Івано-Франківський національний університет нафти і газу

Інститут інженерної механіки

Кафедра комп'ютеризованого машинобудування

Бережанський Вадим Вікторович

(прізвище, ім'я, по батькові)

УДК 32.816

(індекс)

БАКАЛАВРСЬКА РОБОТА

Навчально-ігрова мехатронна система "Змагання роботів"

(назва роботи)

Інженерія мехатронних сиситем

(назва освітньої програми)

131- Прикладна механіка

(шифр і назва спеціальності)

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

Здобувач освітнього ступеня Бережанський В.В

(підпис, ініціали та прізвище здобувача)

Науковий керівник Копей В.Б. доктор техн. наук, професор

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

Професор Панчук В.Г
(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

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

Івано-Франківськ
2023

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Інститут Інженерної Механіки

Кафедра Комп'ютеризованого машинобудування

Освітній рівень Бакалавр

Спеціальність 131- Прикладна механіка

(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____
«____» _____20__ року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Бережанському Вадиму Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Навчально-ігрова мехатронна система «Змагання роботів» ,
керівник роботи Копей В.Б. доктор техн. наук, професор ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "___"_____ 20__ року №_____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: література з питань проектування промислових роботів, маніпуляторів, механізмів захоплення.

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

1 ОГЛЯДОВА ЧАСТИНА 1.1 Робототехніка. Історія розвитку 1.2 Роботизованні системи 1.3 Види змагань роботів 1.4 Принципи участі у змаганні роботів 1.5 Огляд симуляторів роботів 1.6 Можливе практичне використання навиків, отриманих у змаганні роботів

2. ПІДБІР ДЕТАЛЕЙ ТА ПРОТОТИПУ РОБОТА 2.1 Підбір елементів для складання прототипу робота 2.2 Прототип робота для змагань

3 МОДЕЛЮВАННЯ ЗМАГАНЬ РОБОТІВ 3.1 Принципи моделювання змагань роботів 3.2 Програма на основі Nodebox і Pymunk для моделювання змагання роботів і відлагодження керуючих програм

4 ПРОЕКТУВАННЯ МОБІЛЬНОГО РОБОТА 4.1 Проектування конструкції. Параметрична модель робота для змагань у SolidWorks 4.2 Керуюча програма робота

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

1. СК робота з захоплювачем – 1 лист А1. 2. СК робота з маніпулятором – 1 лист А1.

3. Схема змагань роботів – 1 лист А1. 4. 3D моделі деталей – 1 лист А1. 5. Принципова електрична схема – 1 лист А1.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Копей В.Б. доктор техн. наук, професор		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання		
2	Збір матеріалу, підготовка оглядової частини		
3	Програма для симуляції змагань роботів		
4	Проектування конструкції мобільного робота		
5	Розробка програми для керування роботом		
6	Оформлення пояснювальної записки і графічної частини		
7	Відгук керівника дипломного проєкту		
8	Рецензування і перевірка на плагіат		
9	Захист дипломної роботи		

Студент _____ __Бережанський В.В_
(підпис) (прізвище та ініціали)

Керівник роботи _____ _Копей В.Б._
(підпис) (прізвище та ініціали)

Анотація

Бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра на тему: «Навчально-ігрова мехатронна система «Змагання роботів»».

Бакалаврська робота складається з пояснювальної записки на 85 аркушах формату А4 з додатками, містить 58 рисунків та 5 таблиць. Графічна частина проєкту містить 5 креслень формату А1.

Виконано аналіз робототехнічних платформ, видів змагань роботів та симуляторів роботів. Підібрано прототип і компоненти робота для змагань роботів, в яких роботи намагаються ідентифікувати об’єкт і виштовхнути його за межі кола. Розроблена Python-програма на основі Nodebox for OpenGL і Pymunk для моделювання змагання роботів і відлагодження керуючих програм. За допомогою SOLIDWORKS побудовано параметричну модель робота з механізмом захоплення для змагань роботів. Розроблено керуючу програма робота на основі Python і Firmata.

Ключові слова: Arduino, мехатроніка, програмування, змагання роботів, робототехніка, система керування, тривимірне моделювання.

Annotation

Bachelor's thesis for the educational and qualification level of a bachelor on the topic: "Educational and gaming mechatronic system "Competitions of robots"".

Bachelor's work consists of an explanatory note on 85 sheets of A4 format with applications, contains 58 figures and 5 tables. The graphic part of the project contains 5 A1 format drawings.

The analysis of robotic platforms, types of robot competitions and robot simulators was carried out. The prototype and components of the robot were selected for robot competitions, in which the robots try to identify an object and push it out of the circle. A Python program based on Nodebox for OpenGL and Pymunk has been developed for simulating robot competition and debugging control programs. With the help of SOLIDWORKS, a parametric model of robot for robot competitions with a gripping mechanism was built. A robot control program based on Python and Firmata has been developed.

Keywords: Arduino, mechatronics, Python, robot competition, robotics, control system, 3D modeling.

Зміст

ВСТУП 7
1 ОГЛЯДОВА ЧАСТИНА 9
1.1 Робототехніка. Історія розвитку 9
1.2 Роботизованні системи 10
1.2.1 Платформа TETRIX 11
1.2.2 Платформа Mindstorms Education EV3 WeDo 11
1.2.3 Платформа VEX IQ, VEX EDR 13
1.2.4 Платформа Fischer technik 14
1.2.5 Платформа Huna MRT 15
1.2.6 Платформа Bioloid 16
1.2.7 Платформа Arduino 17
1.3 Види змагань роботів 18
1.4 Перші змагання з робототехніки Noosphere Engineering Race в Україні 20
1.5 Клуб робототехніки та змагання роботів 21
1.6 Принципи участі у змаганні роботів 22
1.7 Огляд симуляторів роботів 23
1.8 Можливе практичне використання навиків, отриманих у змаганні роботів 25
2. ПІДБІР ДЕТАЛЕЙ ТА ПРОТОТИПУ РОБОТА 27
2.1 Підбір елементів для складання прототипу робота 27
2.2 Прототип робота для змагань 37
3 МОДЕЛЮВАННЯ ЗМАГАНЬ РОБОТІВ 40
3.1 Принципи моделювання змагань роботів 40
3.2 Програма на основі Nodebox і Pymunk для моделювання змагання роботів і відлагодження керуючих програм 42
4 ПРОЕКТУВАННЯ МОБІЛЬНОГО РОБОТА 50
4.1 Параметрична модель робота для змагань у SolidWorks 50
4.2 Керуюча програма робота 62
ВИСНОВКИ 71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 73
Додаток А – Принципова електрична схема робота 75
Додаток Б – Схема сканування ультразвуковим датчиком HC-SR04 76
Додаток В – Програма для моделювання змагань роботів 77
Додаток Г – Керуюча програма робота 81

ВСТУП

Будь-які змагання роботів дозволяють студентам підвищити мотивацію до вивчення мехатроніки, конструювання роботів і програмування. В роботі удосконалено навчальну мехатронну систему для змагань мобільних роботів на основі платформи Arduino і мови програмування Python.

Правила змагань роботів наступні:

1.У змаганнях беруть участь мобільні роботи, які зібрані і запрограмовані учасниками змагань і керуються бездротовим методом. Кожний учасник може тримати у таємниці алгоритм робота.

2.Роботи змагаються в межах кола з визначеним організаторами радіусом.

3.Робот повинен ідентифікувати світлий циліндричний об’єкт розміром 100х100 мм, який є тільки один, і виштовхнути його за межі кола. В цьому випадку учаснику нараховується один бал.

4.Якщо робот виштовхує за межі кола темний об’єкт, яких усього 10, з учасника знімається один бал.

5.Перемагає учасник, який набрав найбільшу кількість балів.

6.Допустимі параметри роботів і рівень складності змагань визначаються організаторами. Можуть бути такі рівні складності:

I (простий) робот працює частково автономно. Для запобігання зіткнення роботів, або виїзду за межі може бути застосоване ручне керування.

II (середній) робот працює автономно. Для запобігання зіткнення роботів, або виїзду за межі їм передається загальна інформація про розташування роботів.

III (складний) – робот працює повністю автономно.

Метою роботи є розроблення навчальної мехатронної системи для змагань мобільних роботів за описаними вище правилами із використанням першого рівня складності. Для досягнення мети потрібно розв’язати наступні задачі:

1.Виконати огляд сучасного стану робототехніки і платформ для розроблення роботів.

2.Виконати огляд відомих змагань роботів і принципів участі в цих змаганнях.

3.Виконати огляд принципів моделювання і симуляторів роботів.

4.Підібрати прототип робота для змагань і його деталі.

5.Розробити програму для симуляції змагання роботів і відлагодження керуючих програм.

6.Спроектувати конструкцію робота для змагань.

7.Розробити і відлагодити керуючу програму робота для змагань.

1 ОГЛЯДОВА ЧАСТИНА

1.1 Робототехніка. Історія розвитку

Сьогодні нам важко уявити своє життя без роботів і різних автоматичних пристроїв, які замінили людини в самих різних сферах промисловості і повсякденному житті. Незважаючи на відносно недовгу історію робототехніки, вона міцно увійшла в наше життя, замінивши людини в найбільш небезпечних умовах роботи.

Робототехніка – це прикладна наука, яка опікується проєктуванням, розробкою, будівництвом, експлуатацією та використанням роботів, а також систем для їх контролю.

Робототехніка базується на створенні роботів для автоматизації технічних операцій і процесів, для заміни людини під час виконання небезпечної, важкої і довго тривалої роботи.

Робототехніка поділяється на такі сфери використання:

– будівельна;
– промислова;
– побутова;
– авіаційна;
– екстремальна (військова, космічна, підводна).

За даними Національної асоціації учасників ринку робототехніки, у світі на 10 тисяч працівників, 2015 року доводилося у середньому 69 промислових роботів.

Розвиток робототехніки бере свій початок ще з давній часів. Першими роботами можна вважати механізми які допомагали людині збільшувати силу й швидкість переміщення. Такі механізми приводились в рух примітивно за допомогою: води, пари, зубчастих коліс. Одним з найвідоміших прикладів можна вважати Архімедів гвинт.

Слово робототехніка походить від слова робот, яке було представлено публіці чеським письменником Карелом Чапеком у його п'єсі R.U.R. (Россумські

Універсальні Роботи) 1920 року. Саме Айзек Азімов вперше сформував три закони робототехніки:

– робот не може заподіяти шкоду людині, або своєю бездіяльністю дозволити, щоб людині була заподіяна шкода;

– робот повинен підкорятися наказам людини, за винятком тих, котрі суперечать першому пункту;

– робот повинен захищати самого себе, якщо тільки його дії не суперечать першому і другому пунктам [1].

1.2 Роботизованні системи

Незважаючи на те, що освітня робототехніка - це напрямок відносно новий, розвивається вона стрімкими темпами. Розробкою робототехнічних платформ для освіти займаються десятки зарубіжних компаній. Всі виробники освітньої робототехніки тісно співпрацюють з кращими фахівцями освітньої галузі з усього світу для виявлення основних проблем освітнього процесу. Основне завдання всіх платформ освітньої робототехніки – це підвищення мотивації та підготовки компетентних фахівців, здатних вирішувати складні інженерно-технічні завдання. Різні платформи вирішують цю задачу по-різному. Основні робототехнічні платформи світу наведені в таблиці 1.1.

Таблиця 1.1- Робототехнічні платформи

№	Виробник	Країна	Назва платформи
1	Pitsco	США	TETRIX
2	Lego	Данія	Mindstorms Education EV3 WeDo
3	VEXRobotics	США	VEX IQ, VEX EDR
4	FischerTechnik	Германія	Fischer technik
5	MRT International	Південна Корея	Huna MRT
6	Robotis	Південна Корея	Bioloid
7	Arduino Software	Італія	Arduino

1.2.1 Платформа TETRIX

Основними конструкційними елементами TETRIX MAX є алюмінієвий канал з маркуванням отворів з товарним знаком, який дозволяє з'єднання з кратним 45°. Деталі та складні одиниці кріпляться за допомогою болтів. Серед інших будівельних елементів можна візначити шестерні, колеса, двигуни і серводвигуни (рисунок 1.1).

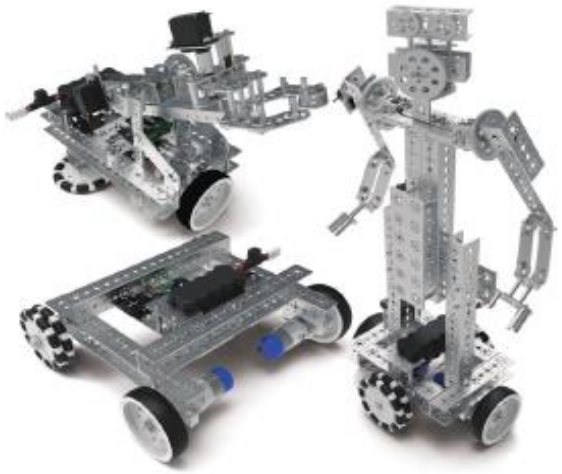


Рисунок 1.1 - Платформа TETRIX MAX

TETRIX MAX може взаємодіяти з комплектом NXT Lego Mindstorms, щоб інтелектуальний блок NXT контролював двигуни і сервоприводи TETRIX. Зв'язок не тільки механічний, а й організаційний: в США LEGO Education North America є спільним підприємством Pitsco, Inc. і освітнього підрозділу LEGO Group.

TETRIX PRIME був розроблений для використання в середній школі, але також може використовуватися з компонентами TETRIX MAX. Ці системи також можуть використовуватися з контролерами R / C або різними або програмованими параметрами, такими як myRIO і Arduino.

1.2.2 Платформа Mindstorms Education EV3 WeDo

Чималу роль серед навчальних роботів в даний час мають Лего-конструктори. Серед роботів серії Lego виділяють наступні навчальні роботи:

- Lego WeDo;

- Lego Mindstorms.

Освітні рішення LEGO® Education WeDo 2.0 (рисунок 1.2) забезпечують міцний зв'язок між нудною теорією з курсу навколишнього світу і технології з реальним світом за допомогою практичних завдань, цікавих проєктних робіт і сучасних технологій.

Рисунок 1.2 - LEGO® Education WeDo 2.0

Проєктна діяльність формує у дітей знання, вміння і навички в області технології, фізики, технічних і природничо-наукових дисциплін, а також інформатики. Ці сучасні освітні рішення являють собою унікальне поєднання кубиків LEGO, програмного забезпечення, цікавих, що відповідають освітнім стандартам науково-дослідних проєктів. Все це допомагає дітям розвинути не тільки ключові компетенції XXI століття, а й навички ведення науково-дослідної діяльності, а також впевненість у своїх силах і знаннях. Учні мають інструмент, за допомогою якого вони навчитися ставити запитання, формулювати завдання і розробляти власні рішення, тому що радість наукового відкриття виявиться в їх власних руках.

Моделювання, конструювання та випробування роботів, здатних виконувати складні команди, реєструвати дані в ході спостережень, реагувати на зміни зовнішніх умов – це спосіб викладати математику простіше і наочніше.

Рисунок 1.3 - LEGO MINDSTORMS Education

Набори LEGO MINDSTORMS Education (рисунок 1.3) оживляють технологію, фізику і обчислення за допомогою наочно-практичного навчання, в основу якого покладено практико-орієнтовані проєктні завдання з використанням кращих робото технічних рішень [2].

1.2.3 Платформа VEX IQ, VEX EDR

Набори VEX EDR (рисунок 1.4) складаються з перфорованих металевих елементів. Відмінною особливістю даної лінійки є наявність в мікроконтролері портів аналогового і цифрового типу. Це дозволяє використовувати конструктор для вивчення базових елементів мікроелектроніки.

Рисунок 1.4 - Платформа VEX EDR

Програмне забезпечення для EDR представлено у вигляді трьох пакетів: MPLAB, easyC і ROBOTC. На практиці ж в якості основної платформи використовують, як правило, тільки останню, найбільш універсальну середу ROBOTC.

1.2.4 Платформа Fischer technik

Найбільшого поширення конструктор FISHERTECHIK (рисунок 1.5) отримав в професійних навчальних закладах і технічних ВНЗ. Набори комплектуються фірмовими контролерами, двигунами, датчиками і блоками живлення. Кожна версія конструктора містить в коробці досить об'ємний блок-контролер з пазами і виступом типу «ластівчин хвіст».

Рисунок 1.5 - Конструктор FISHERTECHIK

Конструктори FISHERTECHIK використовуються в різних країнах усього світу для демонстрації принципів роботи механізмів і машин в середніх, спеціальних і вищих навчальних закладах, а також для моделювання виробничих процесів і презентаційних цілей.

Основним елементом цих конструкторів є блок з пазами і виступом типу «ластівчин хвіст». Така форма дає можливість з'єднувати елементи практично в будь-яких комбінаціях. Також в комплекти конструкторів входять програмовані контролери, двигуни, різні датчики і блоки живлення, що дозволяє приводити механічні конструкції в рух, створювати роботів і програмувати їх за допомогою

комп'ютера. Для розробки керуючих програм для контролера ROBO TX використовується середу візуального програмування ROBO Pro. Програми завантажуються в контролер через інтерфейси USB або Bluetooth.

1.2.5 Платформа Huna MRT

HUNA-MRT для початківців (рисунок 1.6) - це набори серії FUN & BOT і KICKY (MRT2). Всі деталі конструкторів пластмасові, яскраві, електроніки мінімум. Це попередній, що не програмований етап знайомства з робототехнікою для дітей 6-8 років. Набори навчають основам конструювання, простим механізмам і з'єднанням.

Рисунок 1.6 - Конструктор HUMA MRT

Роботи цього рівня не програмуються і це плюс для дітей дошкільного віку діти отримують швидкий результат своєї роботи, не витрачаючи час на розробку алгоритму та написання програми. При цьому конструктори включають електронні елементи: датчики, мотори, пульт управління - все це дозволяє вивчити основи робототехніки.

Набори супроводжуються докладними інструкціями і методичними матеріалами.

Весь матеріал викладено в ігровій формі - це казки, розповіді, приклади з навколишнього життя.

1.2.6 Платформа Bioloid

Robotis Bioloid (рисунок 1.7) набір для створення робота, вироблений корейською

фірмою Robotis Набор призначений для освітніх цілей а також для тих, хто захоплюється робототехнікою. Набір Bioloid включає в себе невеликі серводвигун – Dynamixels і є самостійним модулем, за допомогою якого можуть бути зібрані роботи різної конструкції, наприклад, колісні або крокуючі роботи.

Рисунок 1.7 – Robotis Bioloid

Набір Bioloid схожий з наборами LEGO Mindstorms від компанії LEGO і VexRobotics Design System від компанії VEX Robotics. Набір використовується в Військово-морський академії США як навчальне обладнання в курсі машинобудування. Так само набір Bioloid часто використовують учасники міжнародних змагань RoboCup.

У комплект Bioloid входять серводвигуни Dynamixels, набір сенсорів, програмне забезпечення, що включає себе середу 3D моделювання і середовище програмування на C-подібному мовою. Кількість приводів достатня, щоб виготовити механізм з вісімнадцятьма ступенями свободи [3].

1.2.7 Платформа Arduino

Arduino - найпоширеніша платформа для дорослої робототехніки та електроніки, друга за поширеністю серед дітей.

Це унікальний електронний конструктор, всесвітньо відома програма, яка використовується для отримання базових знань в області мікроелектроніки.

За своєю суттю Arduino - це міні-комп'ютер, який дозволяє писати програми, а вже з їх допомогою успішно управляти різними електронними пристроями, починаючи від світлодіодних конструкцій, закінчуючи робототехнікою.

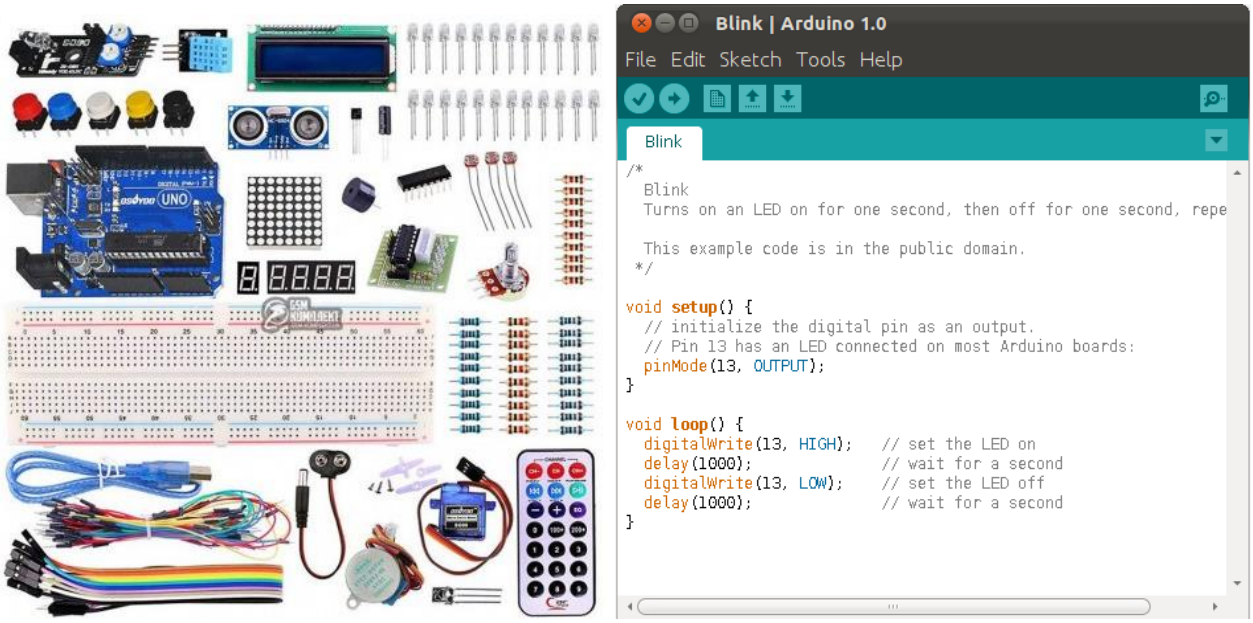


Рисунок 1.8 – Платформа Arduino

Використання Arduino в загальноосвітніх закладах допомагають дітям у створенні нескладних або великих і масштабних електронних проєктів. Хлопці зможуть створювати корисні для будинку пристрої, роботів, елементи для розумного будинку, пристрої, які будуть виконувати різні функції. Дітям належить отримати знання візуального програмування Scratch for Arduino і освоїти мову програмування С ++.

Arduino - потужний процесор, робота з яким полегшена тим, що є можливість підключати периферійні пристрої: датчики температури, освітлення, прискорення, ультразвуку, тиску тощо. Можуть бути використані також мотори і серводвигун, модулі управління Bluetooth або за допомогою Інтернету. Щоб всі

зазначені периферійні пристрої нормально взаємодіяли між собою, на мікроконтролер записуються алгоритми роботи і взаємодії [4].

1.3 Види змагань роботів

Існує багато типів змагань роботів, кожен зі своїми правилами та цілями. Ось кілька прикладів:

Робототехніка сумо: у цьому змаганні роботи намагаються виштовхнути один одного з рингу або арени. Виграє робот, який довше пробуде в рингу.

Роботи, що слідують за лінією: у цьому змаганні роботи повинні слідувати за лінією на трасі, керуючи поворотами, щоб дістатися до фінішу.

Роботи, які розгадують лабіринт: у цьому змаганні роботи повинні пройти лабіринт, щоб знайти кінець. Перемагає робот, який найшвидше пройде лабіринт.

Робот-футбол: це змагання, де дві команди роботів грають у футбол одна проти одної. Роботи повинні вміти орієнтуватися на полі, уникати перешкод і забивати голи.

Робототехнічний порятунок: у цьому змаганні роботи призначені для навігації в симульованому сценарії лиха, наприклад землетрусу чи пожежі, щоб знаходити та рятувати жертв. Перегони безпілотників: це змагання, де пілоти змагаються зі своїми безпілотниками через смугу перешкод, часто в окулярах від першої особи, щоб побачити, що бачить дрон.

Бойові роботи: у цьому змаганні роботи призначені для боротьби один з одним, часто за допомогою такої зброї, як пилки, молотки або вогнемети. Перемагає останній робот.

Танці роботів: у цьому змаганні роботи запрограмовані танцювати під музику, а судді оцінюють роботів на основі їх креативності та продуктивності.

Це лише кілька прикладів багатьох типів змагань роботів, які існують. Кожне змагання має свої унікальні правила та завдання, і всі вони вимагають поєднання інженерії, програмування та креативності, щоб досягти успіху.

Є кілька відомих і престижних змагань роботів, які широко визнані в спільноті робототехніки. Ось кілька прикладів:

FIRST Robotics Competition: це щорічне змагання для учнів середньої школи, яке кидає виклик командам розробити та створити роботів, щоб змагатися в інженерних змаганнях.

RoboCup: це міжнародне змагання, яке фокусується на розробці автономних роботів, які можуть грати у футбол. Мета конкурсу — розробити команду повністю автономних гуманоїдних роботів, які зможуть перемогти чемпіонів світу з футболу серед людей до 2050 року.

DARPA Robotics Challenge: це було змагання, організоване Агентством передових оборонних дослідницьких проектів (DARPA), яке кидало виклик командам розробити роботів, здатних реагувати на стихійні лиха та надзвичайні ситуації.

VEX Robotics Competition: це змагання, яке кидає завдання командам студентів розробити, створити та запрограмувати роботів для змагання в інженерних змаганнях.

Міжнародний конкурс повітряної робототехніки (IARC): це змагання, яке ставить перед командами завдання розробити автономних повітряних роботів, здатних виконувати складні завдання, наприклад ідентифікувати об’єкти та взаємодіяти з ними.

Amazon Robotics Challenge: це було змагання, організоване Amazon, яке кидало виклик командам розробити роботів, які могли б автономно збирати та пакувати товари на складі.

Ці змагання є надзвичайно конкурентоспроможними та залучають найталановитіших та інноваційних робототехніків з усього світу. Перемога в цих змаганнях є великим досягненням і може створити додаткові можливості в галузі робототехніки [5].

1.4 Перші змагання з робототехніки Noosphere Engineering Race в Україні

За словами організаторів події «Асоціації Ноосфера», NER — це змагання принципово нового формату. Вони об'єднують у собі елементи навчання та, власне, змагання. Учасники — а це діти та підлітки віком від 7 до 16 років, пройшли навчання починаючи з основ робототехніки та знайомства з конструктором до програмування на VEX та розуміння всіх циклів та послідовностей. Таким чином, у змаганнях беруть участь навіть ті діти та підлітки, які не мають наразі особливих знань з робототехніки та доступу до конструктора VEX. Провідні експерти-практики надають учасникам необхідне програмне забезпечення, проводять майстер-класи та вебінари з робототехніки VEX.

«ГО "Асоціація Ноосфера" прагне розвивати робототехнічний напрямок в Україні, залучати до цього руху талановитих школярів, мотивувати їх отримувати нові знання. Віримо, що в майбутньому ці талановиті робототехніки стануть рушіями інноваційних інженерних ідей та приборкувачами усіх роботів світу[7].

Рисунок 1.9 – Змагання Noosphere Engineering Race

1.5 Клуб робототехніки та змагання роботів

1 грудня, в перший день зими в легкій та невимушеній обстановці в коледжі стартував Клуб Робототехніки, що об’єднав студентів-однодумців, захоплених ідеями робототехніки.

З привітальним словом виступив директор Василь Береговський. Викладачі коледжу Олександр Аронець, Михайло Бідасюк та Олександр Іванов дали можливість поринути у світ роботів та окреслили основні напрямки роботи клубу, Володимир Лісафін наголосив на перспективах використання роботів у нафтогазовій галузі. Студенти мали змогу побачити та самостійно керувати мобільними роботами, роботами-андроїдами та роботами-маніпуляторами.

Учасникам клубу відкривається можливість розробляти, складати, програмувати та вдосконалювати найрізноманітніших роботів та систем автоматики.

Клуб стане основою для створення та реалізації проєктів з програмування, автоматики, механіки та електроніки.

Також у день відкриття, учасники провели конкурс змагання роботів. Студенти мали можливість випробувати свої вміння у програмуванні та керуванні мобільними роботами[8].

Рисунок 1.10 – Змагання в клубі робототехніки

1.6 Принципи участі у змаганні роботів

Участь у змаганні, де роботи ідентифікують предмет і виштовхують його з кола, може бути захоплюючою справою. Хоча конкретний процес може відрізнятися залежно від конкурсу, який вас цікавить, ось кілька загальних кроків, які допоможуть вам розпочати:

Дослідницькі змагання: шукайте змагання роботів, які зосереджуються на ідентифікації об’єктів і завданнях штовхання. У всьому світі проводяться різні змагання з робототехніки, наприклад RoboCup, FIRST Robotics Competition, або місцеві заходи, організовані університетами, клубами робототехніки чи галузевими групами. Перегляньте їхні веб-сайти або зверніться до організаторів, щоб дізнатися, чи пропонують вони певний конкурс, який вас цікавить.

Зрозумійте правила: прочитайте та ознайомтеся з правилами та рекомендаціями конкурсу. Звертайте пильну увагу на вимоги щодо ідентифікації об’єкта, розмір і форму кола, правила штовхання об’єкта та будь-які інші спеціальні вказівки, які застосовуються. Переконайтеся, що ви розумієте критерії перемоги, підрахунок балів і будь-які накладені обмеження.

Зберіть команду: змагання роботів часто включають командну роботу. Знайдіть однодумців із додатковими навичками, такими як програмування, електроніка, машинобудування та дизайн. Створення сильної команди збільшить ваші шанси на успіх і дозволить вам ефективно розподіляти завдання.

Спроектуйте та створіть свого робота. Коли у вас буде команда, продумайте мозковий штурм і розробіть робота, який зможе ефективно ідентифікувати об’єкти та виштовхувати їх із кола. Враховуйте такі фактори, як мобільність, датчики (наприклад, камери, лідар або датчики наближення), механізми захоплення та загальну стратегію. Залежно від конкуренції у вас можуть бути певні обмеження щодо розміру, ваги або джерела живлення робота, тому переконайтеся, що ваш дизайн відповідає цим обмеженням.

Розробіть алгоритми розпізнавання об’єктів і штовхання: запровадьте алгоритми, які дозволять вашому роботу точно ідентифікувати об’єкти в колі. Це

може включати методи комп’ютерного зору, машинне навчання або інші методи виявлення та класифікації об’єктів. Крім того, запрограмуйте рухи робота та механізми штовхання для ефективної навігації та штовхання об’єктів за межі кола.

Тестуйте та повторюйте: проведіть широке тестування, щоб точно налаштувати продуктивність вашого робота. Створіть контрольоване середовище, яке максимально точно імітує умови змагань. Повторюйте свій дизайн, алгоритми та стратегії на основі результатів тестування. Зосередьтеся на підвищенні точності, швидкості та надійності.

Підготуйте документацію: багато змагань вимагають від учасників надати документацію з детальним описом конструкції, алгоритмів і процесу розробки свого робота. Створіть вичерпну документацію, яка описує технічні аспекти вашого робота, включаючи обґрунтування конструкції, архітектуру програмного забезпечення та будь-які унікальні підходи чи інновації.

Зареєструйтеся та беріть участь: коли ви відчуєте впевненість у можливостях свого робота, зареєструйтеся на змагання згідно з інструкціями організатора. Зверніть увагу на терміни реєстрації та вимоги. Переконайтеся, що у вас є все необхідне обладнання, запасні частини та інструменти для дня змагань.

Співпрацюйте та спілкуйтеся: взаємодійте з іншими учасниками, командами та організаторами під час змагань. Спілкування з колегами-ентузіастами-робототехніками може призвести до цінного обміну знаннями, можливостей для співпраці та дружби в спільноті робототехніки.

Пам’ятайте, що участь у змаганнях роботів – це не лише перемога, а й навчання, набуття досвіду та задоволення. Приймайте виклики та насолоджуйтеся проектуванням, будівництвом і змаганням із вашим роботом.

1.7 Огляд симуляторів роботів

Створення симуляторів для змагань роботів може бути цінним інструментом для тестування та вдосконалення продуктивності вашого робота перед участю у справжньому змаганні. Ось деякі засоби та підходи, які слід враховувати під час розробки симуляторів для змагань роботів:

Симулятори на основі фізики: симулятори на основі фізики забезпечують реалістичне уявлення про фізичні взаємодії та динаміку в середовищі змагань. Ці симулятори імітують закони фізики, дозволяючи вам точно моделювати поведінку вашого робота та об’єктів. Деякі популярні симулятори на основі фізики включають Gazebo, Webots і MuJoCo.

Ігрові механізми. Ігрові механізми пропонують потужні інструменти для створення віртуальних середовищ і імітації змагань роботів. Unity3D і Unreal Engine є популярними ігровими движками, які забезпечують широке моделювання фізики, реалістичну графіку та можливості сценаріїв. Вони дозволяють створювати інтерактивні 3D-середовища, де можна програмувати та перевіряти продуктивність свого робота.

Симулятори Robotics Framework: багато робототехнічних фреймворків, таких як ROS (Robot Operating System), мають вбудовані симулятори, які дозволяють моделювати та перевіряти поведінку вашого робота. Наприклад, симулятор Gazebo зазвичай використовується з ROS для створення реалістичних імітацій роботів. Ці симулятори забезпечують інтеграцію з бібліотеками робототехніки, інтерфейсами керування та моделюванням датчиків, що робить їх придатними для моделювання змагань роботів.

Спеціальні симулятори: залежно від складності та вимог змагання, ви можете вибрати розробку індивідуального тренажера, спеціально адаптованого до змагального завдання. Цей підхід дає вам повний контроль над середовищем моделювання, дозволяючи вам створити більш точне уявлення про правила та динаміку змагань. Спеціальні симулятори можна реалізувати за допомогою таких мов програмування, як Python [18], C++ або MATLAB, разом із відповідними бібліотеками фізики чи графіки.

Симулятори доповненої реальності (AR): доповнену реальність можна використовувати для створення симуляторів, які накладають віртуальні об’єкти та середовища на реальний світ. Використовуючи технології доповненої реальності, ви можете симулювати змагальні завдання в реальному світі, дозволяючи вашому роботу взаємодіяти з віртуальними об’єктами, одночасно сприймаючи реальне

середовище. Фреймворки AR, такі як ARCore (для Android) і ARKit (для iOS), надають інструменти для розробки симуляторів роботів на основі AR.

1.8 Можливе практичне використання навиків, отриманих у змаганні роботів

Знання і навики, отримані студентом під час участі в змаганнях роботів в подальшому можуть бути використані в практичних цілях. Зокрема сьогодні дуже актуальним є розміновування об’єктів. У цій галузі можуть бути використані конструкції роботів, апаратні і програмні засоби ідентифікації об’єктів, віддаленого керування роботом, алгоритми автоматичної роботи та інші знання і навики, отримані в змаганнях роботів.

Розміновування — процес повного знешкодження та видалення мін, мін-пасток, саморобних вибухових пристроїв, які не розірвалися, вибухових предметів з певного району місцевості з метою забезпечення безпеки цивільного населення. На морі для розмінування часто використовують — тральники, а для очищення сухопутної ділянки залучаються інженерні підрозділи, саперів, службу МНС (в складі якої є відповідні підрозділи). Розмінування може проводитися вручну або механічним способом з допомогою спецмашин.

Гуманітарне розмінування спрямоване на зменшення шкідливого фактору дії вибухових речовин на життєдіяльність людей.

Мета розмінування полягає в тому, щоб знизити мінну небезпеку до рівня, при якому люди можуть жити безпечно; при якому економічний, соціальний і фізіологічний розвиток може здійснюватися безперешкодно, не наражаючись впливу обмежень, що викликаються впливом наземних мін.

Гуманітарне розмінування в Україні здійснює зокрема, Державне підприємство «Укроборонсервіс», яке володіє широкою базою обладнання:

- Робот для розмінування «Дозор-М» рисунок 1.11,
- Броньовані машини з протимінної захистом «КОЗАК»,
- Машини швидкого реагування,

- Оптичні системи позиціонування TRIMBLE 5600 [9].

Рисунок 1.11 – Робот для розмінування «Дозор-М»

2. ПІДБІР ДЕТАЛЕЙ ТА ПРОТОТИПУ РОБОТА

2.1 Підбір елементів для складання прототипу робота

Змагання роботів-машинок на платформі Arduino може бути цікавим проектом для людей, які цікаві в розвитку своїх знань з програмування та електроніки. Для проекту «Змагання роботів» вам як мінімум знадобиться наступне обладнання:

- Arduino (або Arduino-сумісна платформа)
- Два мотори (наприклад, сервоприводи)
- Колеса для моторів
- Батарейний блок для живлення

Крім того, знадобитися деяке програмне забезпечення для написання коду для Arduino. Можна використовувати Arduino IDE або будь-який інший текстовий редактор, який підтримує мову програмування C++. Робот може управлятись програмою, яка виконується на ПК, наприклад Python-програмою. Це значно розширює його алгоритмічні і обчислювальні можливості. Для комунікації Arduino і Python можна використати відомий протокол Firmata.

Основний принцип роботи полягає в тому, щоб за допомогою моторів забезпечити рух коліс машини, а потім контролювати їх швидкість і напрямок. Для віддаленого керування роботом можна використати Bluetooth-модулі.

Ви можете додати більше функціональних можливостей до своєї машини, наприклад, сенсори розпізнавання кольору, додаткові датчики для вимірювання швидкості або відстані, фоторезистори та інше.

Потрібно почати з найпростіших проектів з Arduino, щоб отримати певний досвід перед тим, як перейти до більш складних проектів. Також не забувайте про безпеку при роботі з електричним обладнанням, дотримуйтеся інструкцій і не допускайте пожежі або інших небезпечних ситуацій.

Основою робота буде мікроконтролер, який користується популярністю сьогодні моделі Arduino UNO R3.

Arduino UNO - це мікроконтролер, який заснована на 8-бітному процесорі ATmega328P. Також він складається з інших компонентів такі як кварцовий генератор, послідовний канал зв'язку та регулятор напруги. Arduino UNO має 14 цифрових input/output пінів, 6 аналогових входів (які також можна використовувати як цифрові) та USB-з'єднання.

Arduino можна використовувати для спілкування з комп’ютером або іншими мікроконтролерами. На рис. 2.1 представлено мікроконтролер Arduino UNO R3.

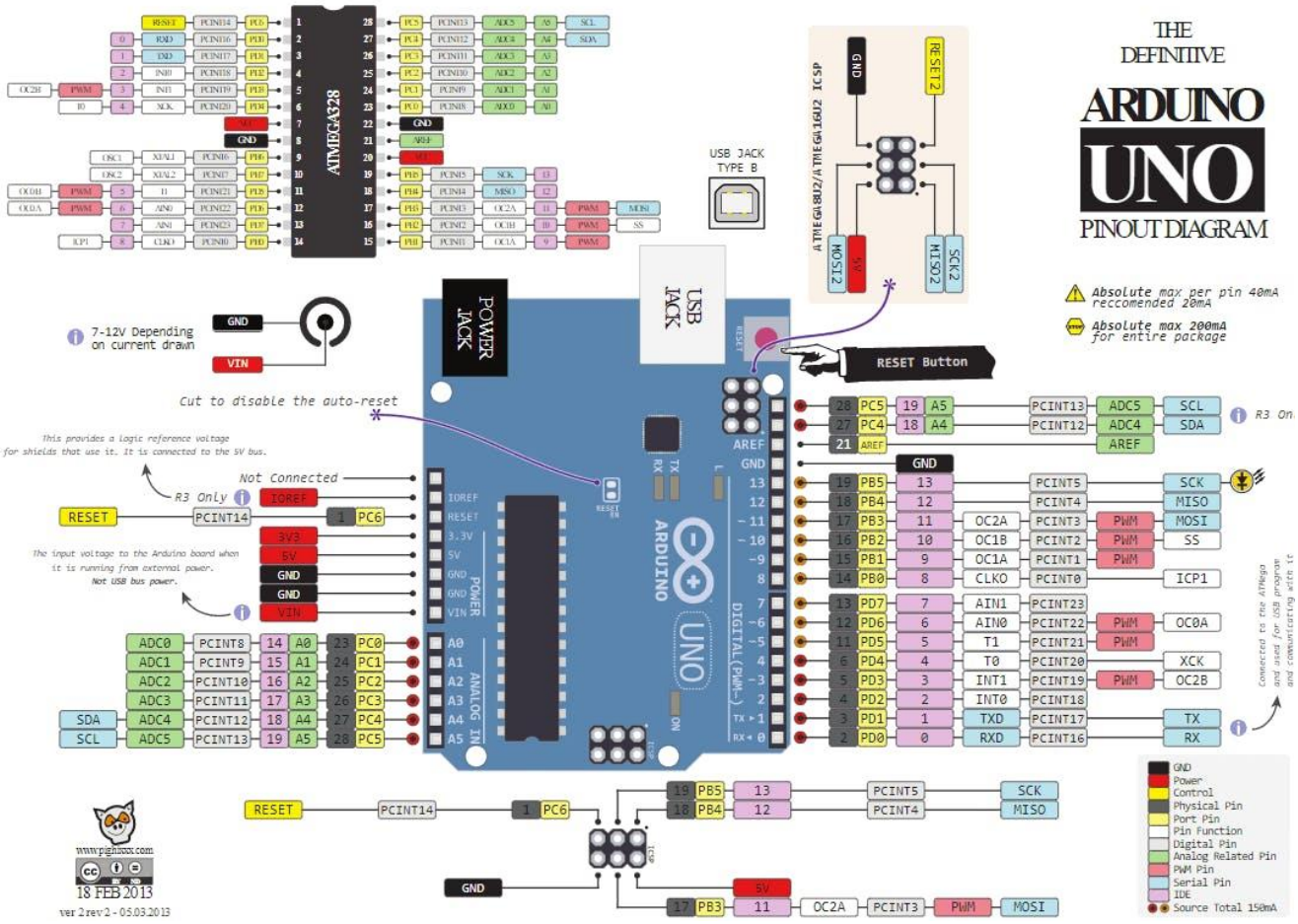


Рисунок 2.1 - Arduino UNO R3 [17]

Для зручності підключення елементів і більших можливостей, використаємо плату розширення Sensor Shield V5 (рис. 2.2). Основні можливості:

- Окреме живлення;
- Цифрового входи — виходи D0-D13;
- Аналогові входи — виходи A0-A5;
- Кнопка скидання RESET;

- Паралельний інтерфейс LCD;
- Послідовний інтерфейс LCD;
- Інтерфейс UART;
- Інтерфейс SD;
- Інтерфейс Bluetooth [10].

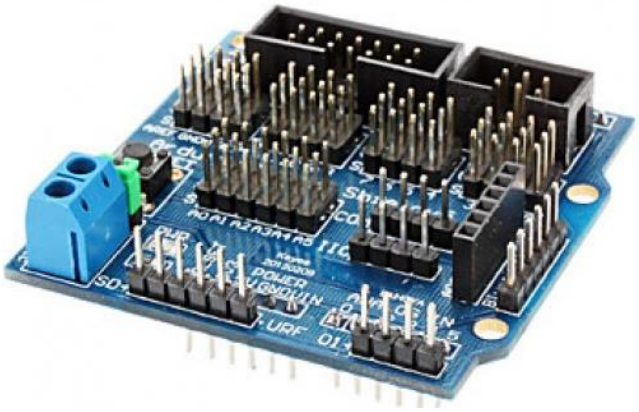


Рисунок 2.2 – Плата Sensor Shield V5

Двигуни, марки DC Gear 48 (рис. 2.3) приводитимуть в рух робота. Їх можна придбати в комплекті з пластмасовими колесами, резиновими накладками.

Рисунок 2.3 – Колесо з двигуном DC Gear 48

Драйвер L298N (рис. 2.4) потрібен для підключення двигунів типу DC Gear 48. Він може управляти двома моторами або одним кроковим двигуном.

Керуюче живлення для моторів VMS: 5 ~ 35В, сила струму 2А на міст. Драйвер L298N підтримує роботу з керуючим мікроконтролером з напругою рівнів 3.3В.

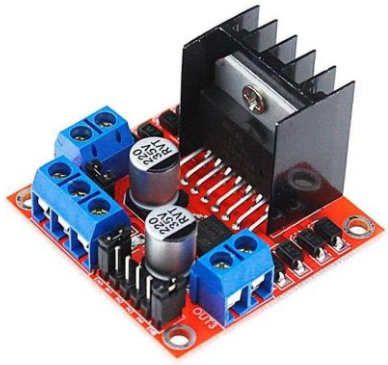


Рисунок 2.4 - Драйвер двигунів L298N

Популярний драйвер двигунів на базі чіпа L298, може управляти двома моторами або одним кроковим двуфазним двигуном. Підтримує роботу з керуючим мікроконтролером з напругою рівнів 3.3В. Керуюче живлення для моторів VMS: 5 ~ 35В, сила струму 2А на міст [11].

Основні характеристики і призначення контактів драйвера двигунів L298N наведено в таблиці 1.

Таблиця 2.1 - Основні характеристики драйвера двигунів L298N.

Основні характеристики:	Призначення контактів:
- Напруга живлення вбудованої логіки: 5В; - Споживаний струм вбудованої логіки:0 - 36мА; - Напруга живлення драйвера: 5 - 35В (max 46В); - Робочий струм драйвера: 2А (піковий струм 3А); - Максимальне споживання енергії: 25 Вт; - Робоча температура: -20 ° С - + 135 ° С; - Габарити: 43,5 х 43,2 х 29,4 мм;	- Vcc - Зовнішнє живлення двигунів; - Живлення логіки - +5 ; - GND – Загальний; - IN1, IN2, IN3, IN4 - контакти керування двигунами; - OUT1, OUT2 - вихід першого двигуна; - OUT3, OUT4 - вихід другого двигуна.

Поворотне, пластмасове колесо на металевій основі з підшипником (рис 2.5) використовуємо в якості заднього колеса.

Рисунок 2.5 - Поворотне колесо

Ультразвуковий датчик відстані HC-SR04 (рис. 2.6) використовуємо для розпізнавання об’єктів за розміром. Він стабільний та точний, не має "сліпих зон". Може вимірювати відстань від 0 см до 1500мм, точність досягає 3 мм[12].

Рисунок 2.6 - Ультразвуковий датчик відстані HC-SR04

Основні характеристики ультразвукового датчика відстані HC-SR04 наведено в таблиці 2.2.

Таблиця 2.2 - Основні характеристики датчика

Основні характеристики	Показники
- Робоча напруга:	3.8 - 5.5В
- Тип:	HC-SR04
- Струм:	8 мА
- Частота:	40 кГц
- Максимальна дистанція:	1500 мм
- Ширина імпульсів:	10 мкс
- Мінімальна дистанція:	0 см
- Зовнішні габарити:	37х20х15 мм
- Роздільна здатність:	3 мм

Принцип роботи ультразвукового датчика відстані HC-SR04.

1. На вихід trig (тригер) посилаємо високий рівень протягом як мінімум 10мкс

2. Модуль починає посилати ультразвукові імпульси з частотою 40 кГц і приймає їх назад, якщо в зоні видимості є будь-які перешкоди

3. Якщо сигнал повертається, модуль встановлює низький рівень на виході echo на 150мс. За часом, який минув з п.1 до низького рівня на виході echo можна розрахувати відстань до перешкоди за формулою:

Відстань = (time * sound velocity)/2

де time - виміряний час імпульсу,

sound velocity - швидкість звуку (340 м/с)

Приводитиметься в рух ультразвуковий датчик буде за допомогою серводвигунів моделі SG90 (рис. 2.7). Рух по горизонталі і вертикалі. Основні характеристики наведено в таблиці 2.3.

Таблиця 2.3- Характеристика серводвигуна моделі SG90

Основні характеристики	Показники
- Швидкість без навантаження:	0.12 сек / 60 град. при живленні 4.8В;
- Крутний момент:	2 кг / см;
- Температурний діапазон:	0 to + 50'C;
- Ширина мертвої зони:	4 мікросекунди;
- Робоча напруга живлення:	3.5-5 В;
- Споживаний струм в русі:	50-80 мА;
- Споживаний струм в утриманні:	5-10 мА;
- Кут повороту:	180 градусів;
- Розміри:	3.3 см х 3 см х 1.3 см;

Найбільш важливі переваги сервоприводів:

- висока потужність в порівнянні з розмірами і вагою двигуна.

- дозвіл визначається за допомогою енкодера.

- висока ефективність: може досягти 90% при невеликих навантаженнях.

- високий крутний момент по відношенню до інерції. Працює з високим прискоренням.

Рисунок 2.7 - Серводвигун моделі SG90

- резервує енергію для підтримки живлення на короткий період.

- резервує крутний момент для підтримки обертання на короткий період.

- двигун залишається прохолодним. Струм споживання пропорційний навантаженні.

- високий крутний момент на високій швидкості.

- тиха робота на високих швидкостях.

- відсутність явищ резонансу і вібрації.

Також необхідні пластмасові деталі, для утворення механізму (рис. 2.8) і передачі крутного моменту (рис. 2.9), ідуть в комплекті з серводвигуном[13].

Рисунок 2.8 – Пластмасові деталі, для утворення механізму

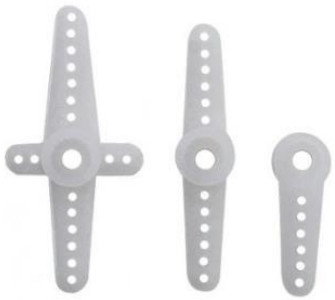


Рисунок 2.9 – Деталі, для передачі руху серводвигуна

Механізм, що керує напрямком ультразвукового датчика відстані в зборі зображено на рисунку 2.10.

Рисунок 2.10 - Механізм в зборі

Прототип живитиметься за допомогою батарейного блоку 6V, розрахованого на чотири батарейки типу AA (рис. 2.11).

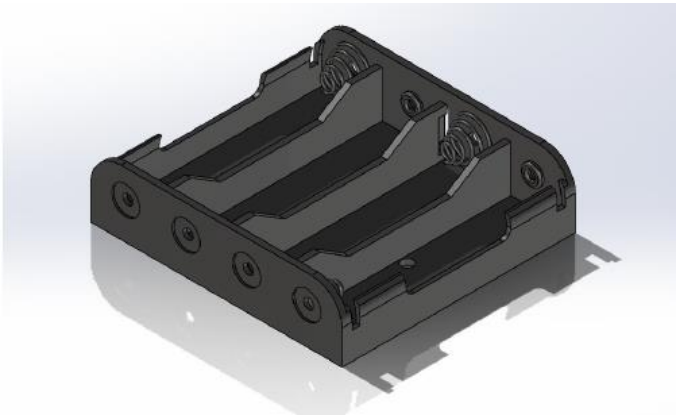


Рисунок 2.11- Батарейний блок

Шасі робота в зборі наведено на рисунку 2.12. Основою робота буде акрилова пластина.

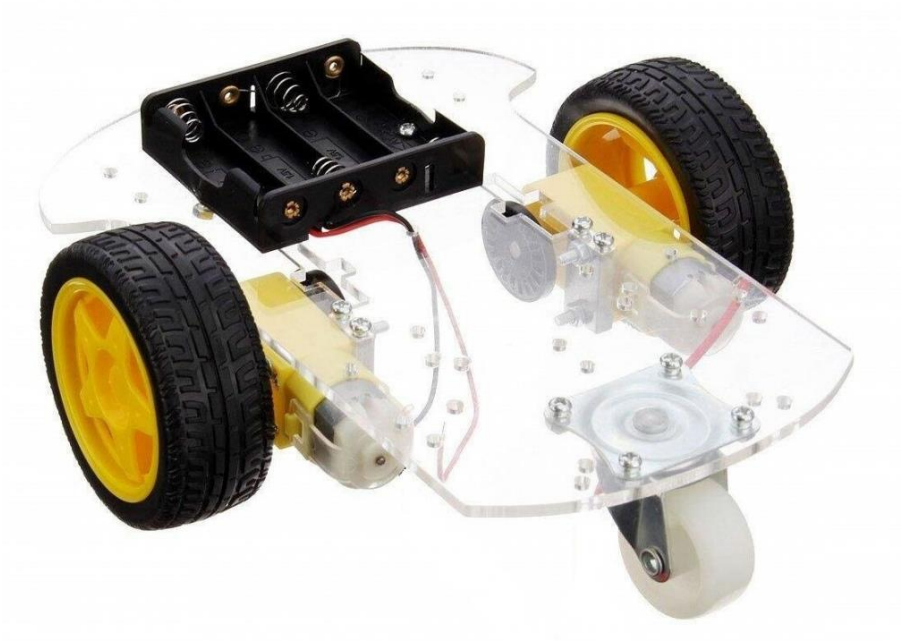


Рисунок 2.12 – Шасі робота в зборі

Провідники типу pin: Female to Female, Male to Male і Male to Female використовуємо для кріплення елементів до мікроконтролера (рис. 2.13).

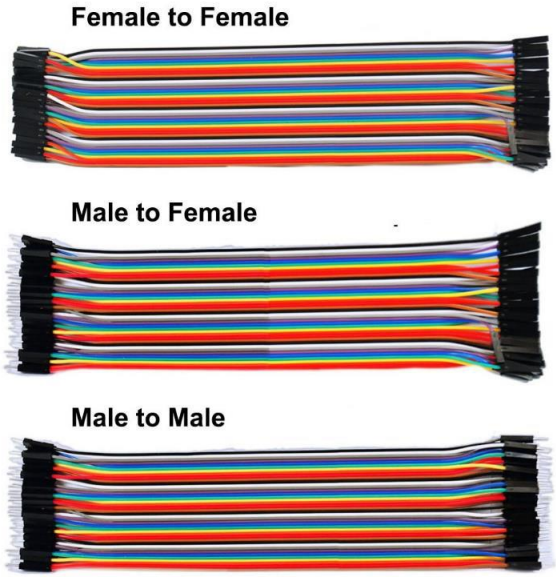


Рисунок 2.13 – Провідники типів Female to Female, Male to Male і Male to Female

Зв’язок з ПК буде організовано за допомогою Bluetooth модуля HC-05 (рис 2. 14).

Основні характеристики:

- Протокол зв'язку Bluetooth Specification v2.0 + EDR;
- Частота GFSK (Gaussian Frequency Shift Keying);
- Потужність відправки ≤4dBm, Class 2;

- Потужність прийому ≤-84dBm at 0.1% BER;

- Швидкість асинхронна 2.1Mbps (Max) / 160 kbps, синхронна 1Mbps / 1Mbps;

- Безпека Authentication and encryption;

- Живлення + 5VDC 50mA;

- Робочі температури -20 ~ +75 C;

- Розміри 26.9мм х 13 мм х 2,2 мм.

Призначення контактів:

- STATE - сюди дублюється сигнал з вбудованого світлодіода, коли модуль активний світлодіод блимає, коли зв'язок встановлено - горить;

- RXD - на цьому піні модуль приймає дані (тобто в вашому скетчі сюди треба відсилати дані) ;

- TXD - сюди модуль відправляє дані;

- GND – земля;

- VCC - живлення 5В;

- EN - вкл / викл, якщо подати сюди логічну одиницю (або просто 5В), то модуль вимкнеться, якщо логічний нуль (або просто не підключати цей пін) буде працювати.

Рисунок 2.14 – Bluetooth модуль HC-05

Придбати всі елементи, для робота, простіше всього готовим комплектом (рис. 2.15), до якого не входять тільки провідники та Bluetooth модуль HC-05. Кріпильні елементи йдуть в комплекті.

Рисунок 2.15 – Комплект для створення робота

2.2 Прототип робота для змагань

Даний робот був куплений мною на Aliexpress як набір деталей для самостійного складання (рис. 2.15). Робота в зборі, зображено на рисунку (рис. 2.16).

Кожна деталь відповідає своєму номеру: 1) Акрилова пластина; 2) Bluetooth модуль HC-05; 3) Батарейний блок; 4) Arduino UNO R3; 5) Sensor Shield V5; 6, 22) Пластмасове колесо; 7) Пластмасова деталь 1; 8) Пластмасова деталь 5; 9) Серводвигун SG90 1; 10) Пластмасова деталь 2; 11) Серводвигун SG90 2; 12) Пластмасова деталь 4; 13) Ультразвуковий датчик відстані HC-SR04; 14) Пластмасова деталь 6; 15) Пластмасова деталь 3; 16) Пластмасова деталь 3; 17) Поворотне колесо; 18) Драйвер двигунів L298N; 19, 22) Двигун DC Gear 48; 20, 21, 23, 24) Акрилові елементи, для кріплення двигунів.

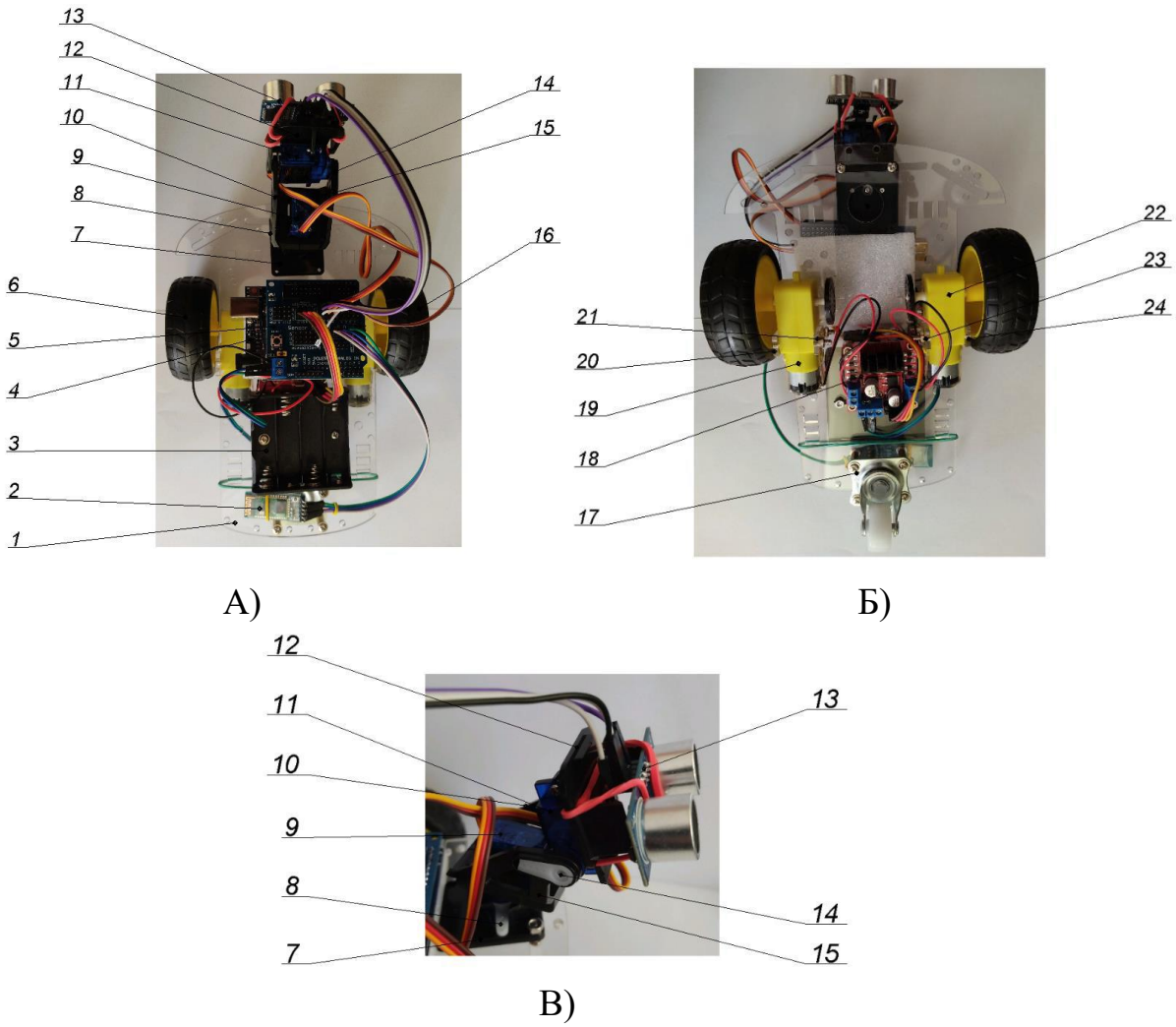


Рисунок 2.16 – Робот для змагань роботів: а) Вигляд зверху; б) Вигляд знизу; в) Механізм керування ультразвуковим датчиком відстані

Колеса 6 і 16 кріпляться до двигунів 19 і 22. За допомогою акрилових деталей 20, 21 і 23, 24 кріпляться до акрилової пластини 1. Поворотне колесо 17 кріпиться до акрилової пластини 1, кріпильними елементами. Двигуни 13 і 22 з’єднуються з драйвером двигунів L298N 18, до виходів «output A» і «output B», провідниками.

Батарейний блок 3 під’єднаний до виходів «+12V power» і «power GND» драйвера двигунів L298N 18, провідниками. Драйвер двигунів L298N 18, з’єднується з Sensor Shield V5 5 контактами керування IN1, IN2, IN3 і IN4 до Digital Ports 8, 9, 10 і 11, провідниками. Sensor Shield V5 3 встановлений на мікроконтролер Arduino UNO R3 4 і провідники живлення батарейного блоку підключені до його виходів «GND» і «5V». Bluetooth модуль 2, підключений до Sensor Shield V5 5 по принципу: «RX» до «TX», «TX» до «RX», «GND» до «-» і «VCC» до «+»,

провідниками. Серводвигуни 9 і 11 підключені до Sensor Shield V5 3, до Digital Ports 5, 6, провідниками. Ультразвуковий датчик відстані HC-SR04 13 підключений до Sensor Shield V5 5, до Digital Ports 7, провідниками, при цьому виходи Echo та Trig є замкнутими між собою.

Батарейний блок 3 прикріплений до акрилової пластини 1, за допомогою кріпильних елементів. Драйвер двигунів L298N 18 прикріплений до акрилової пластини 1, за допомогою кріпильних елементів.

Механізм, що керує напрямком ультразвукового датчика зібраний по принципу з’єднання пластмасових елементів 7, 8, 10, 15, 12 і 14 та серводвигунів 9 і 11 (рис. 2.16, в), кріпильними елементами та канцелярської резинки. При цьому пластмасова деталь 7 кріпиться до акрилової пластини 1, кріпильними елементами. Робот в зборі зображений на рисунку 2.17.

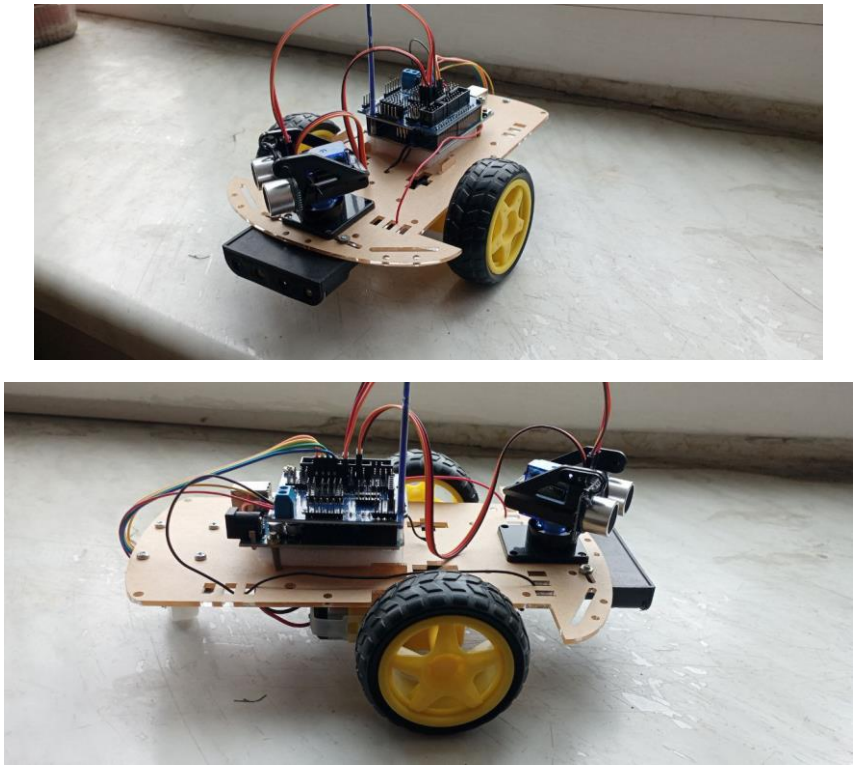


Рисунок 2.17 – Робот для змагань в зборі

3 МОДЕЛЮВАННЯ ЗМАГАНЬ РОБОТІВ

3.1 Принципи моделювання змагань роботів

Створюючи тренажери для змагань роботів, враховуйте наступні аспекти:

Моделювання середовища: точно проектуйте та моделюйте середовище змагань, включаючи арену, об’єкти, умови освітлення та будь-які відповідні перешкоди.

Моделювання робота: створіть 3D-модель свого робота, включаючи його фізичні розміри, датчики, приводи та механізми захоплення. Переконайтеся, що поведінка та можливості робота точно представлені в симуляторі.

Фізичне моделювання: запровадьте або налаштуйте механізм фізичного моделювання для точного моделювання динаміки та взаємодії між роботом, об’єктами та навколишнім середовищем.

Симуляція датчиків: імітуйте датчики робота, такі як камери, лідари або датчики наближення, для створення реалістичних даних датчиків. Це може включати емуляцію виявлення об’єктів, сприйняття глибини або будь-яких інших відповідних можливостей сприйняття.

Керування та створення сценаріїв: розробіть алгоритми керування або сценарії, які дозволяють програмувати поведінку та дії робота в симуляторі. Це може передбачати інтеграцію з API або бібліотеками, що надаються симулятором або робототехнікою.

Візуалізація та аналіз: реалізуйте інструменти візуалізації та аналізу даних для моніторингу та оцінки продуктивності робота під час моделювання. Це може включати відображення в реальному часі, реєстрацію даних датчиків і показники продуктивності.

Створюючи симулятори для змагань роботів, ви можете ефективніше повторювати дизайн і алгоритми свого робота, оцінювати різні стратегії та оптимізувати його продуктивність без обмежень і ризиків фізичного тестування [6].

Можна використовувати Python [18] і пакет Pymunk для створення симуляторів для змагань роботів. Pymunk — це бібліотека двовимірної фізики для Python, яка забезпечує зручний інтерфейс для фізичного механізму Chipmunk [14]. Хоча Pymunk в основному зосереджений на симуляції 2D фізики, його все ще можна використовувати для створення базових симуляторів змагань роботів.

Ось схема того, як ви можете використовувати Python і Pymunk для створення простого симулятора змагань роботів:

Встановіть Pymunk: почніть із встановлення пакета Pymunk у вашому середовищі Python. Ви можете скористатися такою командою, щоб встановити його через pip:

```
pip install pymunk
```

Моделювання середовища: спроектуйте та створіть двовимірне представлення середовища змагань за допомогою Pymunk. Це передбачає визначення меж, об’єктів та будь-яких інших відповідних елементів у симуляторі.

Моделювання робота: створіть представлення свого робота за допомогою Pymunk. Визначте форму, розміри та фізичні властивості робота, такі як маса, сила тертя та форма зіткнення. Ви можете використовувати класи Pymunk Body and Shape, щоб створити та налаштувати фізичне тіло й форму робота.

Фізичне моделювання: використовуйте можливості фізичного механізму Pymunk для моделювання динаміки та взаємодії між роботом, об’єктами та навколишнім середовищем. Визначте фізичні властивості, такі як сила тяжіння, засоби обробки зіткнень і обмеження (якщо необхідно), щоб точно представити фізику змагання.

Симуляція датчиків: реалізуйте моделювання датчиків за допомогою Pymunk або спеціальної логіки. Наприклад, ви можете імітувати датчик камери, виявляючи перекриття або зіткнення між формою робота та об’єктами в навколишньому середовищі.

Керування та створення сценаріїв: розробіть алгоритми керування або сценарії за допомогою Python для програмування поведінки робота в симуляторі.

Це може включати визначення дій, рухів, виявлення об’єктів і прийняття рішень на основі змодельованих даних датчика.

Візуалізація та аналіз. Використовуйте бібліотеки Python, такі як Nodebox for OpenGL, для створення візуалізацій симулятора та відображення відповідної інформації, наприклад показань датчиків, положення робота та стану об’єктів. Крім того, запровадьте інструменти аналізу для оцінки продуктивності робота та збору даних для подальшої оптимізації.

Хоча Pymunk може не пропонувати всі розширені функції та можливості спеціальних симуляторів робототехніки, він може служити відправною точкою для створення простого симулятора змагань роботів у двовимірному середовищі. Якщо ваша конкуренція потребує складнішої динаміки, 3D-симуляції або спеціалізованих функцій, вам може знадобитися дослідити інші бібліотеки, фреймворки чи ігрові движки, які надають більш повні можливості робототехнічного моделювання.

Не забудьте звернутися до документації та прикладів Pymunk, щоб отримати глибше розуміння функціональності та використання бібліотеки під час створення симуляцій на основі фізики.

3.2 Програма на основі Nodebox і Pymunk для моделювання змагання роботів і відлагодження керуючих програм

Дана програма призначена для імітації змагань роботів (рисунок 3.1). Тобто для того щоб кожен учасник змагань з використанням даної програми міг відшліфувати свої стратегію. Звичайно ж що на віртуальному роботі набагато легше відшліфувати стратегію змагань ніж на реальному.

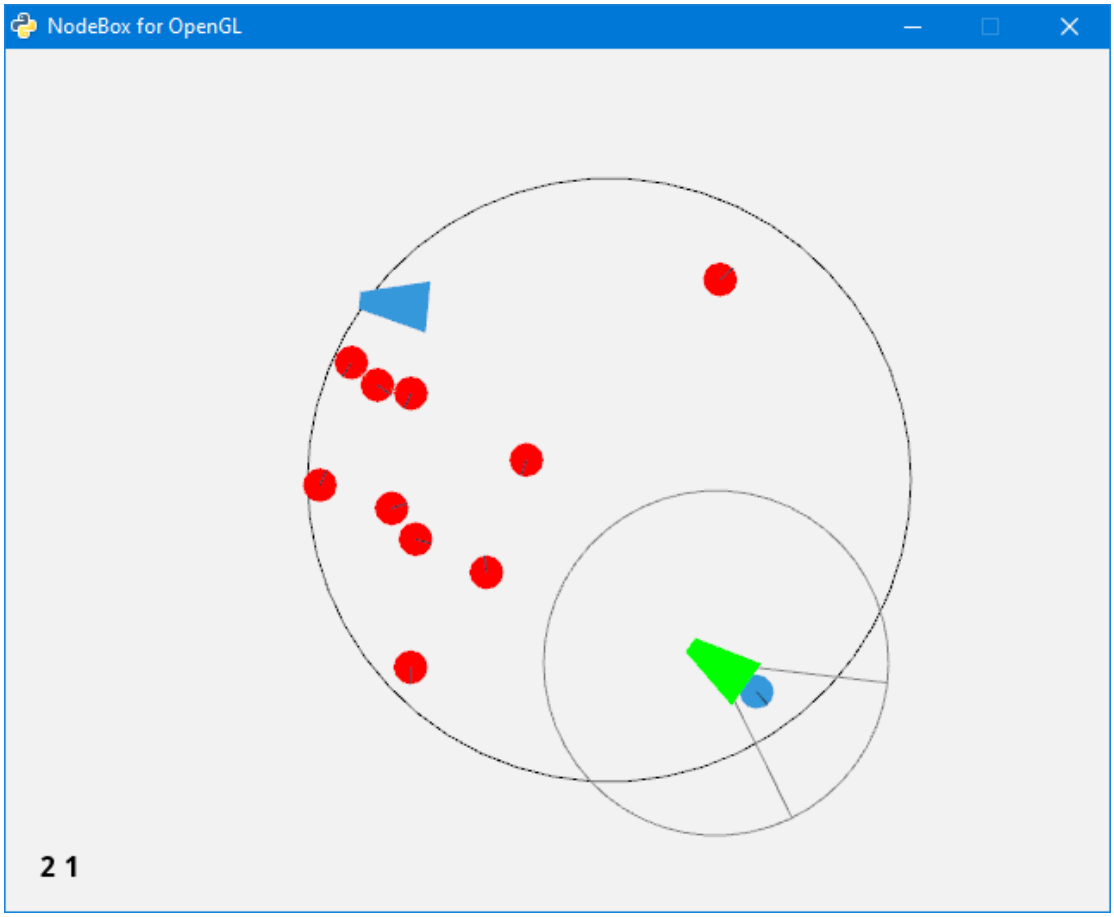


Рисунок 3.1 – Імітація змагань роботів (зелений робот перемагає)

У першу чергу нам необхідні відповідні модулі для роботи програми. В першу чергу нам потрібен пакет *nodebox* який реалізує графік нашої програми, і пакет *pymunk* який дозволяє реалізувати симуляцію фізики:

```
from __future__ import division
from nodebox.graphics import *
import pymunk
import pymunk.pyglet_util
import random, math
```

Функція створення об’єкту space. Це простір з усіма фізичними об’єктами, які потрібні фізичному рушію для симуляції фізики:

```
space = pymunk.Space()
```

Функція яка створює об’єкт заданої форми. Цей об’єкт буде розташований в точці x,y.

```
def createBody(x,y,shape,*shapeArgs):
    body = pymunk.Body()
    body.position = x, y
    s = shape(body, *shapeArgs)
    s.mass = 1
    s.friction = 1
    space.add(body, s)
    return s
```

Функція створює об’єкта робота, який називається s0 і він розташований в точці 300, 300. Також створюються інші об’єкти.

```
s0=createBody(300, 300, pymunk.Poly, ((-20,-5),(-20,5),(20,15),(20,-15)))
s0.score=0
s3=createBody(200, 300, pymunk.Poly, ((-20,-5),(-20,5),(20,15),(20,-15)))
s3.color = (0, 255, 0, 255)
s3.score=0
s1=createBody(300, 200, pymunk.Circle, 10, (0,0))
S2=[]
for i in range(10):
    s2=createBody(350, 250, pymunk.Circle, 10, (0,0))
    s2.color = (255, 0, 0, 255)
    S2.append(s2)
```

Наступна функція потрібна для того, щоб можна було визначити кут нахилу за двома точками:

```
def getAngle(x,y,x1,y1):
    return math.atan2(y1-y, x1-x)
```

Функція дозволяє розрахувати відстань між двома точками:

```
def getDist(x,y,x1,y1):
    return ((x-x1)**2+(y-y1)**2)**0.5
```

Функція повертає істину, якщо об’єкт в межах кола:

```
def inCircle(x,y,cx,cy,R):
    if (x-cx)**2+(y-cy)**2 < R**2:
        return True
    return False
```

Функція повертає істину, якщо робот знаходиться в межах сектора:

```
def inSector(x,y,cx,cy,R,a):
    angle=getAngle(cx,cy,x,y)
    if inCircle(x,y,cx,cy,R) and a-0.5<angle<a+0.5:
        return True
    return False
```

Функція реалізує найпростішу стратегію робота:

```
def strategy(b=s3.body):
    v=100
```

```
    a=b.angle
    b.velocity=v*cos(a), v*sin(a)
    x,y=b.position
    R=getDist(x,y,350,250)
    #print R, b.angle
    line(x,y,*s1.body.position,stroke=Color(0))
    if canvas.frame%100==0:
        if R>180:
            b.angle=getAngle(x,y,350,250)
        else:
            b.angle=getAngle(x,y,*s1.body.position) #2*math.pi*random.random()
```

Складніша функція стратегії стратегія з випадковим рухом робота, який сканує сектор ультразвуковим сенсором:

```
def strategy2(b=s3.body):
    v=100
    a=b.angle
    b.velocity=v*cos(a), v*sin(a)
    x,y=b.position
    R=getDist(x,y,350,250)
    ellipse(x, y, 200, 200, stroke=Color(0.5))
    #line(x,y,x+100*cos(a),y+100*sin(a),stroke=Color(0.5))
    line(x,y,x+100*cos(a+0.5),y+100*sin(a+0.5),stroke=Color(0.5))
    line(x,y,x+100*cos(a-0.5),y+100*sin(a-0.5),stroke=Color(0.5))
    inS=inSector(s1.body.position[0], s1.body.position[1], x, y, 100, a)
    if inS:
        b.angle=getAngle(x,y,*s1.body.position)
        print b.angle
    if canvas.frame%100==0:
```

```
        if R>180:
            b.angle=getAngle(x,y,350,250)
        elif not inS:
            b.angle=2*math.pi*random.random()
```

Наступна функція потрібна для того щоб визначити якій з роботів переміг в змаганнях. Вона використовується в функції *def score():*

```
def scr(s,s0,s3,p=1):
    bx,by=s.body.position
    s0x,s0y=s0.body.position
    s3x,s3y=s3.body.position
    if not inCircle(bx,by,350,250,180):
        if getDist(bx,by,s0x,s0y)>getDist(bx,by,s3x,s3y):
            s0.score=s0.score+p
        else:
            s3.score=s3.score+p
        s.body.position=random.randint(200,400),random.randint(200,300)
```

Функція визначення переможця:

```
def score():
    scr(s1,s0,s3)
    for s in S2:
        scr(s,s0,s3,p=-1)
```

Функція керування роботом з мишки або клавіатури:

```
def manualControl():
    v=10 # швидкість
    b=s0.body
```

```
    a=b.angle
    x,y=b.position
    vx,vy=b.velocity
    if canvas.keys.char=="a":
        b.angle-=0.1
    if canvas.keys.char=="d":
        b.angle+=0.1
    if canvas.keys.char=="w":
        b.velocity=vx+v*cos(a), vy+v*sin(a)
    if canvas.mouse.button==LEFT:
        b.angle=getAngle(x,y,*canvas.mouse.xy)
        b.velocity=vx+v*cos(a), vy+v*sin(a)
```

Функція для симуляції тертя:

```
def simFriction():
    for s in [s0,s1,s3]+S2:
        s.body.velocity=s.body.velocity[0]*0.9, s.body.velocity[1]*0.9
        s.body.angular_velocity=s.body.angular_velocity*0.9
```

Головна функція, яка рисує кожен кадр анімації:

```
def draw(canvas):
    canvas.clear()
    fill(0,0,0,1)
    text("%i %i"%(s0.score,s3.score),20,20)
    nofill()
    ellipse(350, 250, 350, 350, stroke=Color(0))
    manualControl()
    strategy2()
    score()
    simFriction()
```

```
    space.step(0.02)
    space.debug_draw(draw_options)
```

Команди які відповідають за розміри вікна і за запуск процесу анімації:

```
canvas.size = 700, 500
canvas.run(draw)
```

4 ПРОЕКТУВАННЯ МОБІЛЬНОГО РОБОТА

4.1 Параметрична модель робота для змагань у SolidWorks

Для проектування конструкції робота використано SolidWorks. SolidWorks - це продукт компанії SolidWorks Corporation (зараз — компанія Dassault Systèmes), САПР, інженерного аналізу та підготовки виробництва будь-якої складності та призначення [19, 20]. Програма з'явилась в 1993 році та склала конкуренцію таким продуктам як AutoCAD та Autodesk Mechanical Desktop, SDRC I-DEAS і Pro/ENGINEER, Solid Edge [15].

На рисунку 4.1 показана 3d-модель робота, який буде брати участь в змаганнях. Основні деталі робота були куплені на AliExpress.

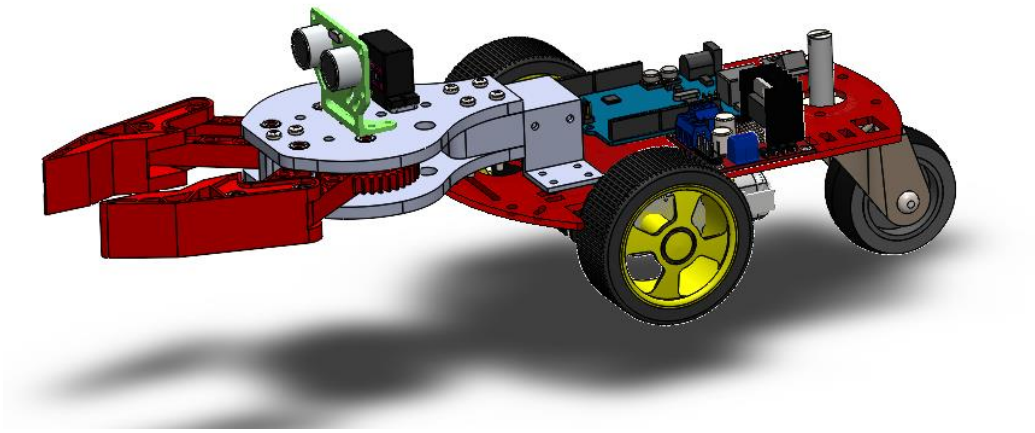


Рисунок 4.1 – 3D модель робота для змагань

Першим кроком розробляємо деталь номер 1 – основу (рисунок 4.2). Вона є найпростішою. Для неї використовуємо операцію видавлювання. На рисунку 4.2 також показано колесо для робота. Діаметр колеса 65 мм. В збірці їх потрібно дві штуки.

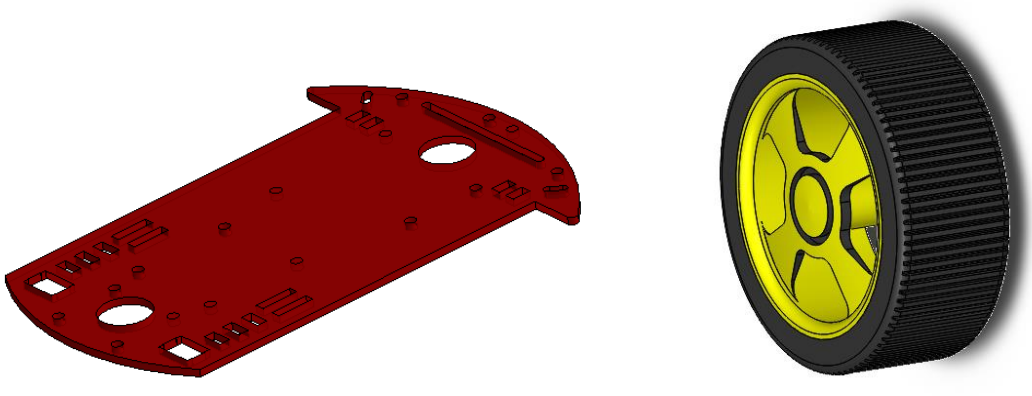


Рисунок 4.2 – 3d-моделі основи і колеса

На рисунку 4.3 показане кріплення для захватного механізму. Його виконуємо за допомогою операцій видавлювання, отвори за допомогою операції видавлення отворів.

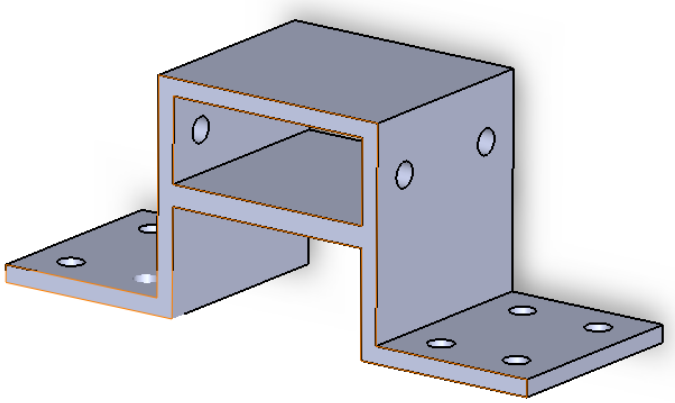


Рисунок 4.3 – Кріплення захватного механізму

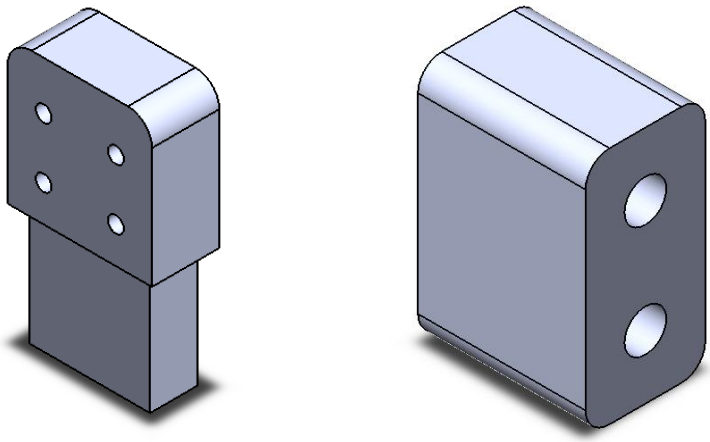


Рисунок 4.4 – Деталі для захватного механізму

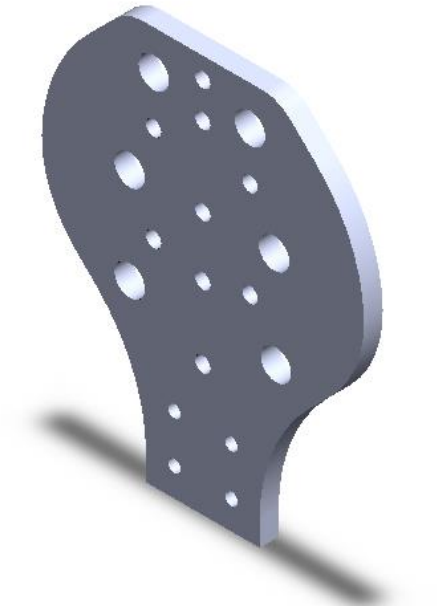


Рисунок 4.5 – Основа захватного механізму

На рисунку 4.5 показано основа захватного механізму. В збірці їх використовують дві деталі. Виготовляємо основу за допомогою операції видавлювання.

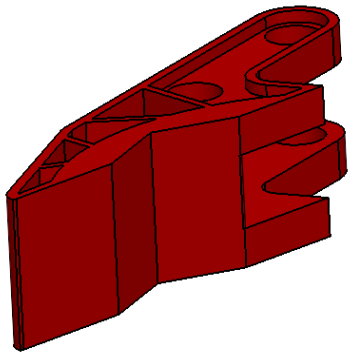


Рисунок 4.6 – Клешня маніпулятора

Рисуємо ескіз клешні, і за допомогою операцій видавлювання, витягуємо 3d модель. На рисунку 4.7 показані допоміжні деталі для захватного механізму.

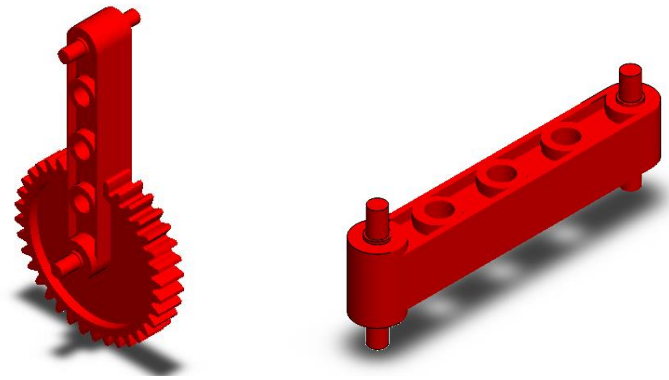


Рисунок 4.7 – Допоміжні деталі захватного механізму

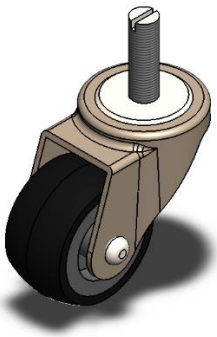


Рисунок 4.8 – Колесо на підшипнику

Розробляємо прототип ще одного робота для змагань за допомогою SolidWorks. Починаємо з проектування платформи рисунок 4.9.

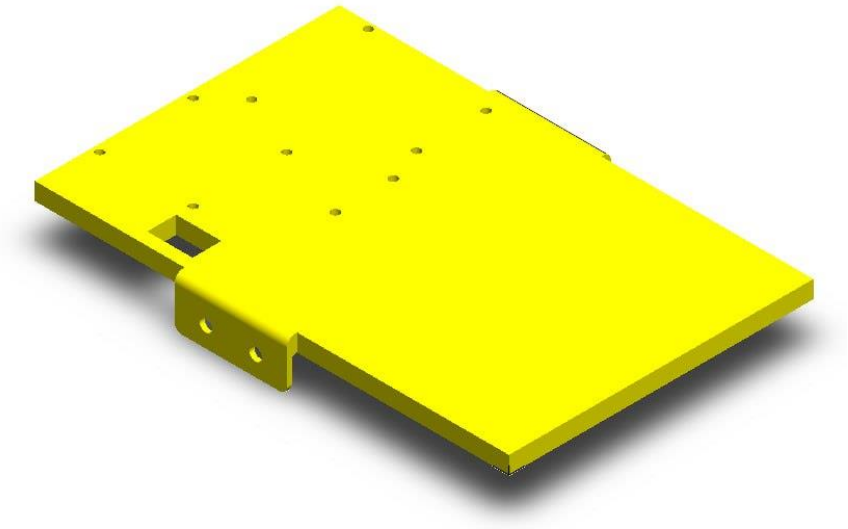


Рисунок 4.9 – Платформа робота

Платформа повинна бути стійкою, щоб витримати всю подальшу конструкцію робота. Габаритні розміри 190х130мм.

Корпус робота показано на рисунку 4.10, кріпиться до платформи за допомогою болтів M5x 0.5x 8 типу.

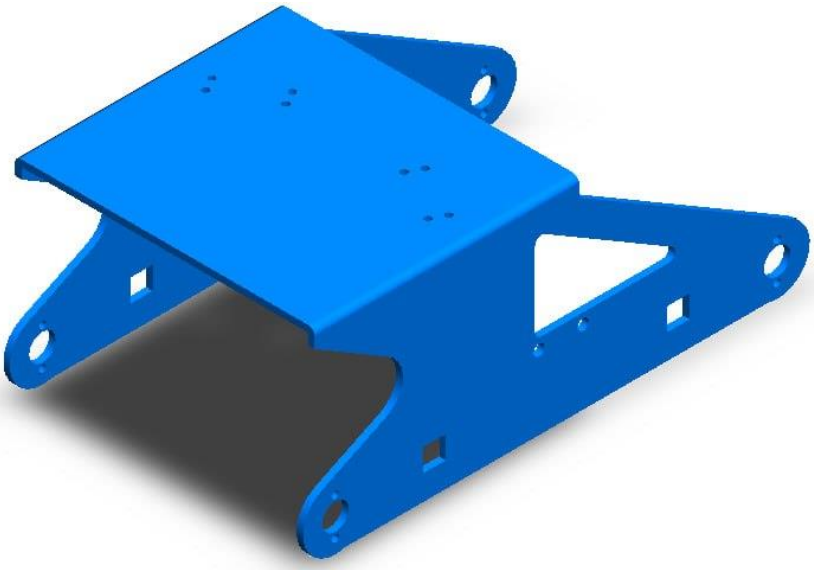


Рисунок 4.10- Модель корпуса робота

Модель тримача ультразвукового датчика HC-SR04 зроблена за допомогою операцій видавлювання та вирізання отворів (рисунок 4.11).

Рисунок 4.11 - Модель тримача датчик відстані HC-SR04

На рисунку 4.12 зображено модель опори для підтримки платформи. Габаритні розміри 10x10x156 мм.

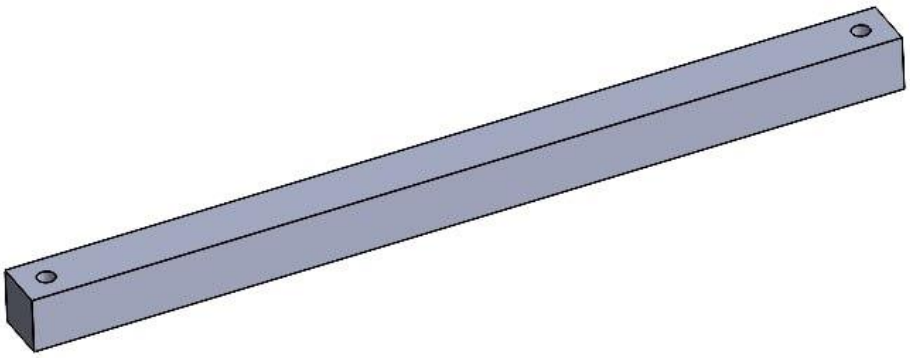


Рисунок 4.12 - Модель опори для підтримання платформи

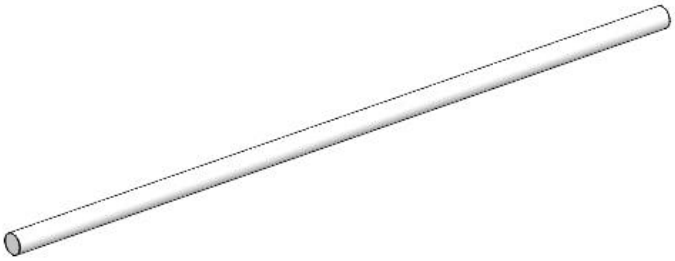


Рисунок 4.13 – Модель передньої колісної вісі

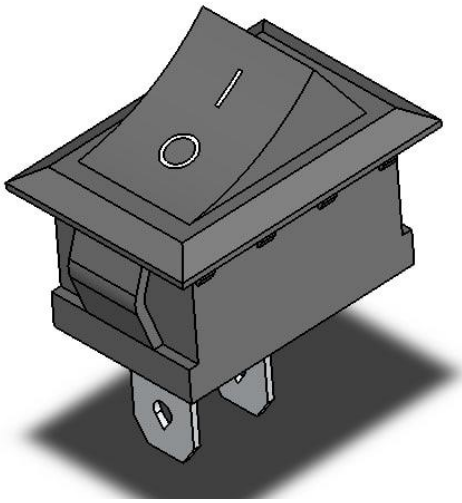


Рисунок 4.14– Модель кнопки

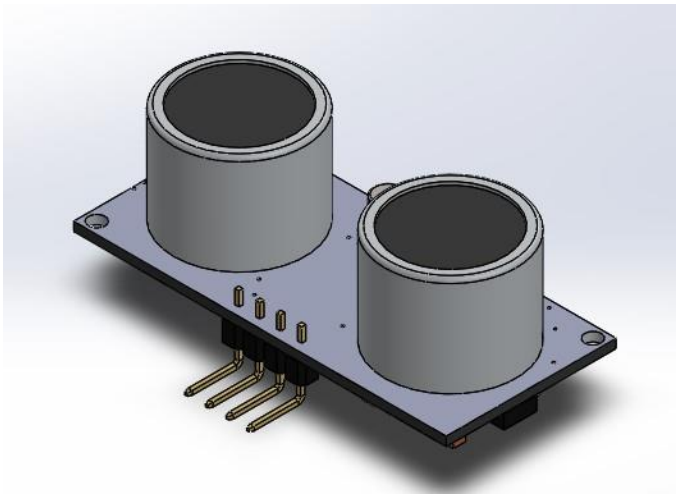


Рисунок 4.15- Модель ультразвукового датчика відстані HC-SR04

Рисунок 4.16- Модель колеса робота

Рисунок 4.17- Модель батарейного блоку 3x AAA

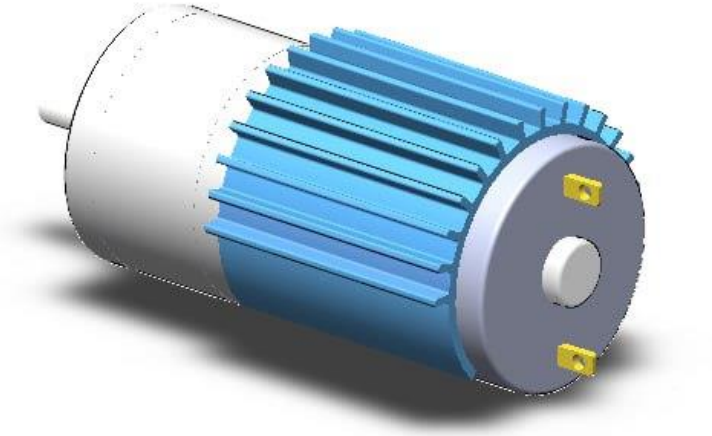


Рисунок 4.18 - Модель двигуна 25ga-370-280-rpm-dc

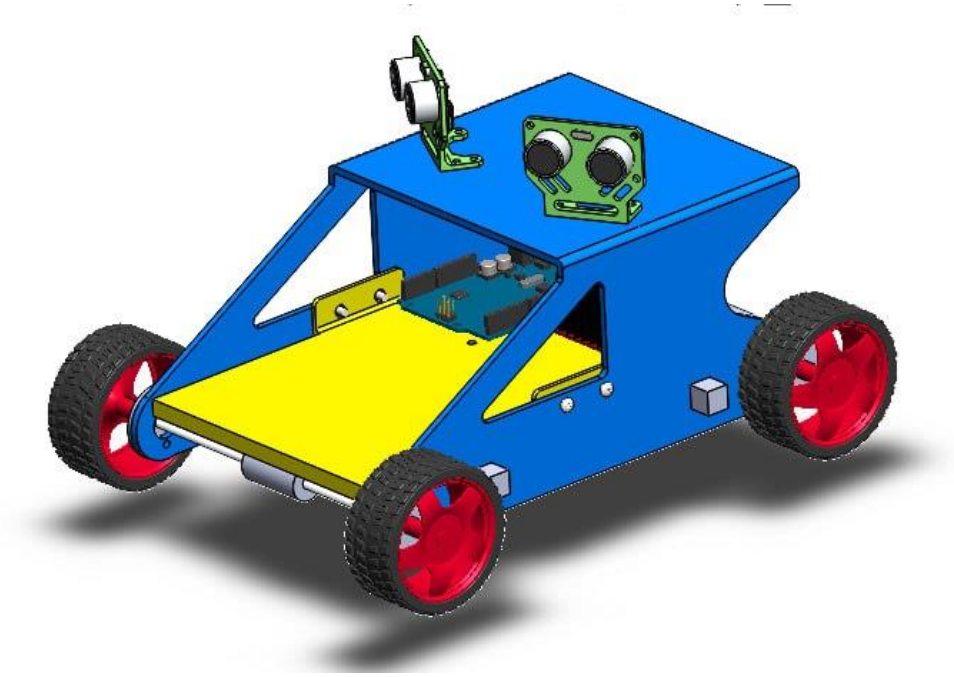


Рисунок 4.19 – Спроєктована модель робота

На рисунку 4.19, показано 3d моделі які в збірці, рисунок 4.20 утворюють основу маніпулятора. Габаритні розміри основ: Діаметр 98 мм, товщина 3 мм.

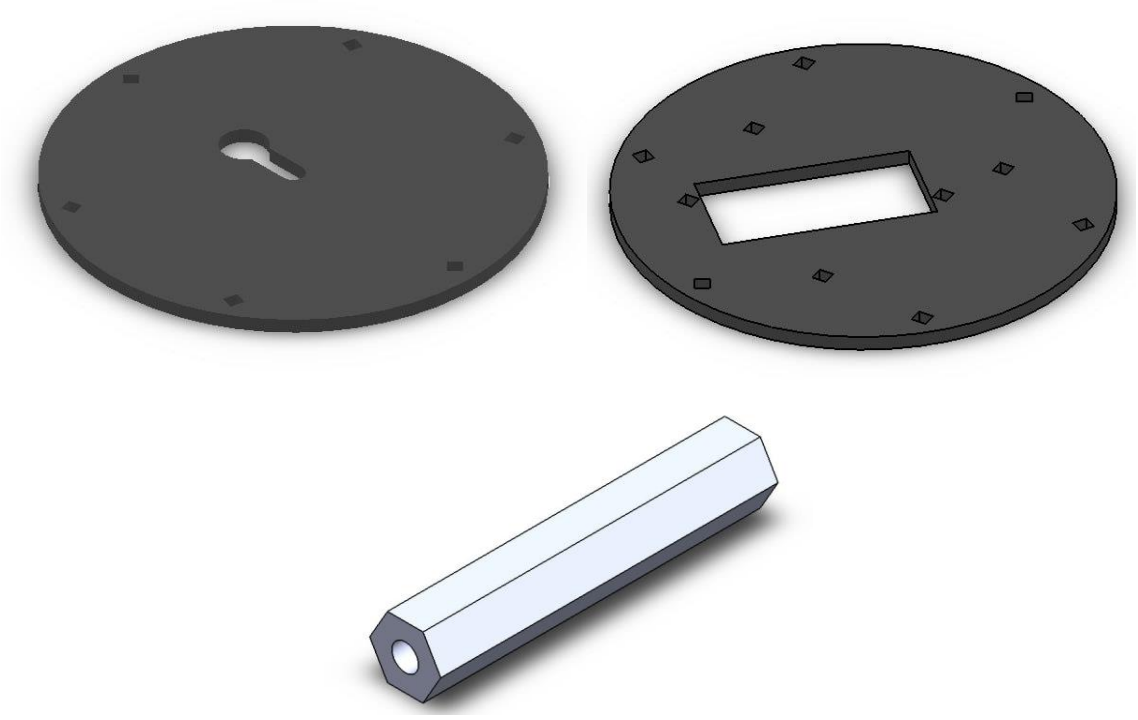


Рисунок 4.20 – Деталі для основи маніпулятора

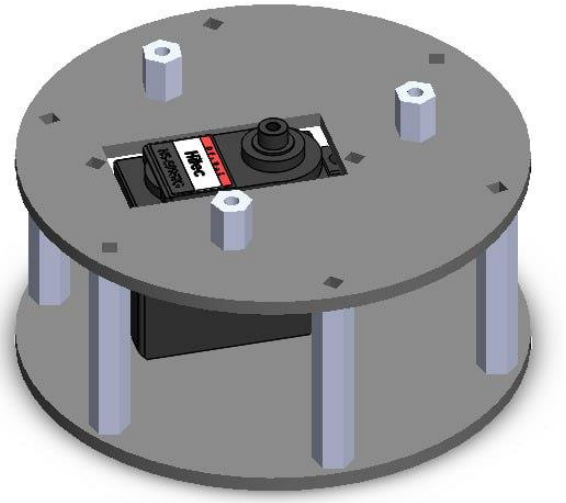


Рисунок 4.21 – Основа маніпулятора

На рисунку 4.22 зображено деталі для кріплення поворотного механізму маніпулятора.

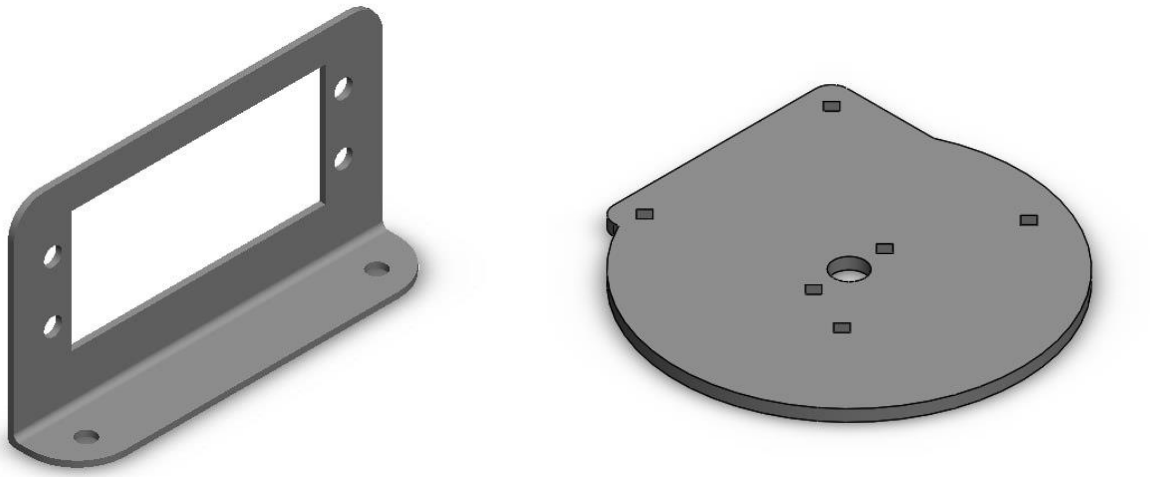


Рисунок 4.22 – Деталі для поворотного механізму

На рисунку 4.23 показано поворотний механізм.

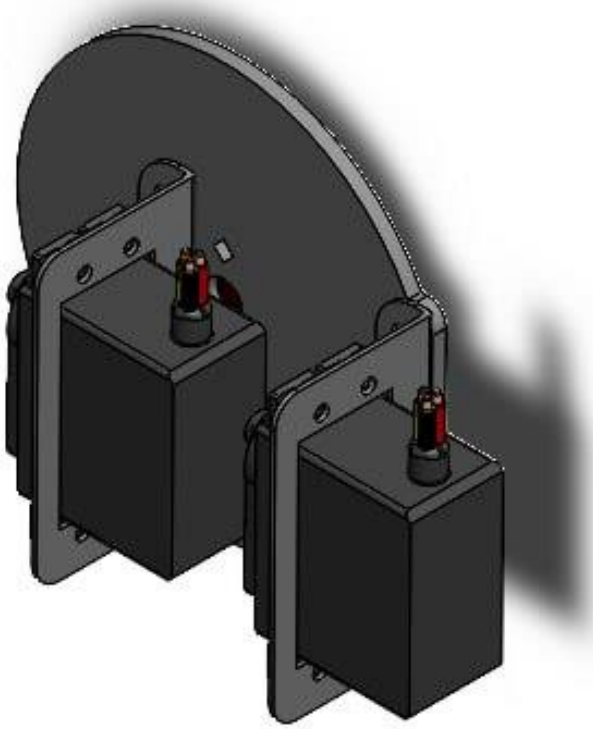


Рисунок 4.23 – Поворотний механізм маніпулятора

На рисунку 4.24 показана вертикальна стійка маніпулятора. Він призначений для підняття на певний рівень висоти захватну конструкцію.

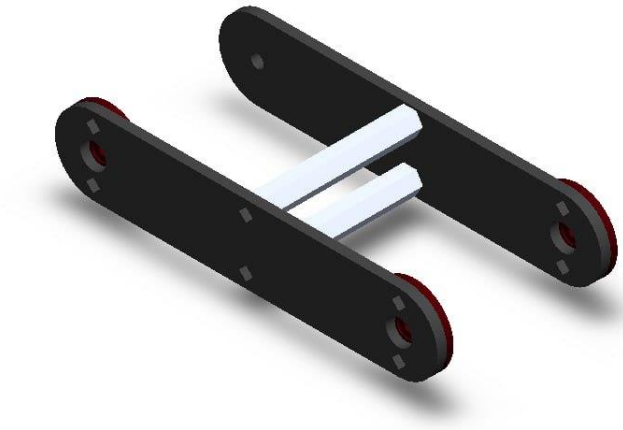


Рисунок 4.24 – Вертикальна стійка

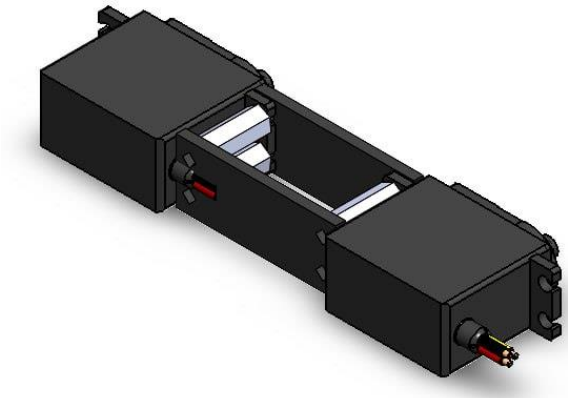


Рисунок 4.25 – Горизонтальна стійка

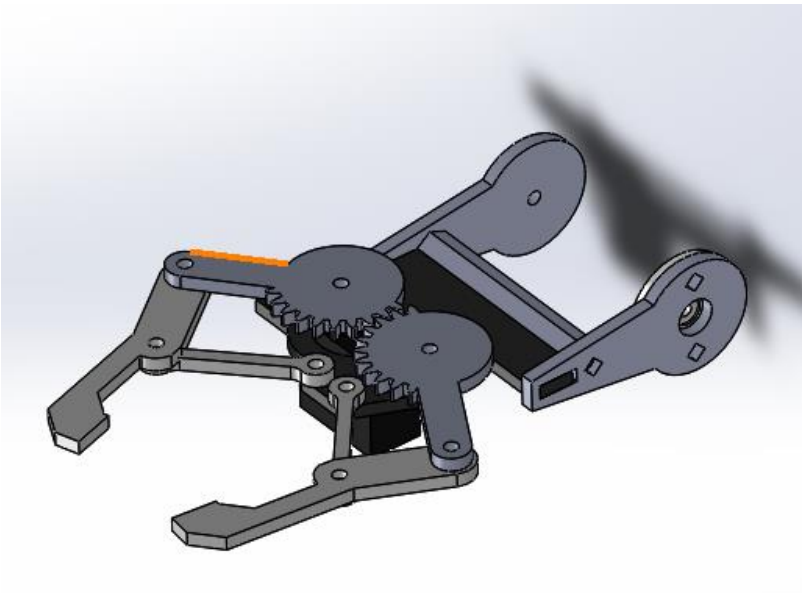


Рисунок 4.26 – Захватний механізм маніпулятора

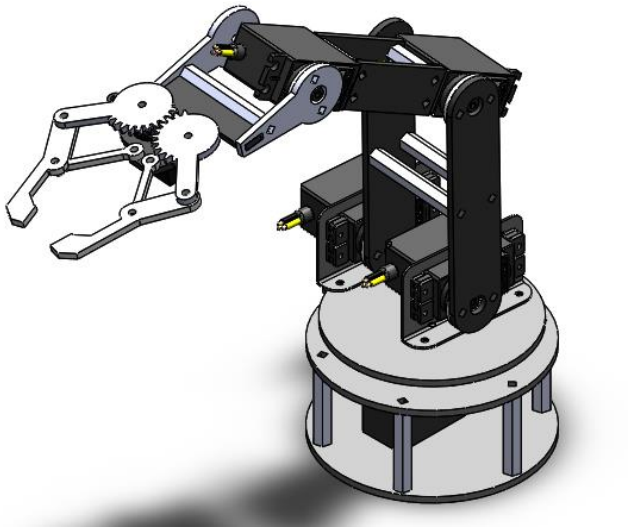


Рисунок 4.27 - 3D збірка повноцінної моделі маніпулятора

Основу маніпулятора прикріплюємо до платформи машинки за допомогою болтів.

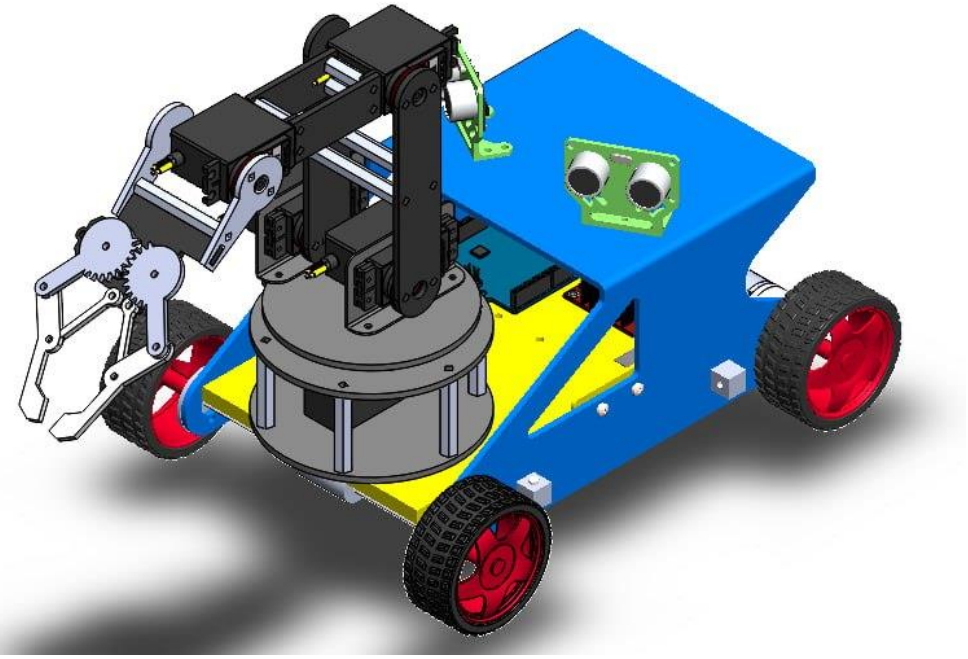


Рисунок 4.28 - 3D збірка повноцінної моделі робота

Для перевірки працездатності шестерні виконаємо симуляцію напруженого стану деталі за допомогою SOLIDWORKS Simulation [19]. Граничні умови показано на рис. 4.29. На зуб шестерні діє сила 5 Н. Помітно (рис. 4.29), що

напруження не перевищують 24 МПа, що менше допустимих 40 МПа для пластикових деталей.

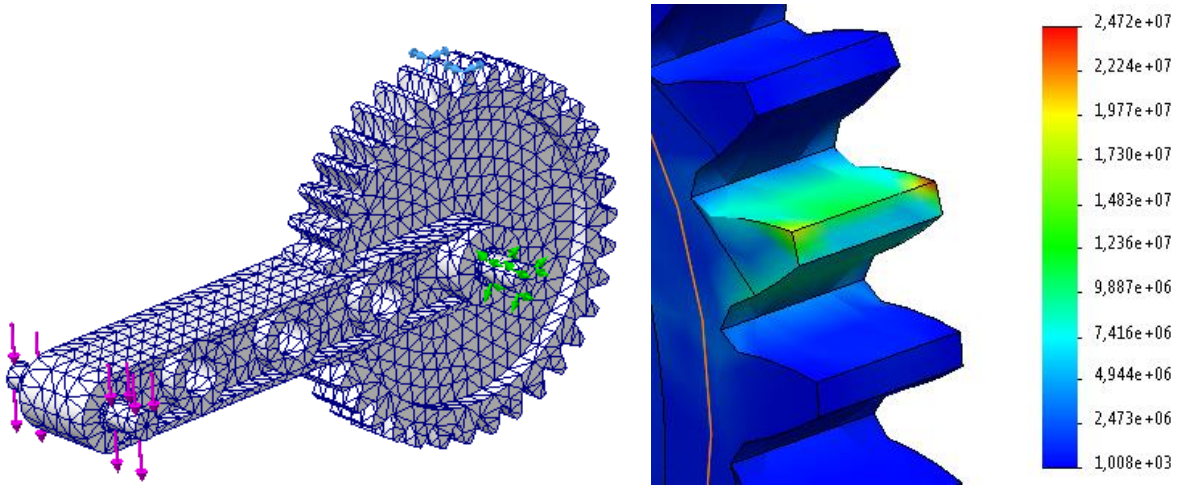


Рисунок 4.29 – Сітка, граничні умови і напруження Мізеса (Па)

4.2 Керуюча програма робота

В загальному програма керування роботом може мати довільну реалізацію, може бути створена будь-якою мовою програмування, використовувати будь-які сенсори і актуатори та способи зв’язку з Arduino.

Опис Python-програми для керування мобільним робота. Програма має назву ArduinoRoboCarTESTman.py.

Початок програми містись вказівку кодування символів:

```
# -*- coding: utf-8 -*-
```

Імпортуємо необхідні для роботи програми модулі, у першу чергу модуль *pyfirmata* який призначений для обміну інформацією плати Arduino і Python програми через протокол *firmata*. Цей протокол дозволяє керувати мобільним роботом з Python і не прошивати щоразу плату Arduino новою прошивкою. Таким чином ми відгладжуємо тільки Python програму. А сам скетч залишається стандартним.

```
import random
from pyfirmata import Arduino, util
```

Для того щоб робити зупинки програми на певну кількість секунд, потрібен модуль *time.*

Створюємо об’єкт під назвою *board*. Цей об’єкт описує плату Arduino, яка з’єднана з Python-програмою через порт *COM7* з швидкістю 57600. Якщо з’єднання вдале то буде виведена інформація за допомогою *print board*. Для нормальної роботи ультразвукового сенсора додати команди *it = util.Iterator(board)* та *it.start().*

```
board = Arduino('COM7', baudrate=57600)#, baudrate=9600
print board
it = util.Iterator(board)
it.start()
```

Дані команди призначенні для створення об’єктів для створення певних pin Arduino і їх налагодження:

```
pin8 = board.get_pin('d:8:o')
pin9 = board.get_pin('d:9:o')
pin10 = board.get_pin('d:10:o')
pin11 = board.get_pin('d:11:o')
servo1=board.get_pin('d:5:s')
servo2=board.get_pin('d:6:s')
echo_pin = board.get_pin('d:7:o')
autho=True
```

Наступна функція повертає середнє значення відстані, виміряної ультразвуковим сенсором де n - кількість вимірювань. Зазвичай значення n задається числом більше двох:

```
def ping(n):
    return sum([util.ping_time_to_distance(echo_pin.ping()) for i in [0]*n])/n
```

Наступна функція призначена для сканування ультразвуковим сенсором певного кута, сканування відбувається тільки по горизонталі:

```
def scan():
   servo2.write(70) # 70-130
   angle=0
   X=[]
   Y=[]
   while angle<=130:
      servo1.write(angle)
      time.sleep(0.9)
      dist=ping(3)
      print angle, dist
      X.append(angle)
      #Y.append(dist)
      Y.append(int(dist<40 and dist!=0))
      #print board.analog[0].read()
      angle+=13
      root.update_idletasks()
      root.update()
   return X,Y
```

Функція повний стоп для двигунів:

```
 def stop(t=0):
 pin8.write(0)
 pin9.write(0)
    pin10.write(0)
    pin11.write(0)
```

```
    if t: time.sleep(t)
```

Функція для руху робота прямо ліворуч:

```
def LF(t, s=True):
    pin10.write(0)
    pin11.write(1)
    time.sleep(t)
    if s: stop()
```

Функція для руху робота назад вліво:

```
def LB(t, s=True):
    pin10.write(1)
    pin11.write(0)
    time.sleep(t)
    if s: stop()
```

Функція для руху робота прямо праворуч:

```
def RF(t, s=True):
    pin8.write(0)
    pin9.write(1)
    time.sleep(t)
    if s: stop()
```

Функція для руху робота назад праворуч:

```
def RB(t, s=True):
    pin8.write(1)
```

```
    pin9.write(0)
    time.sleep(t)
    if s: stop()
```

Функція для руху робота прямо:

```
def F(t, s=True):
    RF(0.01, False)
    LF(t)
    if s: stop()
```

Функція для руху робота назад:

```
def B(t, s=True):
    RB(0.01, False)
    LB(t)
    if s: stop()
```

Функція яка дозволяє роботу розвертатися на певний кут:

```
def Rot(angle, ar):
    # розвернутись на кут
    k=0.01 # емпіричний коефіцієнт
    if (0<=angle<=ar/2.):
        LF(angle*k) # ліворуч
    elif (ar/2.<angle<=ar):
        RF((angle-ar/2.)*k) # праворуч
```

Функція визначення повороту оптимального кута повороту за результатами сканування:

```
def opt_angle(X,Y):
    angles=[] # оптимальні напрямки
    for x,y in zip(X,Y): # для усіх результатів сканування
        if y: # якщо в цьому напрямку щось є
            angles.append(x) # то додати до оптимальних кутів
    #angle=random.choice(angles) # вибрати випадковий напрямок
    angle=angles[len(angles)//2] # середній
    return angle
```

Функція стратегії робота. Ця функція реалізує стратегію для перемоги робота в змаганнях:

```
def strategy(i=0):
    X,Y=scan() # сканує
    visualize(Y)
    print X
    print Y
    if any(Y): # якщо щось знайдено
        angle=opt_angle(X,Y)
        print "angle=", angle
        Rot(angle, 130) # повернутись до об'єкта
        if i==1:
            F(2) # штовхати прямо
            B(2) # повернутись назад
        else:
            strategy(1) # рекурсія
    else: # якщо не знайдено
        angle=180*random.random() # випадковий напрямок
        Rot(angle, 180) # повернутись
```

```
    F(0.1) # їхати прямо
    #strategy() # рекурсія
```

Функція викликається кожну секунду функції стратегії.

```
def update():
    strategy()
    if autho: root.after(1000, update)
```

Функція яка запускається під час натискання певної клавіші з клавіатури:

```
def key_handler(event=None):
    global autho
    s=None
    if event:
        k=event.keycode
        print "keycode=",k
        if   k==39: s=RF # <вправо>
        elif k==37: s=LF # <вліво>
        elif k==40: s=B # <вниз>
        elif k==38: s=F # <вверх>
        elif k==34: servo1.write(servo1.value-10) # <PageDown>
        elif k==33: servo1.write(servo1.value+10) # <PageUp>
        elif k==36: servo2.write(servo2.value-10) # <Home>
        elif k==35: servo2.write(servo2.value+10) # <End>
        elif k==32: time.sleep(5) # <Space>
        elif k==80: autho=False # <P>
        elif k==83: autho=True; update() # <S>
        elif k==27: stop(); board.exit(); r.destroy() # вихід <Esc>
        else: s=None
        time.sleep(0.1)
```

```
        if s:
            s(0.2)
            time.sleep(0.1)
            print s.__name__
```

Обробка подій клавіатури з вікном:

```
class Evnt:
    def __init__(self, keycode):
        self.keycode=keycode
```

Функція призначення для створенням кнопки на вікні:

```
def button(text, keycode, relx, rely):
    btn=tk.Button(root, text=text, command=lambda: key_handler(Evnt(keycode)))
    btn.place(relx=relx, rely=rely, relwidth=0.3, relheight=0.1)
```

Функція візуалізація роботи ультразвукового сенсора. Вона рисує на вікні картину сканування. Код цієї функції був згенерований за допомогою чату GPT:

```
canvas_width = 100
canvas_height = 100
def visualize(sensor_data = [30, 50, 20, 40, 10, 35, 45]):
    scale_factor = canvas_height / (max(sensor_data)+1)
    canvas.delete("all")
    for i, distance in enumerate(sensor_data):
        # Calculate the x-coordinates for the bar
        x1 = i * (canvas_width / len(sensor_data))
        x2 = (i + 1) * (canvas_width / len(sensor_data))
        # Calculate the y-coordinates for the bar
        y1 = canvas_height
```

```
        y2 = canvas_height - (distance * scale_factor)
            # Draw the bar on the canvas
        canvas.create_rectangle(x1, y1, x2, y2, fill="blue")
```

Функція виконується у випадку запуску модуля:

```
if __name__=="__main__":
import Tkinter as tk
root = tk.Tk()
button("^", 38, 0.3, 0.0)
button("<", 37, 0.0, 0.1)
button(">", 39, 0.6, 0.1)
button("v", 40, 0.3, 0.2)
canvas = tk.Canvas(root, width=canvas_width, height=canvas_height, bg="white")
canvas.place(relx=0, rely=0.3, relwidth=1, relheight=0.7)
root.bind('<Key>', key_handler)
root.after(0, update)  # begin updates
root.mainloop()
stop()
board.exit()
```

Даний код потребує тестування з конкретним роботом. Для конкретного робота потрібно тонке налаштування цього коду.

ВИСНОВКИ

1.На основі огляду сучасного стану робототехніки і різних платформ для розроблення роботів виявлено, що платформа Arduino є однією з найпопулярніших і найдешевших платформ, яка має усі можливості застосовуватись в змаганнях мобільних роботів.

2.На основі огляду відомих змагань роботів і принципів участі в цих змаганнях виявлено, що найпростішим для реалізації типом змагань є змагання мобільних роботів, які намагаються ідентифікувати і перемістити об’єкт.

3.Виконано огляд принципів моделювання і симуляторів роботів. Для даного типу змагань вирішено застосувати для створення симулятора мову Python та її пакет для симуляції 2D-фізики Pymunk.

4.Підібрано прототип мобільного робота для змагань роботів і його деталі з використанням недорогих компонентів і платформи Arduino. Представлено характеристики технічного забезпечення, загальні відомості про сервоприводи та ультразвуковий датчик відстані HC-SR04. Наведено приклад встановлення з'єднання з ПК за допомогою Bluetooth модуля HC-05.

5.З використанням мови програмування Python, її графічного пакету Nodebox та пакету симуляції фізики Pymunk розроблено програму для симуляції змагання роботів. У першу чергу цю програму доцільно використовувати учасникам для відлагодження їхніх керуючих програм. За допомогою симулятора розроблені алгоритми ігрової стратегії робота.

6.З використанням SOLIDWORKS спроектовано конструкцію робота для змагань. Проектування прототипу мобільного робота характеризується зручними можливостями параметризації SOLIDWORKS. Модель робота є простою в налаштуванні і удосконаленні. Удосконалено конструкцію шляхом додання механізму захоплення.

7.На основі мови Python та протоколу Firmata розроблено і відлагоджено керуючу програму робота для змагань. Програма дозволяє ручне керування і

автономне, а також дозволяє розширити автономні можливості робота завдяки підключення додаткових Python-пакетів.

8. Розроблена навчально-ігрова мехатронна система «Змагання роботів» призначена для використання у навчальному процесі для підвищення мотивації студентів вивчення робототехніки. Але розроблені моделі, алгоритми і програми можуть бути використані і в практичних цілях. Наприклад для використання в складських задачах або для розмінування місцевості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іванов А. А. Основи робототехніки: навчальний посібник. Москва : "ФОРУМ", 2015. 224 с.

2. Попадюха Ю.А. Сучасні роботизовані комплекси, системи та пристрої: навчальний посібник. «Центр навчальної літератури», 2016. 324с.

3. Аронець О. Arduino для початківців : навчальний посібник. Івано-Франківськ : Симфонія форте, 2018. 192 с.

4. Гусляков О. М. Аналіз світового досвіду застосування та тенденції розвитку військових робототехнічних комплексів // Військово-технічний збірник. №6. 2012. С. 120–127

5. Круглов А. И. Классификация наземных мобильных роботов // Механіко-технологічні системи та комплекси // Вісник НТУ «ХПІ». 2017. №33(1255).

6. Robot competition. URL: https://en.wikipedia.org/wiki/Robot_competition (access 20.06.23).

7. Noosphere Engineering Race: як пройшли перші в Україні змагання на основі конструкторів VEX. URL: https://noosphereglobal.com/uk/noosphere-engineering-race-yak-projshli-pershi-v-ukrayini-zmagannya-na-osnovi-konstruktoriv-vex

8. Офіційний сайт відокремленого структурного підрозділу «Фаховий коледж електронних приладів Івано-Франківського національного технічного університету нафти і газу». URL: https://kep.nung.edu.ua/

9. Петренко Є. С. Обладнання, інструменти та пристрої для ручного розмінування // Спеціальна техніка, 2002. № 1.

10. Monk S. Programming Arduino: Getting Started with Sketches (3rd ed.). McGraw-Hill Education. 2022. ISBN 978-1264676989.

11. Єгоров О.Д. Практична енциклопедія Auduino : навчальний посібник. Запоріжжя: ЗНТУ, 2016. 166 с.

12. Макаров С. Л. Arduino Uno та Raspberry Pi 3: навчальний посібник. ДМК Пресс, 2010. 202 с.

13. Сервоприводи, будова та характеристики. URL: https://autohome.org.ua/market/moto_servo/servo-sg90-9g-detail

14. Micromouse. URL: https://en.wikipedia.org/wiki/Micromouse (access 20.06.23)

15. Козяр М.М., Фещук Ю.В., Парфенюк О.В. Комп'ютерна графіка SolidWorks : Навчальний посібник. Херсон: Олді-Плюс, 2018. 252 с.

16. Основні елементи SolidWorks 2011. Training. SolidWorks Corporation, 2011. 541 с.

17. Arduino Uno pinout. URL: https://www.circuito.io/blog/arduino-uno-pinout/

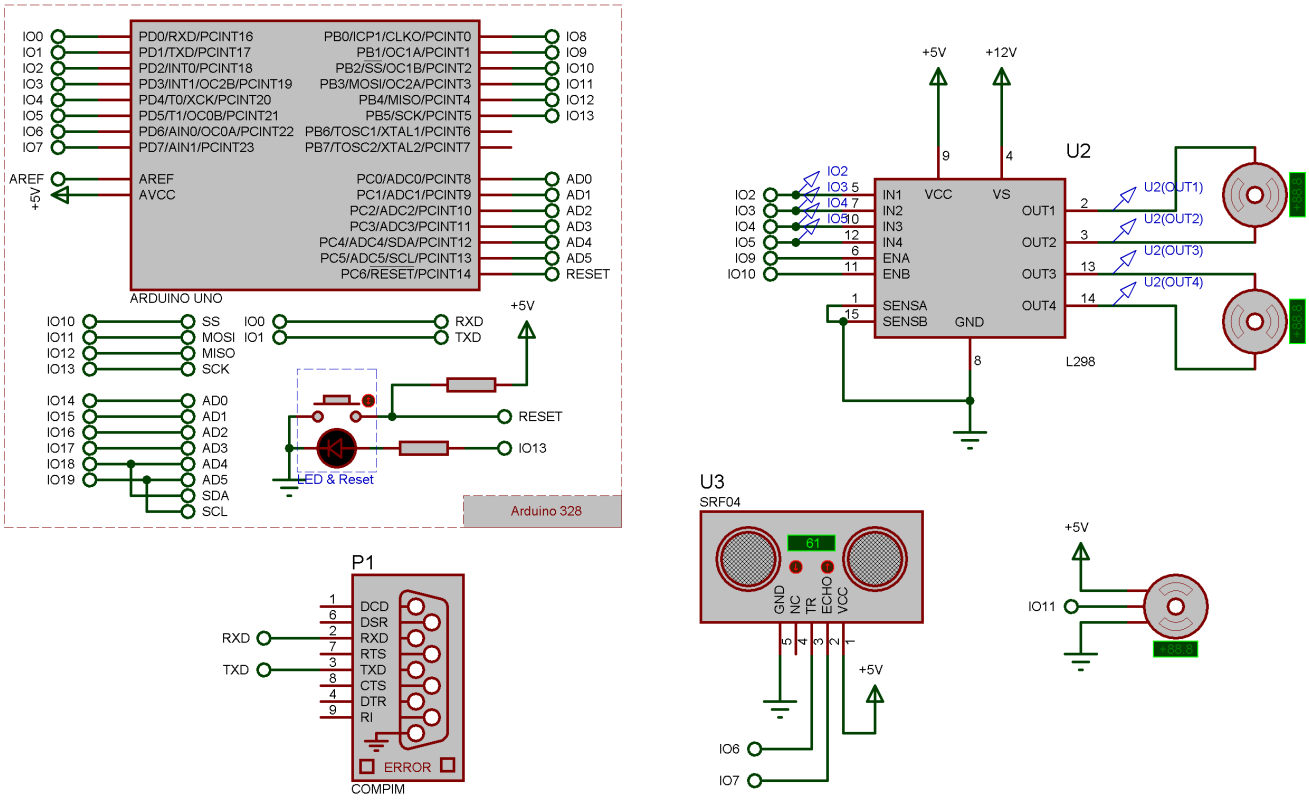
18. Копей В. Б. Мова програмування Python для інженерів і науковців : навчальний посібник. Івано-Франківськ : ІФНТУНГ, 2019. 272 с.

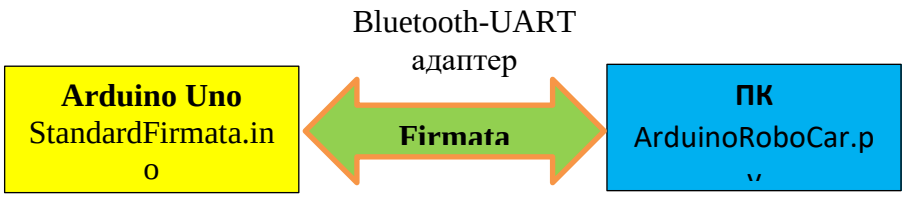
19. Копей Б.В., Копей В.Б. Використання методу скінченних елементів та тривимірного комп'ютерного моделювання для конструювання та оптимізації параметрів нафтогазового обладнання: Навчальний посібник. Івано-Франківськ : Факел, 2008. 117 с.

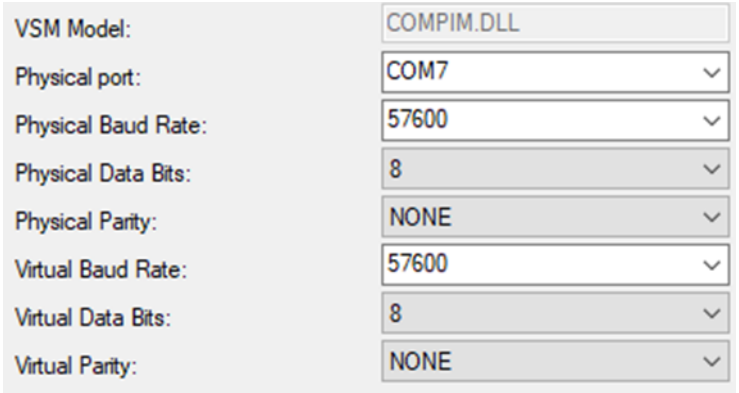
20. Копей В.Б., Онисько О.Р., Борущак Л.О., Роп'як Л.Я. Автоматизоване проектування різальних інструментів: Навчальний посібник. Івано-Франківськ : ІФНТУНГ, 2012. 208 с.

ДОДАТКИ

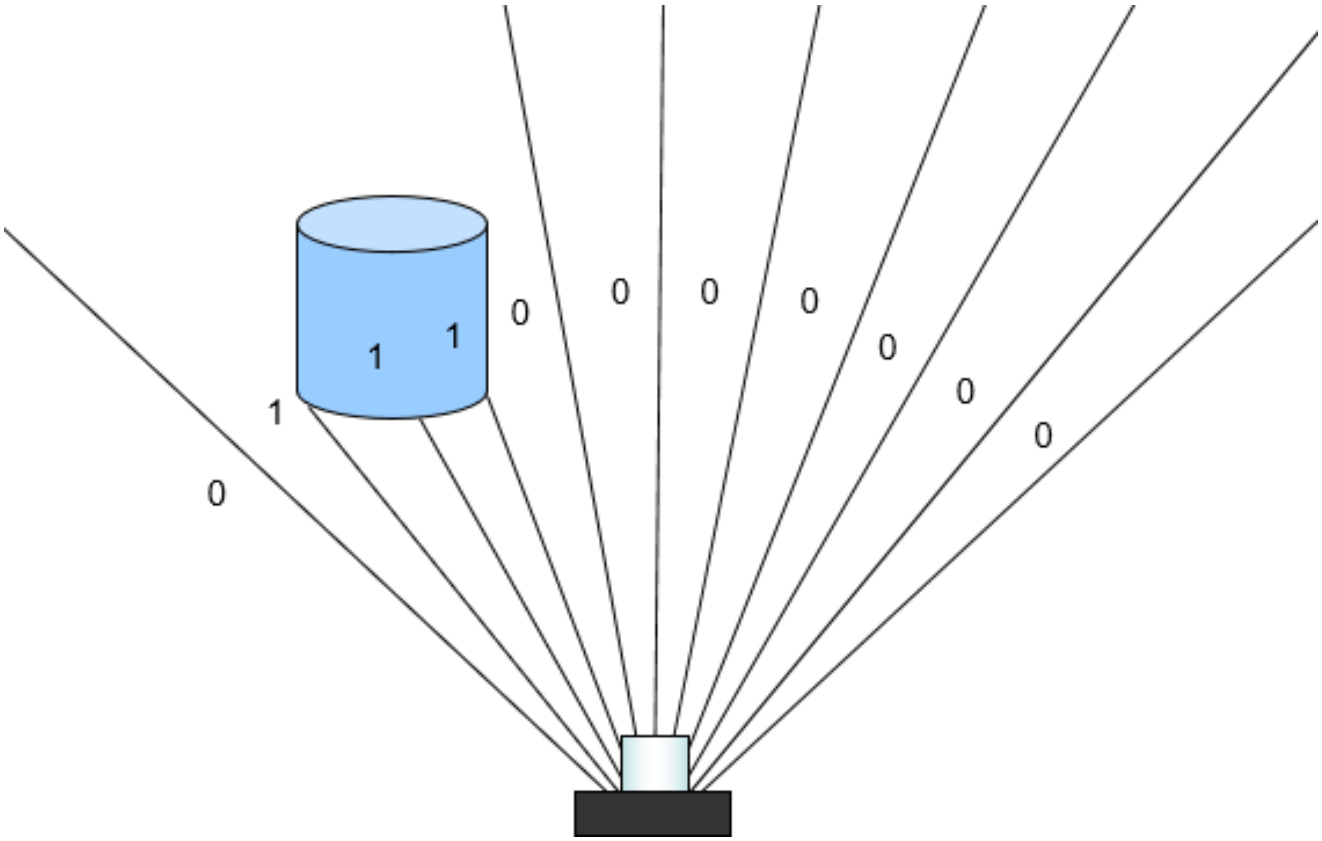
Додаток А – Принципова електрична схема робота







Додаток Б – Схема сканування ультразвуковим датчиком HC-SR04



Додаток В – Програма для моделювання змагань роботів

```
#encoding: utf-8
from __future__ import division
from nodebox.graphics import *
import pymunk
import pymunk.pyglet_util
import random, math

space = pymunk.Space()

def createBody(x,y,shape,*shapeArgs):
    body = pymunk.Body()
    body.position = x, y
    s = shape(body, *shapeArgs)
    s.mass = 1
    s.friction = 1
    space.add(body, s)
    return s #shape!!!

s0=createBody(300, 300, pymunk.Poly, ((-20,-5),(-20,5),(20,15),(20,-15)))
s0.score=0
s3=createBody(200, 300, pymunk.Poly, ((-20,-5),(-20,5),(20,15),(20,-15)))
s3.color = (0, 255, 0, 255)
s3.score=0
s1=createBody(300, 200, pymunk.Circle, 10, (0,0))
S2=[]
for i in range(10):
    s2=createBody(350, 250, pymunk.Circle, 10, (0,0))
    s2.color = (255, 0, 0, 255)
    S2.append(s2)


def getAngle(x,y,x1,y1):
    return math.atan2(y1-y, x1-x)

def getDist(x,y,x1,y1):
    return ((x-x1)**2+(y-y1)**2)**0.5

def inCircle(x,y,cx,cy,R):
    if (x-cx)**2+(y-cy)**2 < R**2:
        return True
    return False
```

```
def inSector(x,y,cx,cy,R,a):
    angle=getAngle(cx,cy,x,y)
    if inCircle(x,y,cx,cy,R) and a-0.5<angle<a+0.5:
        return True
    return False

def strategy(b=s3.body):
    v=100
    a=b.angle
    b.velocity=v*cos(a), v*sin(a)
    x,y=b.position
    R=getDist(x,y,350,250)
    #print R, b.angle
    line(x,y,*s1.body.position,stroke=Color(0))
    if canvas.frame%100==0:
        if R>180:
            b.angle=getAngle(x,y,350,250)
        else:
            b.angle=getAngle(x,y,*s1.body.position) #2*math.pi*random.random()

def strategy2(b=s3.body):
    u"""Стратегія з випадковим рухом робота, який сканує сектор ультразвуковим сенсором"""
    v=100
    a=b.angle
    b.velocity=v*cos(a), v*sin(a)
    x,y=b.position
    R=getDist(x,y,350,250)
    ellipse(x, y, 200, 200, stroke=Color(0.5))
    #line(x,y,x+100*cos(a),y+100*sin(a),stroke=Color(0.5))
    line(x,y,x+100*cos(a+0.5),y+100*sin(a+0.5),stroke=Color(0.5))
    line(x,y,x+100*cos(a-0.5),y+100*sin(a-0.5),stroke=Color(0.5))
    inS=inSector(s1.body.position[0], s1.body.position[1], x, y, 100, a)
    if inS:
        b.angle=getAngle(x,y,*s1.body.position)
        print b.angle
    if canvas.frame%100==0:
        if R>180:
            b.angle=getAngle(x,y,350,250)
        elif not inS:
            b.angle=2*math.pi*random.random()

def scr(s,s0,s3,p=1):
```

```
    bx,by=s.body.position
    s0x,s0y=s0.body.position
    s3x,s3y=s3.body.position
    if not inCircle(bx,by,350,250,180):
        if getDist(bx,by,s0x,s0y)>getDist(bx,by,s3x,s3y):
            s0.score=s0.score+p
        else:
            s3.score=s3.score+p
        s.body.position=random.randint(200,400),random.randint(200,300)

def score():
    u"""визначає переможця"""
    scr(s1,s0,s3)
    for s in S2:
        scr(s,s0,s3,p=-1)


def manualControl():
    u"""Керування роботом з мишки або клавіатури"""
    v=10 # швидкість
    b=s0.body
    a=b.angle
    x,y=b.position
    vx,vy=b.velocity
    if canvas.keys.char=="a":
        b.angle-=0.1
    if canvas.keys.char=="d":
        b.angle+=0.1
    if canvas.keys.char=="w":
        b.velocity=vx+v*cos(a), vy+v*sin(a)
    if canvas.mouse.button==LEFT:
        b.angle=getAngle(x,y,*canvas.mouse.xy)
        b.velocity=vx+v*cos(a), vy+v*sin(a)

def simFriction():
    for s in [s0,s1,s3]+S2:
        s.body.velocity=s.body.velocity[0]*0.9, s.body.velocity[1]*0.9
        s.body.angular_velocity=s.body.angular_velocity*0.9
        #s.body.update_velocity(s.body, (0.,0.), 0.9, 0.02)

draw_options = pymunk.pyglet_util.DrawOptions()

def draw(canvas):
    canvas.clear()
```

```
    fill(0,0,0,1)
    text("%i %i"%(s0.score,s3.score),20,20)
    nofill()
    ellipse(350, 250, 350, 350, stroke=Color(0))
    manualControl()
    strategy2()
    score()
    simFriction()
    space.step(0.02)
    space.debug_draw(draw_options)

canvas.size = 700, 500
canvas.run(draw)
```

Додаток Г – Керуюча програма робота

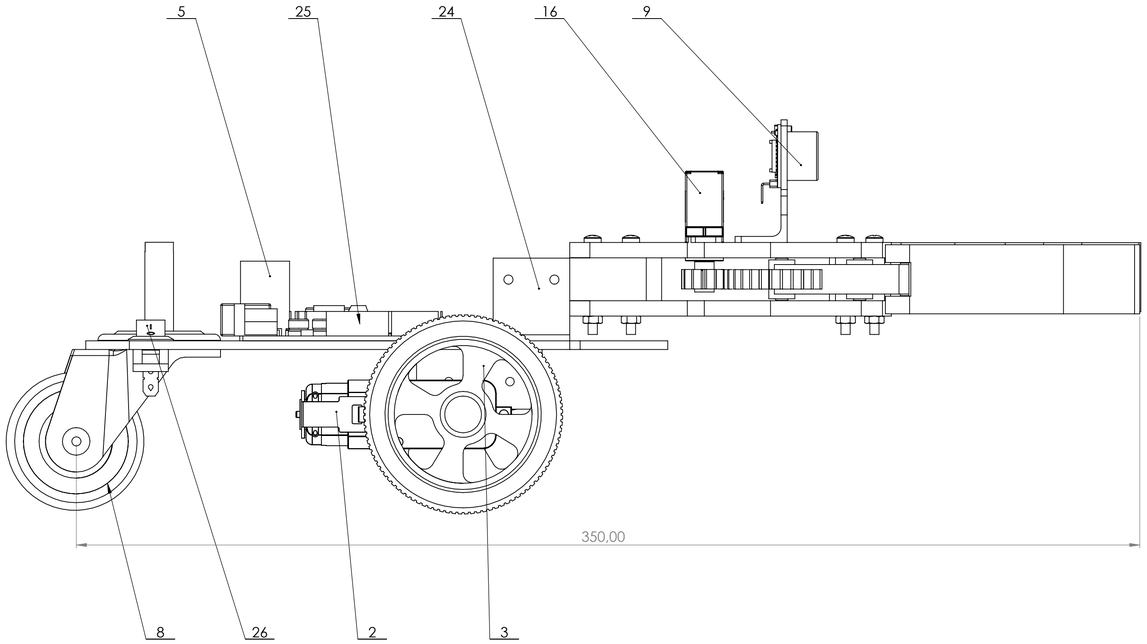
```
# -*- coding: utf-8 -*-

import random
from pyfirmata import Arduino, util
time=util.time
board = Arduino('COM7', baudrate=57600)#, baudrate=9600
print board
it = util.Iterator(board)
it.start()
#board.analog[0].enable_reporting()
pin8 = board.get_pin('d:8:o')
pin9 = board.get_pin('d:9:o')
pin10 = board.get_pin('d:10:o')
pin11 = board.get_pin('d:11:o')
servo1=board.get_pin('d:5:s')
servo2=board.get_pin('d:6:s')
echo_pin = board.get_pin('d:7:o')
autho=True

def ping(n):
   "Повертає середнє значення відстані, виміряної ультразвуковим сенсором, n - кількість вимірювань"
   return sum([util.ping_time_to_distance(echo_pin.ping()) for i in [0]*n])/n

def scan():
   servo2.write(70) # 70-130
   angle=0
   X=[]
   Y=[]
   while angle<=130:
      servo1.write(angle)
      time.sleep(0.9)
      dist=ping(3)
      print angle, dist
      X.append(angle)
      #Y.append(dist)
      Y.append(int(dist<40 and dist!=0))
      #print board.analog[0].read()
      angle+=13
      root.update_idletasks()
      root.update()
   return X,Y
```

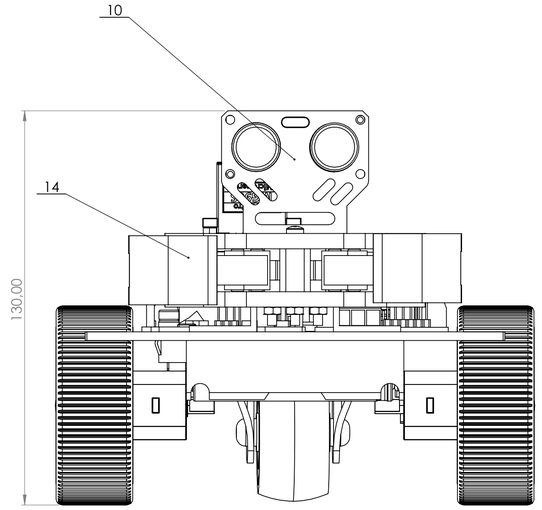
```
def stop(t=0):
    pin8.write(0)
    pin9.write(0)
    pin10.write(0)
    pin11.write(0)
    if t: time.sleep(t)

def LF(t, s=True):
    pin10.write(0)
    pin11.write(1)
    time.sleep(t)
    if s: stop()

def LB(t, s=True):
    pin10.write(1)
    pin11.write(0)
    time.sleep(t)
    if s: stop()

def RF(t, s=True):
    pin8.write(0)
    pin9.write(1)
    time.sleep(t)
    if s: stop()

def RB(t, s=True):
    pin8.write(1)
    pin9.write(0)
    time.sleep(t)
    if s: stop()

def F(t, s=True):
    RF(0.01, False)
    LF(t)
    if s: stop()

def B(t, s=True):
    RB(0.01, False)
    LB(t)
    if s: stop()

def Rot(angle, ar):
    # розвернутись на кут
```

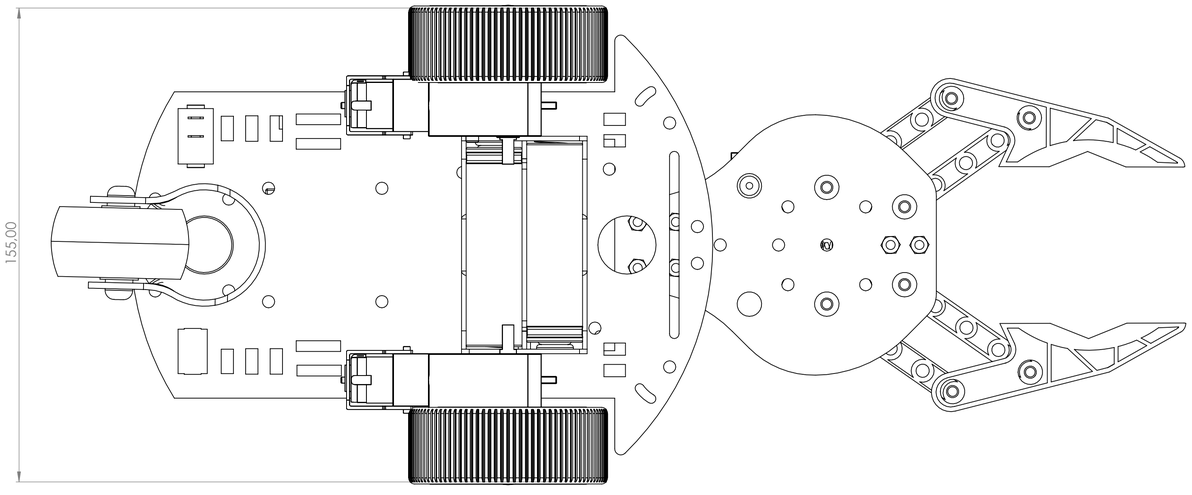
```
    k=0.01 # емпіричний коефіцієнт
    if (0<=angle<=ar/2.):
        LF(angle*k) # ліворуч
    elif (ar/2.<angle<=ar):
        RF((angle-ar/2.)*k) # праворуч

def opt_angle(X,Y):
    """Оптимальний кут повороту за результатами сканування"""
    angles=[] # оптимальні напрямки
    for x,y in zip(X,Y): # для усіх результатів сканування
        if y: # якщо в цьому напрямку щось є
            angles.append(x) # то додати до оптимальних кутів
    #angle=random.choice(angles) # вибрати випадковий напрямок
    angle=angles[len(angles)//2] # середній
    return angle

def strategy(i=0):
    """Стратегія робота"""
    X,Y=scan() # сканує
    visualize(Y)
    print X
    print Y
    if any(Y): # якщо щось знайдено
        angle=opt_angle(X,Y)
        print "angle=", angle
        Rot(angle, 130) # повернутись до об'єкта
        if i==1:
            F(2) # штовхати прямо
            B(2) # повернутись назад
        else:
            strategy(1) # рекурсія?
    else: # якщо не знайдено
        angle=180*random.random() # випадковий напрямок
        Rot(angle, 180) # повернутись
        F(0.1) # іхати прямо
        #strategy() # рекурсія?

def update():
    """Викликається кожну секунду"""
    strategy()
    if autho: root.after(1000, update)

def key_handler(event=None):
    global autho
```

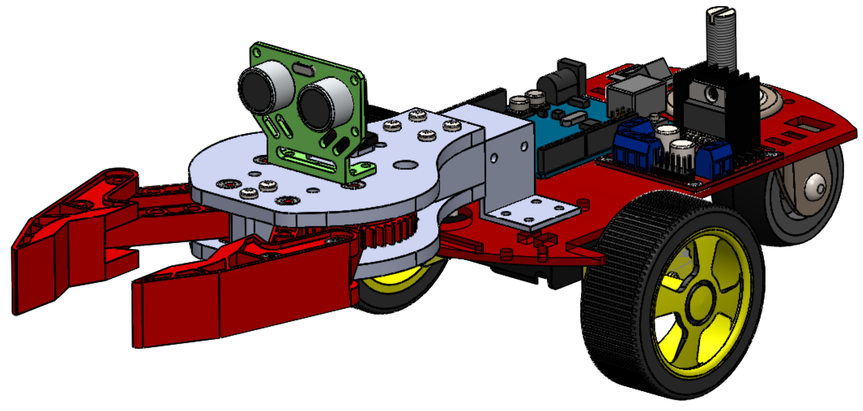
```
    s=None
    if event:
        k=event.keycode
        print "keycode=",k
        if   k==39: s=RF # <вправо>
        elif k==37: s=LF # <вліво>
        elif k==40: s=B # <вниз>
        elif k==38: s=F # <вверх>
        elif k==34: servo1.write(servo1.value-10) # <PageDown>
        elif k==33: servo1.write(servo1.value+10) # <PageUp>
        elif k==36: servo2.write(servo2.value-10) # <Home>
        elif k==35: servo2.write(servo2.value+10) # <End>
        elif k==32: time.sleep(5) # <Space>
        elif k==80: autho=False # <P>
        elif k==83: autho=True; update() # <S>
        elif k==27: stop(); board.exit(); r.destroy() # вихід <Esc>
        else: s=None
        time.sleep(0.1)
        if s:
            s(0.2)
            time.sleep(0.1)
            print s.__name__

# while True:
#     ball.draw()
#     tk.update_idletasks()
#     tk.update()

class Evnt:
    def __init__(self, keycode):
        self.keycode=keycode

def button(text, keycode, relx, rely):
    btn=tk.Button(root, text=text, command=lambda: key_handler(Evnt(keycode)))
    btn.place(relx=relx, rely=rely, relwidth=0.3, relheight=0.1)

canvas_width = 100
canvas_height = 100
def visualize(sensor_data = [30, 50, 20, 40, 10, 35, 45]):
    scale_factor = canvas_height / (max(sensor_data)+1)
    canvas.delete("all")
    for i, distance in enumerate(sensor_data):
        # Calculate the x-coordinates for the bar
        x1 = i * (canvas_width / len(sensor_data))
```

```
        x2 = (i + 1) * (canvas_width / len(sensor_data))

        # Calculate the y-coordinates for the bar
        y1 = canvas_height
        y2 = canvas_height - (distance * scale_factor)

        # Draw the bar on the canvas
        canvas.create_rectangle(x1, y1, x2, y2, fill="blue")

if __name__=="__main__":
    import Tkinter as tk
    root = tk.Tk()
    button("^", 38, 0.3, 0.0)
    button("<", 37, 0.0, 0.1)
    button(">", 39, 0.6, 0.1)
    button("v", 40, 0.3, 0.2)
    canvas = tk.Canvas(root, width=canvas_width, height=canvas_height, bg="white")
    canvas.place(relx=0, rely=0.3, relwidth=1, relheight=0.7)
    root.bind('<Key>', key_handler)
    root.after(0, update)  # begin updates
    root.mainloop()
    stop()
    board.exit()
```









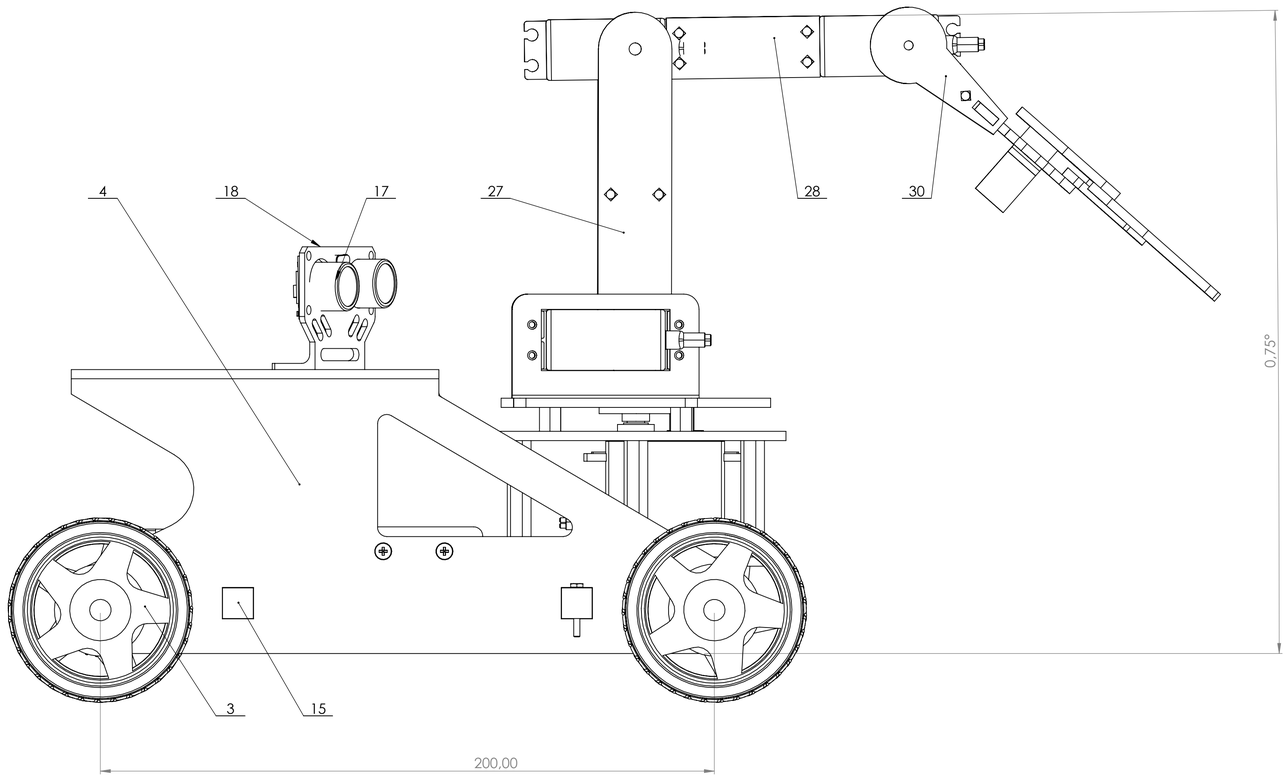
						БК.ПМІ-02.00.00.001 СК	
Изм.	Лист	№ докум.	Подп.	Дата	Робот для змагань з механізмом захоплення	Лит.	Масса / Масштаб
Разраб.		Бережанський					1:2
Пров.		Копей В.Б				Лист 1	Листов 1
Т. контр.		Копей В.Б.				ІФНТУНГ ПМІ-21-1к	
Н. контр.							
Утв.		Панчук В.Г					

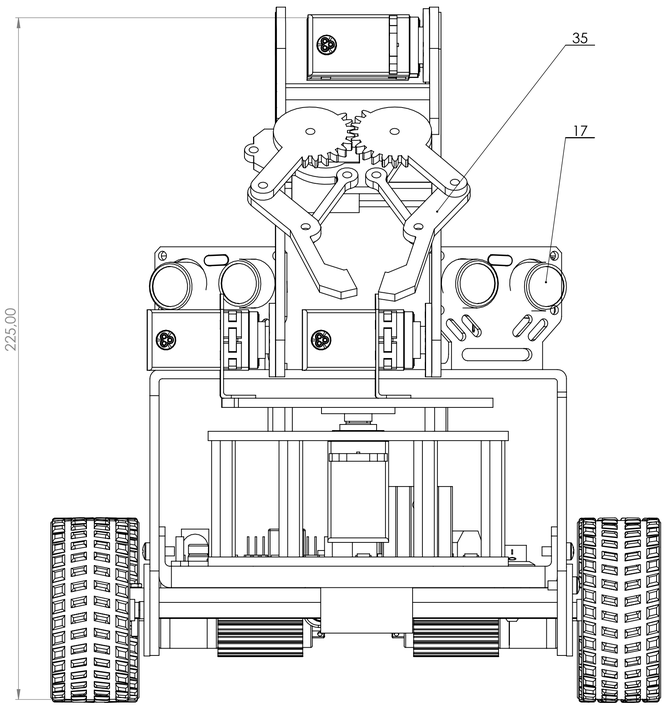
Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

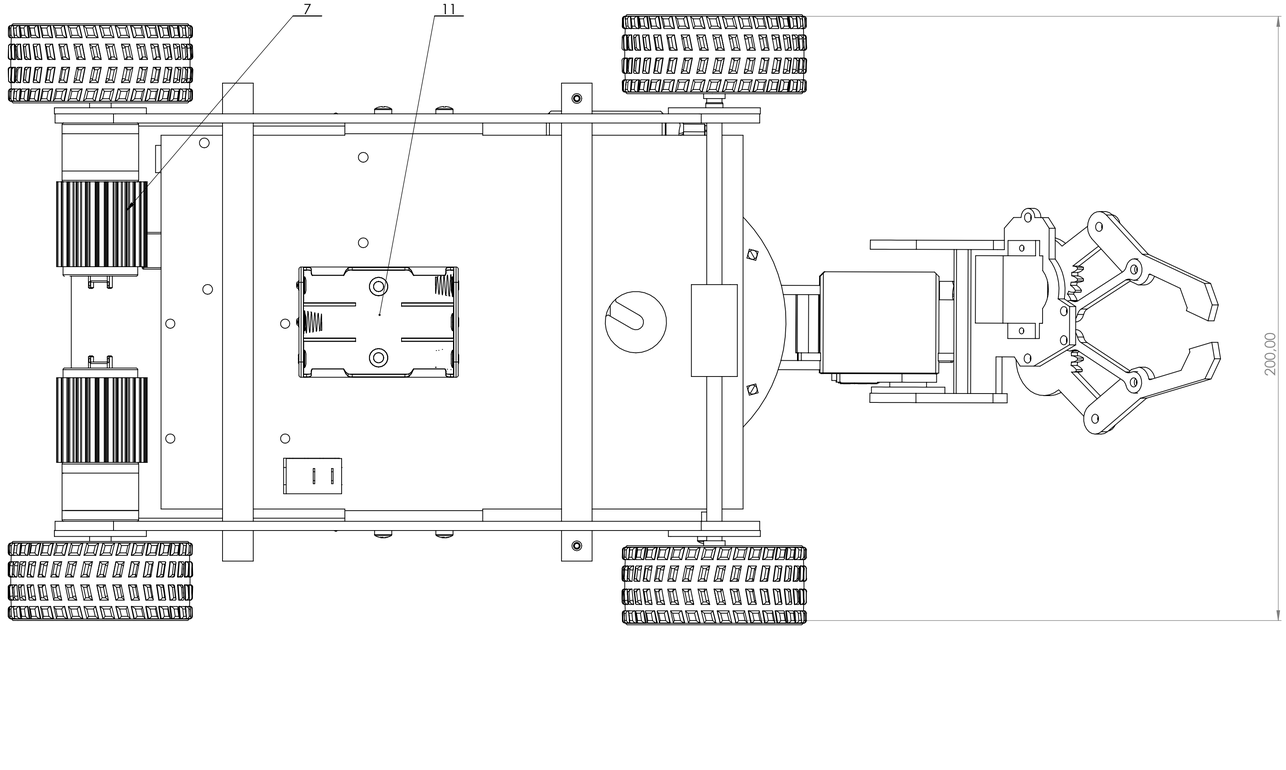
Копировал | Формат А1

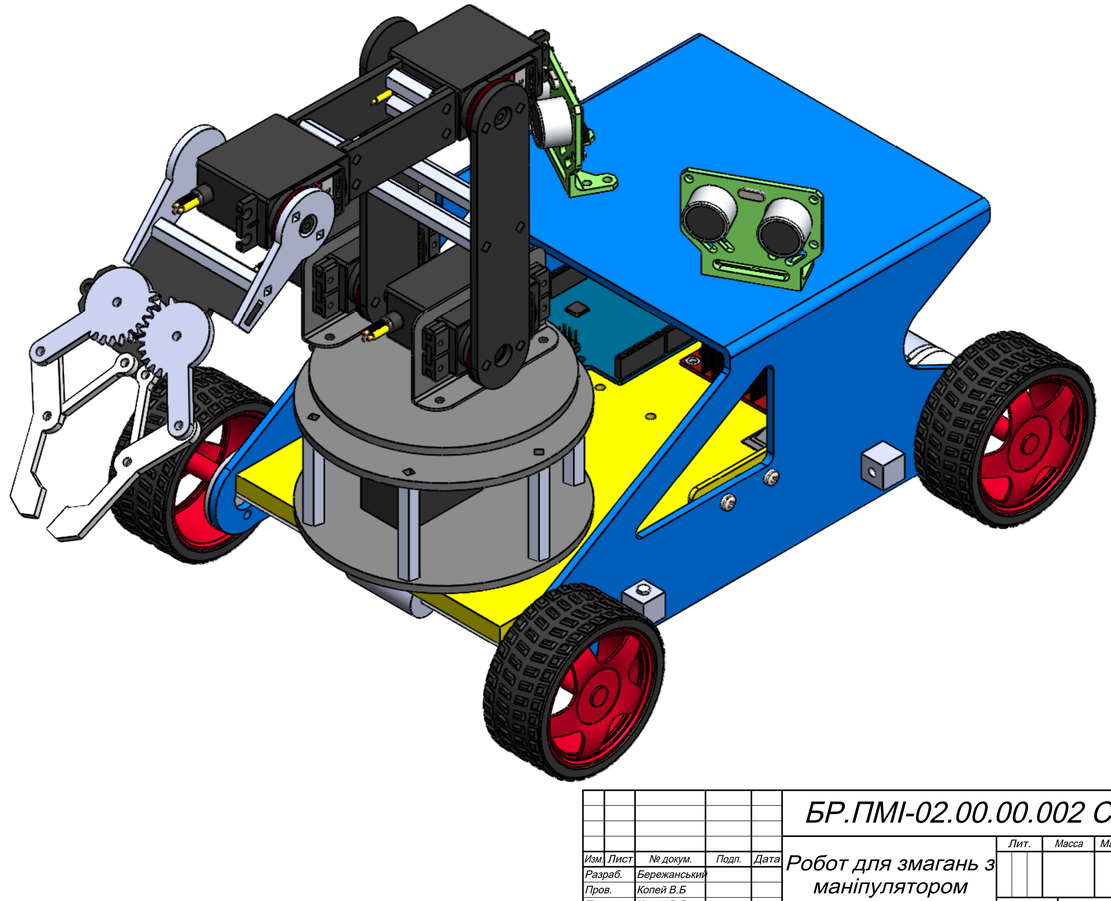
Файл: fire fighting robot

БР.ПМІ-02.00.00.002 СК









						БР.ПМІ-02.00.00.002 СК	
Изм.	Лист	№ докум.	Подп.	Дата	Робот для змагань з маніпулятором	Лит.	Масса / Масштаб
Разраб.		Бережанський					1:5
Пров.		Копей В.Б				Лист 1	Листов 1
Т. контр.		Копей В.Б.				ІФНТУНГ ПМІ-21-1к	
Н. контр.							
Утв.		Панчук В.Г.					

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал | Формат А1

Файл: Робот з маніпулятором А1-2

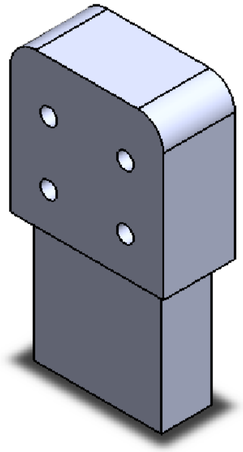


Рисунок 1.1 - 3D молель для захоплювача

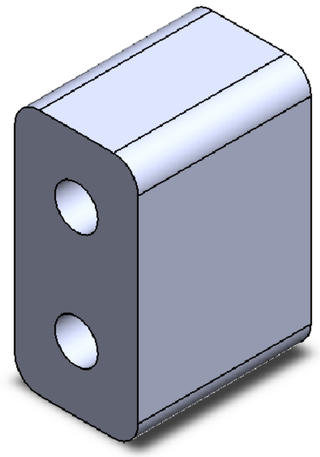


Рисунок 1.2 - 3D молель для захоплювача

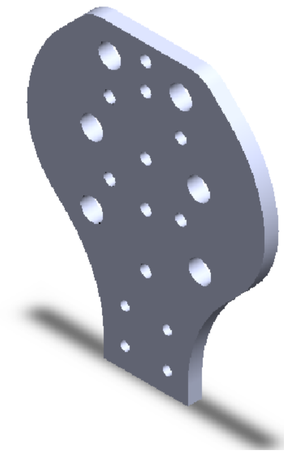


Рисунок 1.3 - 3D молель основи для захоплювача

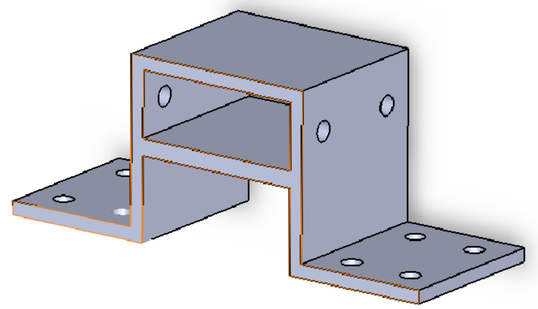


Рисунок 1.4 - 3D молель тримача для захоплювача

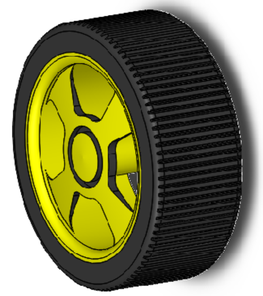


Рисунок 1.5 - 3D молель колеса

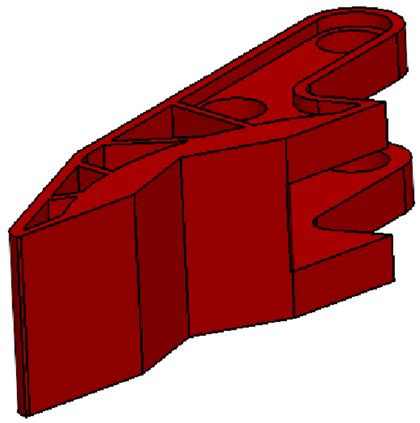


Рисунок 1. 6 - 3D молель клешні

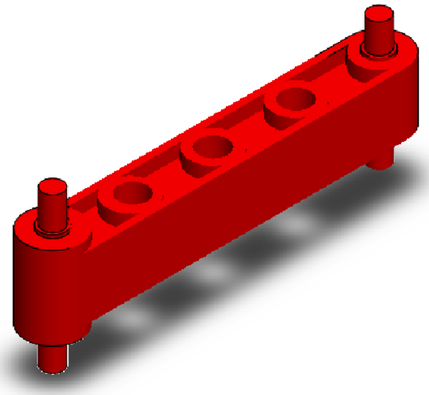


Рисунок 1. 7 - 3D молель деталі захватного механізму

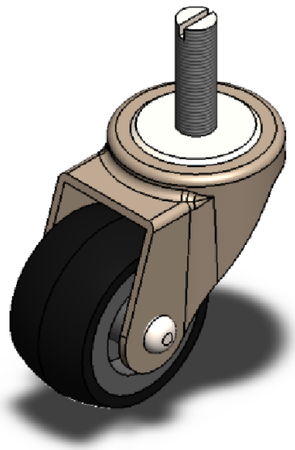


Рисунок 1. 8 - 3D молель колеса з підшипником

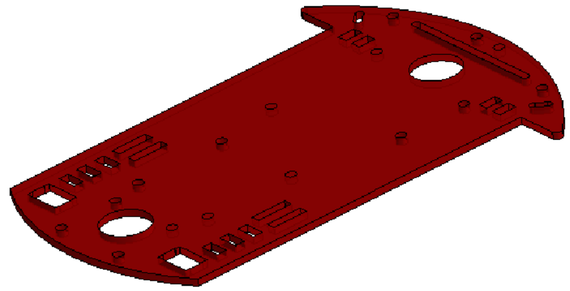


Рисунок 1.9 - 3D молель основи

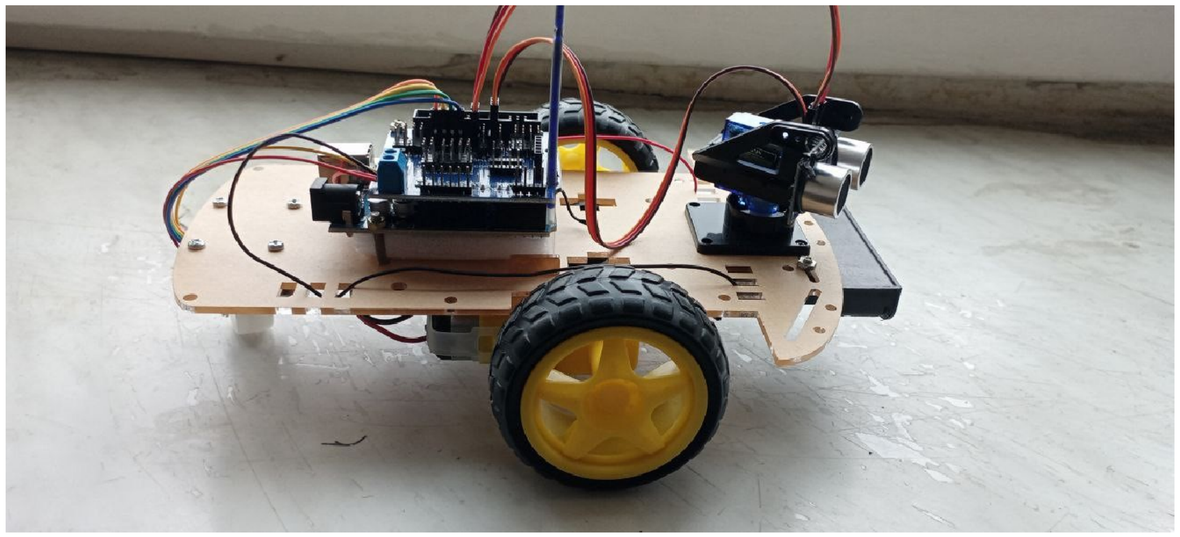


Рисунок 1.12 - Робот для змагань роботів

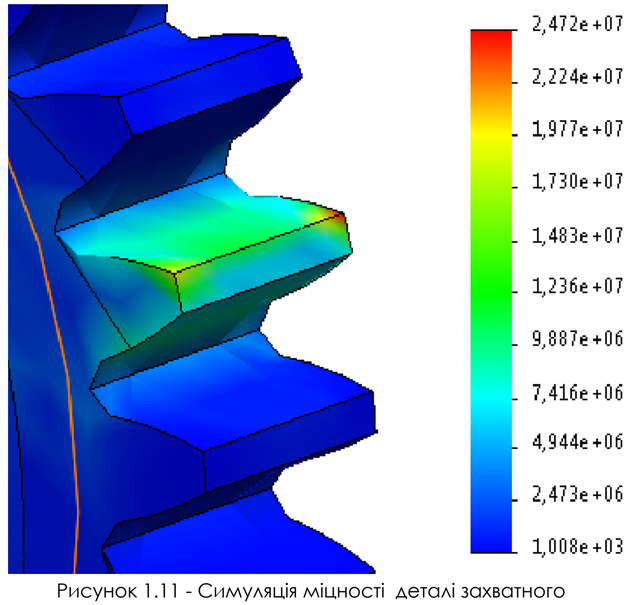


Рисунок 1.11 - Симуляція міцності деталі захватного механізму

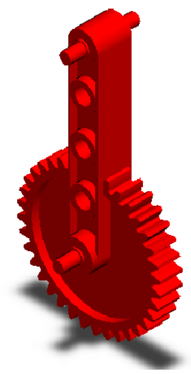


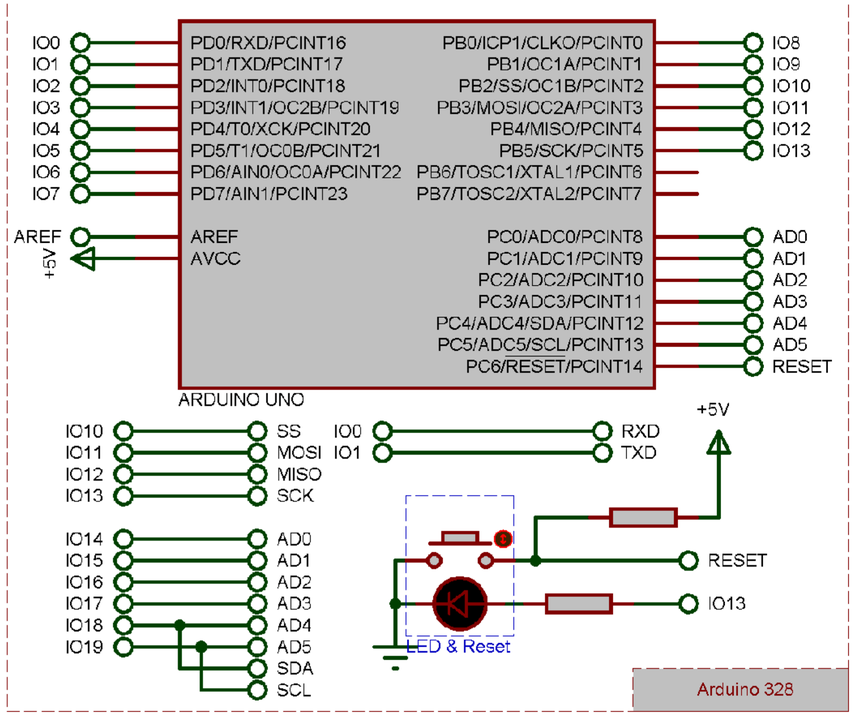
Рисунок 1.10 - 3D модель деталі захватного механізму

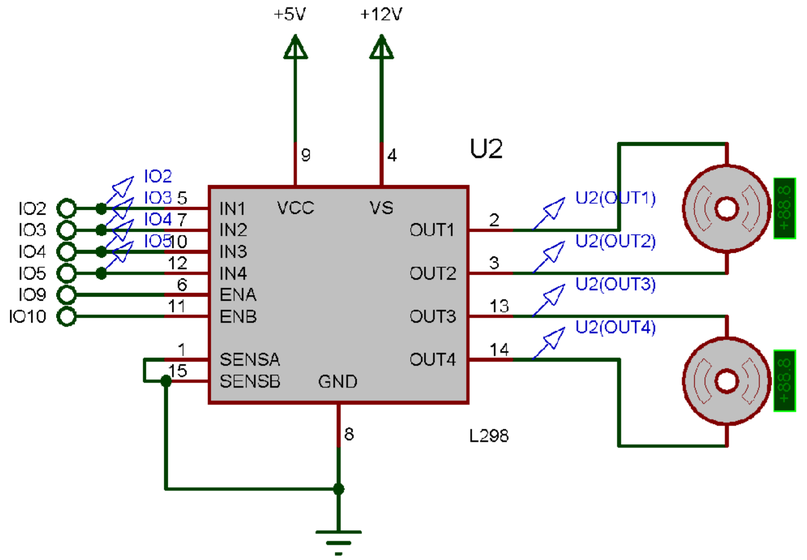
						БР.ПМІ-02.00.00.005	
Изм.	Лист.	№ докум.	Подп.	Дата			
Разраб.		Бережанський				Проектування деталей робота в SOLIDWORKS	Лит. / Масса / Масштаб: 1:1
Пров.		Копей В.Б.					
Т. контр.		Копей В.Б.					Лист 1 / Листов 1
Н. контр.							
Утв.		Панчук В.Г.					

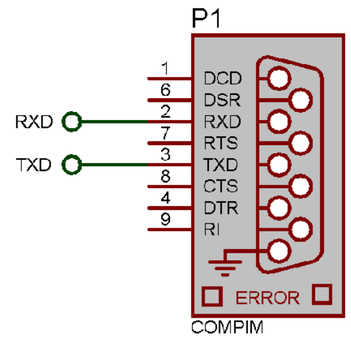
Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

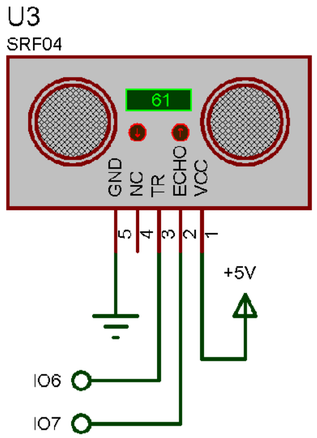
Копировал | Формат А1

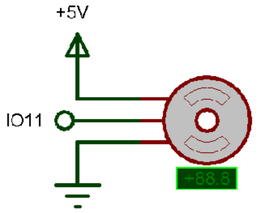
Файл: Деталі робота А1-5

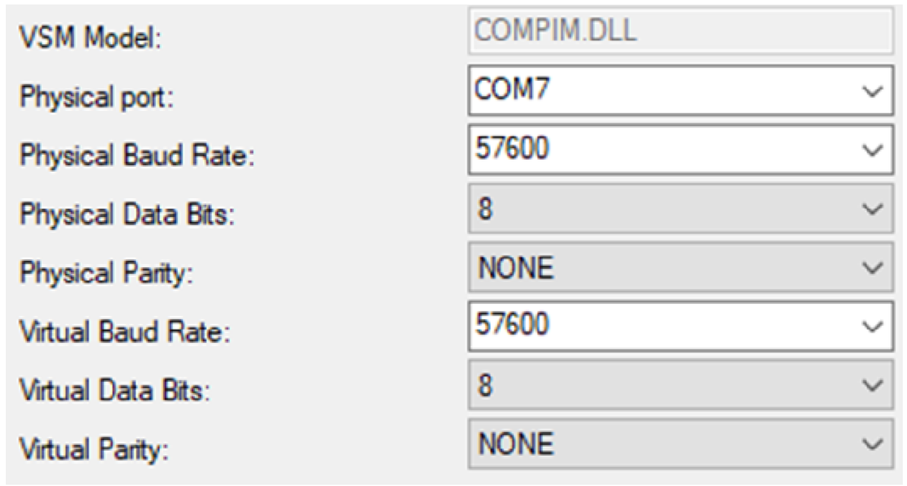


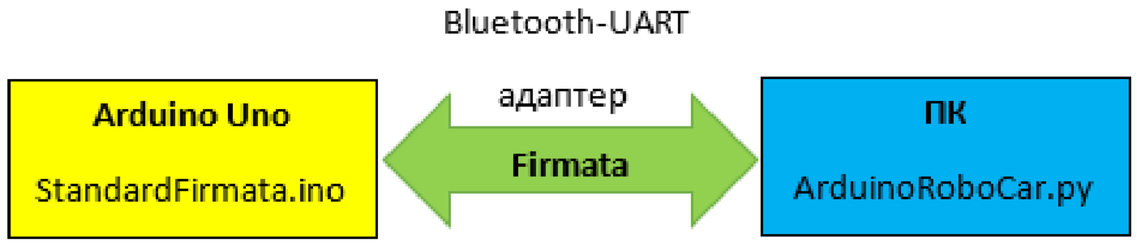










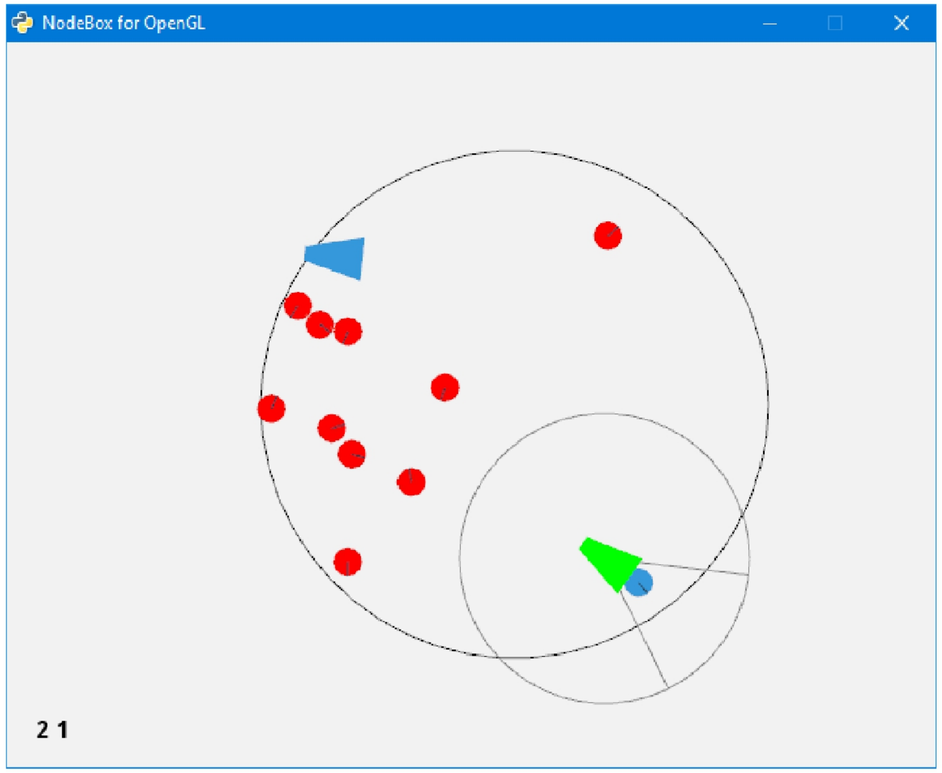


						БР.ПМІ-02.00.00.004	
Изм.	Лист	№ докум.	Подп.	Дата			
Разраб.		Бережанський			Принципова схема	Лит.	Масса	Масштаб
Пров.		Копей В.Б						1:1
Т. контр.		Копей В.Б.				Лист 1	Листов 1	
Н. контр.						ІФНТУНГ ПМІ-21-1К	
Утв.		Панчук В.Г.					

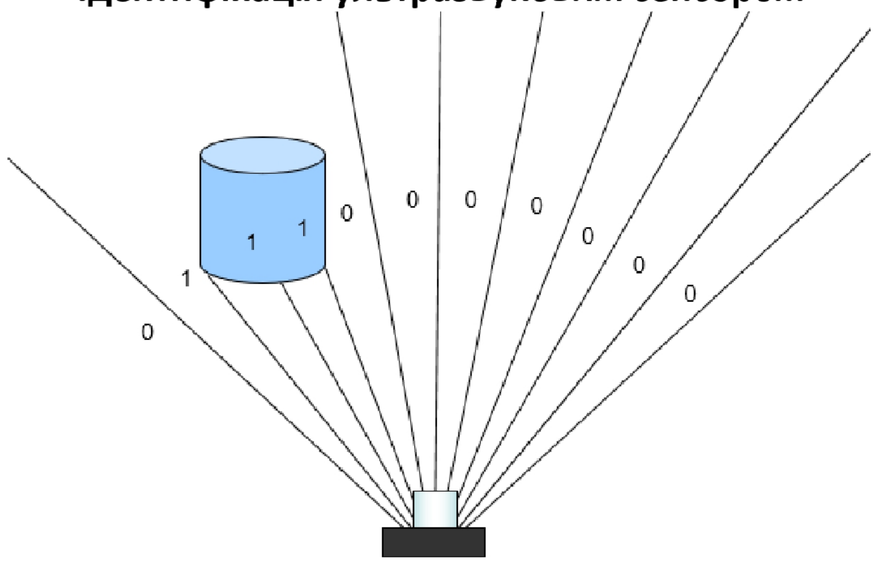
Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал | Формат А1 | Файл: Лист А1-4

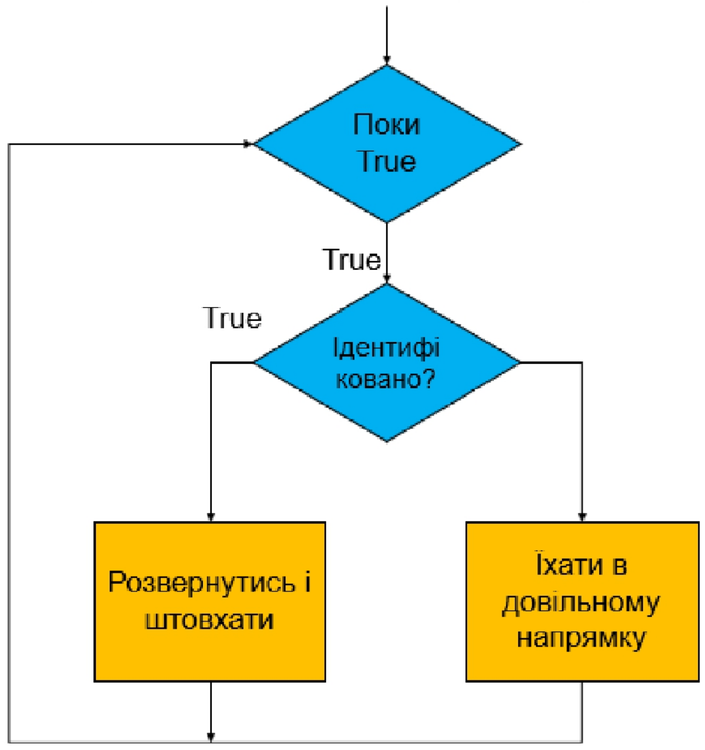
Імітація змагань роботів (зелений робот перемагає)



Ідентифікація ультразвуковим сенсором



Алгоритм найпростішої стратегії робота



Функція створення об’єкта для Pymunk

```
def createBody(x,y,shape,*shapeArgs):
    body = pymunk.Body()
    body.position = x, y
    s = shape(body, *shapeArgs)
    s.mass = 1
    s.friction = 1
    space.add(body, s)
    return s
```

Правила змагань

1.Роботи змагаються в межах кола.

2.Робот повинен ідентифікувати світлий (синій) об’єкт і виштовхнути його за межі. Тоді йому нараховується один бал.

3.Якщо робот виштовхує за межі кола темний (червоний) об’єкт, з нього знімається один бал.

4.Перемагає робот, який набрав найбільшу кількість балів

					БР.ПМІ-02.00.00.003		
Изм.	Лист	№ докум.	Подп.	Дата	Симулятор змагань роботів	Лит.	Масса	Масштаб
Разраб.		Бережанський						1:1
Пров.		Копей В.Б.				Лист 1	Листов 1	
Т. контр.		Копей В.Б.						
Н. контр.						ІФНТУНГ ПМІ-21-1К		
Утв.		Панчук В.Г.						

Копировал

Формат А1

Файл: Симулятор змагань роботів