МАГІСТЕРСЬКА РОБОТА

МР.ПМКм–11.00.00.000 ПЗ

Група ПМКм-23-1

Ліуш Руслан

Русланович

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інженерної механіки та робототехніки

Кафедра: комп'ютеризованого машинобудування

Ліуш Руслан Русланович
(прізвище, ім'я, по батькові)

УДК 621.3
(індекс)

МАГІСТЕРСЬКА РОБОТА

Система керування моделі марсохода
(назва роботи)

Комп'ютеризовані та роботизовані технології машинобудування
(назва освітньої програми)

131 – Прикладна механіка
(шифр і назва спеціальності)

Р.Р. Ліуш
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Шуляр Богдан Романович, кандидат технічних наук, доцент кафедри комп'ютеризованого машинобудування
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту
Завідувач кафедри
професор_____Панчук В. Г._
(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

_____
(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

м. Івано-Франківськ — 2024 рік

Івано-Франківський національний технічний університет нафти і газу
(повне найменування закладу вищої освіти)

Інститут інженерної механіки та робототехніки
Кафедра комп'ютеризованого машинобудування
Освітній рівень магістр
Спеціальність 131 – Прикладна механіка
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри _____

_____

«____» _____20__ року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТОВІ

Ліушу Руслану Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Система керування моделі марсохода

керівник роботи Шуляр Богдан Романович, кандидат технічних наук, доцент кафедри комп'ютеризованого машинобудування
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "___"_____ 2024 року № _____

2.Строки подання студентом роботи грудня 2024 р.

3. Вихідні дані до роботи: 3D модель та спрощена мініатюра марсохода Perceverance.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз марсохода Perceverance. 2. Аналіз апаратниих компонентів. 3. Розробка алгоритму керування марсоходом. 4. Виготовлення блоку керування Марсоходом. 5. Експериментальна частина.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1.Структурна схема – 1 аркуш А4. 2. Блок-схема – 1 аркуш А4. 3. Лістинг коду – 16 аркушів А4. 4. Креслення корпуса – 2 аркуші А3. 5. Креслення кришки – 1 аркуш А3.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Шуляр Б.Р. доцент кафедри КМВ		
2	Шуляр Б.Р. доцент кафедри КМВ		
3	Шуляр Б.Р. доцент кафедри КМВ		
4	Шуляр Б.Р. доцент кафедри КМВ		
5	Шуляр Б.Р. доцент кафедри КМВ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Термін виконання етапів роботи	Примітки
1	Загальна характеристика	01.04.2024	
2	Опис і конструкція навчального проєкту	01.08.2024	
3	Проектна частина	05.10.2024	
4	Конструкторська частина	18.11.2024	
5	Дослідницька частина	24.11.2024	
6	Захист магістерської роботи	27.12.2024	

Студент _____ Ліуш Р.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шуляр Б.Р.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська робота на тему «Система керування моделі марсохода» присвячена створенню системи керування для моделі марсохода, що імітує функції справжніх планетохідних апаратів. Дана робота складається з 65 аркушів. В неї входять 25 рисунків та 5 додатків.

Основною метою дослідження є розробка інтерактивної платформи, яка може використовуватися для освітніх, дослідницьких і демонстраційних цілей, сприяючи глибшому розумінню принципів робототехніки, систем керування та навігації.

Актуальність роботи зумовлена необхідністю створення доступних і функціональних моделей для вивчення основних концепцій сучасних робототехнічних систем. Розроблена модель марсохода дозволяє вивчати принципи дистанційного управління, алгоритми автоматизації, роботу датчиків та обробку даних. Крім того, вона є прикладом інтеграції апаратного і програмного забезпечення для виконання реальних завдань.

Даний проєкт спрямований на те, щоб забезпечити студентів, інженерів-початківців і науковців практичною платформою для тестування своїх ідей та отримання практичного досвіду. Модель марсохода дає змогу досліджувати різноманітні сценарії роботи: від основ управління рухом до складних задач автономної навігації й аналізу середовища.

Таким чином, ця робота створює основу для вивчення сучасних технологій та їхнього застосування, водночас сприяючи розвитку практичних навичок у робототехніці, програмуванні й інтеграції систем.

SUMMARY

The master's thesis titled "Control System of a Mars Rover Model" is dedicated to developing a control system for a Mars rover model that simulates the functions of actual planetary exploration vehicles. This work comprises 65 pages, includes 25 figures, and features 5 appendices.

The primary goal of the research is to create an interactive platform that can be utilized for educational, research, and demonstration purposes, promoting a deeper understanding of robotics, control systems, and navigation principles.

The relevance of the work lies in the need to develop accessible and functional models for studying the fundamental concepts of modern robotic systems. The developed Mars rover model facilitates the study of remote control principles, automation algorithms, sensor operation, and data processing. Additionally, it serves as an example of hardware and software integration to perform real-world tasks.

This project aims to provide students, novice engineers, and researchers with a practical platform to test their ideas and gain hands-on experience. The Mars rover model enables the exploration of various operational scenarios, ranging from basic motion control to complex tasks such as autonomous navigation and environmental analysis.

Thus, this thesis lays the foundation for studying modern technologies and their applications while fostering the development of practical skills in robotics, programming, and systems integration.

ЗМІСТ

					МР.ПМКм–11.00.00.000 ПЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Ліуш Р.Р.			Система керування моделі марсохода	Літ.	Арк.	Аркушів
Перевір.		Шуляр Б.Р.						
Реценз.						ІФНТУНГ ПМКм–23–1		
Н. Контр		Шуляр Б.Р.						
Затверд.		Панчук В.Г.						

ВСТУП

Освоєння космосу є одним із найважливіших напрямків сучасної науки і техніки, що вимагає розробки інноваційних рішень у галузі робототехніки, електроніки та програмування. Особливу увагу привертають марсоходи – автономні роботизовані платформи, які здійснюють дослідження поверхні Марса. Одним із найбільш передових прикладів таких пристроїв є марсохід Perseverance, створений NASA.

Мета даної магістерської роботи полягає у розробці системи керування для зменшеної та спрощеної моделі марсохода Perseverance. Це включає проектування електронної та програмної частини, що дозволяють забезпечити базову функціональність моделі: рух, навігацію, виконання простих задач і взаємодію з користувачем.

У ході роботи будуть досліджені ключові принципи проектування роботизованих систем, а також реалізовані алгоритми керування, що враховують особливості руху марсоходів у реальних умовах. Результати досліджень та розробок можуть бути використані для освітніх і дослідницьких цілей, сприяючи подальшому вдосконаленню технологій робототехніки.

1 АНАЛІЗ МАРСОХОДА PERSEVERANCE

1.1 Історія проєктування та виробництво марсохода Perseverance

Марсохід Perseverance (рисунок 1.1), створений NASA, є одним із найвідоміших та найсучасніших роботизованих дослідників Марса. Його розробка почалася в рамках програми Mars 2020, що є частиною ширшої ініціативи NASA з дослідження Червоної планети та підготовки до майбутніх пілотованих місій.

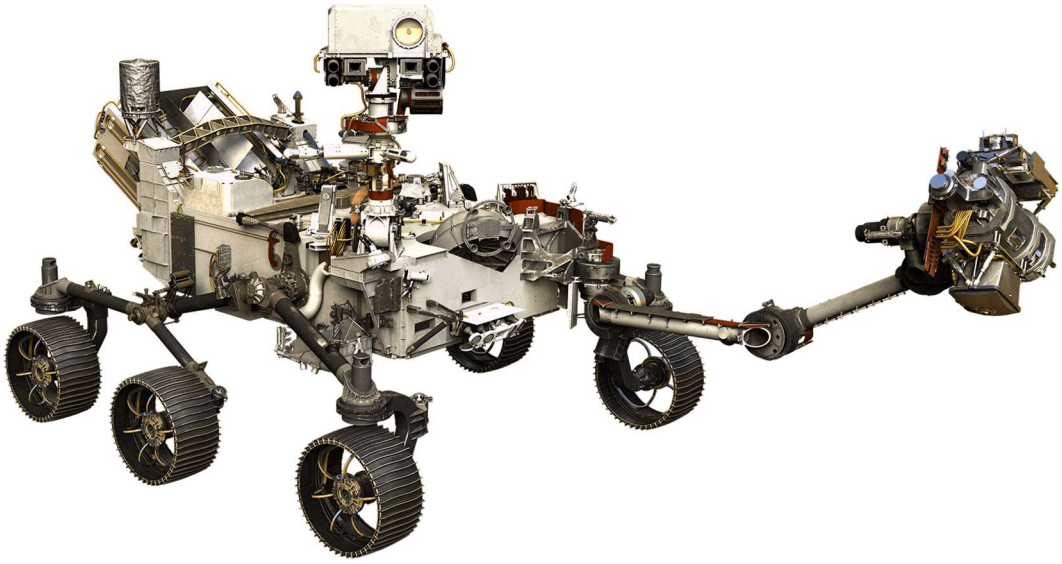


Рисунок 1.1 – Комп’ютерна візуалізація марсоходу Perseverance

У 2012 році NASA почала розглядати можливість створення нового марсохода після успіху місії Curiosity. Ідея полягала у створенні апарата, який не лише продовжить дослідження, але й зробить значний внесок у вивчення можливості існування життя на Марсі, збору зразків і підготовки до майбутніх польотів людини.

Основою для Perseverance став дизайн Curiosity (рисунок 1.2), однак були додані нові інструменти та покращені технології.

Рисунок 1.2 – Комп’ютерна візуалізація марсохода Curiosity

Процес проєктування і виробництва марсохода Perseverance був складним і багатоступеневим, спираючись на попередній досвід NASA, зокрема успіх місії Curiosity. Розробка почалася у 2013 році в Лабораторії реактивного руху (Jet Propulsion Laboratory, JPL) у Каліфорнії. Основна ідея полягала в тому, щоб створити марсохід, який не тільки продовжить наукову спадщину Curiosity, але й відкриє нові горизонти в дослідженні Марса. Конструктивно Perseverance зберіг основні риси свого попередника, але зазнав значних технологічних вдосконалень.

На початкових етапах інженери визначали ключові завдання, які мав виконувати марсохід. Це включало пошук слідів давнього життя, збір зразків ґрунту, випробування нових технологій і забезпечення високої автономності. Після формулювання цих завдань команда почала працювати над розробкою концепції, використовуючи комп’ютерне моделювання для тестування різних дизайнів і систем.

Однією з найскладніших частин роботи було проектування наукових інструментів, таких як SHERLOC і PIXL. Ці пристрої вимагали надзвичайної точності, оскільки їх основним завданням є аналіз хімічного складу поверхневих матеріалів і пошук органічних сполук. Для цього потрібно було створити

інструменти, які зможуть працювати в екстремальних умовах Марса, витримуючи температурні коливання, низький тиск і високу радіацію.

Виробництво системи коліс було іншим важливим аспектом. Попередній марсохід, Curiosity, зазнав певних труднощів через пошкодження коліс на нерівній марсіанській поверхні. Тому для Perseverance розробили нові колеса з міцнішого матеріалу з модифікованим дизайном, які змогли краще витримувати навантаження і протистояти зносу.

Окрему увагу приділили системі збору зразків. Ця інноваційна технологія вперше дозволила зберігати ґрунтові зразки у герметичних контейнерах, які в майбутньому зможуть бути доставлені на Землю іншими місіями. Це рішення потребувало розробки надточних маніпуляторів, здатних працювати автономно з мінімальною участю оператора із Землі.

Паралельно інженери працювали над системами енергозабезпечення та комунікації. Perseverance оснащений радіоізотопним термоелектричним генератором (RTG), який забезпечує постійне живлення навіть у складних марсіанських умовах. Системи комунікації базуються на комбінації антен високої та низької потужності, що дозволяє марсоходу передавати великі обсяги даних через орбітальні станції, такі як Mars Reconnaissance Orbiter.

Після завершення проєктування марсохід зібрали і провели масштабні випробування. Perseverance тестували на стійкість до вібрацій, вакуумних умов, низьких температур і впливу радіації. Зокрема, під час тестування симулювали всі етапи місії, від запуску ракетою до приземлення на поверхню Марса. Особливу увагу приділили перевірці роботи гелікоптера Ingenuity і пристрою MOXIE, який перетворює вуглекислий газ марсіанської атмосфери на кисень.

Процес виробництва і тестування зайняв близько семи років, і до моменту запуску у липні 2020 року Perseverance став одним із найбільш передових інженерних творінь у світі.

1.2 Запуск та початок місії марсохода Perseverance

Запуск марсохода Perseverance став визначною подією, яка відбулася 30 липня 2020 року. Для виведення апарата в космос використали потужну ракету Atlas V 541 виробництва United Launch Alliance. Запуск (рисунок 1.3) здійснювався з мису Канаверал, штат Флорида, з космічного стартового комплексу SLC-41. Завдяки ретельній підготовці та суворим випробуванням, місія розпочалася успішно, без технічних проблем.

Рисунок 1.3 – Запуск ракети Atlas V 541 30 липня 2020 року

Після виходу на траєкторію польоту до Марса Perseverance розпочав семимісячну подорож довжиною близько 470 мільйонів кілометрів. Під час польоту система забезпечення життєздатності марсохода працювала автономно, підтримуючи оптимальні умови для апарата, а команда на Землі регулярно перевіряла його стан. Важливо, що апарат мав обмежену можливість корекції траєкторії, і будь-які маневри виконувалися лише за необхідності.

Кульмінацією цієї подорожі стало приземлення 18 лютого 2021 року в кратері Єзеро, який було обрано завдяки його геологічній унікальності (рисунок 1.4). Колись цей регіон був озером, і його відкладення можуть зберігати сліди давнього життя, якщо воно існувало на Марсі. Посадка марсохода стала

однією з найбільш складних частин місії, адже для цього необхідно було пройти через "сім хвилин жаху" — період, коли апарат автономно виконував усі операції, і зв’язок із Землею був неможливим через затримку сигналу.

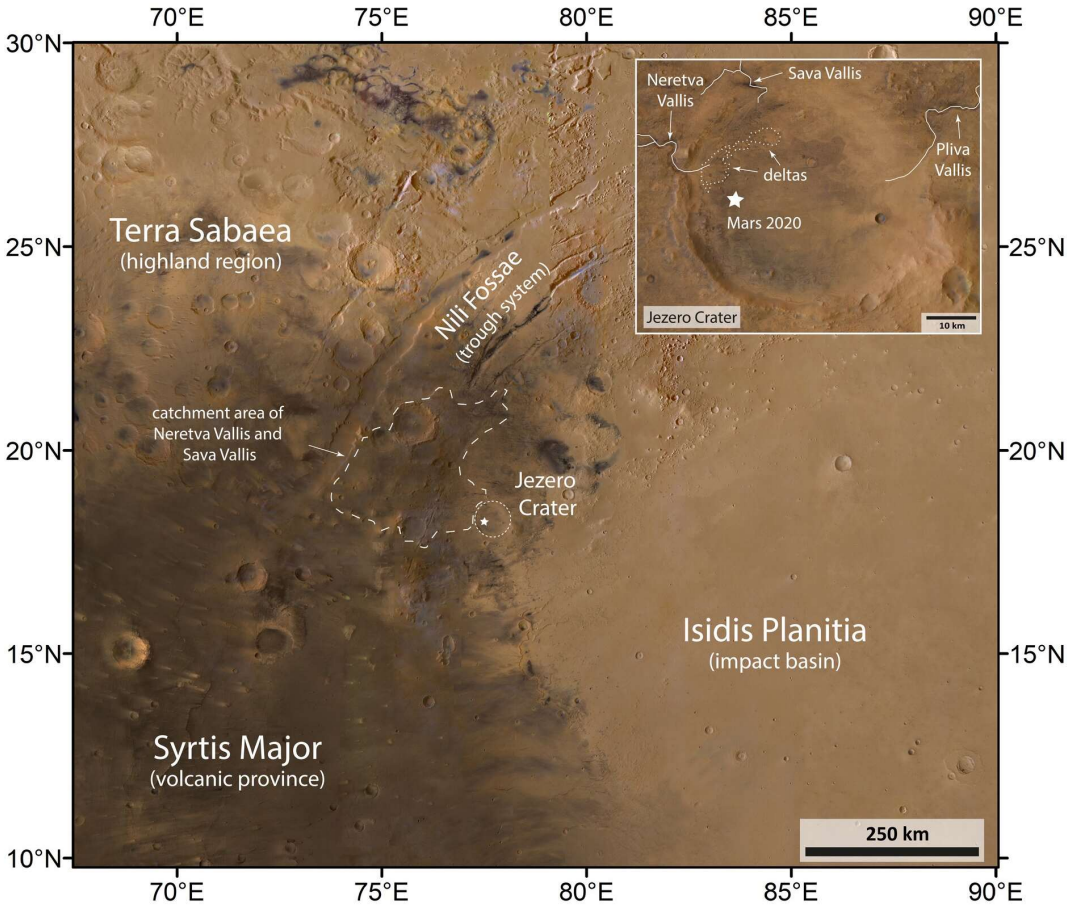


Рисунок 1.4 – Знімок рельєфу Марсу біля кратеру Єзеро

Для успішного приземлення (рисунок 1.5)[2] Perseverance використав систему Sky Crane, яка вже застосовувалася під час місії Curiosity. Ця система дозволила апарату повільно опуститися на поверхню за допомогою тросів, забезпечуючи м’яке і точне приземлення. Інноваційною особливістю посадки стало використання технології Terrain-Relative Navigation (TRN), яка дозволила марсоходу "бачити" місце посадки за допомогою камер і уникати перешкод у реальному часі.

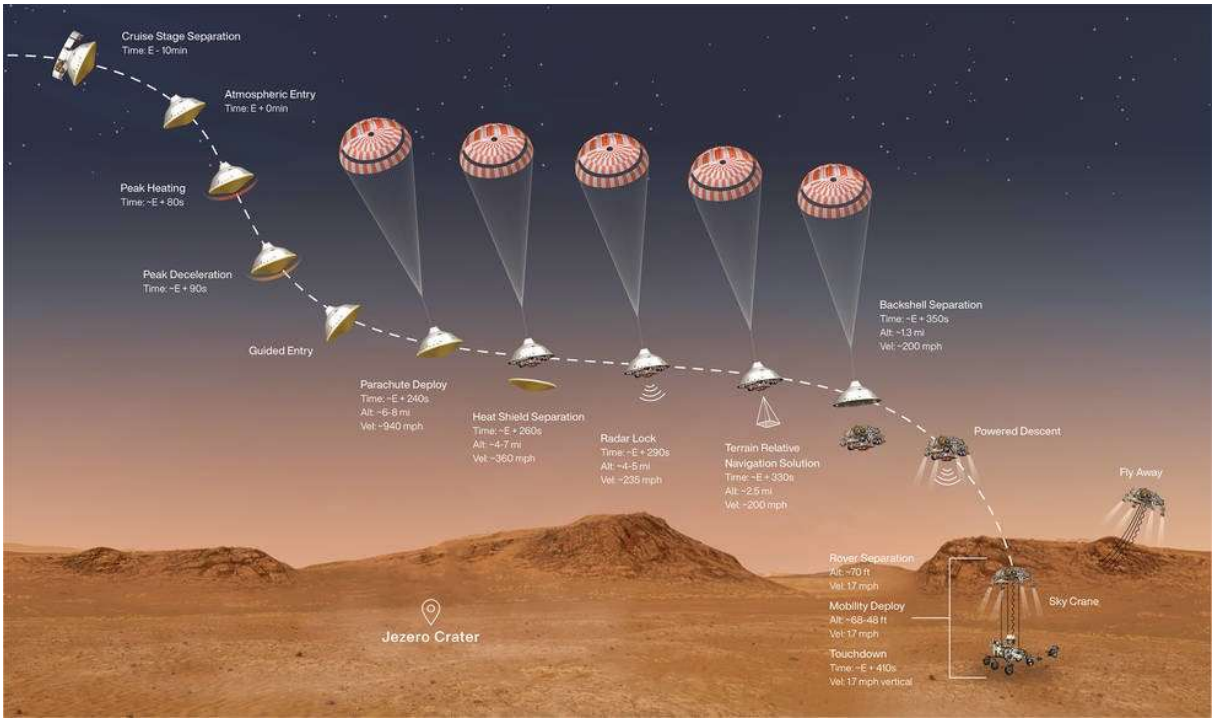


Рисунок 1.5 – Процес приземлення марсоходу Perseverance

Після приземлення марсохід розгорнув антену для встановлення стабільного зв’язку з орбітальними апаратами, а потім із Землею. Першими завданнями стали перевірка всіх систем і виконання початкових діагностичних тестів. Протягом перших кількох днів Perseverance надіслав перші зображення Марса, підтвердивши свою працездатність і готовність до виконання місії.

Одним із найзахопливіших моментів початкового етапу стало розгортання та тестування гелікоптера Ingenuity (рисунок 1.6). У квітні 2021 року він здійснив свій перший політ, ставши першим літальним апаратом, який злетів на іншій планеті. Це був важливий крок для підтвердження можливості аеродинамічного руху в розрідженій атмосфері Марса.

Рисунок 1.6 – Марсіанський вертоліт Ingenuity

Після початкових налаштувань марсохід розпочав виконання своїх основних наукових завдань: збирання зразків, дослідження кратера Єзеро та випробування нових технологій, таких як MOXIE. Зібрані дані вже зараз допомагають вченим краще зрозуміти геологічну історію Марса, а також перспективи майбутньої колонізації Червоної планети. [2]

1.2 Аналіз демонстраційної моделі марсохода

Марсохід, розроблений за прикладом Perseverance, є масштабованою моделлю, що поєднує елементи складної механіки, електроніки та програмного забезпечення. Основу конструкції становить шасі, виготовлене з легкого й міцного матеріалу, що забезпечує стійкість і надійність в умовах пересування по нерівних поверхнях. Шасі має інтегровану шестиколісну систему з незалежною підвіскою, яка дозволяє марсоходу долати перешкоди, імітуючи рух справжнього марсохода NASA.

Кожне колесо приводиться в рух за допомогою редукторних двигунів постійного струму, що забезпечують високу тягу і стабільний рух навіть на складному рельєфі. Рама марсохода спеціально адаптована для монтажу

електроніки та інших функціональних компонентів, зокрема камер і маніпуляторів.

Електронна частина включає в себе мікроконтролер, такий як Arduino або ESP32, що відповідає за обробку сигналів і керування всіма системами. Важливою складовою є плата драйвера двигунів, яка регулює швидкість і напрямок руху коліс. Також система оснащена модулем для дистанційного керування, що забезпечує інтерактивність і можливість отримання даних у реальному часі.

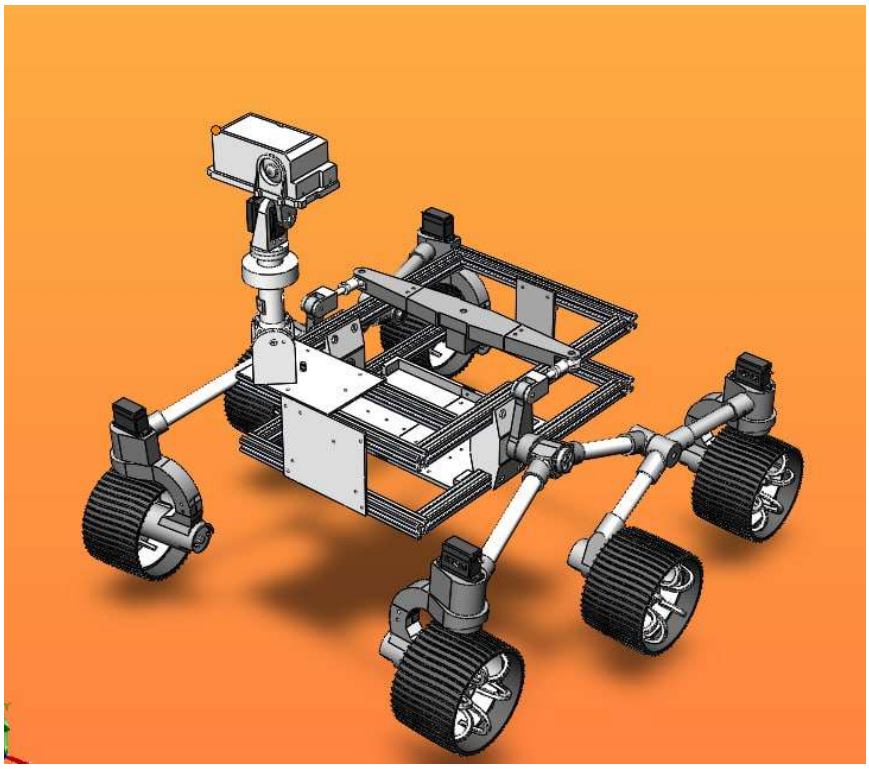


Рисунок 1.7 – Масштабована модель марсохода

Система керування моделі марсохода реалізує повертання за принципом геометрії Акермана (рисунок 1,8), що є ключовим підходом для забезпечення плавного і точного маневрування. Такий метод керування використовується для імітації поведінки автомобільного типу шасі, де передні колеса можуть змінювати напрямок руху, а задні залишаються прямолінійними.

У цій моделі поворотні колеса з’єднані з сервоприводом, який контролює кут повороту коліс. За допомогою мікроконтролера, розраховується необхідний кут для кожного поворотного колеса залежно від радіуса повороту. Цей кут

базується на принципі, що внутрішнє колесо повинно мати більший кут повороту порівняно з зовнішнім, щоб забезпечити їхню відповідність єдиній кривій траєкторії.

Додатково система може бути інтегрована з алгоритмами для автоматичного уникнення перешкод. Наприклад, якщо модель наближається до об'єкта, датчики відстані (ультразвукові або інфрачервоні) можуть виявити перешкоду і відповідно скоригувати траєкторію повороту, використовуючи геометрію Акермана.

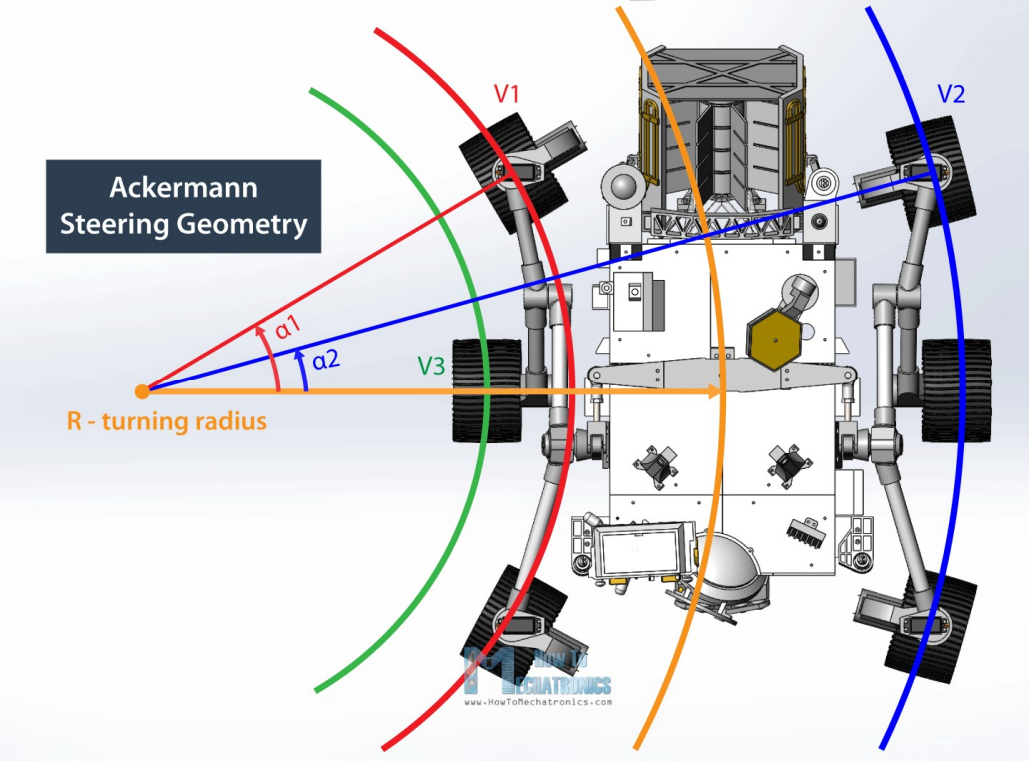


Рисунок 1.8 – Геометрія повороту при принципі Акермана

Такий підхід до керування забезпечує високу стійкість моделі на поворотах, знижує втрати енергії через ковзання коліс і дозволяє моделі маневрувати більш реалістично, імітуючи рух справжнього транспортного засобу. Система також демонструє наочність і практичну ефективність, що робить її ідеальним вибором для навчальних і дослідницьких проєктів.

2 АНАЛІЗ АПАРАТНИИХ КОМПОНЕНТІВ

2.1 Розробка структурної схеми

Структурна схема (Додаток А) демонструє систему керування моделі марсохода. Центральним компонентом системи є мікроконтролер ESP32, що забезпечує інтеграцію з платформою Blynk для дистанційного керування. Весь проєкт орієнтований на забезпечення точного та гнучкого керування приводами марсохода, включаючи його серводвигуни та колекторні мотори, які відповідають за рух та маніпуляції.

У цьому проєкті пристрій з Blynk виступає як основний інтерфейс керування. Це може бути смартфон або планшет, підключений до бездротової мережі, що забезпечує передачу команд до ESP32. За допомогою Blynk користувач може в реальному часі керувати всіма функціями марсохода, такими як його рух, положення маніпуляторів або навіть додатковими модулями.

ESP32, що є центральним обчислювальним модулем, отримує ці команди, обробляє їх та передає на периферійні модулі, такі як 16-канальний PWM генератор. Останній використовується для створення широтно-імпульсних сигналів, необхідних для керування приводами. У випадку марсохода такі сигнали забезпечують плавність і точність роботи моторів, що особливо важливо для маневрування на складному рельєфі або під час виконання складних механічних завдань.

Сервоприводи в цій схемі відповідають за поворот коліс. Вони підключені до PWM генератора, що дозволяє точно задавати їх положення відповідно до отриманих команд.

Колекторні двигуни забезпечують основний рух марсохода. Вони з'єднані з драйверами, які контролюють подачу сигналів і живлення. Кожен драйвер відповідає за пару моторів, забезпечуючи необхідну потужність для обертання коліс. Така конфігурація дозволяє забезпечити незалежне керування кожним двигуном, що необхідно для реалізації поворотів, руху вперед або назад.

2.2 Вибір компонентів для приладу

2.2.1 Сервомотор DS3225

DS3225 — це високопотужний цифровий сервопривід (рисунок 2.1), який широко використовується в робототехніці, радіокерованих моделях, проєктах DIY і автоматизації. Він вирізняється своїми великими крутним моментом, точністю і довговічністю. Нижче наведено детальний опис характеристик цього сервопривода.

Рисунок 2.1 – Сервомотор DS3225

Характеристики:

- вага: близько 66 г;
- розміри: 40.5 × 20 × 40 мм;
- крутний момент вала: 25 кгс*см (при живленні 7,4В), 20 кгс*см(6 В)
- Напруга живлення: 4,8 В – 7,4 В

2.2.2 ШІМ драйвер PCA9685

PCA9685 — це 16-канальний драйвер широтно-імпульсної модуляції (PWM), розроблений для управління сервоприводами, світлодіодами або іншими пристроями, які потребують точного регулювання сигналу. Він часто використовується у проєктах робототехніки, IoT та автоматизації завдяки своїй гнучкості, простоті використання та можливості одночасного керування багатьма каналами. (рисунок 2.2).

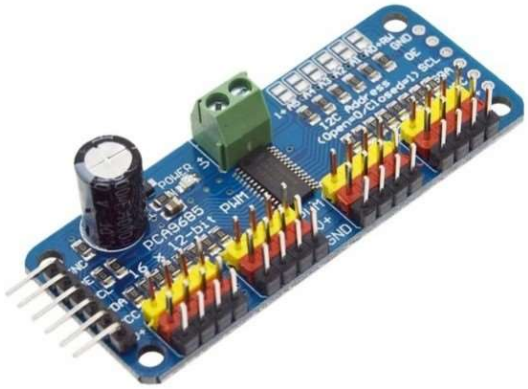


Рисунок 2.2 – 16-канальний 12-bit ШІМ шилд

Характеристики:

- кількість каналів: 16 незалежних каналів PWM;
- розрядність ШІМ-сигналу: 12 біт, що дає можливість задавати до 4096 рівнів регулювання ширини імпульсу;
- напруга живлення: логічна частина – 2,3-5,5 В, виходи для підключення пристроїв – до 6 В;
- інтерфейс зв’язку: I2C;
- частота PWM: Регульована в діапазоні від 24 Гц до 1526 Гц.

- можливість встановлення до 62 унікальних адрес за допомогою перемикачів, що дозволяє підключити до 992 виходів (16 каналів × 62 модулі) в одну систему;
- розміри: 62 мм х 25 мм х 15 мм.

2.2.3 ESP32 LuaNode32

ESP32 LuaNode32 — це програмоване Wi-Fi та Bluetooth модульне рішення, засноване на чіпі ESP32, яке популярне серед розробників для створення IoT-проєктів. LuaNode32 має підтримку програмування мовою Lua, що дозволяє швидко і просто писати сценарії для інтеграції пристрою з іншими системами або хмарними сервісами. Цей модуль відрізняється високою продуктивністю, енергоефективністю та багатим функціоналом.

Рисунок 2.3 – ESP32 LuaNode32

Характеристики

- наявність 2 процесорів: подвійне ядро Tensilica LX6 із тактовою частотою до 240 МГц та вбудований копроцесор для виконання малопотужних завдань у сплячому режимі;
- 520 КБ SRAM;

- підтримка зовнішньої флеш-пам’яті (зазвичай 4–16 МБ, залежно від модуля);
- підтримка Wi-Fi IEEE 802.11 b/g/n (режими STA, AP і STA+AP);
- підтримка Bluetooth класичного та BLE (Bluetooth Low Energy);
- 36 портів введення/виведення GPIO, що підтримують PWM, ADC, DAC, I²C, SPI, UART та інші інтерфейси;
- 18 каналів ADC(аналого-цифровий перетворювач) (12 біт) із діапазоном 0–3.3 В;
- 2 канали DAC (цифро-аналоговий перетворювач) (8 біт)
- робоча напруга 3.3 В (з інтегрованим регулятором для 5 В);
- попередньо встановлене середовище Lua або підтримка інших мов, таких як Python (MicroPython), Arduino IDE, або C/C++;
- дуже низьке енергоспоживання в сплячому режимі (менше 5 мкА);
- розміри: 27x55 мм.

2.2.4 Драйвер колекторних двигунів TB6612FNG

TB6612FNG — це двоканальний драйвер двигуна постійного струму (рисунок 2.4), розроблений для ефективного керування моторами в проектах робототехніки, автоматизації та IoT. Він підтримує роботу з низьковольтними двигунами та забезпечує точне регулювання швидкості й напрямку обертання. Завдяки своїй компактності, енергоефективності та зручному інтерфейсу, TB6612FNG широко використовується у проектах на базі мікроконтролерів, таких як Arduino, ESP32 або Raspberry Pi.

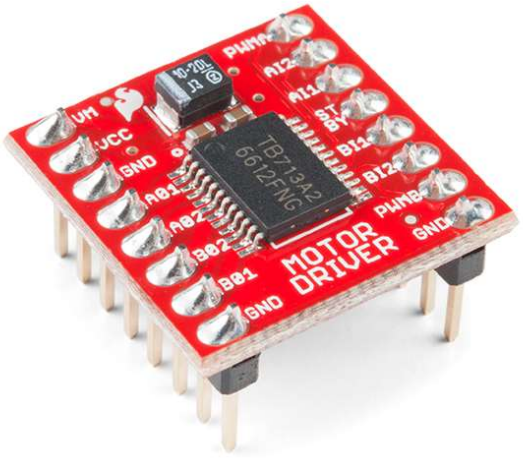


Рисунок 2.4 – TB6612FNG

Характеристики:

- підтримка двох окремих двигунів постійного струму;
- живлення логічної частини – 2,7-5,5 В, моторної – 4,5-13,5В;
- номінальний вихідний струм – 1,2 А, піковий – 3,2 А;
- керування швидкістю відбувається через входи PWM, для визначення напрямку обертання призначені цифрові піни;
- частота PWM до 100 кГц;
- захист від перевантаження по струму;
- захист від перегріву;
- захист від низької напруги;
- розміри: 21x18x3 мм.

2.2.5 Двигун колекторний XD-37GB555

XD-37GB555 — це компактний редукторний двигун постійного струму (рисунок 2.5), який широко використовується в робототехніці, автоматизації та інших проєктах, що вимагають низькошвидкісного, але висококрутного

обертання. Завдяки своїм характеристикам, цей двигун ідеально підходить для механізмів, які потребують стабільного моменту на низьких обертах.

Рисунок 2.5 – XD-37GB555

Характеристики;

- швидкість обертання: 50 об/хв (без навантаження);
- напруга живлення – 6-12 В;
- вбудований металевий редуктор забезпечує передачу високого крутного моменту (10 кг·см) на низькій швидкості;
- діаметр корпусу 37 мм (відповідає серії 37GB);
- циліндричний вал 6мм із плоскою частиною для надійного з'єднання з муфтами або іншими компонентами;
- середній струм споживання – 0,2-0,8 А, залежно від навантаження;
- маса приблизно 200 г.

3 РОЗРОБКА АЛГОРИТМУ КЕРУВАННЯ МАРСОХОДОМ

3.1 Розроблення алгоритму роботи системи

Алгоритм роботи програмного забезпечення, яке керує рухом та функціоналом марсохода зображено в додатку Б, створеного на платформі Arduino. Програма тісно взаємодіє з платформою Blynk, що дозволяє віддалено контролювати марсохід через інтернет або мобільний додаток.

На початку роботи програма ініціалізує всі необхідні компоненти марсохода: сервоприводи, відповідальні за рух коліс та маніпуляторів, а також двигуни, що забезпечують рух марсохода. Далі встановлюється з'єднання з платформою Blynk для отримання команд управління.

Після ініціалізації програма входить у безперервний цикл. На кожній ітерації циклу здійснюється зчитування нових команд з платформи Blynk. Ці команди можуть містити інформацію про бажаний напрямок руху, швидкість, а також інші параметри роботи марсохода.

Отримавши команду, програма аналізує її і відповідно змінює положення сервоприводів та керує роботою двигунів. Таким чином, марсохід виконує задану команду.

3.2 Розробка інтерфейсу керування через середовище Blynk

Blynk — це багатофункціональна платформа для створення проєктів Інтернету речей (IoT), яка надає змогу легко керувати й моніторити підключені пристрої через мобільний додаток. Завдяки простоті використання й широким можливостям, вона стала популярною як серед ентузіастів DIY-проєктів, так і серед професійних розробників.

Blynk працює через мобільний додаток, доступний для iOS та Android, який забезпечує інтуїтивно зрозумілий інтерфейс для взаємодії з IoT-пристроями. У цьому додатку користувач може створити власний графічний інтерфейс із

кнопками, повзунками, графіками чи іншими елементами управління. Для обміну даними між пристроями і додатком використовується сервер, який може бути як хмарним, так і локальним, що надає гнучкість у виборі середовища.

Ця платформа підтримує широкий спектр апаратного забезпечення, включаючи мікроконтролери ESP32, ESP8266, Arduino та Raspberry Pi. Для передачі даних вона використовує різноманітні протоколи зв'язку, такі як Wi-Fi, Ethernet, GSM/3G/4G, Bluetooth і BLE. Завдяки цьому пристрої можуть легко інтегруватися в існуючу інфраструктуру.

Для роботи з Blynk розробники використовують бібліотеку, яка доступна для багатьох середовищ, наприклад, Arduino IDE. Основою взаємодії є віртуальні піни, через які відбувається обмін даними між пристроєм і додатком. Ця бібліотека дозволяє створювати гнучкі системи керування з простим програмуванням.

Функціональність Blynk охоплює як керування пристроями (увімкнення чи вимкнення ламп, моторів тощо), так і моніторинг стану датчиків у реальному часі. Інтегрований інструмент сповіщень дозволяє отримувати push-повідомлення про стан системи, а автоматизація допомагає створювати сценарії, наприклад, для запуску обладнання за розкладом або за певними умовами. Також можна збирати дані від пристроїв і аналізувати їх у зручному форматі.

Перевагами Blynk є простота використання, універсальність і гнучкість. Користувачі можуть швидко створити інтерактивні інтерфейси, налаштувати автоматизацію і керувати пристроями з будь-якої точки світу. Крім того, активна спільнота користувачів і розробників допомагає знаходити рішення для будь-яких завдань.

Blynk ідеально підходить для широкого спектра застосувань. У сфері розумного дому його можна використовувати для керування освітленням, обігрівачами або камерами. У робототехніці платформа стане чудовим рішенням для моніторингу й управління роботами. Для екологічного моніторингу вона дозволяє відстежувати параметри довкілля, такі як температура, вологість чи

якість повітря. Завдяки підтримці автоматизації Blynk використовується також у промислових системах і DIY-проєктах.

Інтерфейс керування марсоходом потребує задання напряму руху вперед-назад, вправо-вліво, тому було обрано готовий модуль джойстика в меню конфігурування. Користувацький інтерфейс на смартфоні зображено на рисунку 3,1.

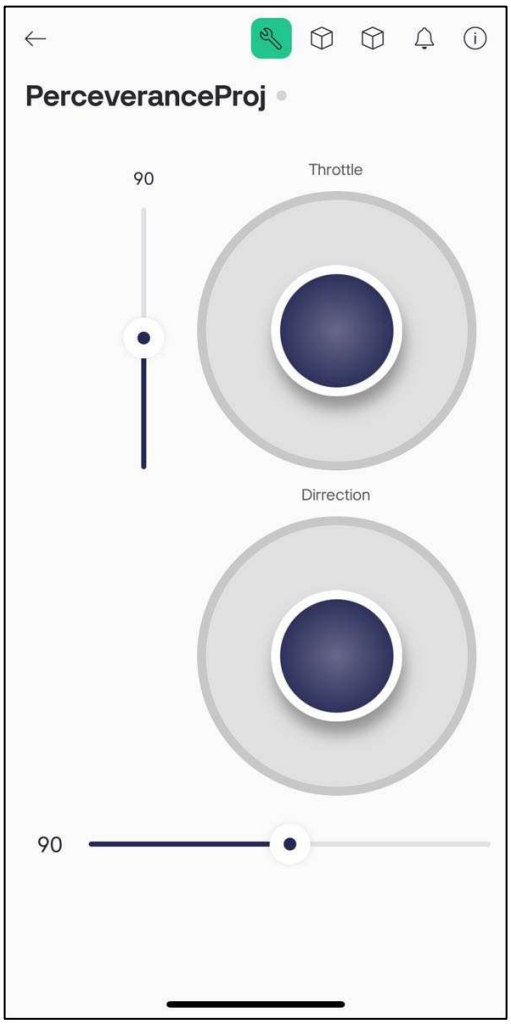


Рисунок 3.1 – Користувацький інтерфейс програми керування марсоходом

3.2 Розроблення програмної частини системи

Згідно з наведеним алгоритмом, програмний код, з урахуванням технічних аспектів програмування мікроконтролера, написаний на мові C/C++, наведено в додатку В.

Далі будуть розглянуті основні компоненти програмного коду для реалізації проєкту.

Код починається з підключення необхідних бібліотек, які забезпечують роботу з платформою Blynk, модулем WiFi, а також з контролером Adafruit PCA9685 для управління сервоприводами. Визначаються константи для ідентифікації пінів, підключених до моторів, сервоприводів та інших компонентів. Це дозволяє спростити доступ до апаратних ресурсів і зробити код більш читабельним. Також задаються мінімальні та максимальні значення сигналів для сервоприводів, які відповідають кутам повороту 0–180 градусів.

```
#include <SPI.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
```

У наступному блоці коду визначаються параметри для підключення до WiFi-мережі та Blynk-платформи, такі як ім'я мережі, пароль і токен автентифікації. Ініціалізується об'єкт для роботи з PCA9685, який використовується для генерації сигналів PWM для сервоприводів. Далі оголошуються глобальні змінні, що зберігають поточні стани пристрою, такі як кути повороту сервоприводів, швидкості моторів і фізичні параметри марсохода, наприклад відстані між колесами. Ці змінні також включають значення X і Y, які отримуються через Blynk для визначення напрямку руху марсохода.

```
#define SERVO_1 0
#define SERVO_3 1
...
#define MOTOR_6_IN2 16
```

Функція `setPWMViaPCA9685()` відповідає за генерацію PWM-сигналів для сервоприводів, перетворюючи значення в мікросекундах на відповідні імпульси. У функції setup виконується початкове налаштування пристрою, включаючи

ініціалізацію з'єднання з Blynk, встановлення частоти роботи PCA9685 та конфігурацію станів пінів. Також у коді є функція `motorStop`, яка використовується для зупинки всіх моторів, забезпечуючи безпечне завершення роботи пристрою.

Команди, що надходять через Blynk, обробляються для керування рухом марсохода. У кожному циклі виконуються обчислення для визначення напрямку та швидкості обертання моторів залежно від вхідних даних. Код враховує як апаратні параметри робота, так і програмні обмеження, щоб забезпечити плавну та ефективну роботу пристрою.

У функції `calculateServoAngle()` реалізовано розрахунок геометрія Акермана. Це досягається тим, що кожне кероване колесо повертається на свій специфічний кут, забезпечуючи перетин всіх осей обертання коліс в одній точці - центрі повороту. Це реалізовано через розрахунок різних кутів повороту для внутрішніх та зовнішніх коліс відносно центру повороту. При цьому використовується радіус повороту r та геометричні параметри платформи (d1-d4), які відповідають реальним відстаням між колесами марсохода у зменшеному масштабі.

Швидкості коліс при повороті також розраховуються за принципом Акермана - кожне колесо має свою швидкість пропорційну відстані від центру повороту. Це видно у функції `calculateMotorsSpeed()`, де:

`speed1` - максимальна швидкість для зовнішніх коліс

`speed2` - зменшена швидкість для передніх і задніх внутрішніх коліс

`speed3` - найменша швидкість для середнього внутрішнього колеса, яке найближче до центру повороту

Така система керування точно відтворює принципи руху реального марсохода, дозволяючи платформі здійснювати складні маневри з максимальною точністю та мінімальним зносом механічних частин.

4 ВИГОТОВЛЕННЯ БЛОКУ КЕРУВАННЯ МАРСОХОДОМ

4.1 3D моделювання корпусу за допомогою програми Autodesk Inventor

Autodesk Inventor – це потужне професійне програмне забезпечення, призначене для 3D-проектування. Воно широко використовується інженерами-конструкторами та дизайнерами для створення детальних моделей, складальних одиниць та робочих креслень. Програма надає широкий спектр інструментів, які дозволяють ефективно вирішувати складні інженерні завдання.

За допомогою Autodesk Inventor можна створювати реалістичні тривимірні моделі деталей та складальних одиниць, використовуючи різноманітні методи моделювання. Створені моделі можуть бути використані для створення детальних робочих креслень з усіма необхідними вимірами, допусками та написом. Програма також дозволяє проводити симуляції для перевірки міцності конструкцій та оптимізації їх характеристик.

Однією з ключових переваг Autodesk Inventor є його інтуїтивно зрозумілий інтерфейс, що дозволяє швидко освоїти програму навіть новачкам. Крім того, програма інтегрується з іншими продуктами Autodesk, такими як AutoCAD та Revit, що забезпечує безперебійний обмін даними між різними проектами.

Застосування Autodesk Inventor:

- Машинобудування: Проектування деталей машин, механізмів, приладів.
- Автомобільна промисловість: Розробка автомобільних компонентів та цілих автомобілів.
- Аерокосмічна промисловість: Створення моделей літаків, космічних апаратів та їх компонентів.
- Медична промисловість: Проектування медичного обладнання та імплантатів.

Переваги Autodesk Inventor:

- Широкий набір інструментів для 3D-моделювання, складання, створення креслень та симуляцій.
- Можливість налаштування програми під конкретні потреби користувача.
- Швидке виконання складних операцій моделювання.
- Простий та зрозумілий інтерфейс, що дозволяє швидко освоїти програму.
- Можливість інтеграції з іншими програмами Autodesk та сторонніми додатками.

.

На рисунку 4.1 зображений інтерфейс програмного забезпечення.

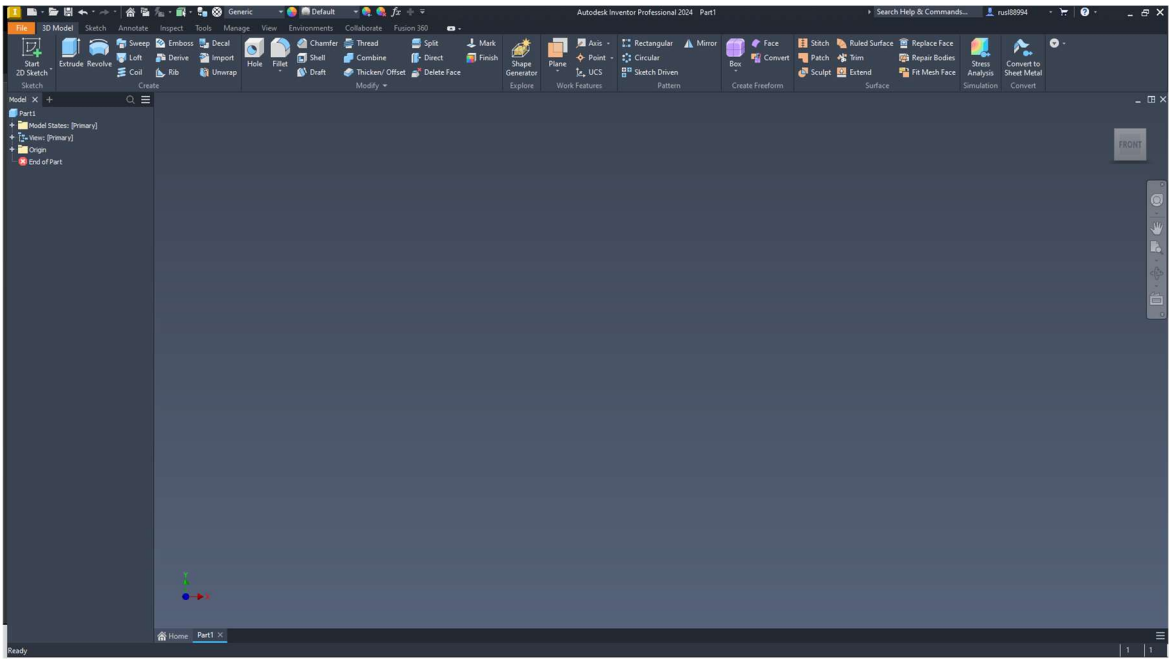


Рисунок 4.1 – Програмний інтерфейс програми Autodesk Inventor

На рисунках 4.2 та 4.3 зображено спроєктований корпус для електроніки керування марсоходом в програмному інтерфейсі Autodesk Inventor. Кресленики деталей корпусу наведені в додатках Г, Д.

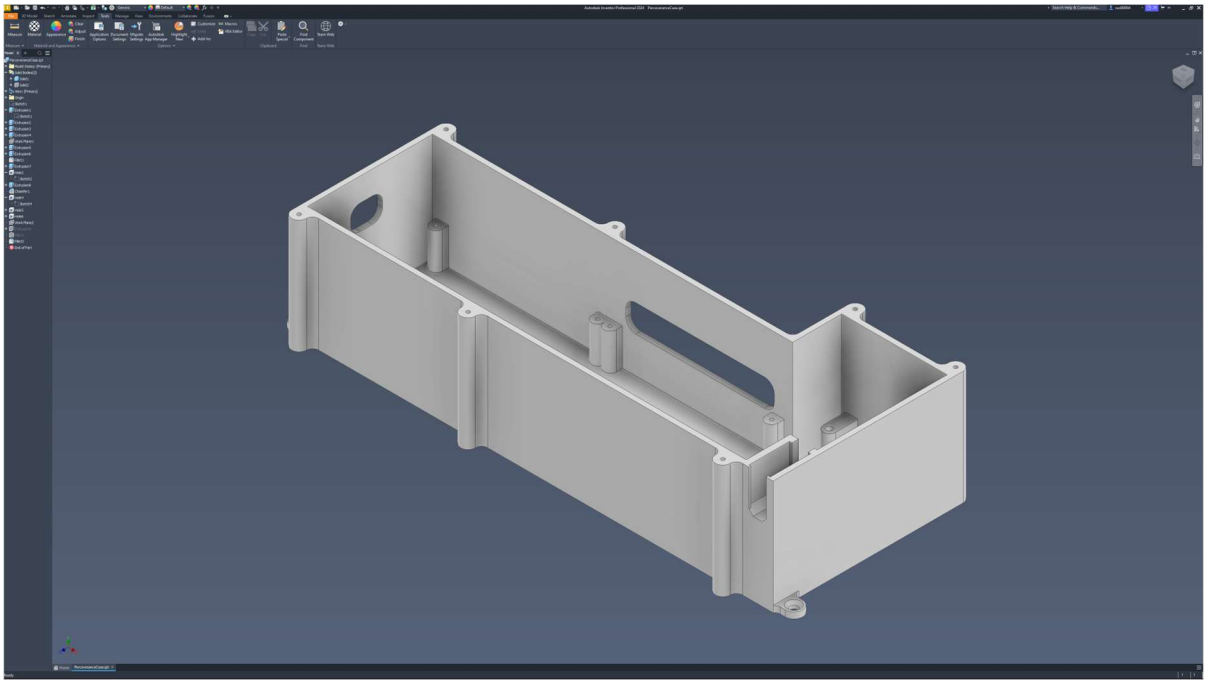


Рисунок 4.2 – Нижня частина корпусу для електроніки

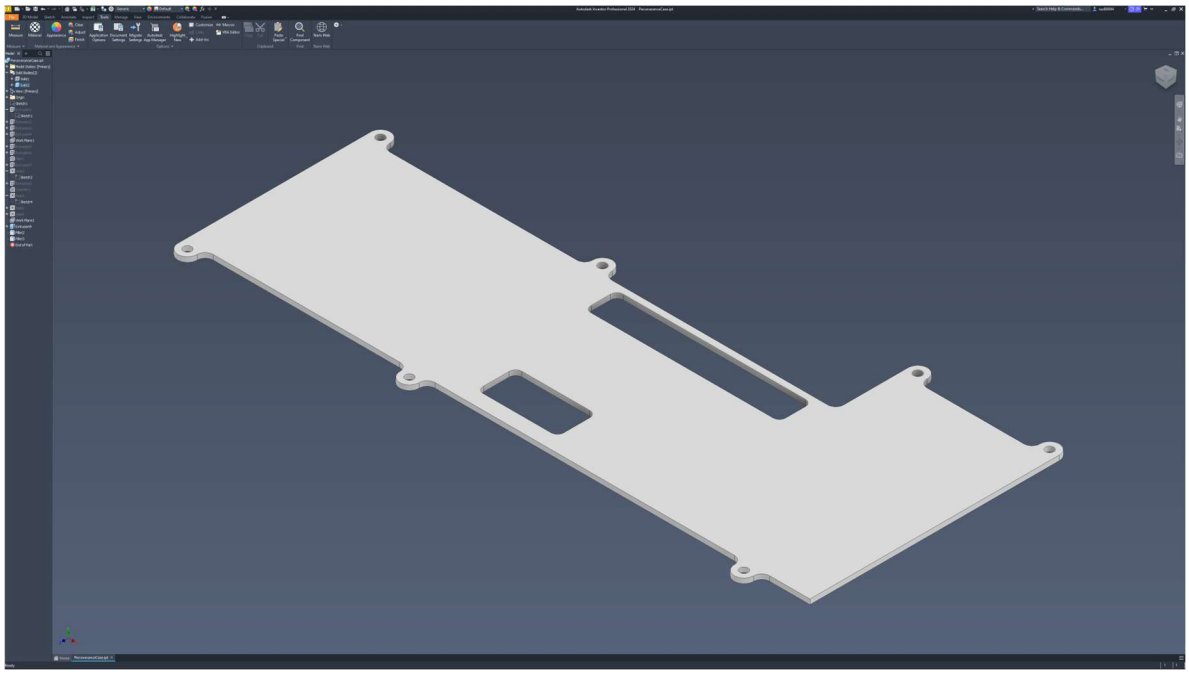


Рисунок 4.3 – Кришка корпусу для електроніки

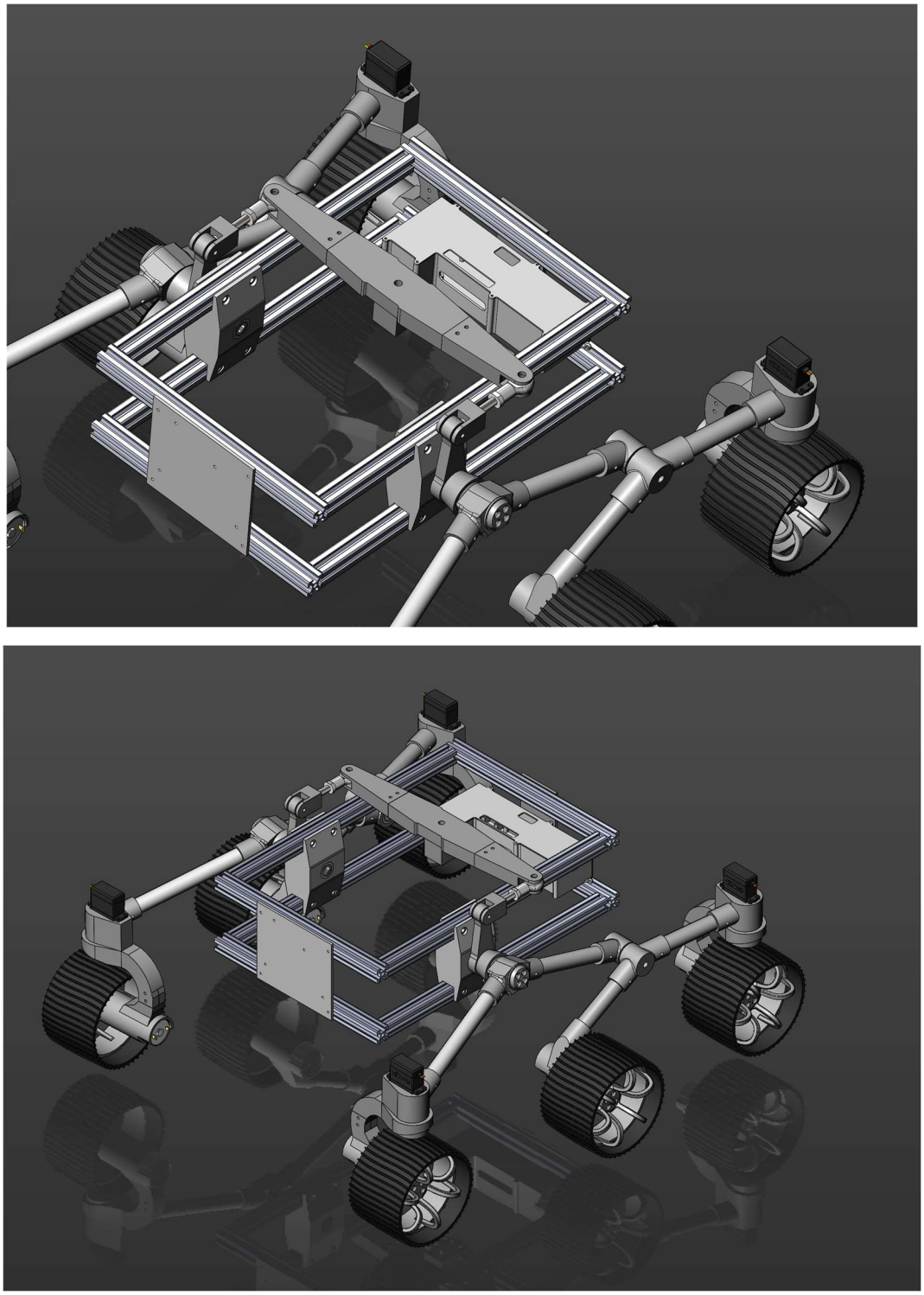


Рисунок 4.4 – Корпус для електроніки в моделі марсохода

4.2 Виготовлення корпусу для електроніки

4.2.1 Короткі відомості про технологію 3D друку

3D друк – це технологія, яка дозволяє створювати фізичні об'єкти безпосередньо з цифрових 3D моделей. Процес відбувається шляхом пошарового нанесення матеріалу, такого як пластик, метал чи смола, на платформу за заданою комп'ютерною моделлю. Ця технологія дозволяє втілити в життя практично будь-який дизайн, від простих прототипів до складних інженерних конструкцій.

На відміну від традиційних методів виробництва, 3D друк не вимагає використання спеціальних форм чи інструментів. Це значно спрощує процес створення невеликих партій виробів та дозволяє швидко вносити зміни в конструкцію. Завдяки цьому, 3D друк широко використовується в різних галузях, таких як медицина, автомобілебудування, аерокосмічна промисловість, ювелірна справа та архітектура.

Основними перевагами 3D друку є:

- Можливість створення унікальних виробів за індивідуальними замовленнями.
- Швидке створення прототипів та невеликих партій виробів.
- Можливість створення об'єктів з дуже складною геометрією.
- Відсутність необхідності у дорогих формах та інструментах.

Корпус був виготовлений методом FDM. Готові деталі зображені на рисунку 4.5.

FDM (Fused Deposition Modeling) – це одна з найпоширеніших технологій тривимірного друку, яка передбачає створення об'ємних предметів шляхом послідовного наплавлення розплавленого матеріалу. Процес починається з цифрової 3D-моделі, яку ділять на тонкі шари. Далі, спеціальний пристрій – екструдер – розплавляє термопластичний матеріал (найчастіше PLA або ABS), який подається у вигляді нитки, і наносить його на платформу шаром за шаром, точно повторюючи контури цифрової моделі. Кожен наступний шар

охолоджується і склеюється з попереднім, формуючи таким чином тривимірний об'єкт.

FDM відрізняється доступністю обладнання та широким вибором матеріалів, що робить цю технологію популярною як серед любителів, так і в промисловості. До переваг FDM можна віднести відносну простоту використання, можливість друку великогабаритних об'єктів та широкий спектр застосувань. Однак, до недоліків варто віднести відносно низьку швидкість друку та можливі недоліки поверхні готових виробів, які можуть потребувати додаткової обробки.

FDM широко використовується для створення прототипів, моделей, інструментів, деталей для промислового обладнання та багатьох інших виробів. Завдяки своїй універсальності та доступності, FDM стала однією з найпопулярніших технологій адитивного виробництва.

Схематичне зображення FDM технології друку зображено на рисунку 4.4.

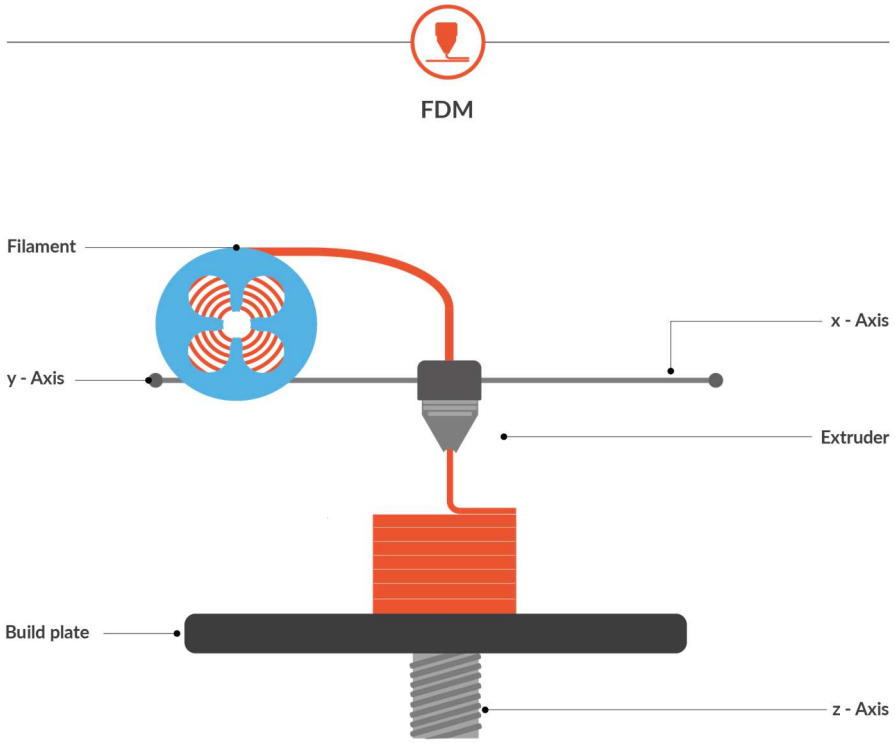


Рисунок 4.4 – Схематичне зображення FDM технології друку

Рисунок 4.5 – Деталі корпусу

5 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

На першому етапі експериментальної роботи було виконано монтаж електронного блоку, який є серцем моделі марсохода. До складу блоку входять такі ключові компоненти: мікроконтролер для обробки даних і управління системою, двигуни для забезпечення руху, датчики для отримання інформації про навколишнє середовище та модулі зв’язку для дистанційного управління.

Особливу увагу було приділено правильному розташуванню компонентів на основі марсохода. Компактне і водночас функціональне розміщення забезпечує ефективність роботи та захист компонентів від зовнішніх впливів. Виконано пайку і з’єднання проводів, дотримуючись вимог електричної сумісності та полярності підключення. Крім того, було враховано енергоспоживання всіх елементів, що дозволило розробити стабільну систему живлення. Виготовлені макетні плати з всіма необхідними компонентами зображено на рисунку 5.1.

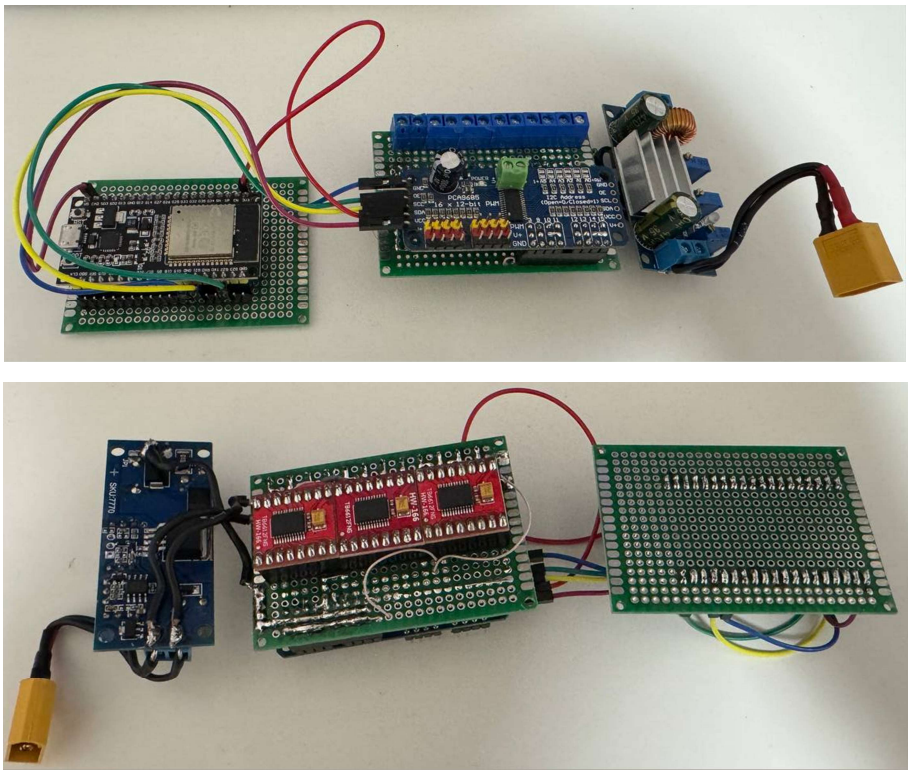


Рисунок 5.1 – Плати керування марсоходом

Для монтажу використовувалися спеціальні кріплення та ізоляційні матеріали, які зменшують ризик пошкодження електроніки під час руху марсохода по нерівному рельєфу. Схема підключення була протестована за допомогою мультиметра, що дозволило перевірити коректність електричних з’єднань перед початком тестування.

Також прокладено джгути для керування приводами, що зображено на рисунку 5.2.

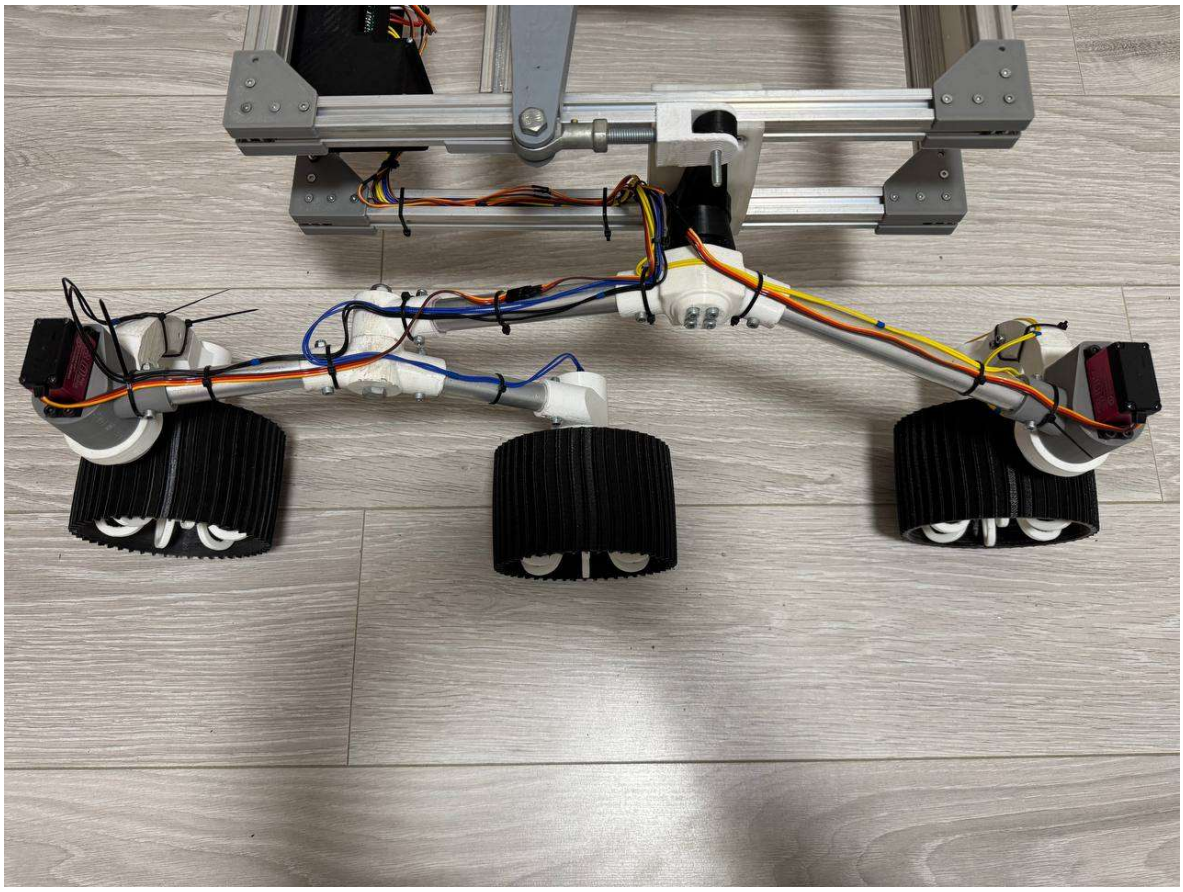


Рисунок 5.2 – Система кабелів для керування

На рисунку 5.3 зображено підключені джгути до змонтованого блоку керування.

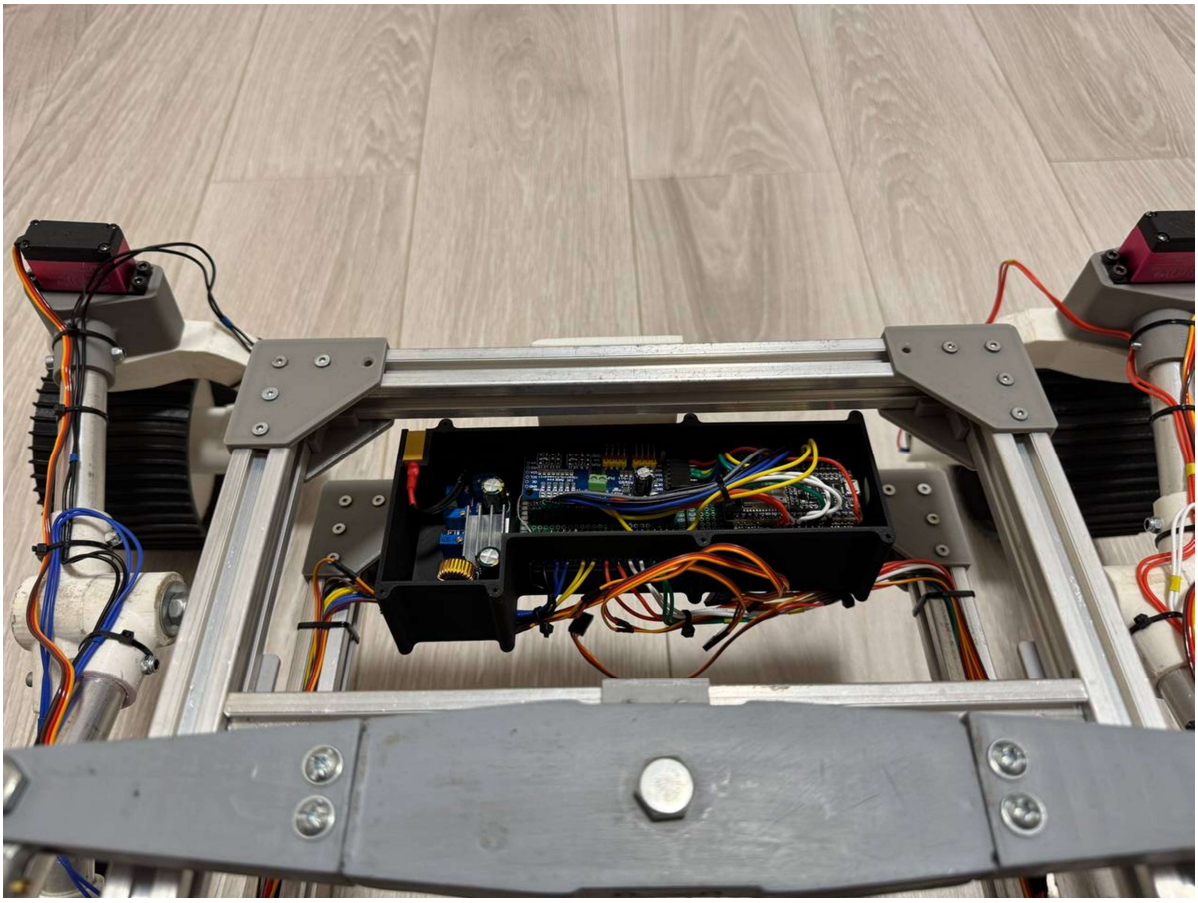


Рисунок 5.3 – Блок керування з підключеними проводами

На наступному етапі проведено комплексне тестування системи керування, яке включало окрему перевірку функціонування кожного компонента та їхньої інтеграції в єдину систему. На рисунках 5.4, 5.5 можна спостерігати тестування логіки повороту коліс з геометрією Акермана.

Також, перевірено роботу двигунів. Для цього було протестовано приводи на різних швидкостях і в різних напрямках. За допомогою мікроконтролера виконано калібрування двигунів для забезпечення точного управління рухом.

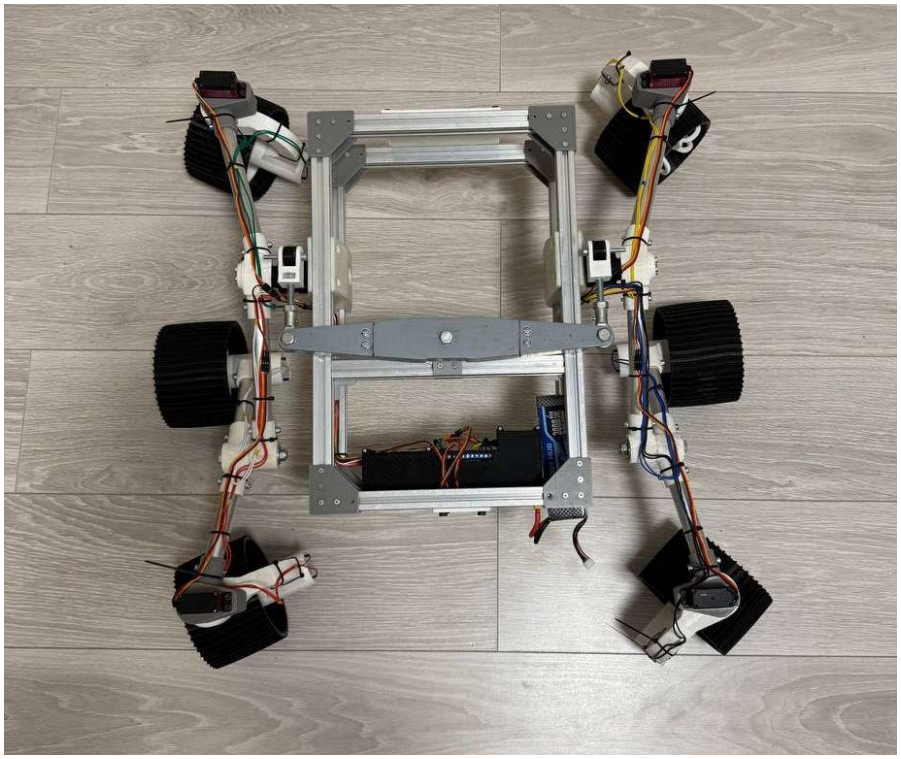


Рисунок 5.4 – Тестування повороту праворуч

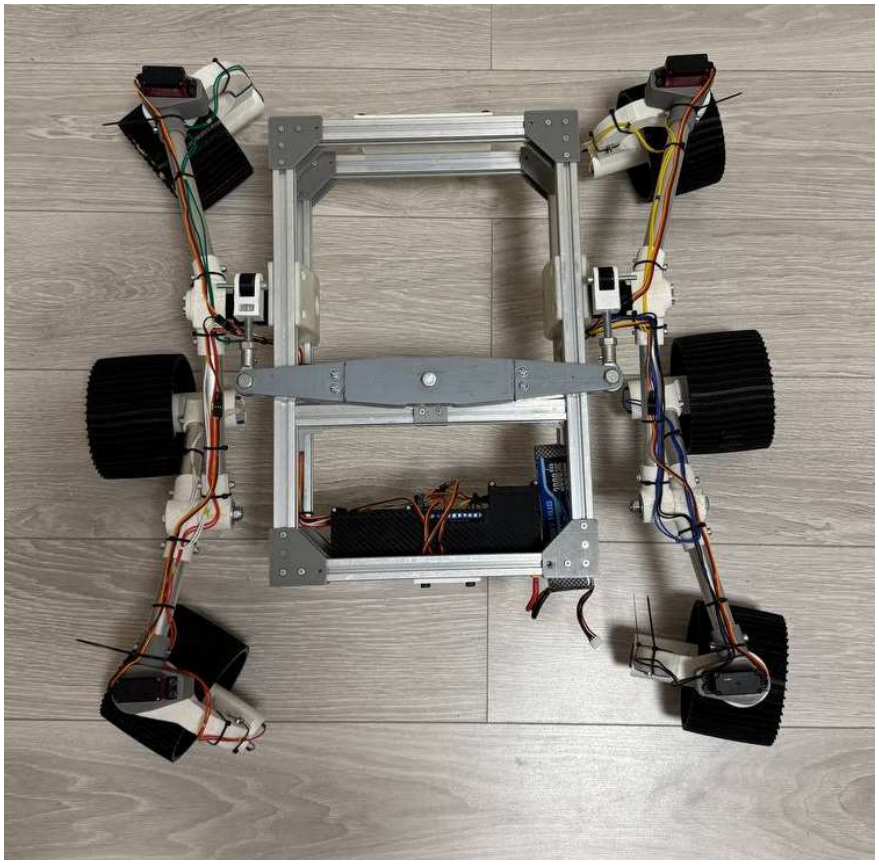


Рисунок 5.4 – Тестування повороту ліворуч

ВИСНОВКИ

У ході виконання магістерської роботи на тему "Система керування моделі марсохода" було успішно розроблено та реалізовано комплексну систему, яка включає всі основні етапи проєктування, моделювання, виготовлення та інтеграції компонентів.

На початковому етапі виконано розробку структурної схеми, яка забезпечила логічну організацію взаємодії між основними компонентами системи. На основі цієї схеми створено блок-схему, що деталізувала функціонування кожного блоку системи. Це дозволило чітко визначити алгоритми роботи та взаємозв’язки між апаратними і програмними компонентами.

Для реалізації системи керування було написано програмний код, що забезпечує повноцінне функціонування моделі марсохода. Код оптимізовано для ефективного використання ресурсів мікроконтролера і враховує особливості використаного обладнання.

Корпус електроніки було спроєктовано за допомогою сучасних засобів комп’ютерного моделювання, що дозволило створити конструкцію з урахуванням ергономічності, захисту компонентів та зручності монтажу. Корпус виготовлено, протестовано і адаптовано до умов експлуатації моделі.

Електронна частина системи була виготовлена на макетній платі, що дозволило забезпечити надійність з’єднань та зручність у проведенні тестувань. Остаточно змонтований блок електроніки інтегровано в модель марсохода, а проведені випробування підтвердили його працездатність і відповідність технічним вимогам.

Таким чином, виконання даної роботи дозволило створити функціональну систему керування для моделі марсохода, яка може бути використана для дослідницьких і навчальних цілей. Робота включала повний цикл розробки від концептуальної ідеї до реального впровадження, що підтверджує її практичну значущість. Отримані результати можуть слугувати основою для подальшого вдосконалення системи або розширення її функціональних можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DIY Mars Perseverance Rover Replica – Режим доступу до ресурсу: https://howtomechatronics.com/projects/diy-mars-perseverance-rover-replica-with-arduino/

2. Mars 2020: Perseverance Rover – Режим доступу до ресурсу: https://science.nasa.gov/mission/mars-2020-perseverance/

3. Rajamani, R. "Vehicle Dynamics and Control" (2nd Edition)

4. 16-канальний 12-bit PWM/Servo модуль з I2C інтерфейсом на PCA9685 [Електронний ресурс] Режим доступу до ресурсу: https://arduino.ua/prod1442-16-kanalnii-12-bit-pwmservo-modyl-s-i2c-interfeisom-na-pca9685

5. Wi-Fi модуль ESP32 LuaNode32 [Електронний ресурс] Режим доступу до ресурсу: https://arduino.ua/prod2041-wi-fi-modyl-esp32-luanode32-38-pin

6. Драйвер двигунів двоканальний на TB6612FNG [Електронний ресурс] Режим доступу до ресурсу: https://arduino.ua/prod2377-draiver-dvigatelei-dvyhkanalnii-na-tb6612fng

7. Понижуючий LED драйвер PWM XL4005 DC-DC [Електронний ресурс] Режим доступу до ресурсу: https://beegreen.com.ua/ponizhuyuchij-led-drajver-pwm-xl4005-dc-dc-5a-08-30v-10989

ДОДАТКИ

МР.ПМКм-11.00.00.000 Е1

					МР.ПМКм-11.00.00.000 Е1	
Зм.	Арк.	№ докум.	Підп	Дата	Система керування моделі марсохода Блок-схема	Літ. / Маса / Масштаб: 1:1
Розроб.		Ліуш Р.Р.				Аркуш / Аркушів
Перев.		Шуляр Б.Г.				
Т. контр.		Шуляр Б.Г.				ПМКм-23-1 ІФНТУНГ
Н. контр.		Шуляр Б.Г.				
Затв.		Панчук В.Г.				

Копіював

Формат А4

					МР.ПМКм-11.00.00.000 БС	
Зм.	Арк.	№ докум.	Підп	Дата	Система керування моделі марсохода Блок-схема	Літ. / Маса / Масштаб: 1:1
Розроб.		Ліуш Р.Р.				
Перев.		Шуляр Б.Г.				Аркуш / Аркушів
Т. контр.		Шуляр Б.Г.				
						ПМКм-23-1 ІФНТУНГ
Н. контр.		Шуляр Б.Г.				
Затв.		Шуляр Б.Г.				

ДОДАТОК В. Система керування моделі марсохода.

Лістинг програмного коду

```
#define BLYNK_TEMPLATE_ID "TMPL4JI3RnWx2"
#define BLYNK_TEMPLATE_NAME "Контроллер"
#define BLYNK_AUTH_TOKEN
"HDlO1vk34uTedswQ40NmP8UxGOEEK3Z5"

#include <SPI.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

#define BLYNK_PRINT Serial

//конфігурування каналів керування
#define SERVO_1 0
#define SERVO_3 1
#define SERVO_4 2
#define SERVO_6 3

#define MOTOR_STBY12 13

#define MOTOR_1 8
#define MOTOR_1_IN1 14
#define MOTOR_1_IN2 12

#define MOTOR_2 9
#define MOTOR_2_IN1 27
```

```
#define MOTOR_2_IN2 26

#define MOTOR_STBY34 25

#define MOTOR_3 10
#define MOTOR_3_IN1 33
#define MOTOR_3_IN2 32

#define MOTOR_4 11
#define MOTOR_4_IN1 0
#define MOTOR_4_IN2 4

#define MOTOR_STBY56 19

#define MOTOR_5 12
#define MOTOR_5_IN1 18
#define MOTOR_5_IN2 5

#define MOTOR_6 13
#define MOTOR_6_IN1 17
#define MOTOR_6_IN2 16


#define SERVO_MIN 80    // Мінімальний сигнал для
сервоприводу (~0 градусів)
#define SERVO_MAX 550   // Максимальний сигнал для
сервоприводу (~180 градусів)

// WiFi credentials
char auth[] = BLYNK_AUTH_TOKEN;
```

```
char ssid[] = "RusLaptuk";
char pass[] = "BabaJuravlja";

// PCA9685 servo driver
Adafruit_PWMServoDriver pwm =
Adafruit_PWMServoDriver();


int XPlot = 90;
int YPlot = 0;

int servo1Angle = 90;
int servo3Angle = 90;
int servo4Angle = 90;
int servo6Angle = 90;
int s = 0; // rover speed
int r = 0; // turning radius
float speed1, speed2, speed3 = 0;
float speed1PWM, speed2PWM, speed3PWM = 0;
float thetaInnerFront, thetaInnerBack,
thetaOuterFront, thetaOuterBack = 0;

float d1 = 271; // distance in mm
float d2 = 278;
float d3 = 301;
float d4 = 304;

void setPWMViaPCA9685(uint8_t servoChannel, int
microseconds) {
```

```
  uint16_t pulseLength = map(microseconds, 0, 180,
SERVO_MIN, SERVO_MAX);
  pwm.setPWM(servoChannel, 0, pulseLength);
}

void setMotorPWMViaPCA9685(uint8_t servoChannel, int
microseconds) {
  pwm.setPWM(servoChannel, 0, microseconds);
}

void setup() {
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  // Ініціалізація PCA9685
  if (!pwm.begin()) {
    Serial.println("Помилка ініціалізації PCA9685!");
    while (true); // Зупинка, якщо ініціалізація не
вдалася
  } else {
    Serial.println("PCA9685 успішно ініціалізовано.");
  }

  pwm.setPWMFreq(50);  // Частота для сервоприводів
(50 Гц)

  setPWMViaPCA9685(SERVO_1, servo1Angle);
  setPWMViaPCA9685(SERVO_3, servo3Angle);
  setPWMViaPCA9685(SERVO_4, servo4Angle);
  setPWMViaPCA9685(SERVO_6, servo6Angle);
```

```
pinMode(MOTOR_STBY12, OUTPUT);
pinMode(MOTOR_STBY34, OUTPUT);
pinMode(MOTOR_STBY56, OUTPUT);
pinMode(MOTOR_1_IN1, OUTPUT);
pinMode(MOTOR_1_IN2, OUTPUT);
pinMode(MOTOR_2_IN1, OUTPUT);
pinMode(MOTOR_2_IN2, OUTPUT);
pinMode(MOTOR_3_IN1, OUTPUT);
pinMode(MOTOR_3_IN2, OUTPUT);
pinMode(MOTOR_4_IN1, OUTPUT);
pinMode(MOTOR_4_IN2, OUTPUT);
pinMode(MOTOR_5_IN1, OUTPUT);
pinMode(MOTOR_5_IN2, OUTPUT);
pinMode(MOTOR_6_IN1, OUTPUT);
pinMode(MOTOR_6_IN2, OUTPUT);


digitalWrite(MOTOR_STBY12, HIGH);
digitalWrite(MOTOR_STBY34, HIGH);
digitalWrite(MOTOR_STBY56, HIGH);
digitalWrite(MOTOR_1_IN1, LOW);
digitalWrite(MOTOR_1_IN2, LOW);
digitalWrite(MOTOR_2_IN1, LOW);
digitalWrite(MOTOR_2_IN2, LOW);
digitalWrite(MOTOR_3_IN1, LOW);
digitalWrite(MOTOR_3_IN2, LOW);
digitalWrite(MOTOR_4_IN1, LOW);
digitalWrite(MOTOR_4_IN2, LOW);
digitalWrite(MOTOR_5_IN1, LOW);
```

```
  digitalWrite(MOTOR_5_IN2, LOW);
  digitalWrite(MOTOR_6_IN1, LOW);
  digitalWrite(MOTOR_6_IN2, LOW);
}


BLYNK_WRITE(V1) {
  XPlot = param.asInt(); // Отримати значення кута з
повзунка (0-180)
}

BLYNK_WRITE(V2) {
  YPlot = param.asInt(); // Отримати значення кута з
повзунка (0-180)
}

void motorStop(){
  digitalWrite(MOTOR_1_IN1, LOW);
  digitalWrite(MOTOR_1_IN2, LOW);
  digitalWrite(MOTOR_2_IN1, LOW);
  digitalWrite(MOTOR_2_IN2, LOW);
  digitalWrite(MOTOR_3_IN1, LOW);
  digitalWrite(MOTOR_3_IN2, LOW);
  digitalWrite(MOTOR_4_IN1, LOW);
  digitalWrite(MOTOR_4_IN2, LOW);
  digitalWrite(MOTOR_5_IN1, LOW);
  digitalWrite(MOTOR_5_IN2, LOW);
  digitalWrite(MOTOR_6_IN1, LOW);
  digitalWrite(MOTOR_6_IN2, LOW);
}
```

```
void calculateServoAngle() {
  // Calculate the angle for each servo for the input
turning radius "r"
  thetaInnerFront = round((atan((d3 / (r + d1)))) *
180 / PI);
  thetaInnerBack = round((atan((d2 / (r + d1)))) * 180
/ PI);
  thetaOuterFront = round((atan((d3 / (r - d1)))) *
180 / PI);
  thetaOuterBack = round((atan((d2 / (r - d1)))) * 180
/ PI);

}

void calculateMotorsSpeed() {
  // if no steering, all wheels speed is the same -
straight move
  if (YPlot > 90 && YPlot < 90) {
    speed1 = speed2 = speed3 = s;
  }
  // when steering, wheels speed depend on the turning
radius value
  else {
    // Outer wheels, furthest wheels from turning
point, have max speed
    // Due to the rover geometry, all three outer
wheels should rotate almost with the same speed. They
differe only 1% so we asume they are the same.
    speed1 = s;
```

```
    // Inner front and back wheels are closer to the
turing point and have lower speeds compared to the
outer speeds
    speed2 = s * sqrt(pow(d3, 2) + pow((r - d1), 2)) /
(r + d4);
    // Inner middle wheel is closest to the turning
point, has the lowest speed
    speed3 = s * (r - d4) / (r + d4);
  }

  // speed value from 0 to 100% to PWM value from 0 to
4095
  speed1PWM = map(round(speed1), 0, 100, 0, 4095);
  speed2PWM = map(round(speed2), 0, 100, 0, 4095);
  speed3PWM = map(round(speed3), 0, 100, 0, 4095);
}

void loop() {
  Blynk.run();

  if (XPlot > 91)   {
    r = map(XPlot, 90, 180, 1400, 600); // turining
radius from 1400mm to 600mm
  }
  // Steering left
  else if (XPlot < 88)   {
    r = map(XPlot, 90, 0, 1400, 600); // turining
radius from 600mm to 1400mm
  }
```

```
  s = map(YPlot, 0, 180, 0, 100); // rover speed from
0% to 100%

  calculateMotorsSpeed();
  calculateServoAngle();

    // Steer right
  if (XPlot > 91) {
    // Servo motors
    // Outer wheels
    setPWMViaPCA9685(SERVO_1, servo1Angle +
thetaInnerFront); // front wheel steer right
    setPWMViaPCA9685(SERVO_3, servo3Angle -
thetaInnerBack); // back wheel steer left for overall
steering to the right of the rover
    // Inner wheels
    setPWMViaPCA9685(SERVO_4, servo4Angle +
thetaOuterFront);
    setPWMViaPCA9685(SERVO_6, servo6Angle -
thetaOuterBack);
    if (YPlot > 90) { // Move forward
      // Motor Wheel 1 - Left Front
      setPWMViaPCA9685(MOTOR_1, speed1PWM); // Outer
wheels running at speed1 - max speed
      digitalWrite(MOTOR_1_IN1, HIGH);
      digitalWrite(MOTOR_1_IN2, LOW);

      // Motor Wheel 2 - Left Middle
      setPWMViaPCA9685(MOTOR_2, speed1PWM);
      digitalWrite(MOTOR_2_IN1, HIGH);
```

```
      digitalWrite(MOTOR_2_IN2, LOW);

      // Motor Wheel 3 - Left Back
      setPWMViaPCA9685(MOTOR_3, speed1PWM);
      digitalWrite(MOTOR_3_IN1, HIGH);
      digitalWrite(MOTOR_3_IN2, LOW);

      // right side motors move in opposite direction
      // Motor Wheel 4 - Right Front
      setPWMViaPCA9685(MOTOR_4, speed2PWM); // Inner
front wheel running at speed2 - lower speed
      digitalWrite(MOTOR_4_IN1, HIGH);
      digitalWrite(MOTOR_4_IN2, LOW);

      // Motor Wheel 5 - Right Middle
      setPWMViaPCA9685(MOTOR_5, speed3PWM); // Inner
middle wheel running at speed3 - lowest speed
      digitalWrite(MOTOR_5_IN1, HIGH);
      digitalWrite(MOTOR_5_IN2, LOW);

      // Motor Wheel 6 - Right Back
      setPWMViaPCA9685(MOTOR_6, speed2PWM); // Inner
back wheel running at speed2 - lower speed
      digitalWrite(MOTOR_6_IN1, HIGH);
      digitalWrite(MOTOR_6_IN2, LOW);
    }
    else if (YPlot < 90) {
      // Motor Wheel 1 - Left Front
      setPWMViaPCA9685(MOTOR_1, speed1PWM); // Outer
wheels running at speed1 - max speed
```

```
        digitalWrite(MOTOR_1_IN1, LOW);
        digitalWrite(MOTOR_1_IN2, HIGH);

        // Motor Wheel 2 - Left Middle
        setPWMViaPCA9685(MOTOR_2, speed1PWM);
        digitalWrite(MOTOR_2_IN1, LOW);
        digitalWrite(MOTOR_2_IN2, HIGH);

        // Motor Wheel 3 - Left Back
        setPWMViaPCA9685(MOTOR_3, speed1PWM);
        digitalWrite(MOTOR_3_IN1, LOW);
        digitalWrite(MOTOR_3_IN2, HIGH);

        // right side motors move in opposite direction
        // Motor Wheel 4 - Right Front
        setPWMViaPCA9685(MOTOR_4, speed2PWM); // Inner
front wheel running at speed2 - lower speed
        digitalWrite(MOTOR_4_IN1, LOW);
        digitalWrite(MOTOR_4_IN2, HIGH);

        // Motor Wheel 5 - Right Middle
        setPWMViaPCA9685(MOTOR_5, speed3PWM); // Inner
middle wheel running at speed3 - lowest speed
        digitalWrite(MOTOR_5_IN1, LOW);
        digitalWrite(MOTOR_5_IN2, HIGH);

        // Motor Wheel 6 - Right Back
        setPWMViaPCA9685(MOTOR_6, speed2PWM); // Inner
back wheel running at speed2 - lower speed
        digitalWrite(MOTOR_6_IN1, LOW);
```

```
      digitalWrite(MOTOR_6_IN2, HIGH);
    }
    else motorStop();
  }

  // Steer left
  else if (XPlot < 89) {

    setPWMViaPCA9685(SERVO_1, servo1Angle -
thetaOuterFront);
    setPWMViaPCA9685(SERVO_3, servo3Angle +
thetaOuterBack);
    setPWMViaPCA9685(SERVO_4, servo4Angle -
thetaInnerFront);
    setPWMViaPCA9685(SERVO_6, servo6Angle +
thetaInnerBack);

    if (YPlot > 90) { // Move forward
      // Motor Wheel 1 - Left Front
      setPWMViaPCA9685(MOTOR_1, speed2PWM);
      digitalWrite(MOTOR_1_IN1, HIGH);
      digitalWrite(MOTOR_1_IN2, LOW);

      // Motor Wheel 2 - Left Middle
      setPWMViaPCA9685(MOTOR_2, speed3PWM);
      digitalWrite(MOTOR_2_IN1, HIGH);
      digitalWrite(MOTOR_2_IN2, LOW);

      // Motor Wheel 3 - Left Back
      setPWMViaPCA9685(MOTOR_3, speed2PWM);
```

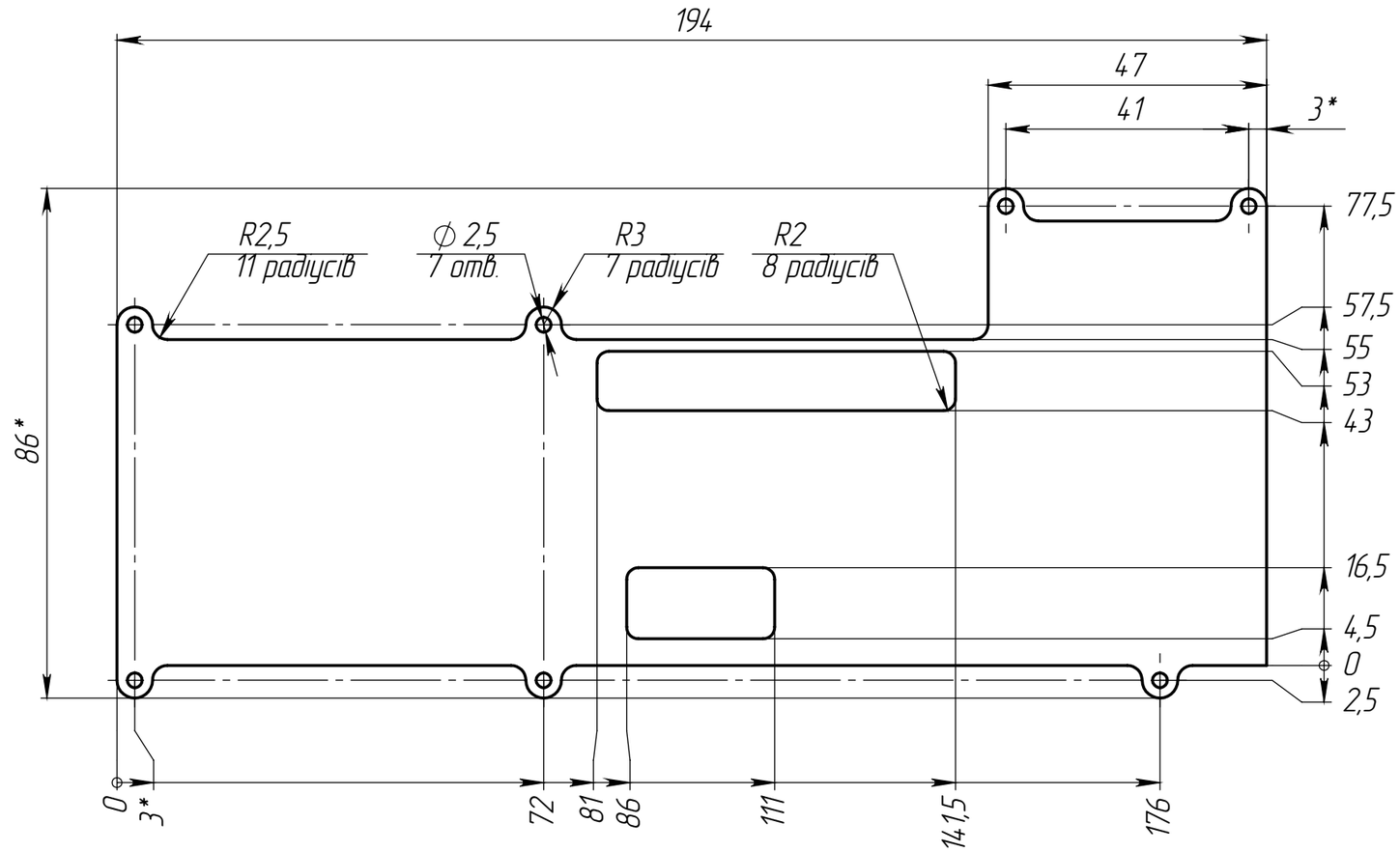
```
    digitalWrite(MOTOR_3_IN1, HIGH);
    digitalWrite(MOTOR_3_IN2, LOW);

    // right side motors move in opposite direction
    // Motor Wheel 4 - Right Front
    setPWMViaPCA9685(MOTOR_4, speed1PWM);
    digitalWrite(MOTOR_4_IN1, HIGH);
    digitalWrite(MOTOR_4_IN2, LOW);

    // Motor Wheel 5 - Right Middle
    setPWMViaPCA9685(MOTOR_5, speed1PWM);
    digitalWrite(MOTOR_5_IN1, HIGH);

    // Motor Wheel 6 - Right Back
    setPWMViaPCA9685(MOTOR_6, speed1PWM);
    digitalWrite(MOTOR_6_IN1, HIGH);
    digitalWrite(MOTOR_6_IN2, LOW);
  }
  else if (YPlot < 90) {
    // Motor Wheel 1 - Left Front
    setPWMViaPCA9685(MOTOR_1, speed2PWM);
    digitalWrite(MOTOR_1_IN1, LOW);
    digitalWrite(MOTOR_1_IN2, HIGH);

    // Motor Wheel 2 - Left Middle
    setPWMViaPCA9685(MOTOR_2, speed3PWM);
    digitalWrite(MOTOR_2_IN1, LOW);
    digitalWrite(MOTOR_2_IN2, HIGH);

    // Motor Wheel 3 - Left Back
```

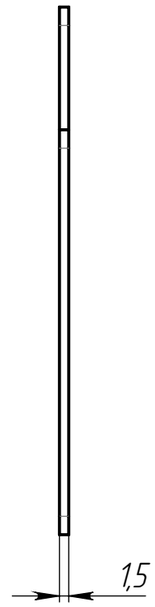
```
      setPWMViaPCA9685(MOTOR_3, speed2PWM);
      digitalWrite(MOTOR_3_IN1, LOW);
      digitalWrite(MOTOR_3_IN2, HIGH);

      // right side motors move in opposite direction
      // Motor Wheel 4 - Right Front
      setPWMViaPCA9685(MOTOR_4, speed1PWM);
      digitalWrite(MOTOR_4_IN1, LOW);
      digitalWrite(MOTOR_4_IN2, HIGH);

      // Motor Wheel 5 - Right Middle
      setPWMViaPCA9685(MOTOR_5, speed1PWM);
      digitalWrite(MOTOR_5_IN1, LOW);
      digitalWrite(MOTOR_5_IN2, HIGH);

      // Motor Wheel 6 - Right Back
      setPWMViaPCA9685(MOTOR_6, speed1PWM);
      digitalWrite(MOTOR_6_IN1, LOW);
      digitalWrite(MOTOR_6_IN2, HIGH);
    }
    else motorStop();
  }
  // Move straight
  else {
    setPWMViaPCA9685(SERVO_1, servo1Angle);
    setPWMViaPCA9685(SERVO_3, servo3Angle);
    setPWMViaPCA9685(SERVO_4, servo4Angle);
    setPWMViaPCA9685(SERVO_6, servo6Angle);

    if (YPlot > 90) { // Move forward
```

```
// Motor Wheel 1 - Left Front
setMotorPWMViaPCA9685(MOTOR_1, speed1PWM);
digitalWrite(MOTOR_1_IN1, HIGH);
digitalWrite(MOTOR_1_IN2, LOW);

// Motor Wheel 2 - Left Middle
setMotorPWMViaPCA9685(MOTOR_2, speed1PWM);
digitalWrite(MOTOR_2_IN1, HIGH);
digitalWrite(MOTOR_2_IN2, LOW);

// Motor Wheel 3 - Left Back
setMotorPWMViaPCA9685(MOTOR_3, speed1PWM);
digitalWrite(MOTOR_3_IN1, HIGH);
digitalWrite(MOTOR_3_IN2, LOW);

// right side motors move in opposite direction
// Motor Wheel 4 - Right Front
setMotorPWMViaPCA9685(MOTOR_4, speed1PWM);
digitalWrite(MOTOR_4_IN1, HIGH);
digitalWrite(MOTOR_4_IN2, LOW);

// Motor Wheel 5 - Right Middle
setMotorPWMViaPCA9685(MOTOR_5, speed1PWM);
digitalWrite(MOTOR_5_IN1, HIGH);
digitalWrite(MOTOR_5_IN2, LOW);

// Motor Wheel 6 - Right Back
setMotorPWMViaPCA9685(MOTOR_6, speed1PWM);
digitalWrite(MOTOR_6_IN1, HIGH);
digitalWrite(MOTOR_6_IN2, LOW);
```

```
}
else if (YPlot < 90) {
  // Motor Wheel 1 - Left Front
  setMotorPWMViaPCA9685(MOTOR_1, speed1PWM);
  digitalWrite(MOTOR_1_IN1, LOW);
  digitalWrite(MOTOR_1_IN2, HIGH);

  // Motor Wheel 2 - Left Middle
  setMotorPWMViaPCA9685(MOTOR_2, speed1PWM);
  digitalWrite(MOTOR_2_IN1, LOW);
  digitalWrite(MOTOR_2_IN2, HIGH);

  // Motor Wheel 3 - Left Back
  setMotorPWMViaPCA9685(MOTOR_3, speed1PWM);
  digitalWrite(MOTOR_3_IN1, LOW);
  digitalWrite(MOTOR_3_IN2, HIGH);

  // right side motors move in opposite direction
  // Motor Wheel 4 - Right Front
  setMotorPWMViaPCA9685(MOTOR_4, speed1PWM);
  digitalWrite(MOTOR_4_IN1, LOW);
  digitalWrite(MOTOR_4_IN2, HIGH);

  // Motor Wheel 5 - Right Middle
  setMotorPWMViaPCA9685(MOTOR_5, speed1PWM);
  digitalWrite(MOTOR_5_IN1, LOW);
  digitalWrite(MOTOR_5_IN2, HIGH);

  // Motor Wheel 6 - Right Back
  setMotorPWMViaPCA9685(MOTOR_6, speed1PWM);
```

```
        digitalWrite(MOTOR_6_IN1, LOW);
        digitalWrite(MOTOR_6_IN2, HIGH);
      }
      else motorStop();
    }

  }
```





1. *Розміри для довідок.
2. Деталь виготовляти за допомогою 3Д друку методом FDM.
3. Граничні відхилення розмірів, форми і розташування поверхонь згідно з ОСТ1 00022-80.

Зм.	Арк.	№ докум.	Підпис	Дата	Кришка	Літ.	Маса	Масштаб
Розроб.		Ліуш Р.Р.					0.017	1:1
Перев.		Шуляр Б.Р.				Аркуш	Аркушів 1	
Т. контр.		Шуляр Б.Р.			Monofilament PLA	ПМКм-23-1 ІФНТУНГ		
Н. контр.		Шуляр Б.Р.						
Затв.		Панчук В.Г.						

Копіював

Формат А3

Перв. застосув.

Довідковий №

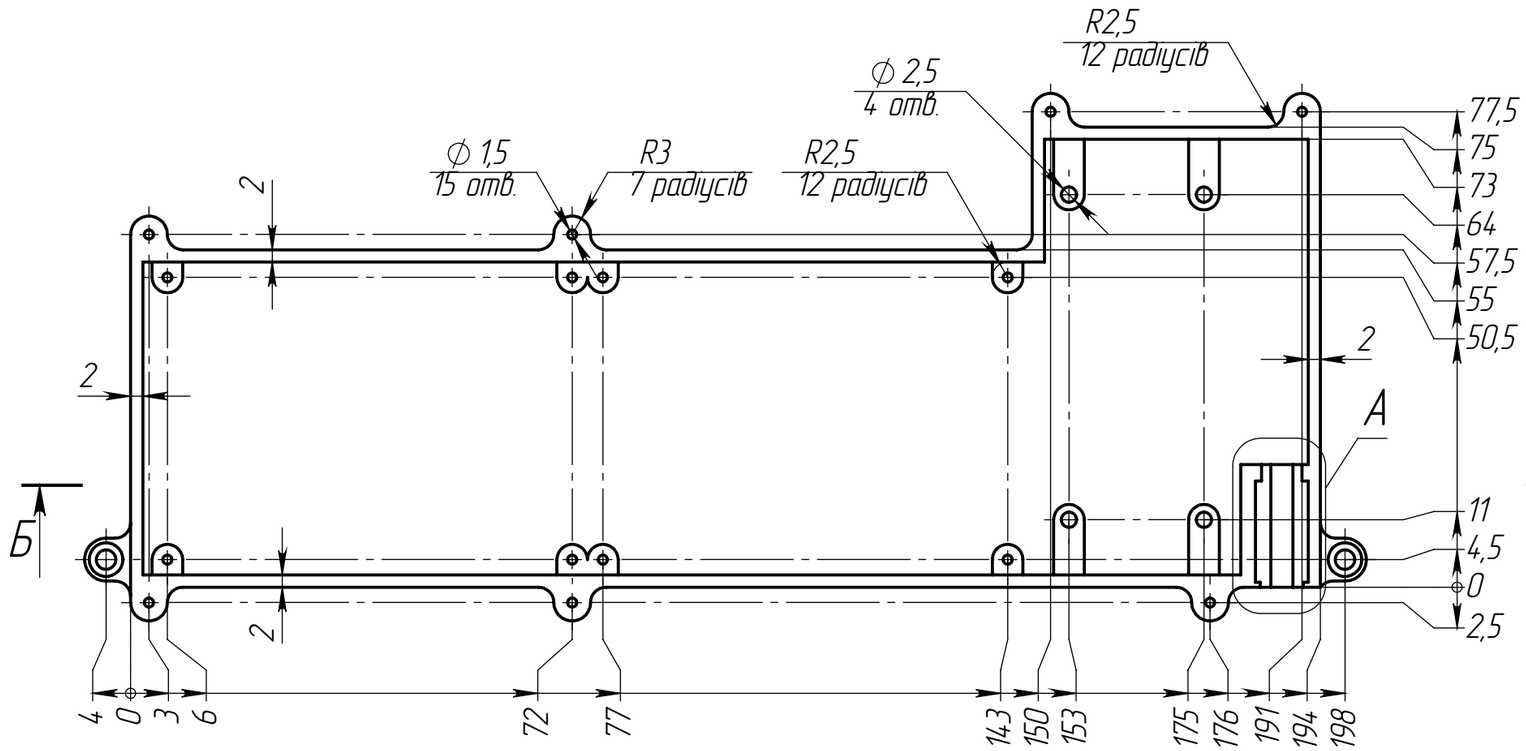
Підпис та дата

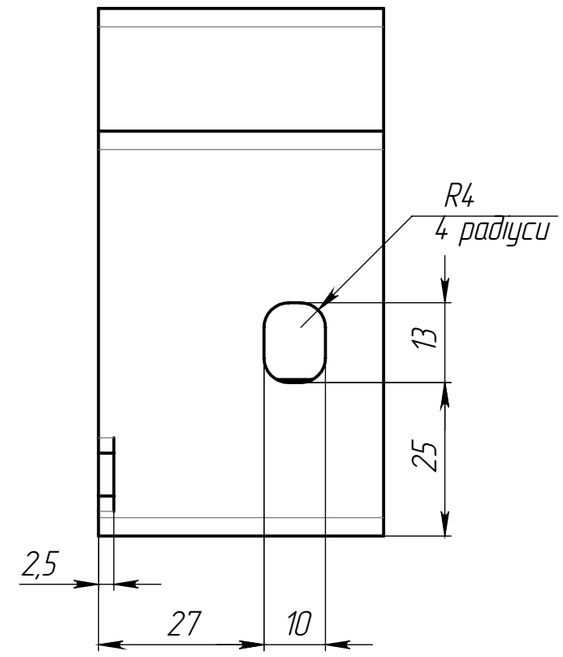
Інв. № дубл.

Замість. інв. №

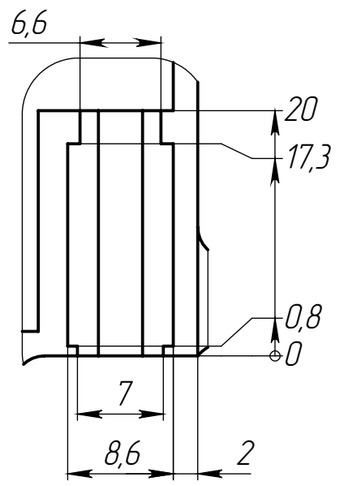
Підпис та дата

Інв. № правдн.





A (2 : 1)



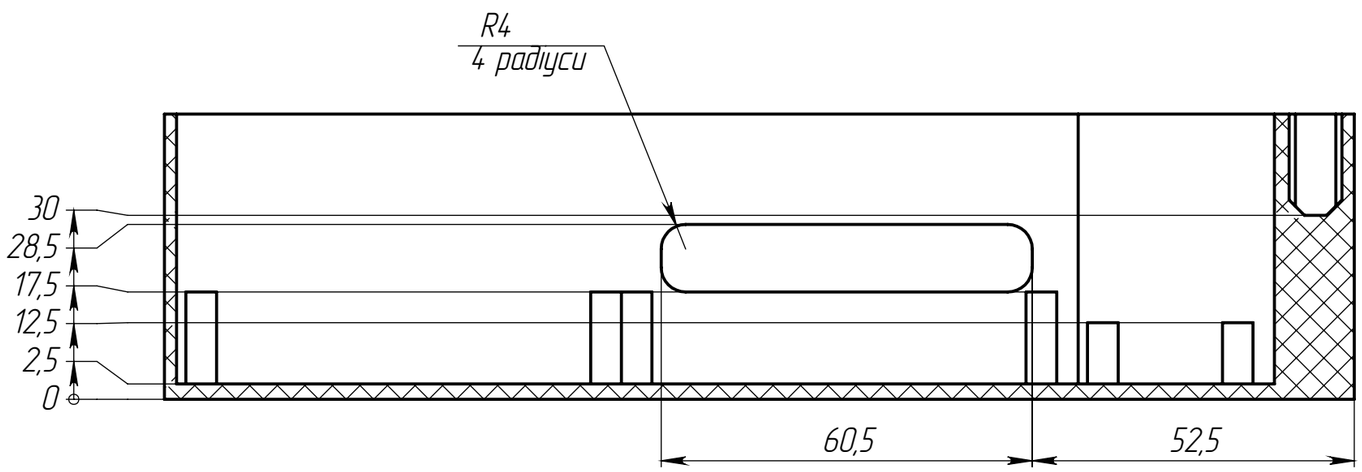
1. *Розміри для довідок.
2. Деталь виготовляти за допомогою 3Д друку методом FDM.
3. Граничні відхилення розмірів, форми і розташування поверхонь згідно з ОСТ1 00022-80.

Зм.	Арк.	№ докум.	Підпис	Дата	Корпус	Літ.	Маса	Масштаб
Розроб.		Ліуш Р.Р.					0.097	1:1
Перев.		Шуляр Б.Р.				Аркуш	Аркушів 2	
Т. контр.		Шуляр Б.Р.			Monofilament PLA	ПМКм-23-1 ІФНТУНГ		
Н. контр.		Шуляр Б.Р.						
Затв.		Панчук В.Г.						

Копіював Формат А3

Перев. застосув.
Довідковий №
Підпис та дата
Інв. № дубл.
Замість інв. №
Підпис та дата
Інв. № правдн.

Б–Б



БІБЛІОГРАФІЧНА ДОВІДКА

Тема: Система керування моделі марсохода

Обсяг ПЗ – 65 аркушів, , 7 використаних джерел.

Перелік креслень графічної частини:

- Структурна схема
- Блок-схема
- Корпус
- Кришка

Виконав: _____ _____

(підпис) (дата)