows 10 має багато корисних функцій, які дозволяють користувачам працювати з операційною системою з більшою продуктивністю та зручністю. Наприклад, вона має новий інтерфейс користувача, який називається "Start menu", який дозволяє користувачам швидше знайти та запускати програми. Вона також має новий браузер, який називається "Microsoft Edge", який дозволяє користувачам переглядати веб-сторінки з більшою швидкістю та зручністю.

Windows 10 також має багато корисних інструментів для розробників, таких як Visual Studio Code, який дозволяє розробникам писати та відлагоджувати код на різних мовах програмування, включаючи Python. Вона також має інструменти для автоматизації різних задач, таких як робота з файлами та мережею.

Як середовище розробки було обрано VS Code [17] через свою багатофункціональність.

Visual Studio Code (VS Code) - це безкоштовний та відкритий редактор коду, розроблений Microsoft. Він підтримує широкий спектр мов програмування та має велику екосистему розширень, що робить його одним з найпопулярніших редакторів коду на ринку. У цьому тексті детально розглянуто можливості та особливості VS Code.

Інтерфейс VS Code спроектований для максимальної ефективності та зручності. Він складається з кількох основних елементів:

- Бічна панель: Містить вкладки для перегляду файлів, пошуку, налагодження та розширень.
- Редактор коду: Основна область для написання та редагування коду.
- Рядок стану: Відображає інформацію про поточний файл та статус компіляції.
- Меню та панель інструментів: Надають доступ до основних функцій та налаштувань.

Навігація в VS Code здійснюється за допомогою комбінацій клавіш, панелі команд та контекстних меню. Це дозволяє швидко переміщуватися між файлами, методами та класами.

VS Code підтримує широкий спектр мов програмування, включаючи:

- JavaScript, TypeScript, Node.js
- Python, Java, C++, C#
- PHP, Ruby, Go, Rust
- HTML, CSS, SCSS, Less
- XML, JSON, YAML
- SQL, Markdown, LaTeX

Для кожної мови VS Code надає базову підтримку, таку як підсвічування синтаксису, автодоповнення коду та навігація. Більш просунута підтримка забезпечується за допомогою розширень.

Одна з найсильніших сторін VS Code - це його розширювана екосистема. Розширення дозволяють додавати нові можливості, інтегрувати зовнішні інструменти та налаштовувати редактор під конкретні потреби.

Деякі популярні розширення включають:

- ESLint, Prettier - для форматування та перевірки коду
- GitLens, Git Graph - для роботи з Git
- Live Share - для спільної роботи над кодом в реальному часі
- Docker - для роботи з контейнерами

- Azure Tools - для роботи з хмарними сервісами Microsoft

Налаштування VS Code здійснюється за допомогою JSON-файлів. Це дозволяє зберігати налаштування в системі контролю версій та переносити їх між різними машинами.

VS Code тісно інтегрується з популярними системами контролю версій, такими як Git. Вбудовані можливості Git включають:

- Перегляд змін у файлах
- Коміти, пуші та пули
- Вирішення конфліктів
- Перегляд історії змін
- Розширення, такі як GitLens, додають ще більше можливостей, включаючи анотації коміту, перегляд гілок та віддалених репозиторіїв.

VS Code включає вбудовані можливості для налагодження коду. Для кожної мови доступні спеціальні налагоджувачі, які дозволяють:

- Встановлювати точки зупинки
- Покрокове виконання коду
- Перегляд змінних та стеку викликів
- Консоль для виконання коду

Запуск додатків також інтегрований в VS Code. Можна налаштувати різні конфігурації запуску для різних середовищ та платформ.

VS Code тісно інтегрується з хмарними сервісами, такими як Azure, AWS та Google Cloud. Розширення дозволяють:

- Керувати ресурсами хмари
- Розгортати додатки безпосередньо з редактора
- Налагоджувати додатки, що працюють в хмарі
- Інтегрувати сервіси штучного інтелекту

Це робить VS Code ідеальним інструментом для розробки хмарних додатків.

VS Code спроектований для максимальної продуктивності. Він включає багато можливостей для підвищення ефективності розробки:

- Швидкий пошук файлів та символів

- Багатокурсорна правка
- Вбудована підтримка рефакторингу
- Інтелектуальне автодоповнення коду
- Вбудована термінальна консоль
- Комбінації клавіш та панель команд дозволяють швидко виконувати часті дії без відриву від клавіатури.
- Крос-платформеність та відкритість

Код VS Code є відкритим, що дозволяє спільноті вносити свій внесок у розвиток редактора. Це призводить до швидкого виправлення помилок та додавання нових можливостей.

Python [23] є інтерпретованою об’єктно-орієнтованою мовою програмування високого рівня зі строгою динамічною типізацією. Високорівневі структури даних разом із динамічною семантикою та динамічним зв’язуванням роблять його привабливим для швидкої розробки додатків і як засіб об’єднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python і стандартні бібліотеки доступні як у скомпільованій формі, так і у вихідному вигляді на всіх основних платформах. Мова програмування Python підтримує декілька парадигм програмування, зокрема: об’єктно-орієнтоване, процедурне, аспектно-орієнтоване та функціональне.

До його основних переваг можна віднести наступне:

- чистий синтаксис (відступи слід використовувати для виділення блоків);
- переносимість програм (що характерно для більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включаючи модуль для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисно для експериментів і вирішення простих задач);
- стандартний дистрибутив має просте, але в той же час досить потужне середовище розробки, яке називається IDLE і яке написано на мові Python;

- зручний для розв'язування математичних задач (має інструменти для роботи з комплексними числами, може оперувати цілими числами будь-якого розміру, може використовуватися як потужний калькулятор у діалоговому режимі);
- відкритий код (можливість його редагування іншими користувачами).

Python має ефективні високорівневі структури даних і простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів і той факт, що це інтерпретована мова, роблять його ідеальним для написання сценаріїв і швидкої розробки додатків у багатьох галузях промисловості на більшості платформ.

Інтерпретатор мови Python і багата стандартна бібліотека (як вихідні тексти, так і двійкові дистрибутиви для всіх основних операційних систем) вільно поширюються. Python підтримує динамічну типізацію, тобто тип змінної визначається лише під час виконання. З основних типів слід відзначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, закодованими в Unicode [18].

Дизайн мови Python побудований навколо моделі об’єктно-орієнтованого програмування. Реалізація ООП [19] у Python є елегантною, потужною та добре продуманою, але в той же час досить специфічною порівняно з іншими об’єктно-орієнтованими мовами.

Особливості та особливості:

- Класи одночасно є об'єктами з усіма наведеними нижче можливостями.
- Спадкування, в тому числі множинне.
- Поліморфізм (всі функції віртуальні).
- Інкапсуляція (два рівні — відкритий і прихований методи і поля). Особливість - приховані учасники доступні для використання та позначаються як приховані лише спеціальними іменами.
- Спеціальні методи контролю життєвого циклу об'єкта: конструктори, деструктори, розподільники пам'яті.

- Перевантаження операторів (усі, крім is, '.', '=' і символічних логічних значень).
- Властивості (моделювання поля за допомогою функцій).
- Управління доступом до полів (емуляція полів і методів, частковий доступ і т.д.).
- Методи керування найпоширенішими операціями (справжнє значення, len(), глибоке копіювання, серіалізація, ітерація по об’єкту, …)
- Метапрограмування (керування створенням класів, тригери створення класів тощо)
- Повна інтроспекція.
- Класові та статичні методи, поля класів.
- Класи, вкладені в функції та інші класи.

Python підтримує парадигму функціонального програмування, зокрема:

- функція є об'єктом;
- функції вищих порядків;
- рекурсія;
- розроблена обробка списків (спискові вирази, операції над послідовностями, ітератори);
- аналог затворів;
- часткове застосування функції;
- можливість реалізації інших засобів у самій мові.

Існує кілька спеціалізованих середовищ розробки для Python, включаючи IDLE, Thonny, PyCharm [20], PyScripter, Spyder (останній призначений для наукової розробки), а також розширення для Visual Studio Code і плагін PyDev для Eclipse.

Пакет tkinter («інтерфейс Tk») є стандартним інтерфейсом Python до інструментарію GUI Tcl/Tk. І Tk, і tkinter доступні на більшості платформ Unix, включаючи macOS, а також у системах Windows.

Запуск python -m tkinter із командного рядка має викликати вікно, яке показує простий інтерфейс Tk, повідомляючи вам, що tkinter правильно встановлено у вашій

системі, а також показує, яку версію Tcl/Tk встановлено, щоб ви могли читати Tcl /Tk документація для цієї версії.

Підтримка Tkinter розділена на кілька модулів. Для більшості програм знадобиться основний модуль tkinter, а також модуль tkinter.ttk, який надає сучасний тематичний набір віджетів і API:

з імпорту tkinter *

з tkinter імпортувати ttk

class tkinter.Tk(screenName=None, baseName=None, className='Tk', useTk=True, sync=False, use=None)

Створіть віджет Tk верхнього рівня, який зазвичай є головним вікном програми, і ініціалізуйте інтерпретатор Tcl для цього віджета. Кожен екземпляр має власний асоційований інтерпретатор Tcl.

Клас Tk зазвичай створюється з використанням усіх значень за замовчуванням. Однак наразі розпізнаються такі аргументи ключових слів:

screenName

Якщо задано (у вигляді рядка), установлює змінну середовища DISPLAY. (лише X11)

назва бази

Ім'я файлу профілю. За замовчуванням baseName походить від імені програми (sys.argv[0]).

назва класу

Ім'я класу віджетів. Використовується як файл профілю, а також як ім’я, з яким викликається Tcl (argv0 в interp).

useTk

Якщо True, ініціалізувати підсистему Tk. Функція tkinter.Tcl() встановлює значення False.

синхронізація

Якщо True, виконувати всі команди X-сервера синхронно, щоб негайно повідомляти про помилки. Можна використовувати для налагодження. (лише X11)

використовуючи

Визначає ідентифікатор вікна, у яке потрібно вставити програму, замість того, щоб створювати його як незалежне вікно верхнього рівня. Ідентифікатор має бути вказаний так само, як значення параметра -use для віджетів верхнього рівня (тобто він має форму, подібну до тієї, яку повертає winfo_id()).

Зверніть увагу, що на деяких платформах це працюватиме належним чином, лише якщо ідентифікатор посилається на фрейм Tk або верхній рівень, для якого ввімкнено параметр -container.

Дякую

Об’єкт програми Tk, створений шляхом створення екземпляра Tk. Це забезпечує доступ до інтерпретатора Tcl. Кожен віджет, до якого приєднано той самий екземпляр Tk, має однакове значення атрибута tk.

майстер

Об’єкт віджета, який містить цей віджет. Для Tk master — це None, оскільки це головне вікно. Терміни master і parent схожі й іноді використовуються як синоніми для імен аргументів; однак виклик winfo_parent() повертає рядок назви віджета, тоді як master повертає об’єкт. parent/child представляє деревоподібний зв’язок, тоді як master/slave представляє структуру контейнера.

дітей

Прямі нащадки цього віджета як dict з іменами дочірніх віджетів як ключами та дочірніми об’єктами екземплярів як значеннями.

tkinter.Tcl(screenName=None, baseName=None, className='Tk', useTk=False)

Функція Tcl() є фабричною функцією, яка створює об’єкт, подібний до створеного класом Tk, за винятком того, що вона не ініціалізує підсистему Tk. Це найбільш корисно під час запуску інтерпретатора Tcl у середовищі, де вам не потрібно створювати вікна третьої сторони верхнього рівня, або де це неможливо (наприклад, системи Unix/Linux без X-сервера). Для об’єкта, створеного об’єктом Tcl(), ви можете створити вікно верхнього рівня (та ініціалізувати підсистему Tk), викликавши його метод loadtk().

Модулі, які забезпечують підтримку Tk, включають:

tkinter

Основний модуль Tkinter.

tkinter.colorchooser

Діалогове вікно, де користувач може вибрати колір.

tkinter.commondialog

Базовий клас для діалогових вікон, визначених в інших модулях, перерахованих тут.

tkinter.filedialog

Загальні діалоги, які дозволяють користувачеві вказати файл для відкриття або збереження.

tkinter.font

Утиліти для допомоги в роботі зі шрифтами.

tkinter.messagebox

Доступ до стандартних діалогових вікон Tk.

tkinter.scrolledtext

Текстовий віджет із вбудованою вертикальною смугою прокручування.

tkinter.simpledialog

Основні діалоги та зручні функції.

tkinter.ttk

Тематичний набір віджетів, представлений у Tk 8.5, надає сучасні альтернативи багатьом класичним віджетам у головному модулі tkinter.

Додаткові модулі:

_tkinter

Бінарний модуль, який містить низькорівневий інтерфейс до Tcl/Tk. Він автоматично імпортується головним модулем tkinter і ніколи не повинен використовуватися безпосередньо розробниками програм. Зазвичай це спільна бібліотека (або DLL), але в деяких випадках може бути статично пов’язана з інтерпретатором Python.

idlelib

Інтегроване середовище розробки та навчання Python (IDLE). На основі tkinter.

tkinter.constants

Символьні константи, які можна використовувати замість рядків під час передачі різних параметрів викликам Tkinter. Автоматично імпортується головним модулем tkinter.

tkinter.dnd

(експериментальний) Підтримка перетягування для tkinter. Це стане застарілим, коли його замінять на Tk DND.

tkinter.tix

(застаріло) Старіший сторонній пакет Tcl/Tk, який додає кілька нових віджетів. Кращі альтернативи для більшості можна знайти на tkinter.ttk.

черепаха

Графіка Turtle у вікні Tk.

Важливою частиною роботи програми є взаємодія з серіал портом. Для цього ми використали import serial. Це дуже корисна бібліотека. Ось деякі її необхідні функції:

write(data):

- Параметри: дані – дані для надсилання.
- Повертає: кількість записаних байтів.
- Тип повернення: int
- Викликає: SerialTimeoutException – якщо для порту налаштовано тайм-аут запису та час перевищено.
- Записує дані байтів у порт. Це має бути тип bytes (або сумісний, наприклад bytearray або memoryview). Рядки Unicode мають бути закодовані (наприклад, 'hello'.encode('utf-8').

read_until(expected=LF, size=None)

- Параметри:

  очікуваний – рядок байтів для пошуку.

  розмір – кількість байтів для читання.
- Повернення: зчитані байти з порту.
- Тип повернення: байти

- Читає, доки не буде знайдено очікувану послідовність («\n» за замовчуванням), розмір перевищено або поки не настане тайм-аут. Якщо встановлено час очікування, він може повертати менше символів, ніж вимагається. Без тайм-ауту він блокуватиметься, доки не буде прочитано запитану кількість байтів.
- Змінено у версії 2.5: повертає екземпляр байтів, якщо доступний (Python 2.6 і новіші версії), і str інакше.
- Змінено у версії 3.5: у попередніх версіях перший аргумент

read(size=1)

- Параметри: розмір – кількість байтів для читання.
- Повертає: байти, зчитані з порту.
- Тип повернення: байти
- Читання байтів розміру з послідовного порту. Якщо встановлено час очікування, він може повертати менше символів, ніж вимагається. Без тайм-ауту він блокуватиметься, доки не буде прочитано запитану кількість байтів.
- Змінено у версії 2.5: повертає екземпляр байтів, якщо доступний (Python 2.6 і новіші версії), і str інакше.

Діаграма класів програми керування наведена на рисунку 3.2.

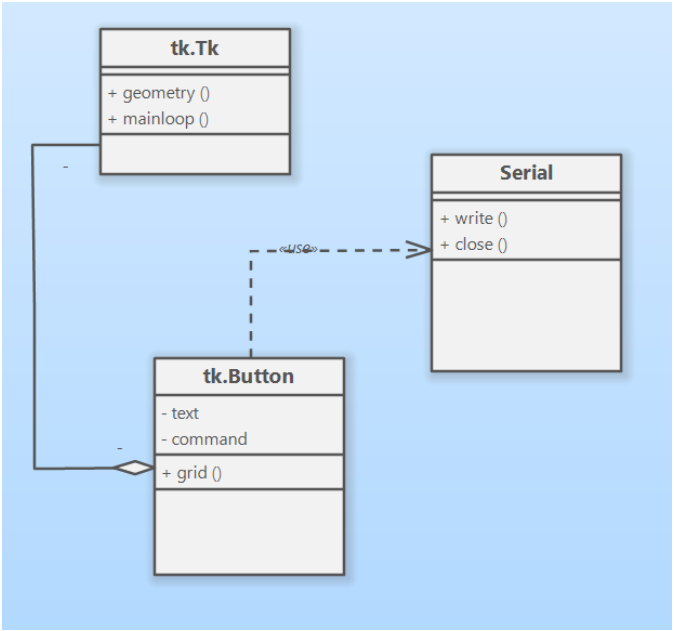

Рисунок 3.2 – Діаграма класів програми керування

До діаграми класів входять такі класи:

- tk.Tk: Представляє головне вікно застосунку.
  - geometry(): Метод для встановлення розміру вікна.
  - mainloop(): Метод для запуску циклу подій Tkinter.

tk.Button: Представляє кнопки в застосунку.

- Атрибути:
  - text: Текст кнопки.
  - command: Функція, яка виконується при натисканні на кнопку.
- Методи:
  - grid(): Метод для розміщення кнопки у вікні з використанням макета сітки.

Serial: Представляє об'єкт серійної комунікації.

- write(): Відправляє дані через серійний порт.
- close(): Закриває серійний порт.

Об'єкти tk.Button налаштовуються з атрибутом command, який викликає відповідні функції при натисканні.

Функції використовують об'єкт Serial (ser) для відправки команд через серійний порт.

Застосунок запускається за допомогою root.mainloop(), який утримує вікно відкритим та реагує на натискання кнопок.

Програма складається з кількох логічних частин. Першою логічною частиною є імпорт бібліотеки для графічного інтерфейсу та бібліотеки для серіал порта :

```
import tkinter as tk
import serial
```

Далі йде налаштування серіал порта :

```
try:
   ser = serial.Serial('COM3', 9600)
   ser.write('0'.encode('utf-8')) #СТОП
except serial.SerialException:
   print('Port is not available. Check the port name and try again.')
```

```
    print(serial.SerialException)
    exit(1)
```

Наступною логічною частиною коду є створення графічного інтерфейсу та додавання кнопок для виклику функцій для надсилання команд роботу :

```
# Спільні стилі для кнопок
button_style = {"width": 8, "height": 4, "bg": "lightblue", "fg": "black"}
root = tk.Tk()
root.geometry("300x200")
butoonVpered = tk.Button(text="В", command = lambda: vpered(),
    **button_style).grid(row=0, column=0)
butoonNazad = tk.Button(text="Н", command = lambda: nazad(),
    **button_style).grid(row=0, column=1)
butoonVlivo = tk.Button(text="Л", command = lambda: vlivo(),
    **button_style).grid(row=0, column=2)
butoonVpravo = tk.Button(text="П", command = lambda: vpravo(),
    **button_style).grid(row=0, column=3)
button0 = tk.Button(text="0", command = lambda: zero(),
    **button_style).grid(row=1, column=0)
button1 = tk.Button(text="1", command = lambda: one(),
    **button_style).grid(row=1, column=1)
button2 = tk.Button(text="2", command = lambda: two(),
    **button_style).grid(row=1, column=2)
button3 = tk.Button(text="3", command = lambda: three(),
    **button_style).grid(row=1, column=3)
```

Код для роботи програми та графічного вікна

```
try:
    root.mainloop()
except:
    ser.write('0'.encode('utf-8')) #СТОП
    ser.close()
```

```
ser.write('0'.encode('utf-8')) #СТОП
ser.close()
```

І ще є такий логічний блок коду як блок функцій які надсилають команди на робота :

Функція vpered() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб робот рухався вперед:

```
def vpered():
    ser.write('0'.encode('utf-8')) #СТОП
    ser.write('F'.encode('utf-8')) #ВПЕРЕД
```

Функція nazad() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб робот рухався назад:

```
def nazad():
    ser.write('0'.encode('utf-8')) #СТОП
    ser.write('B'.encode('utf-8')) #РЕВЕРС
    ser.write('F'.encode('utf-8')) #ВПЕРЕД
```

Функція vlivo() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб робот рухався вліво:

```
def vlivo():
    ser.write('0'.encode('utf-8')) #СТОП
    ser.write('L'.encode('utf-8')) #ВЛІВО
```

Функція vpravo() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб робот рухався впарво:

```
def vpravo():
    ser.write('0'.encode('utf-8')) #СТОП
    ser.write('R'.encode('utf-8')) #ВПРАВО
```

Функція zero() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб робот зупинився:

```
def zero():
    ser.write('0'.encode('utf-8')) #СТОП
```

Функція one() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб роботу задати коефіцієнт швидкості 1/3:

```
def one():
    ser.write('1'.encode('utf-8')) # ШВИДКІСТЬ 1
```

Функція two() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб роботу задати коефіцієнт швидкості 2/3:

```
def two():
    ser.write('2'.encode('utf-8')) # ШВИДКІСТЬ 2
```

Функція three() відповідає за подачу серії відповідних команд на мікроконтролер для того щоб роботу задати коефіцієнт швидкості максимальний:

```
def three():
    ser.write('3'.encode('utf-8')) # ШВИДКІСТЬ 3
```

Суть передачі команд керування полягає в передаванні відповідних команд на мікроконтролер у вигляді символів utf-8.

3.3 Висновки до розділу 3

У цьому розділі було розглянуто програмне забезпечення для мікроконтролера робота та для пульта керування оператора. Було описано алгоритми роботи цих програм, було описано ключові особливості їх роботи та описано можливі збої. Також було описано середовища розробки та описано програмні мови на яких були реалізовані дані програми.

Також було наведено частини програмного коду з їх описом. В результаті вийшли невеликі та нескладні не складні для розуміння проте повністю функціональні та ефективні програми.

4 ТЕСТУВАННЯ КОМПЛЕКСУ РОБОТА-САПЕРА

4.1 Тестування комплексу робота-сапера

Одним з важливих етапів розробки є перевірка правильності та працездатності комплексу. На цьому етапі ми маємо перевірити як окремі компоненти комплексу так і комплекс цілком.

Для початку перевіримо програму для надсилання команд на робота з комп'ютера. Для цього вже готову програму запустимо та перевіримо тестовим приймачем чи правильно видаються команди.

Ми визначили що все передається правильно. Передані дані відповідають відповідним командам оператора.

Далі перевіримо чи правильно приймаються та обробляються команди які приймає робот. Для цього будемо передавати команди на робота і перевіряти чи правильно робот виконує ці команди.

Ми визначили що робот отримує і обробляє команди правильно. Передані оператором команди були перетворені в правильні рухи робота.

Оглянемо загальну правильність конструкції та схеми робота (рис. 4.1). Як видно присутній у великій кількості тимчасовий та навісний монтаж, хоча основні елементи прикріплені жорстко до корпусу.

Також бачимо що робот заживлений від літій-іонного акумулятора якого йому вистачить приблизно на кілька годин роботи.

Видно що робот відносно невеликий проте має міцну раму зварену з міцного металевого кута відносно великої товщини яка дозволяє перевозити важкі вантажі до 200 кг.

Проте варто зазначити що в такому виконанні розводки електроніки є певна небезпека при дощі. Для цього можна закрити електроніку або спеціальним кожухом або підручними засобами.

Рисунок 4.1 – Зображення електроніки робота.

Також необхідно перевірити ходові якості робота. Для цього ми спробуємо проїхатися ним по пересіченій місцевості (рис. 4.2), (рис. 4.3).

Рисунок 4.2 – Зображення повороту робота

Рисунок 4.3 – Зображення подолання роботом перешкоди

Результати показали що робот має непогану прохідність, проте при поганому грунті можливі незначні ускладнення з поворотами.

Розроблений додаток має функціонал, наведений в таблиці 4.1, який було порівняно із функціоналом провідних комплексів галузі: TALON, ANDROS F6A, Dragon Runner.

Таблиця 4.1 – Порівняльна характеристика розробленого ПЗ

Назва комплексу	Модульність	Вантажопідйомність	Резервування керування	Простота використання
Моя розробка	+	+	+	+
TALON	+	+	-	-
ANDROS F6A	+	-	-	-
Dragon Runner	-	-	-	-

4.2 Висновок до розділу 4

Отже, розроблене програмне забезпечення для керування роботизованим комплексом робот-сапер було протестовано на відповідність меті дослідження. Функціонал програмного забезпечення є розширеним по відношенню до існуючих сучасних рішень та відповідає завданню.

ВИСНОВКИ

У ході виконання бакалаврської дипломної було розроблено комплекс управління роботом та робота.

Було проведено аналіз аналогів, а саме наземних роботизованих комплексів роботів-саперів, було розглянуто особливості такого типу комплексів та наведено декілька прикладів таких комплексів. На підставі цього аналізу було розроблено концепт хорошого робота передбачає здешевлення та деяке спрощення без значної шкоди функціоналу.

Були детально розглянуті ключові аспекти проектування робота та основи його конструкції. Було створено та розглянуто електричну схему та структурну схему робота.

Було розроблено програмний модуль робота. Було описано алгоритм роботи цієї програми, описано ключові особливості її роботи. Також було описано мову програмування Wiring та описано середовище розробки. Наведено частину програмного коду із поясненнями.

Було розроблено програмне забезпечення керування роботом. Було описано алгоритми роботи цієї програми, було описано ключові особливості її роботи та описано можливі збої. Також було описано середовище розробки та описано мову програмування Python. Також було наведено частини програмного коду з їх описом.

Було проведено тестування комплексу яке включало в себе перевірку розширення функціональних можливостей програмного забезпечення керування роботизованим комплексом, правильність видачі команд, надійності та прохідності. За результатами тестування можна стверджувати що якість керування дійсно висока, команди керування з пульта на робота передаються правильно та без помилок, конструкція надійна та витривала а прохідність висока, проте все ж недостатня для поганих грунтів.

В результаті було створено роботизований комплекс з мінімізацією витрат, доволі нескладною конструкцією та з розширеними функціональними можливостями програмного забезпечення керування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теза – [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20735

2. PackBot – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/PackBot

3. ANDROS F6A – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/ANDROS

4. Dragon Runner – [Електронний ресурс]. – Режим доступу: https://www.qinetiq.com/en-us/capabilities/robotics-and-autonomy/dragon-runner-small-and-compact-robot

5. Гіроскутер – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%93%D1%96%D1%80%D0%BE%D1%81%D0%BA%D1%83%D1%82%D0%B5%D1%80

6. Безколекторний електричний двигун – [Електронний ресурс]. – Режим доступу: https://drivesystems.com.ua/bldc-2/

7. Мотор-колесо – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Wheel_hub_motor

8. Вентильний електродвигун – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%B4%D0%B2%D0%B8%D0%B3%D1%83%D0%BD

9. Контролер безколекторного двигуна – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D1%80%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80_%D1%85%D0%BE%D0%B4%D1%83

10. Ардуіно – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Arduino

11. Bluetooth HC-06 – [Електронний ресурс]. – Режим доступу: https://arduino.ua/prod241-bluetooth-modyl-hc-06

12. Arduino Uno – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Arduino_Uno

13. C++ – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/C%2B%2B

14. ОС Windows 10 – [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Windows_10

15. Python – [Електронний ресурс]. – Режим доступу: https://www.python.org/

16. Tkinter – [Електронний ресурс]. – Режим доступу: https://docs.python.org/uk/3/library/tkinter.html

17. VS Code – [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/

18. Юнікод – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%AE%D0%BD%D1%96%D0%BA%D0%BE%D0%B4

19. ООП – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

20. PyCharm – [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/ru-ru/pycharm/

21. TALON – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Foster-Miller_TALON

22. Бен Стівенс «The C++ Programming Language», 2013. 1368 с.

23. Девід Бізлі та Браян Джонсон «Python Cookbook», 2013. 704 с.

24. Windows 10 – [Електронний ресурс]. – Режим доступу: https://wikipedia.org/wiki/Windows

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного забезпечення для комплексу робот-сапер

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 88,7% Схожість 11,3%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Дацюк А.І.

Керівник роботи Барабан С.В.

ДОДАТОК Б

Інструкція користувача

Програма керування є доволі простою у використанні. Для її запуску потрібно спочатку підключитися по bluetooth до контролера за допомогою стандартних функцій Windows а далі на комп’ютері запустити програму яка називається RobotControllAppWindows.exe.

Програма має наступний вигляд(рис. Б.1).

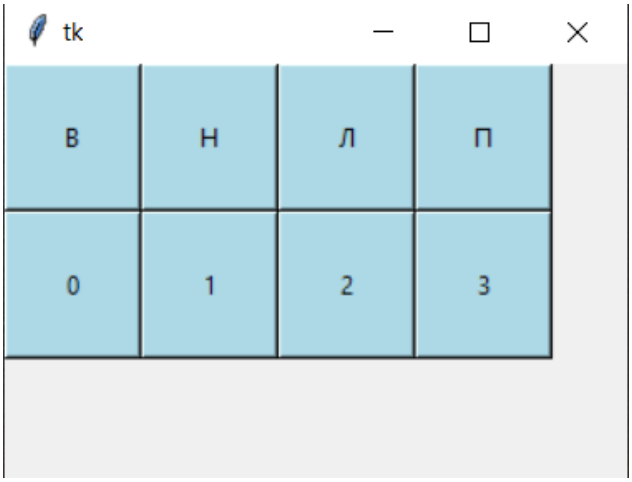

Рисунок Б.1 – Вигляд програми керування роботом

Як видно програма має 8 кнопок для керування. Завдяки ним можна виконувати переміщення робота з різними швидкостями і в різних напрямках. Розглянемо та опишемо їх:

«В» – Кнопка руху вперед, натиснути і відпустити для руху вперед;

«Н» – Кнопка руху назад, натиснути і відпустити для руху назад;

«Л» – Кнопка повороту вліво, натиснути і відпустити для повороту вліво;

«П» – Кнопка повороту вправо, натиснути і відпустити для повороту вправо;

«0» – Кнопка зупинки, натиснути і відпустити для зупинки;

«1» – Кнопка вибору швидкості 1/3;

«2» – Кнопка вибору швидкості 2/3;

«3» – Кнопка вибору максимальної швидкості.

ДОДАТОК В

Лістинг програмного коду

Програма мікроконтролера:

```
#include <SoftwareSerial.h>
#define R_Motor 2
#define Speed   3
#define L_Motor 4
SoftwareSerial BTSerial(10, 11); // RX, TX
float speedCoef = 0.3;
void zero(){
  digitalWrite(R_Motor, HIGH);
  digitalWrite(Speed, LOW);
  digitalWrite(L_Motor, HIGH);
}
void right() {
  digitalWrite(R_Motor, LOW);
  delay(500);
  analogWrite(Speed, 255 * speedCoef);
}
void front() {
  analogWrite(Speed, 255 * speedCoef);
}
void left() {
  digitalWrite(L_Motor, LOW);
  delay(500);
  analogWrite(Speed, 255 * speedCoef);
}
void revers(){
  digitalWrite(R_Motor, LOW);
  digitalWrite(L_Motor, LOW);
  delay(500);
}
void setup() {
  BTSerial.begin(9600); // the baud rate of HC-05 is usually 38400
  Serial.begin(9600);
  pinMode(R_Motor, OUTPUT);
  pinMode(Speed, OUTPUT);
  pinMode(L_Motor, OUTPUT);
}
void loop() {
  if (BTSerial.available()) {
    char c = BTSerial.read();
```

```
    Serial.println©;
    switch©{
      case ‘0’:
        zero();
        break;
      case ‘R’:
        right();
        break;
      case ‘F’:
        front();
        break;
      case ‘L’:
        left();
        break;
      case ‘B’:
        revers();
        break;
      case ‘1’:
        speedCoef = 0.3;
        break;
      case ‘2’:
        speedCoef = 0.5;
        break;
      case ‘3’:
        speedCoef = 1;
        break;
    }
  }
}
```

Програма для керування:

```
import tkinter as tk
import serial
try:
    ser = serial.Serial(‘COM3’, 9600)
    ser.write(‘0’.encode(‘utf-8’)) #СТОП
except serial.SerialException:
    print(‘Port is not available. Check the port name and try again.’)
    print(serial.SerialException)
    exit(1)
def vpered():
    ser.write(‘0’.encode(‘utf-8’)) #СТОП
    ser.write(‘F’.encode(‘utf-8’)) #ВПЕРЕД
def nazad():
```

```
    ser.write('0'.encode('utf-8')) #СТОП
    ser.write('B'.encode('utf-8')) #РЕВЕРС
    ser.write('F'.encode('utf-8')) #ВПЕРЕД
def vlivo():
    ser.write('0'.encode('utf-8')) #СТОП
    ser.write('L'.encode('utf-8')) #ВЛІВО
def vpravo():
    ser.write('0'.encode('utf-8')) #СТОП
    ser.write('R'.encode('utf-8')) #ВПРАВО
def zero():
    ser.write('0'.encode('utf-8')) #СТОП
def one():
    ser.write('1'.encode('utf-8')) #ШВИДКІСТЬ 1
def two():
    ser.write('2'.encode('utf-8')) #ШВИДКІСТЬ 2
def three():
    ser.write('3'.encode('utf-8')) #ШВИДКІСТЬ 3
# Спільні стилі для кнопок
button_style = {«width»: 8, «height»: 4, «bg»: «lightblue», «fg»: «black»}
root = tk.Tk()
root.geometry(«300x200»)
butoonVpered = tk.Button(text=»В», command = lambda: vpered(),
**button_style).grid(row=0, column=0)
butoonNazad = tk.Button(text=»Н», command = lambda: nazad(),
**button_style).grid(row=0, column=1)
butoonVlivo = tk.Button(text=»Л», command = lambda: vlivo(),
**button_style).grid(row=0, column=2)
butoonVpravo = tk.Button(text=»П», command = lambda: vpravo(),
**button_style).grid(row=0, column=3)
button0 = tk.Button(text=»0», command = lambda: zero(), **button_style).grid(row=1,
column=0)
button1 = tk.Button(text=»1», command = lambda: one(), **button_style).grid(row=1,
column=1)
button2 = tk.Button(text=»2», command = lambda: two(), **button_style).grid(row=1,
column=2)
button3 = tk.Button(text=»3», command = lambda: three(), **button_style).grid(row=1,
column=3)
try:
    root.mainloop()
except:
    ser.write('0'.encode('utf-8')) #СТОП
    ser.close()
ser.write('0'.encode('utf-8')) #СТОП
ser.close()
```

ДОДАТОК Г

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМПЛЕКСУ РОБОТ-САПЕР

Виконав студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Дацюк А. І.
(прізвище та ініціали)
Керівник: к.т.н., доцент каф. КН
Барабан С. В.
(прізвище та ініціали)
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Рисунок Г.1 – Робот «TALON»

Рисунок Г.2 – Робот «PackBot»

Рисунок Г.3 – Робот «ANDROS F6A»

Рисунок Г.4 – Робот «Dragon Runner»

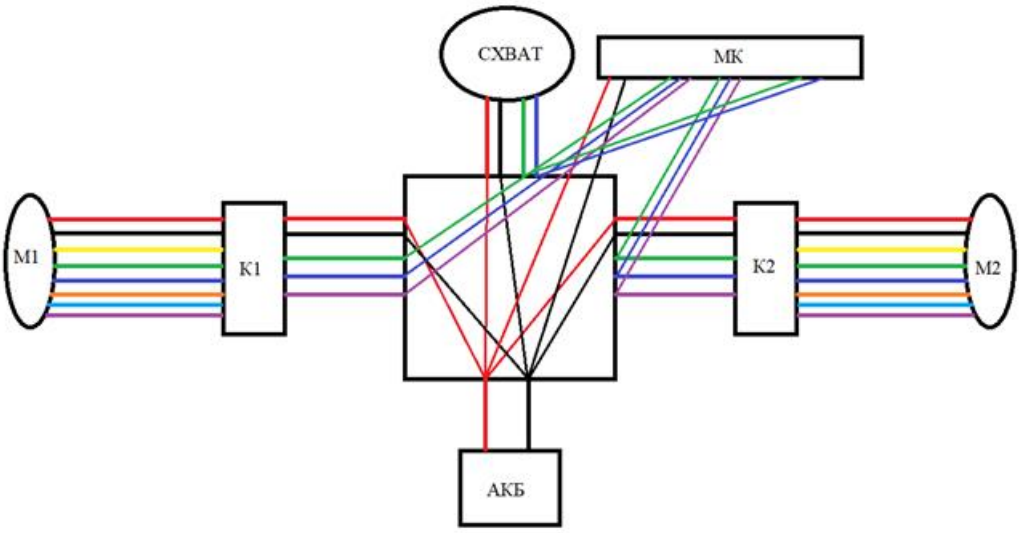

Рисунок Г.5 – Схема електричної проводки робота

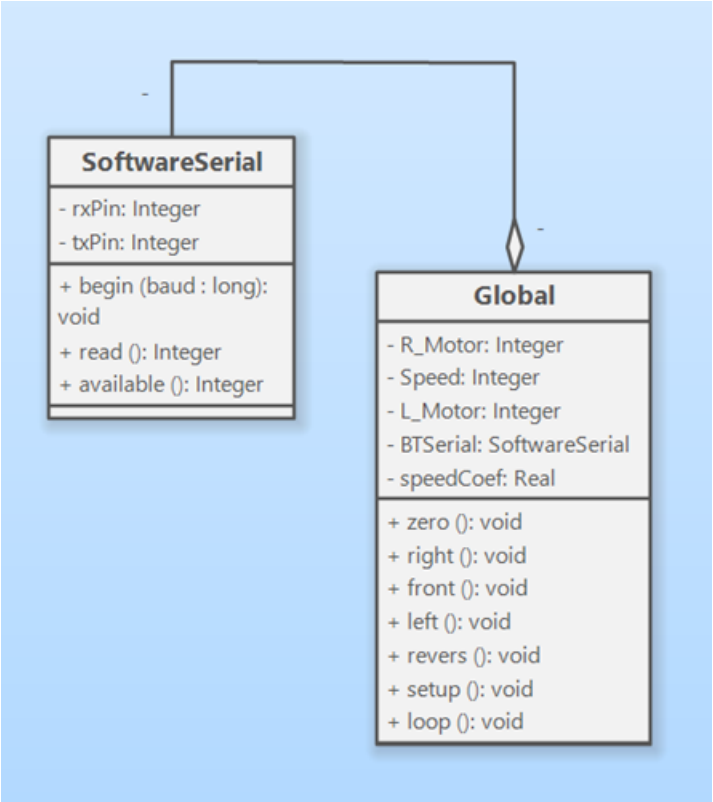

Рисунок Г.6 – Діаграма класів програмного модуля робота

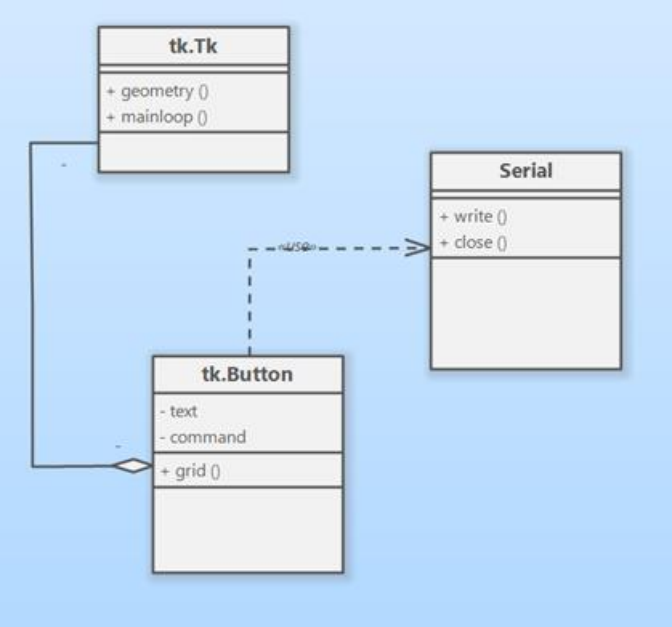

Рисунок Г.7 – Діаграма класів програми керування

Рисунок Г.8 – Зображення електроніки робота

Рисунок Г.9 – Зображення повороту робота

Рисунок Г.10 – Зображення подолання роботом перешкоди

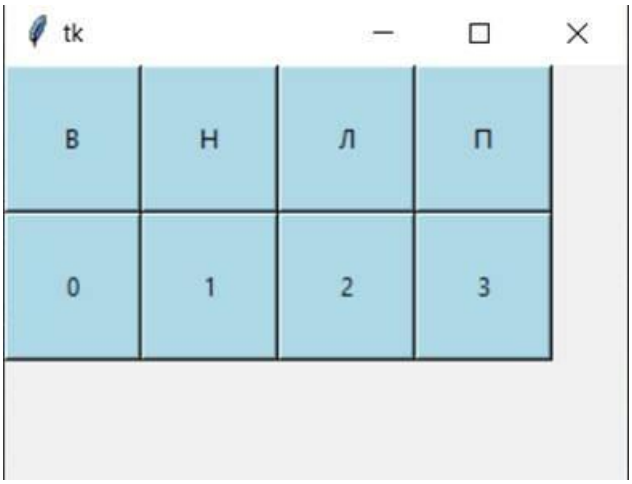

Рисунок Г.11 – Вигляд програми керування роботом