МАГІСТЕРСЬКА РОБОТА

МР.ПМКм-40.00.00.000.ПЗ

Група ПМКм-21-1

Пронюк Ігор

Володимирович

2022

Івано-Франківський національний технічний університет нафти і газу
Інститут інженерної механіки
Кафедра: комп'ютеризованого машинобудування

Пронюк Ігор Володимирович
(прізвище, ім'я, по батькові)

УДК 32.816
(індекс)

МАГІСТЕРСЬКА РОБОТА

Транспортно-складський робот з функцією ідентифікації об'єктів на основі алгоритмів машинного навчання
(назва роботи)

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

131 – Прикладна механіка
(шифр і назва спеціальності)

І.В. Пронюк
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Копей В.Б., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

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

Рецензент

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

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

м. Івано-Франківськ — 2022 рік

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

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

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____
«____» _____20__ року

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

Пронюку Ігорю Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Транспортно-складський робот з функцією ідентифікації об'єктів на основі алгоритмів машинного навчання
керівник роботи Копей В.Б., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "14" жовтня 2022 року № 494/7

2.Строки подання студентом роботи 18 грудня 2022р.

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

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

1. Розробка КП робота; 2. Застосування функції машинного навчання. 3. Безпровідне з'єднання мікроконтролера з ПК; 4. Проектування транспортно-складського робота.

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

1. СК робота – 1 лист А1. 2. СК захоплювача – 1 лист А2. 3. СК механізму рухомого тримача УДВ – 1 лист А2. 4. К «Основа» – 1 лист А1. 5. К «Тримач УДВ нерухомий» – 1 лист А3. 6. К «Тримач УДВ рухомий» – 1 лист А3. 7. К «Основа захоплювача» – 1 лист А3. 8. К «Клешня» – 1 лист А3. 9. К «Лівий ш. з'єднувач» – 1 лист А3. 10. К «Правий ш. з'єднувач» – 1 лист А3. 11. К «З'єднувач» – 1 лист А4. 12. КП робота – 1 лист А1. 13. Платформа тестування роботів – 1 лист А1.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Копей В.Б., професор		
2	Копей В.Б., професор		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Термін виконання етапів роботи	Примітки
1	Загальна характеристика	01.04.2022	
2	Опис і конструкція навчального проєкту	01.06.2022	
3	Оглядова частина	05.08.2022	
4	Основна частина	01.10.2022	
5	Захист магістерської роботи	27.12.2022	

Студент _____ Пронюк І.В.
(підпис) (прізвище та ініціали)

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

“___”_____2022 р.

Реферат

Кваліфікаційної магістерської роботи «Транспортно-складський робот з функцією ідентифікації об'єктів на основі алгоритмів машинного навчання»

Дана робота складається зі 86 аркушів. До неї входять 73 рисунки, 2 додатки, 29 джерел.

Об'єкт дослідження – транспортно-складський робот.

Мета роботи – розробка транспортно-складського робота з функцією ідентифікації об'єктів на основі алгоритмів машинного навчання.

Для досягнення мети потрібно: 1. Провести аналіз транспортно-складських роботів; 2. Провести аналіз механізмів захоплення; 3. Вибрати мікроконтролер; 4. Реалізувати безпровідне з’єднання мікроконтролера з ПК; 5. Розробити керуючу програму з функцією ідентифікації об'єктів на основі алгоритмів машинного навчання; 6. Спроектувати транспортно-складського робота.

Відповідно до поставленої задачі в оглядовій частині магістерської роботи було проаналізовано транспортно-складські роботи та механізми захоплення. Вибрано мікроконтролер, мову програмування, для створення керуючої програми. Протокол передачі даних а також бібліотеки Python для створення функції ідентифікації об'єктів. Проведено огляд програмних продуктів, потрібних для досягнення мети. В основній частині магістерської роботи підібрано елемент та розроблено прототип транспортно-складського робота. Організовано безпровідний зв’язок мікроконтролера з ПК по протоколу Firmata. Створено керуючу програму робота, а також проведено тестування прототипу. Спроектовано повноцінну модель транспортно складського робота, з механізмом захоплення, для навчальних цілей.

В додатках наведена керуюча Python програма та кошторис робота.

Ключові слова: *транспортно-складський робот, промислові роботи, ідентифікація об’єктів, механізм захоплення, мікроконтролер, мова програмування, бібліотеки Python, керуюча програма, протокол передачі даних Firmata.*

Студент Пронюк І.В.

Abstract

of the master's work "Transport and warehouse robot with object identification function based on machine learning algorithms"

Paper: 86 pages, 73 figures, 29 references, 2 appendices.

The object of research is a transport and warehouse robot.

The purpose of the work is the development of a transport and warehouse robot with the function of object identification based on machine learning algorithms.

To achieve the goal, it is necessary to: 1. Conduct an analysis of transport and warehouse robots; 2. Conduct an analysis of capture mechanisms; 3. Select a microcontroller; 4. Implement a wireless connection of the microcontroller with the PC; 5. Develop a control program with the function of object identification based on machine learning algorithms; 6. Design a transport and warehouse robot.

In accordance with the task, in the review part of the master's work, transport and storage operations and capture mechanisms were analyzed. A microcontroller, programming language, to create a control program is selected. A data transfer protocol as well as Python libraries for creating an object identification function. An overview of the software products needed to achieve the goal was conducted. In the main part of the master's thesis, an element was selected and a prototype of a transport and warehouse robot was developed. Wireless communication between the microcontroller and the PC using the Firmata protocol is organized. The control program of the robot was created, and the prototype was tested. A complete model of a transport warehouse robot with a gripping mechanism was designed for educational purposes.

The Python control program and the estimate of the work are given in the appendices.

Keywords: *transport and warehouse robot, industrial robots, object identification, grasping mechanism, microcontroller, programming language, Python libraries, control program, Firmata data transfer protocol.*

Student Pronyuk I.V.

Зміст

ВСТУП

Одним із вищих рівнів розвитку машинної техніки, коли регулювання й керування виробничими процесами здійснюється без участі людини, а лише під її контролем є автоматизація виробництва. Сучасний стан її розвитку привів до нової системи технологічних машин з керуючими засобами, які ґрунтуються на застосуванні програмованих, логічних контролерах, інтелектуальних засобах вимірювання і контролю та інформаційно об'єднаних промислових мережах.

Сьогодні як ніколи раніше є актуальним питання застосування часткових, комплексних та повні автоматизацій виробництв масового типу. Вони переходять в автоматизовані, безперервні лінії. Важливою ланкою таких ліній є промислові роботи, адже часто сучасне управління виробничими процесами, внаслідок їх складності практично недоступні людині, а прості, автоматичні пристрої ефективно замінюють її.

Складські, промислові роботи, що являють собою автоматичні, стаціонарні чи пересувні машини, з пристроями у вигляді маніпуляторів, можуть використовуватися в практично всіх галузях. Обслуговування верстатів, різання, зварювання, сортування, пакування. Перевагами є: можливість працювати без зупинок протягом тривалого часу, здатність працювати з небезпечними для людини матеріалами а також високі надійність, швидкість і точність.

Недоліками, що характеризують важкий перехід до автоматизованого виробництва є висока вартість, для роботизації всієї виробничої лінії і дефіцит персоналу, що може обслуговувати, працювати з нею.

Підвищення продуктивності, покращення дохідності виробництва сьогодення напряму залежить від використання промислових роботів, що взаємодіють між собою.

1. ОГЛЯДОВА ЧАСТИНА

1.1 Аналіз транспортно-складських роботів і постановка завдання

У широкому розумінні робот може бути визначений як технічна система, здатна замінювати людину чи допомагати їй у виконанні різних завдань. При визначенні робота доводиться вдаватися до таких понять як робоче середовище, джерело енергії, необхідне для забезпечення функціонування робота і джерело інформації для опису поставленої людиною-оператором завдання [1].

Робочими органами можуть бути різноманітні інструменти: кліщі, присоски (захоплення), сопла, пальник (у дуговому зварюванні) і тому подібне.

Лідируючим ринком з використання роботів у промисловості є Азія, на неї припадає 70% на 2020-й рік [2].

Найбільш популярними роботами на виробництві є маніпулятори, які представляють собою конструкцію типу «руки», адаптивне керування яких в основному здійснюється програмно й базується на багатоваріантному програмному забезпеченні. При цьому рішення про вибір типу роботи програми приймається роботом на підставі інформації про середовище, описаної детекторами. Як правило, вони спроможні переміщувати об’єкти в просторі трьох і більше координат та виконувати різноманітні виробничі процеси. На рисунку 1.1 зображено промислові роботи KUKA на автомобільному виробництві.

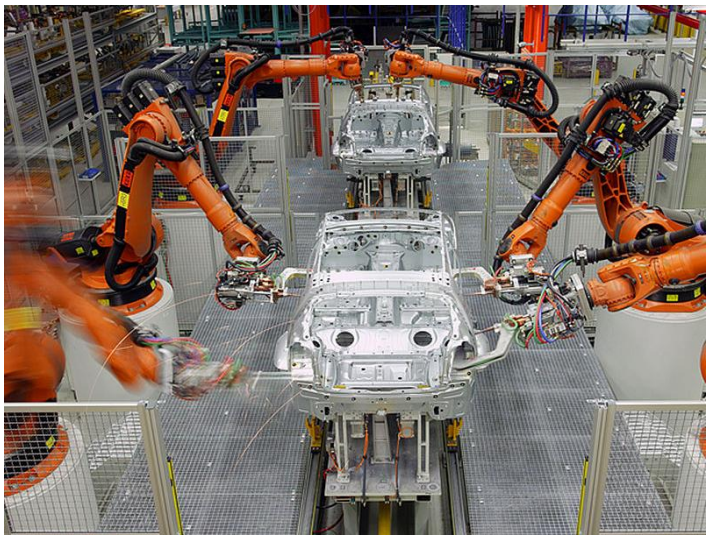


Рисунок 1.1 – Промислові роботи KUKA на автомобільному виробництві

Завдяки таким роботам значно зменшується як час затрачений на виготовлення одиниці продукту, так і підвищується продуктивність виробництва, покращується дохідність виробництва.

Але на заводах по виготовленні корпусів та деталей автомобілів, як правило встановлюють лінію постачання деталей, продукту виробництва. Маніпулятори є статично встановленими. Уже при розгляді підприємств, що виготовляють провідники, чи деталі машин масового виробництва, ситуація не така однозначна. Стоїть питання можливості пришвидшення виробництва, за рахунок скорочення часу між операціями та робочими зонами, в яких ведеться обробка, тощо.

Складські роботи виконують в основному операції сортування, пакування і переміщення об’єктів. Поділяються такі роботи на:

- Самохідні роботи візки;
- Буксири;
- Палетайзери;
- Роботи для сортування;
- Дрони.

Отже транспортно-складські роботи в складах займаються обслуговування стелажів, навантаженням й розвантаження продукції. У місцях зберігання здійснюється за допомогою підіймально-транспортних засобів: палетних шатлів, штабелерів і маніпуляторів [3].

Транспортування вантажів. Конвеєри, конвеєрні системи для палет: рейкові, роликові, підвісні, або автоматично керовані транспортні засоби (AGV) переміщують вантажі з одного місця в інше у самому складі або між складом і виробництвом [3].

Комплектація замовлень. Колаборативні роботи на складах здійснюють автоматичну або напівавтоматичну комплектацію замовлень. До цього типу належать маніпулятори для обробки значних вантажів, пакувальні машини, що здатні розрахувати й підготувати матеріал для пакування продукції певного виду, різноманітні роботи-комплектувальники, а також механічні екзоскелети, що адаптуються до рухів носія і скорочують фізичні зусилля при виконанні рухів. [3].

Транспортно-складські роботи приносять значну цінність складськими операціям. Вони допомагають пом’якшити, усунути помилки, прискорювати час на окремі операції. Зменшити накладні та поточні витрати та сприяти кращому управлінню виробництвом.

1.2 Аналіз механізмів захоплення

Захвати — підсистеми механізмів маніпуляції, які забезпечують тимчасовий контакт із предмет, який потрібно схопити. Вони забезпечують положення та орієнтацію при носінні та підключення об’єкта до транспортно-розвантажувального обладнання. Захоплення досягається силою виробництва і утворюють відповідні елементи. Термін «захват» також використовується у

випадках, коли фактично немає захоплення, а скоріше утримання об’єкта, як напр. у вакуумному відсмоктуванні, де утримання сила може діяти на точку, лінію або поверхню [4].

Існує чотири типи роботизованих захватів: вакуумні захвати, пневматичні захвати, гідравлічні захвати та серво-електричні захвати. Захвати обирають залежно від того, яке застосування потрібно для транспортування та типу матеріалу, що використовується:

1. Вакуумний захват був стандартним у виробництві через його високий рівень гнучкості. Цей тип робота-захвату використовує гумову або поліуретанову присоску для збирання предметів. Деякі вакуумні захвати використовують шар спіненої гуми із закритими порами, а не присоски, щоб завершити застосування.
2. Пневмозахват користується популярністю завдяки своїм компактним розмірам і невеликій вазі. Його можна легко вставити у вузькі приміщення, що може бути корисним у промисловості. Захвати пневматичних роботів можна відкривати або закривати, за що їх називають приводами «бац-бац» через шум, який створюється під час роботи захвату «метал на метал».
3. Гідравлічний захват забезпечує найбільшу міцність і часто використовується для додатків, які потребують значної кількості зусилля. Ці роботизовані захвати створюють свою силу завдяки насосам, які можуть забезпечувати до 2000 фунтів на квадратний дюйм. Незважаючи на те, що гідравлічні захвати міцні, вони брудніші, ніж інші захвати через мастило, яке використовується в насосах. Їм також може знадобитися додаткове технічне обслуговування через пошкодження захвату через зусилля, яке використовується під час застосування.
4. Сервоелектричний захват з’являється все частіше в промислових умовах завдяки тому, що ним легко керувати. Електронні двигуни керують рухом губок захвату. Ці захвати є дуже гнучкими та допускають різні допуски на

матеріал під час роботи з деталями. Сервоелектричні захвати також економічно вигідні, оскільки вони чисті та не мають повітропроводів [5].

Враховуючи всі плюси сервоелектричного захвату, а саме легкість керування, економічна вигідність та гнучкість захоплювача (варіативність використання, хапання об’єктів різних розмірів і форм). Типовий робот маніпулятор з сервоелектричним захоплювачем, моделі xArm 6DOF працюючий на мікроконтролері Arduino зображено на рисунку 1.2.

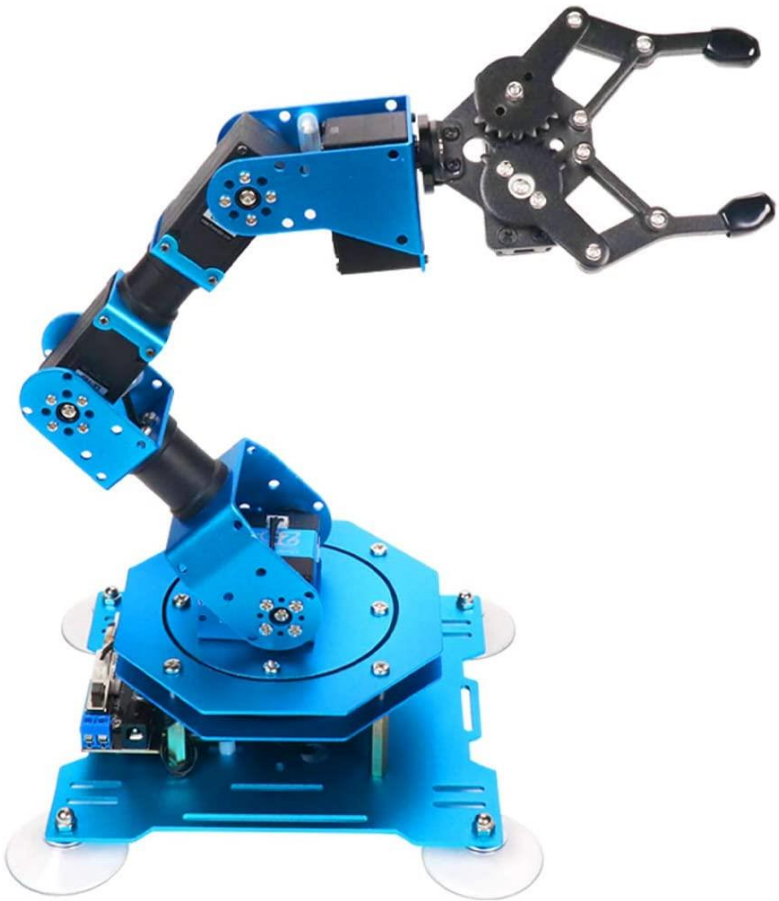


Рисунок 1.2 – Робот маніпулятор з сервоелектричним захватом

Сервоелектричні захвати можуть бути наступних типів [6]:

- Чисте огородження без затиску (рис 1.3, а);
- Часткова посадка форми в поєднанні з затискною силою (рис 1.3, б);
- Чисте силове затискання (рис 1.3, в).

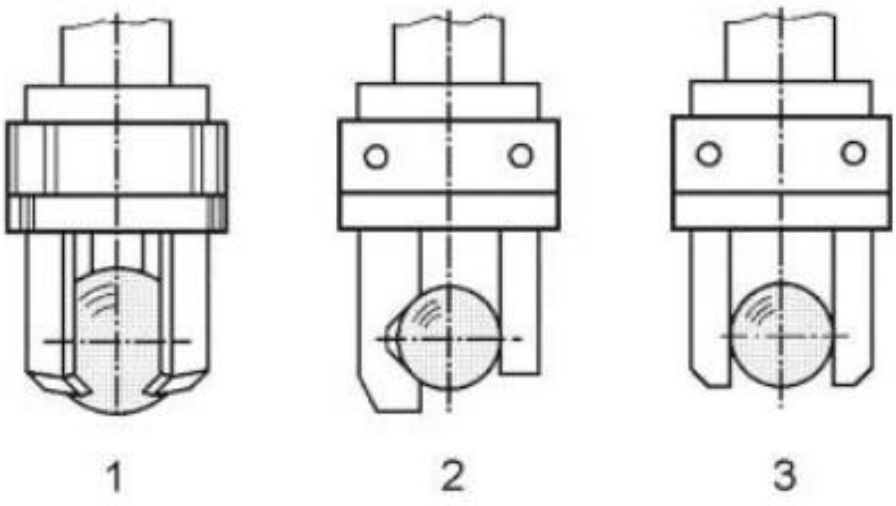


Рисунок 1.3 – Типи захватів:

а) Чисте огородження без затиску; б) Часткова посадка форми в поєднанні з затискною силою; в) Чисте силове затискання.

1.3 Мікроконтролер Arduino UNO

Arduino - це торгова марка апаратно-програмних засобів побудови та прототипування простих систем, моделей та експериментів у галузі електроніки, автоматики, автоматизації процесів та робототехніки.

Мікроконтролер Arduino Uno - це пристрій на основі мікроконтролера ATmega328 (datasheet) (рис 1.4). До його складу входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (з них 6 можуть використовуватися як ШИМ-виходи), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування (ICSP) та кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від AC/DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою кабелю USB [7].

Рисунок 1.4 – Мікроконтролер Arduino Uno

Програмна частина складається з безкоштовної програмної оболонки «IDE» для написання програм, їх компіляції та програмування апаратури. Апаратна частина є набором змонтованих друкованих плат, що продаються як офіційним виробником, так і сторонніми виробниками. Повністю відкрита архітектура системи дозволяє вільно копіювати або доповнювати лінійку продукції Arduino.

Здебільшого використовується для створення автономних об'єктів і підключення до програмного забезпечення через дротові та бездротові інтерфейси. Підходить для початківців з мінімальним вхідним порогом знань у галузі розробки електроніки та програмування.

За допомогою кількох рядків коду ви можете змусити Arduino вмикати або вимикати світло, зчитувати значення датчика та відобразіть його на екрані комп’ютера або навіть використайте для побудови саморобної схеми для ремонту зламаної кухні прилад. Через універсальність Arduino та масову підтримку, доступну в Інтернеті спільноти користувачів Arduino, це привернуло нову породу любителів електроніки, які ніколи не були раніше торкався мікроконтролера, не кажучи вже про програмування [8].

Все програмне забезпечення, необхідне для програмування Arduino, є відкритим кодом (можна використовувати та змінювати безкоштовно). Arduino розробила просту в освоєнні мову програмування (похідну від C++), яка включає різноманітні складні функції програмування в прості команди, які набагато простіше для новачка навчитися [8].

1.4 Мова програмування Python, для створення керуючої програми

Python – інтерпритована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури данних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об’єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Для багатьох основна перевага мови Python полягає в зручності читання, ясності і вищій якості, що відрізняють її від інших інструментів у світі мов програмування. Програмний код Python читати легше, а значить, багаторазове його використання та обслуговування виконується набагато простіше, ніж використання програмного коду на інших мовах програмування. Одноманітність оформлення програмного коду мовою Python полегшує його розуміння навіть для тих, хто не брав участі у його створенні. Крім того, Python підтримує найсучасніші механізми багаторазового використання програмного коду, яким є об'єктно-орієнтоване програмування (ООП) [9].

Порівняно з компілюючими або строго типізованими мовами, такими як C, C++ та Java, Python у багато разів підвищує продуктивність праці розробника.

Об'єм програмного коду Python зазвичай становить третину або навіть п'яту частину еквівалентного програмного коду мовою C++ чи Java. Це означає менший обсяг введення з клавіатури, менша кількість часу на налагодження та менший обсяг трудовитрат на супровід. Крім того, програми на мові Python запускаються відразу ж, минаючи тривалі етапи компіляції та зв'язування, необхідні в деяких інших мовах програмування, що ще більше збільшує продуктивність праці програміста [9].

Більшість програм на мові Python виконується без змін всіх основних платформ. Перенесення програмного коду з операційної системи Linux у Windows зазвичай полягає у простому копіюванні файлів програм з однієї машини в іншу. Більше того, Python надає масу можливостей зі створення графічних інтерфейсів, що переносяться, програм доступу до баз даних, веб-додатків та багатьох інших типів програм. Навіть інтерфейси операційних систем, включаючи спосіб запуску програм та обробку каталогів, у мові Python реалізовані переносимим способом [9].

У складі Python поставляється велика кількість зібраних і переносимих функціональних можливостей, відомих як стандартна Бібліотека. Ця бібліотека надає масу можливостей, затребуваних у прикладних програмах, починаючи від пошуку тексту за шаблоном та закінчуючи мережевими функціями. Крім того, Python допускає розширення як за рахунок ваших власних бібліотек, так і за рахунок бібліотек, створених сторонніми розробниками. З-поміж сторонніх розробок можна назвати інструменти створення веб-сайтів, програмування математичних обчислень, доступ до послідовного порту, розробку ігрових програм та багато іншого. Наприклад, розширення NumPy позиціонується як вільний та потужніший еквівалент системи програмування математичних обчислень Mathlab [9].

За своєю природою Python має простий, легкочитаний синтаксис і ясну модель програмування. Згідно з гаслом, висунутим на недавній конференції з

мови Python, основна його перевага полягає в тому, що Python «кожному по плечу» - характеристики мови взаємодіють обмеженою числом несуперечливих способів і природно випливають із невеликого кола базових концепцій. Це робить мову простою в освоєнні, розумінні та запам'ятовуванні. На практиці програмістам, які використовують мову Python, майже не доводиться вдаватися до довідкових посібників – це несуперечлива система, на виході якої, на подив багатьох, виходить [9].

1.5 StandardFirmata та pyFirmata

StandardFirmata – це «скетч», прошивка для мікроконтролерів, реалізації протоколу зв’язку з програмним забезпеченням на головному комп’ютері [10]. Цей «скетч» доступний для використання в офіційному програмному забезпеченні Arduino IDE (рис 1.5).

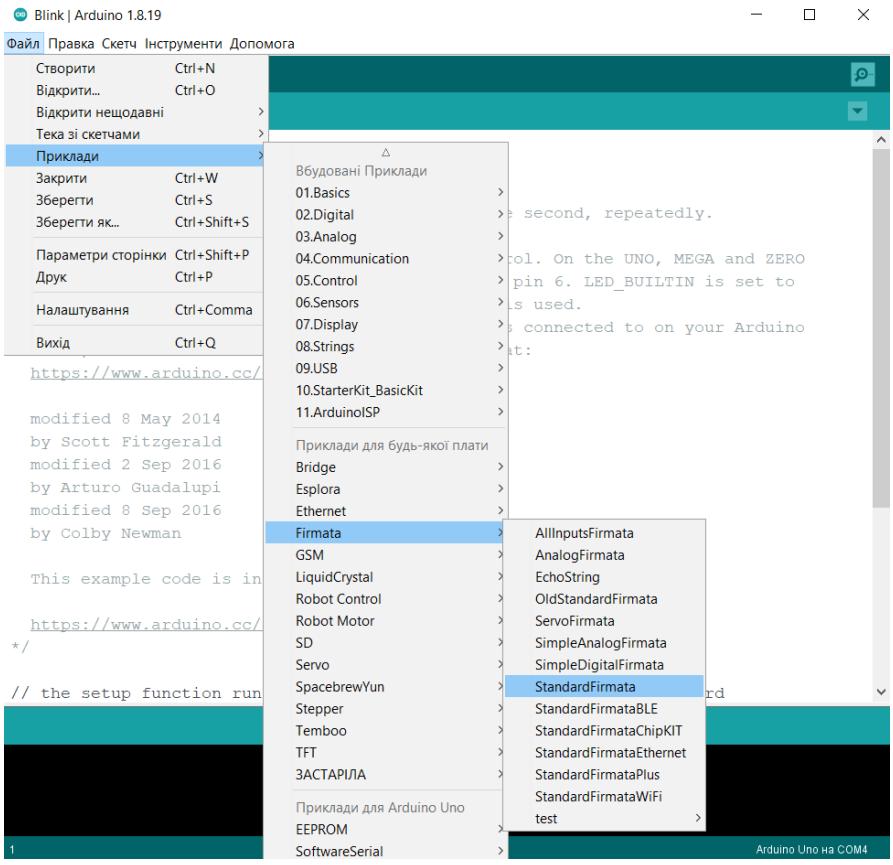


Рисунок 1.5 – StandardFirmata в Arduino IDE

Існує дві основні моделі використання Firmata. В одній моделі автор ескізу Arduino використовує різні методи, надані бібліотекою Firmata, для вибіркової передачі та отримання даних між пристроєм Arduino та програмним забезпеченням, що працює на головному комп'ютері. Наприклад, користувач може надіслати аналогові дані на хост за допомогою:

```
Firmata.sendAnalog(analogPin, analogRead(analogPin)),
```

або надіслати дані, упаковані в рядок, за допомогою:

```
Firmata.sendString(stringToSend)
```

Див. Файл -> Приклади -> Firmata -> AnalogFirmata & EchoString відповідно для прикладів. Друга і більш поширена модель полягає в тому, щоб завантажити ескіз загального призначення під назвою StandardFirmata (або один із варіантів, наприклад StandardFirmataPlus або StandardFirmataEthernet залежно від ваших потреб) на платі Arduino, а потім використовувати головний комп'ютер виключно для взаємодії з платою Arduino. StandardFirmata знаходиться в Arduino IDE у Файл -> Приклади -> Firmata [11].

Отже Arduino діятиме як сервер, отримуватиме запити та надсилатиме дані, а комп'ютер діятиме як клієнт, надсилатиме запити та отримуватиме дані.

Для використання StandardFirmata, успішного компілювання і вивантаження «скетчу» на мікроконтролер Arduino, чи аналогічний, потрібно встановити бібліотеку «Firmata» через менеджер бібліотек Arduino IDE (рис 1.6).

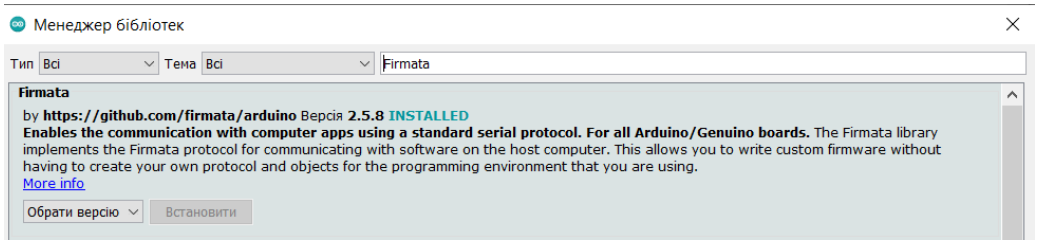


Рисунок 1.6 – Бібліотека Firmata в менеджері бібліотек

Також на однойменному сайті GitHub, в директорії «firmata/Arduino» доступні клієнтські версії бібліотек, для різних потреб. До прикладу модифікована, яка підтримує використання ультразвукових датчиків відстані.

В свою чергу pyFirmata – це модуль, інтерфейс для створення послідовного зв’язку Firmata між скриптом Python на будь-якому комп’ютері та Arduino.

Модуль pyFirmata можна встановити за допомогою pip, командою в консоль Python:

```
pip install pyfirmata
```

Або завантажити його в форматі архіву. [12].

Тобто в цілому Firmata — це проміжний протокол, який з’єднує вбудовану систему з головним комп’ютером, а канал протоколу за замовчуванням використовує послідовний порт. Платформа Arduino є стандартною еталонною реалізацією для Firmata. Arduino IDE поставляється з підтримкою Firmata.

1.6 OpenCV та Scikit-learn, для створення функції ідентифікації об'єктів

OpenCV (Open Source Computer Vision Library) – це бібліотека комп'ютерного зору з відкритим кодом, функцій та алгоритмів комп’ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях [13].

Для розпізнавання об’єктів на зображенні використовуються каскади, які можна завантажити на GitHub в директорії «opencv/opencv» [14].

Бібліотека розроблена Intel і нині підтримується Willow Garag та Itseez. Сирцевий код бібліотеки написаний мовою C++ і поширюється під ліцензією BSD. Біндинги підготовлені для різних мов програмування, таких як Python, Java, Rudy, Matlab, Lua та інших. Може вільно використовуватися в

академічних та комерційних цілях [13].

Бібліотека містить понад 2500 оптимізованих алгоритмів, серед яких повний набір як класичних так і практичних алгоритмів машинного навчання і комп'ютерного зору. Алгоритми OpenCV застосовують у таких сферах: аналіз та обробка зображень, системи з розпізнавання обличчя, ідентифікації об'єктів, розпізнавання жестів, відстежування переміщення камери та багато інших.

Scikit-learn – це безкоштовна програмна бібліотека машинного навчання для мови програмування Python, яка надає функціональність для створення та тренування різноманітних алгоритмів класифікації, регресії та кластеризації, таких як лінійна регресія, random forest, градієнтний бустинг, і працює у зв'язці з бібліотеками NumPy та SciPy. Scikit-learn є однією з найбільш популярних бібліотек машинного навчання [15].

Перша версія бібліотеки була написана Девідом Корнапе влітку 2007 року в рамках програми Google Summer of Code. Пізніше цього ж року Метью Брюхер почав працювати над нею, як частиною своєї дипломної роботи. 2010 року дослідники з INRIA продовжили розробку бібліотеки і 1 лютого 2010 року випустили перший офіційний реліз. Відтоді з'явилося декілька нових релізів бібліотеки, а навколо неї утворилася процвітаюча спільнота з розробників, що продовжують підтримувати та удосконалювати проект. Scikit-learn здебільшого написаний на Python та широко використовує NumPy для розв’язання задач лінійної алгебри та операцій з масивами. Крім того, деякі алгоритми написані на Cython для покращення продуктивності. [15].

А ще Scikit-learn добре інтегрується з багатьма бібліотеками Python. До прикладу Matprolib та plotly для побудови графіків, NumPy для масивів, Pandas, SciPy та багатьма іншими.

Отже основні особливості Scikit-learn [15]:

- Прості та ефективні інструменти для прогнозного аналізу даних;

- Доступний для всіх і придатний для повторного використання в різних контекстах;
- Створено на основі NumPy, SciPy і matplotlib;
- Відкритий вихідний код, комерційне використання - ліцензія BSD.

Модуль scikit можна встановити за допомогою pip, командою в консоль Python:

```
pip install -U scikit-learn
```

Або завантажити його в форматі архіву [12].

1.7 Огляд програмних продуктів

Arduino IDE

Взаємодія з мікроконтролером Arduino UNO R3 відбувається за допомогою інтегрованого середовища розробки Arduino IDE для Windows, MacOS і Linux. Воно розроблене на Ci і C++, призначене для створення та завантаження програм на мікроконтролер по послідовному порті.

Інтерфейс Arduino IDE зображений на рисунку 1.7.

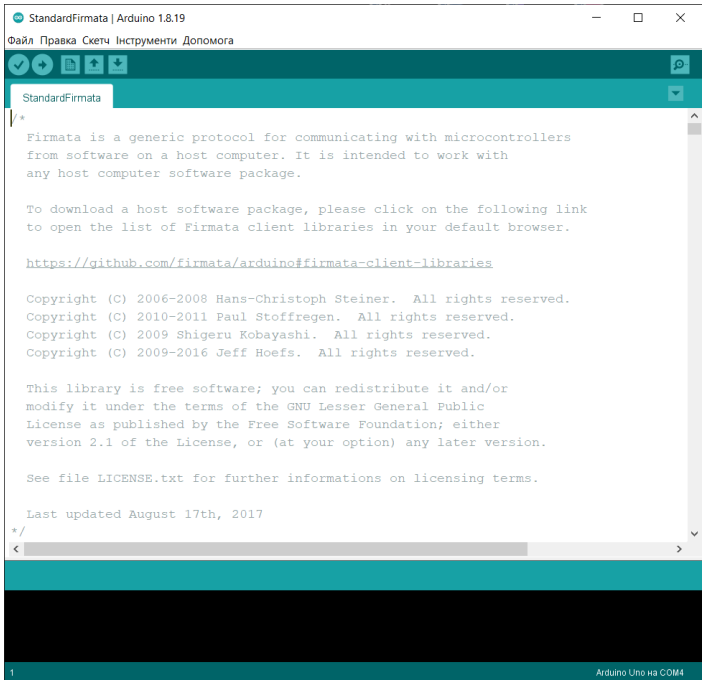


Рисунок 1.7 – Інтерфейс Arduino IDE

Для вивантаження StandardFirmata достатньо вибрати COM порт, по якому підключений мікроконтролер Arduino, чи інший сумісний, до ПК (рис 1.8).

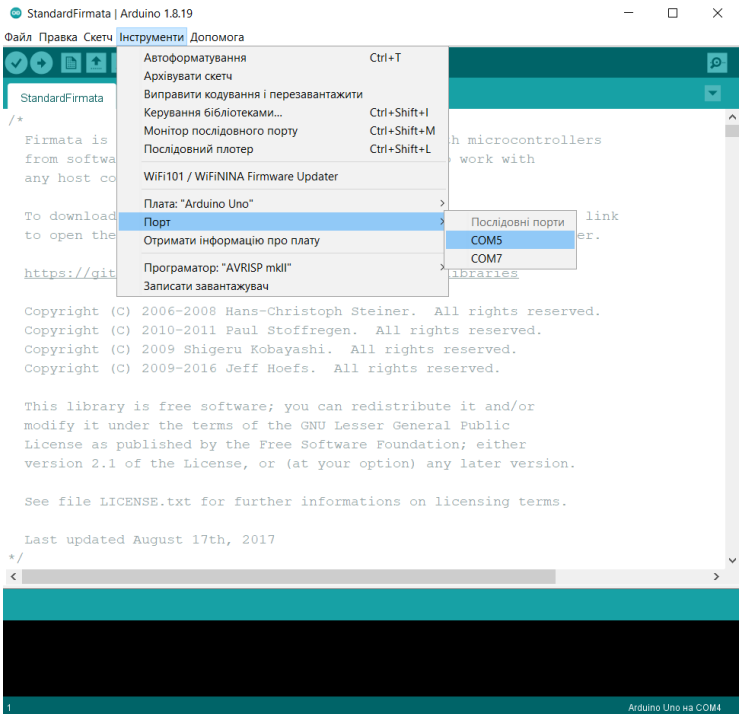


Рисунок 1.8 – Вибір COM порту

Після чого вибрати модель плати (рис 1.9) І нажати кнопку "Вивантажити" (рис 1.10)

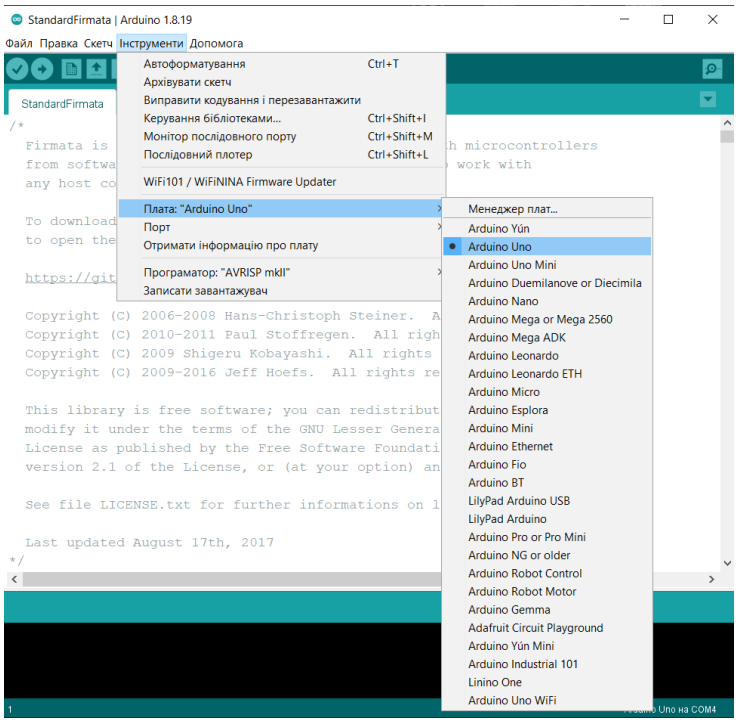


Рисунок 1.9 – Вибір моделі плати

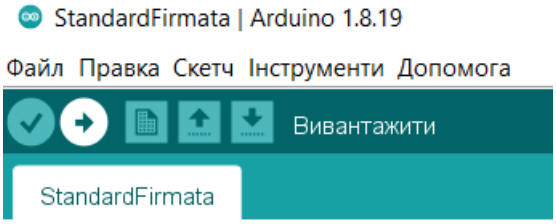


Рисунок 1.10 – Кнопка вивантаження програми

Pyzo

Для написання керуючої Python програми використовується Python версії 2.7.17 [16] в зв'язку з високою стабільністю і великою кількістю якісних, допрацьованих пакетів.

А для спрощення написання програми використовується ПП Pyzo – це безкоштовне обчислювальне середовище з відкритим кодом на основі Python. Середовище розробки Python, яке працює з будь-яким інтерпретатором Python, встановленим у вашій системі, включно з середовищем Conda. IDE націлена на інтерактивність і простоту. Складається з редактора, оболонки та набору інструментів, які допомагають програмісту різними способами [17].

Інтерфейс Pyzo зображений на рисунку 1.11.

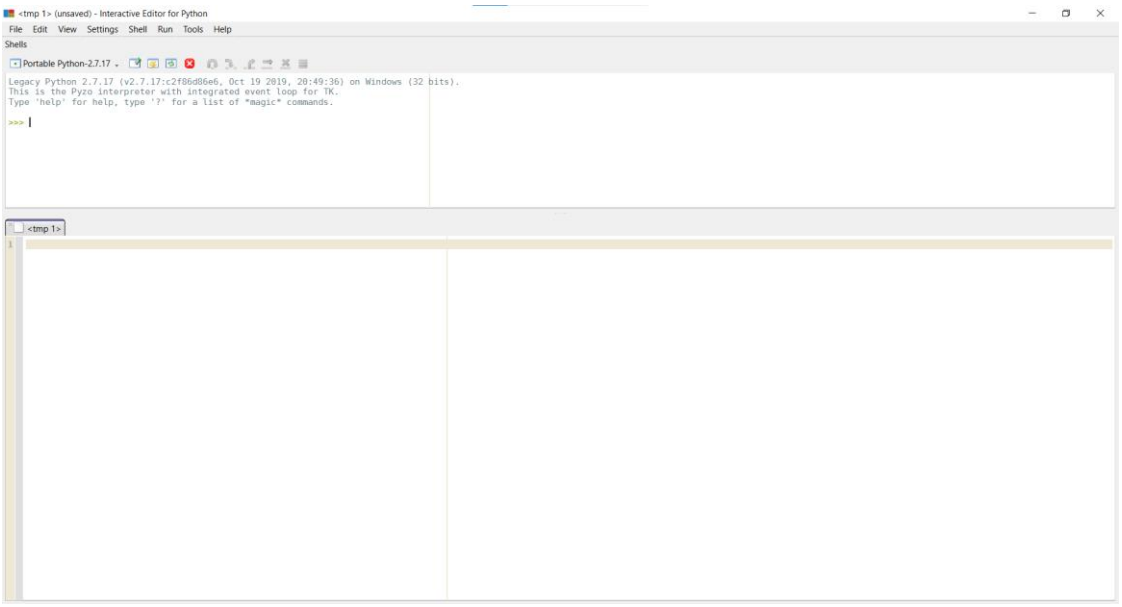


Рисунок 1.11 – Інтерфейс програми Pyzo

Для використання достатньо вказати шлях до python.exe. Перейти в Shell – Edit Shell Configuration… (рис 1.12)

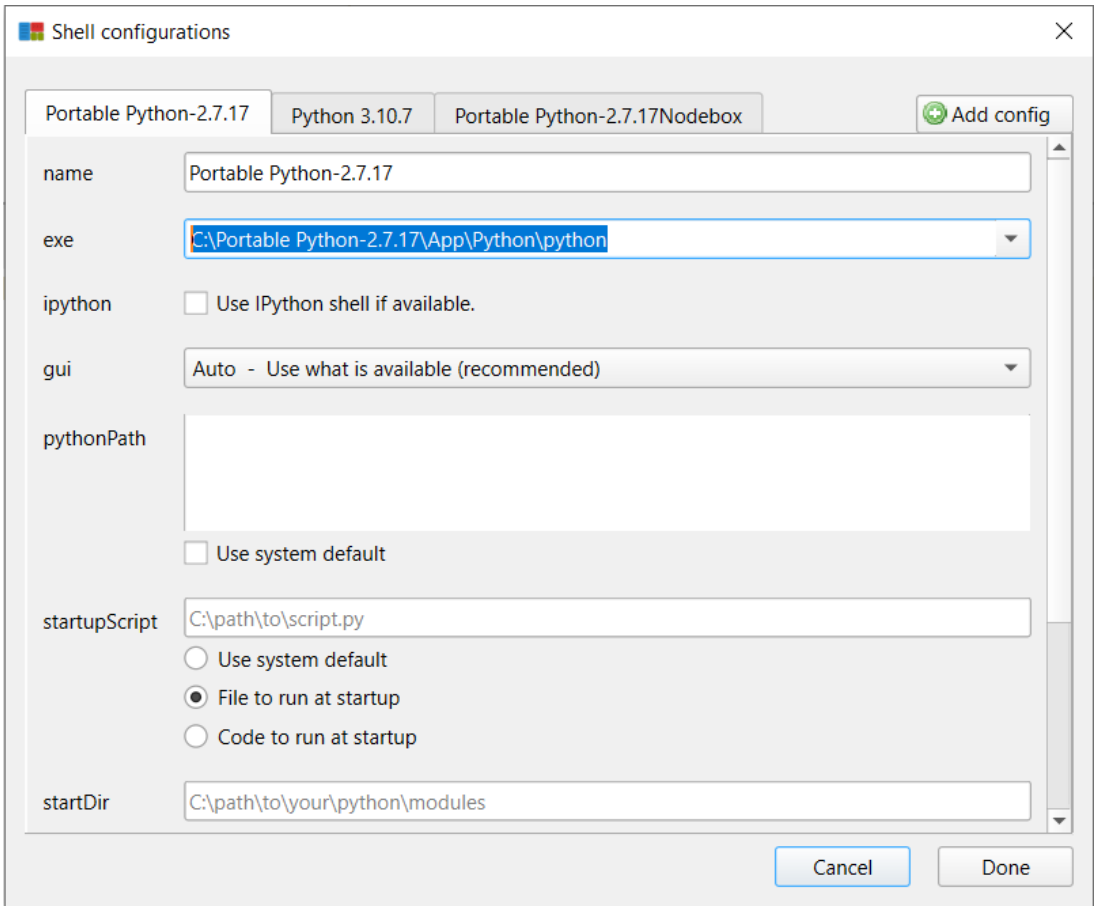


Рисунок 1.12 – налаштування програми Pyzo

SolidWorks

Для створення 3D моделей деталей та 3D моделі збірки повноцінного робота використовується ПП SolidWorks – продукт компанії SolidWorks Corporation (зараз – дочірня компанія Dassault Systèmes), САПР, інженерного аналізу та підготовки виробництва будь-якої складності та призначення.

Завдання які дозволяє вирішити «SolidWorks», при конструкторській підготовці [18]:

- 3D проектування виробів (деталей і зборок) будь-якого ступеня складності з урахуванням специфіки виготовлення;
- Створення конструкторської документації;
- Промисловий дизайн;
- Проектування комунікацій (електроджгутів, трубопроводи тощо.);

- Інженерний аналіз (міцність, стійкість, теплопередача, частотний аналіз, динаміка механізмів, газо / гідродинаміка, оптика і світлотехніка, електромагнітні розрахунки, аналіз розмірних ланцюгів і ін.);
- Експрес-аналіз технологічності на етапі проектування.

А також, при технологічній підготовці виробництва:

- Проектування оснащення і інших засобів технологічного оснащення;
- Аналіз технологічності конструкції виробу;
- Аналіз технологічності процесів виготовлення (лиття пластмас, аналіз процесів штампування, витяжки, гнуття та ін.);
- Розробка технологічних процесів;
- Матеріальне та трудове нормування;
- Механообробка: розробка керуючих програм для верстатів з ЧПУ, верифікація УП, імітація роботи верстата. Фрезерна, токарна, токарно-фрезерна і електроерозійна обробка, лазерна, плазмова і гідроабразивна різання, вирубні штампи, координатно-вимірювальні машини;
- Управління даними і процесами на етапі ТПП.

«SolidWorks» є конструкторською системою твердотільного параметричного моделювання машинобудівних конструкцій спеціально розробленою для використання на персональних комп'ютерах під управлінням операційної системи Windows. Стандартний графічний користувальницький інтерфейс Windows і засоби твердотільного параметричного моделювання дозволяють швидше і легше ніж будь-коли створювати тривимірні моделі деталей, складальні одиниці, генерувати креслення, значно знижуючи терміни проектування і зменшуючи час виходу виробів на ринок. Має ряд модулів: SolidWorks Motion, SolidWorks Routing, SolidWorks Simulation (COSMOSWorks), SolidWorks Simulation, Toolbox SolidWorks та інші.

Інтерфейс програми «SolidWorks» для персональних комп’ютерів є відносно простим і інтуїтивним (рис. 2.13). 3D моделювання здійснюється по

принципу створення ескізів та маніпуляції з ними, такими опціями як: «Extruded», «Revolved», «Hole», «Swept», «Fillet», «Pattern», «Rib» та багатьох інших.

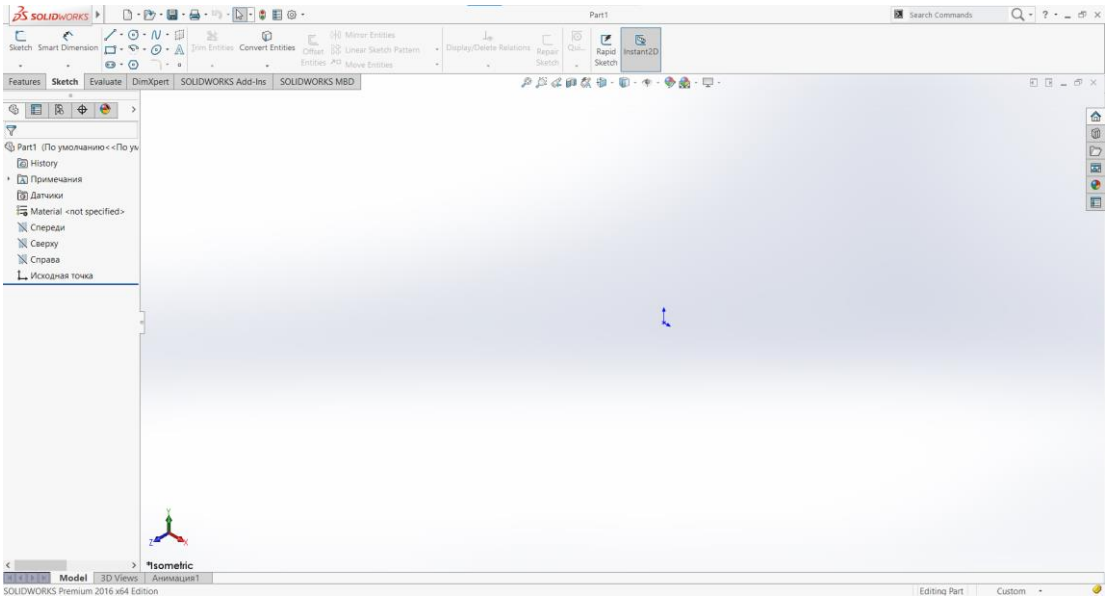


Рисунок 1.13 – Інтерфейс програми «SolidWorks».

Termite

Для налаштування швидкості роботи Bluetooth модуля використовується ПП Termite – простий у використанні та легкий у налаштуванні термінал RS232. Він використовує інтерфейс, схожий на інтерфейс програм «месенджер» або «чат» з великим вікном, яке містить усі отримані дані, і рядком редагування для введення рядків для передачі. Основними перевагами утиліти є простота встановлення (можливо, із попередньо налаштованими параметрами) за допомогою евристичного пошуку відповідного COM-порту та, як уже згадувалося, її зручність для користувача [19].

Функціонал Termite [19]:

- Інтерфейс плагіна для попередньої обробки або альтернативного перегляду даних, реєстрації отриманих даних у файл, додавання панелі

інструментів до Termite, макросів клавіатури та інших функцій, які ви можете придумати;

- Історія введених команд із автозавершенням;
- Головне вікно зі змінним розміром, з опцією «зберігати вікно зверху», багатомовний інтерфейс користувача;
- Можливість запускати з попередньо налаштованим параметрами з носія лише для читання (інсталяція не потрібна);
- Підтримка нестандартних швидкостей передачі даних (MIDI, DMX512);
- Різні кольори для переданих і отриманих даних (синій = передані, зелений = отримані);
- Дані можуть пересилатися між двома портами RS232;
- Діалогове вікно пошуку переданого/отриманого тексту (спливаюче меню, яке відкривається правою кнопкою миші);

Termite 3.4 є захищеним авторським правом програмним забезпеченням, яке є безкоштовним для особистого та комерційного використання. Ви можете використовувати та поширювати його без обмежень. Однак ви не можете видаляти або приховувати авторські права. Немає жодних гарантій чи гарантій; використовуйте його на свій страх і ризик [19]:

Інтерфейс ПП Termite зображено на рисунку 1.14.

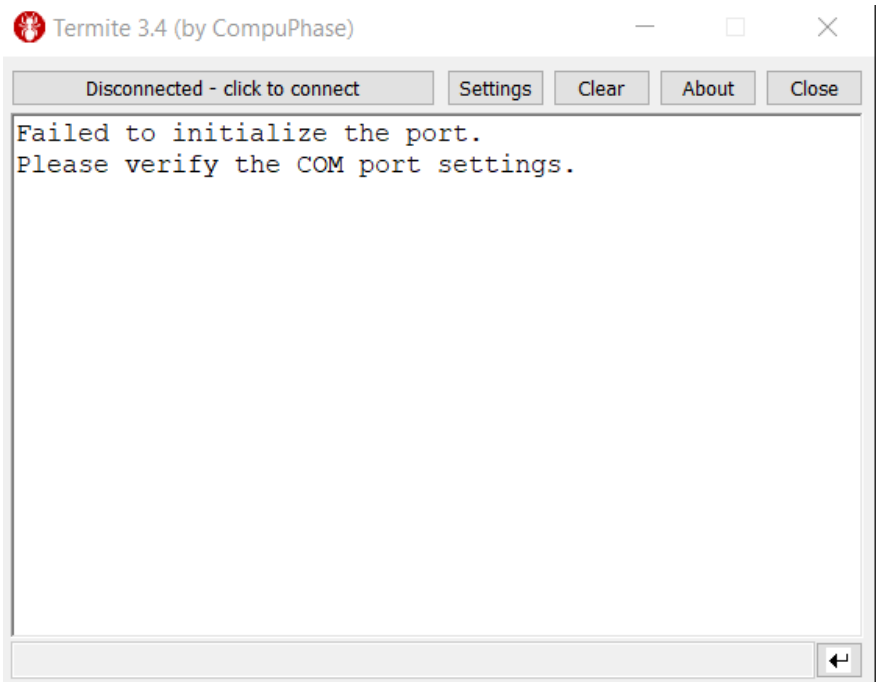


Рисунок 1.14 – Інтерфейс ПП Termite

1.8 Постановка завдання

Завданням є розробити гнучкого, надійного і простого в використанні транспортно-складського робота, що може бути використаний для перевезення об’єктів між робочими зонами, цехами підприємства, їх сортування. Робот повинен бути дешевий в виготовленні і розрахованим для використання на підприємствах, основною діяльністю яких є обробка металів, чи виготовлення деталей машин.

Для підвищення продуктивності і можливості реалізації автономної програми для автоматичних ліній виробництва буде створено функцію ідентифікації об’єктів на основі алгоритмів машинного навчання.

Розробити прототип робота для налагодження алгоритму автономної роботи та ідентифікації об’єктів.

Спроектувати модель повноцінної, дієздатної моделі транспортно-складського робота. Гнучкого, дешевого і простого в використанні на базі

мікроконтролера Arduino UNO R3, як перепрограмованої системи керування. Спроектувати механізм захвату чистого силового затискання, що буде простим у виготовленні.

2. ОСНОВНА ЧАСТИНА

2.1 Підбір елементів, для прототипу робота.

Основою робота буде мікроконтролер, популярної сьогодні моделі Arduino UNO R3 (рис. 2.1). Він є достатньо доступним, підтримується виробником і користується популярністю в ентузіастів, любителів в галузі електроніки, в зв’язку з чим, на базі цього мікропроцесора уже існує багато проектів різноманітної направленості.

Рисунок 2.1 – Arduino UNO R3

Для більших можливостей і зручності підключення елементів, 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.

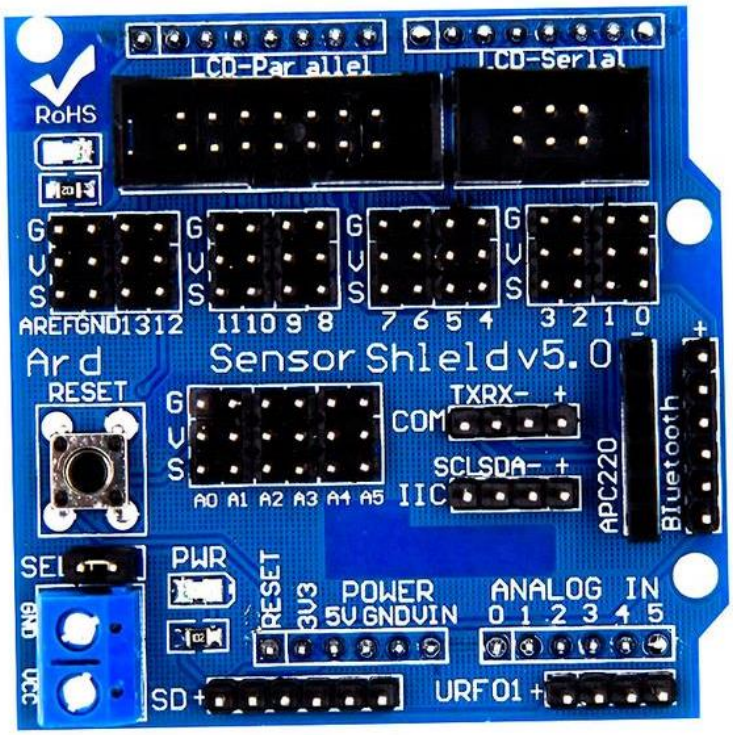


Рисунок 2.2 – Sensor Shield V5

Двигуни, що приводитимуть в рух прототип робота - DC Gear 48 (рис. 2.3). Які продаються в комплекті з пластмасовими колесами, резиновими шинами.

Рисунок 2.3 – Двигун DC Gear 48 з колесом.

Для підключення двигунів типу DC Gear 48 потрібен драйвер L298N (рис. 2.4), який може управляти двома моторами або одним кроковим двигуном.

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

Основні характеристики:

- Напруга живлення вбудованої логіки: 5В;
- Споживаний струм вбудованої логіки: 0 - 36мА;
- Напруга живлення драйвера: 5 - 35В (максимально 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;

Призначення контактів:

- Vcc - Підключення зовнішнього живлення двигунів;
- +5 - Живлення логіки;
- GND – Загальний;
- IN1, IN2, IN3, IN4 - контакти керування двигунами;
- OUT1, OUT2 - вихід першого двигуна;
- OUT3, OUT4 - вихід другого двигуна.

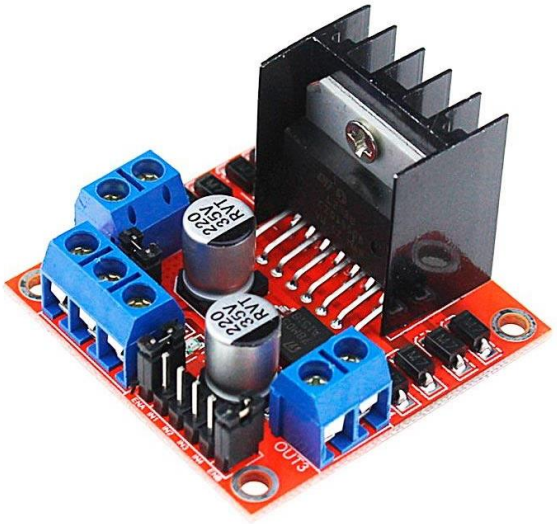


Рисунок 2.4 – Драйвер двигунів L298N.

В якості задніх коліс прототипу виступить поворотне, пластмасове колесо на металевій основі з підшипником (рис 2.5).

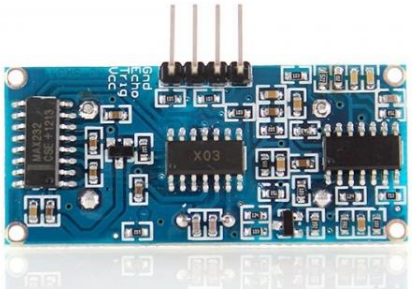
Рисунок 2.5 – Поворотне колесо.

Для реалізації функції розпізнавання об’єктів за розміром використовуватиметься ультразвуковий датчик відстані HC-SR04 (рис. 2.6). Він стабільний та точний, не має "сліпих зон". Може вимірювати відстань від 0 см до 1500мм, точність досягає 3 мм.

Основні характеристики:

- Робоча напруга: 3.8 - 5.5В;
- Тип: HC-SR04;
- Струм: 8 мА;
- Частота: 40 кГц;
- Максимальна дистанція: 1500 мм;
- Мінімальна дистанція: 0 см;
- Роздільна здатність: 3 мм;
- Ширина імпульсів: 10 мкс;
- Кут: 15 градусів;
- Зовнішні габарити: 37x20x15 мм.

а)



б)

Рисунок 2.6 – Ультразвуковий датчик відстані HC-SR04:

а) Фото спереду; б) Фото ззаду.

Приводитиметься в рух ультразвуковий датчик відстані буде за допомогою двох серводвигун моделі SG90 (рис. 2.7). Рух по горизонталі і вертикалі.

Основні характеристики:

- 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 x 3 см x 1.3 см;

Рисунок 2.7 – Серводвигун моделі SG90

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

Рисунок 2.8 – Пластмасові деталі, для утворення механізму

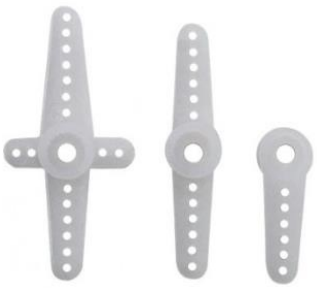


Рисунок 2.9 – Пластмасові деталі, для передачі руху серводвигуна

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

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

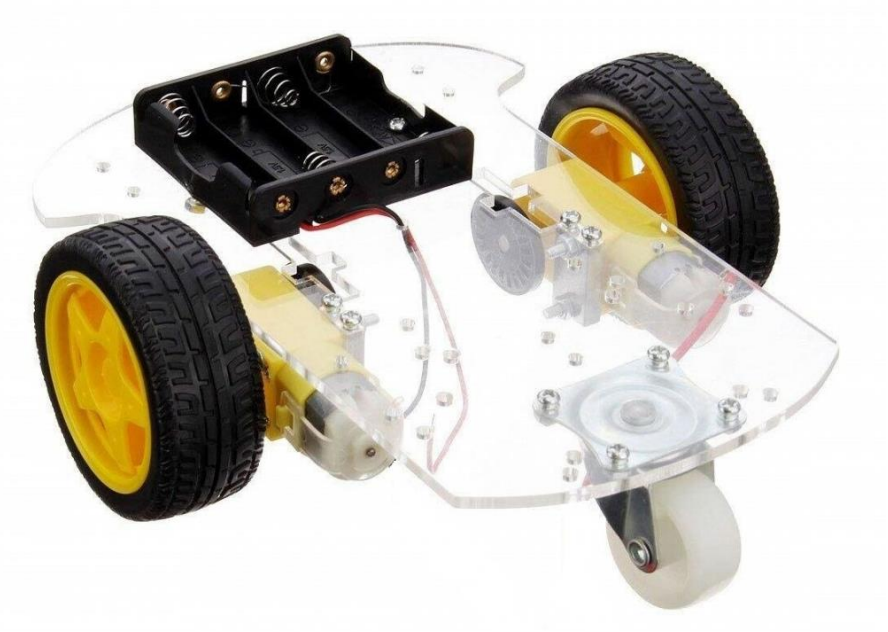


Рисунок 2.12 – Шасі робота в зборі

Для кріплення елементів до мікроконтролера використовуються провідники типу pin: Female to Female, Male to Male і Male to Female (рис. 2.13).

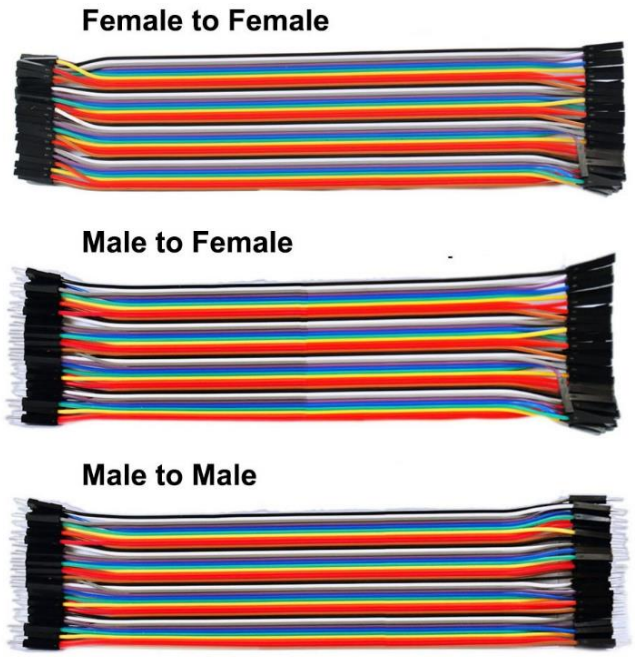


Рисунок 2.13 – Провідники

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;
- Профіль Bluetooth serial port;
- Живлення + 5VDC 50mA;
- Робочі температури -20 ~ +75 C;
- Розміри 26.9мм x 13 мм x 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.

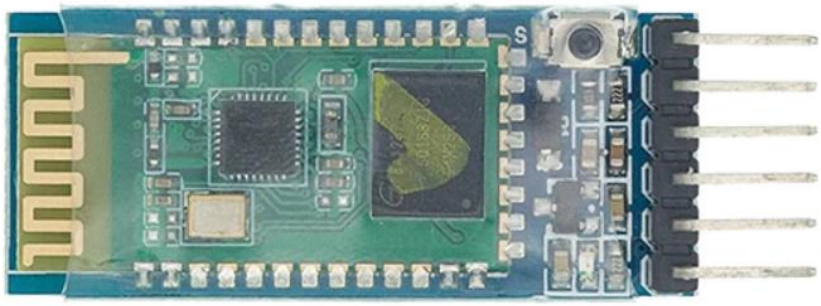


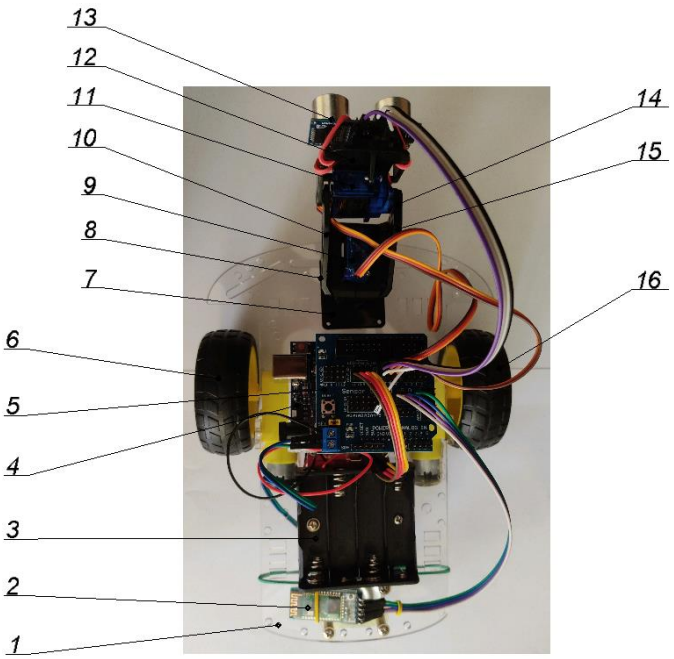
Рисунок 2.14 – Bluetooth модуль HC-05

У висновку, придбати данні елементи простіше всього готовим комплектом (рис. 2.15), в який не входять тільки провідників та Bluetooth модуль HC-05. Кріпильні елементи в комплекті.

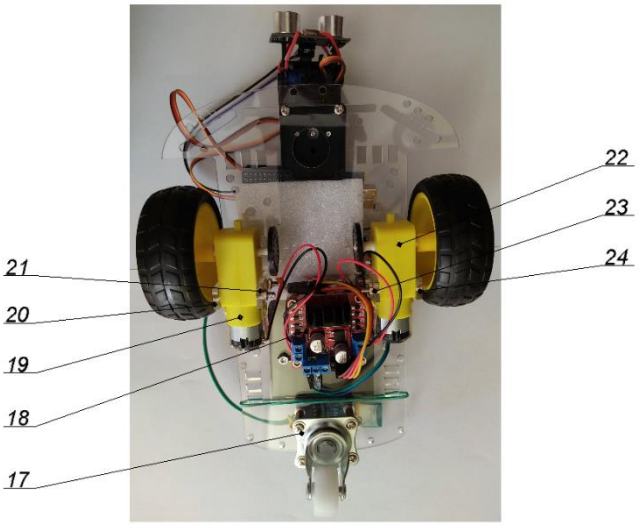
Рисунок 2.15 – Комплект для створення прототипу транспортно – складського робота

2.2 Конструкція прототипу транспортно-складського робота

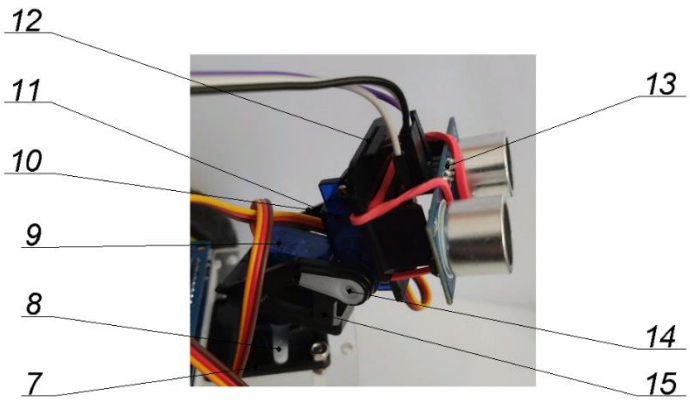
Елементи транспортно-складського робота в зборі, зображено на рисунку (рис. 2.16).



а)



б)



в)

Рисунок 2.16 – Прототип транспортно-складського робота в зборі:

а) Вигляд зверху; б) Вигляд знизу; в) Механізм керування ультразвуковим датчиком відстані.

де: 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.

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.

Прототип робота в зборі зображений на рисунку 2.17.

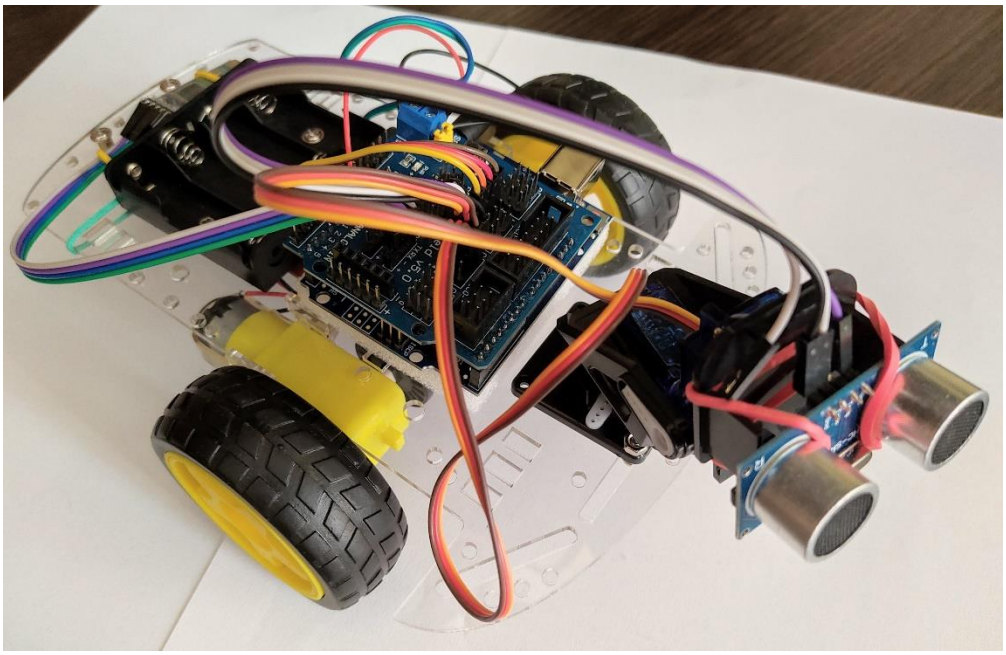


Рисунок 2.17 – Прототип транспортно-складського робота в зборі

2.3 Встановлення з'єднання з ПК, за допомогою Bluetooth модуля HC-05

Для встановлення з’єднання Bluetooth модуля HC-05 з ПК по протоколу Firmata потрібно налаштувати швидкість роботи модуля на швидкість 57600 бод. Для цього підключаємо Bluetooth модуль до USB порту ПК через перехідник USB To TTL UART, наприклад CP2102 (рис 2.18) по принципу: «RX» до «TX», «TX» до «RX», «GND» до «-» і «VCC» до «+», провідниками.

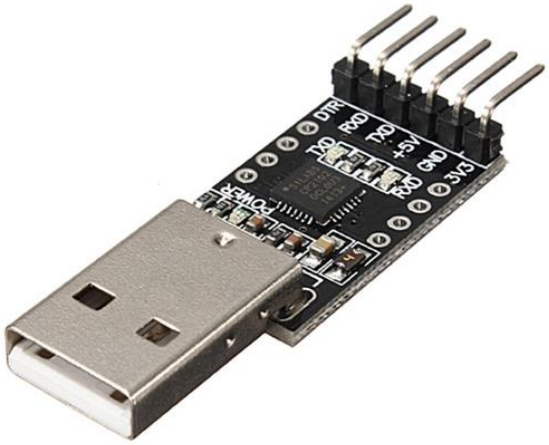


Рисунок 2.18 – Перехідник USB To TTL UART CP2102

Запускаємо ПП Termite і переходимо в налаштування. Вибираємо COM-порт, по якому підключений перехідник USB To TTL UART, швидкість по якій працює Bluetooth модуль в даний момент (по замовчуванню 9600бод, або 38400бод). В розділі «Transmitted text» вибираємо передання тексту «Append CR-LR». Для підтвердження і збереження налаштувань нажимаємо кнопку «OK» (рис. 2.19).

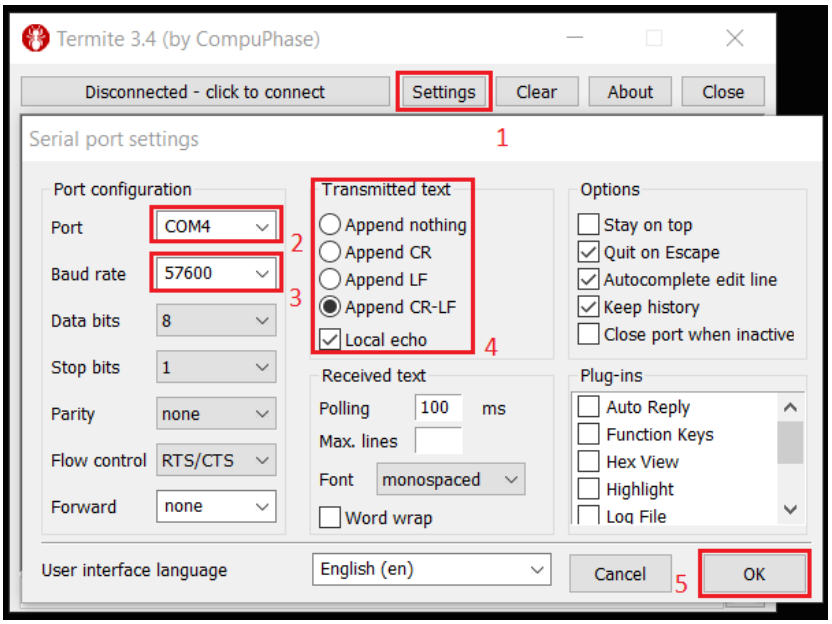


Рисунок 2.19 – Налаштування програми Termite

Після чого можемо вводити команди в термінал. При надсиланні команди слід затиснути кнопку на перехіднику USB To TTL UART. Для перевірки з’єднання можна ввести команду:

AT

Якщо Bluetooth модуль дасть відповідь «OK», значить з’єднання працює правильно. Для зміни швидкості вводимо команду:

```
AT+UART=57600,0,0
```

де «57600» – швидкість роботи модуля.

Для перевіри підтвердження налаштувань перезапускаємо ПП Termite. В налаштуваннях вибираємо «Baud rate» (швидкість роботи модуля) «57600», нажимаємо кнопку «OK» для підтвердження і збереження налаштувань.

В терміналі пишемо і надсилаємо команду (рис. 2.20):

```
AT+UART
```

Якщо відповіддю є «+UART=57600,0,0» налаштування пройшло успішно.

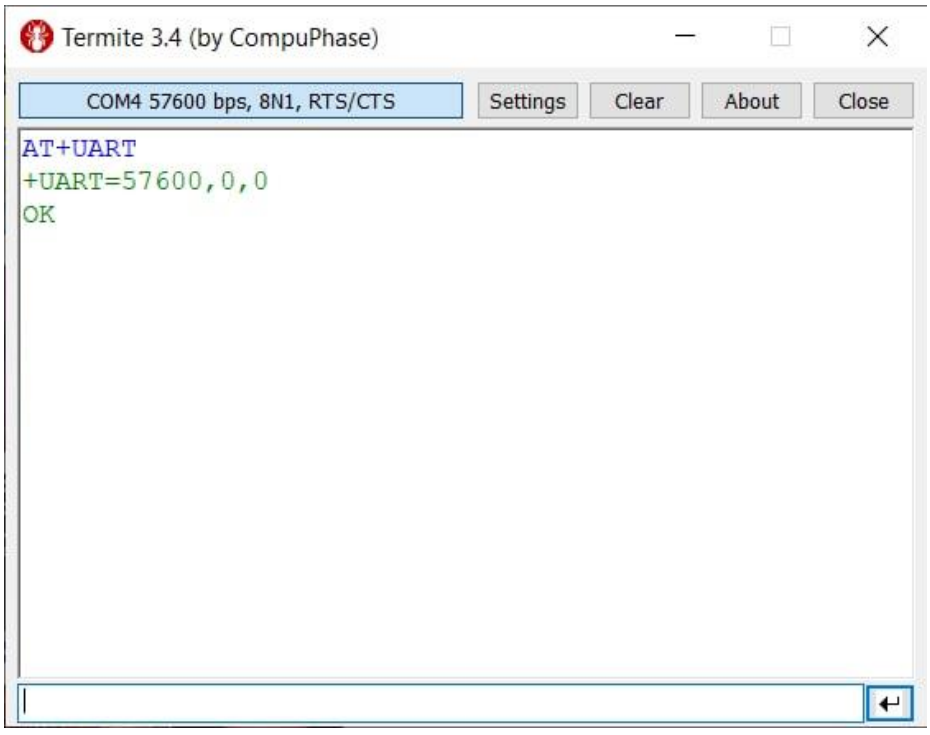


Рисунок 2.20 – Перевірка швидкості роботи Bluetooth модуля

З цього моменту, з’єднання майбутньої керуючої Python програми з «скетчем» StandardFirmata на мікроконтролері Arduino UNO R3 працюватиме відповідним чином. Можна вільно влаштовувати з’єднання з Bluetooth модулем HC-05, за допомогою внутрішніх опцій операційної система Windows, чи інших.

У випадку з операційною системою Windows 10, COM-порт для встановлення з’єднання можна дізнатися перейшовши в розділ «Пристрої Bluetooth та інші пристрої» налаштувань, де вибрати «Інші параметри Bluetooth» (рис. 2.21).

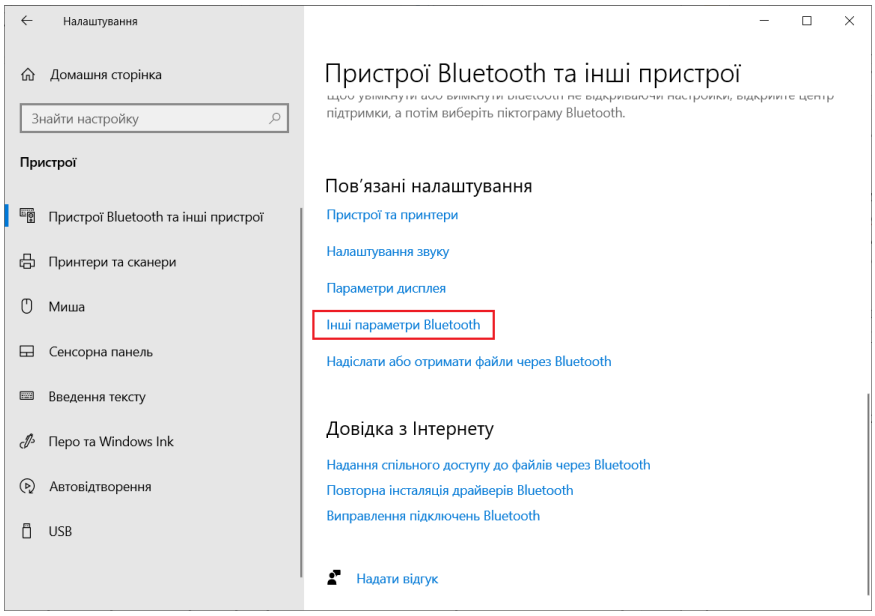


Рисунок 2.21 – Опція «Інші параметри Bluetooth»

В новому вікні перейти на вкладку «COM-порти», де потрібний COM порт буде з відміткою «Port» (рис. 2.22).

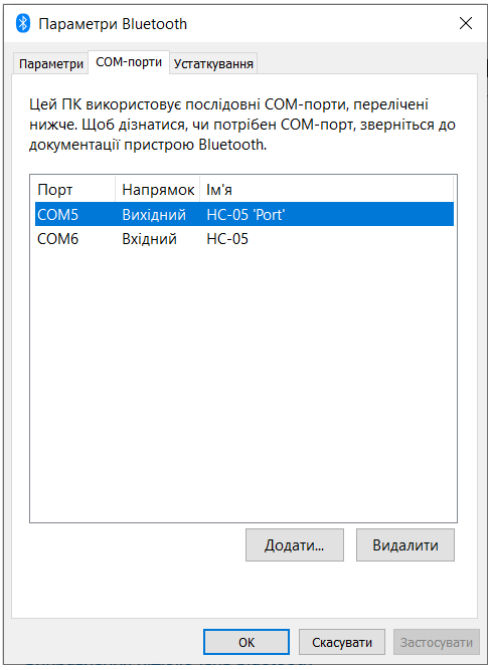


Рисунок 2.22 – COM-порти Bluetooth

2.4 Керуюча програма робота

Опис керуючої програми робота, з ідентифікацією об’єктів за допомогою ультразвукового датчика відстані «Program_V1». Робот ідентифікує об’єкт шириною та висотою 10см.

Початок програми:

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

Імпорт модулей:

```
from pyfirmata import Arduino, util
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
import random

time=util.time
```

Звернення до мікроконтролера Arduino, «пінів», що яких підключені різні пристрої:

```
board = Arduino('COM25')
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')
```

Функція «ping» повертає середнє значення відстані, виміряної ультразвуковим сенсором, n - кількість вимірювань «def ping(n)»:

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

Функція «scan» сканує простір по горизонталі. Цикл, в якому змінюється значення кута і цеклічно викликається функція «ping», щоб виконувати сканування при різних значеннях кута. Крок зміни кута 13°:

```
def scan():
	angle=0
	X=[]
	Y=[]
	while angle<=130:
		servo1.write(angle)
		time.sleep(1)
		dist=ping(3)
		X.append(angle)
		Y.append(dist)
		print board.analog[0].read()
		angle+=13
	return X,Y
```

Функція «scan3D» є аналогічною функції «scan», але вміщує вкладені цикли, які дозволяють сканувати по горизонталі на рівному куті підйому:

```
def scan3D():
      H,V,D=[],[],[]
      for v in [110, 60]: # скануємо знизу і зверху
            h=0
            while h<=130:
                  servo2.write(v) # 70-130
                  time.sleep(0.01)
                  servo1.write(h)
                  time.sleep(1)
                  d=ping(3)
                  H.append(h)
                  V.append(v)
                  D.append(int(d<40 and d!=0))
                  #print h,v,d
                  #print board.analog[0].read(),
                  h+=13
                  return H,V,D
```

Наступні функції керують двигунами робота. Являють собою ряд заготовлених шаблонів пересування, для майбутнього використання. Параметр «t» – час роботи двигуна:

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

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

def RB(t):
      pin10.write(1)
      pin11.write(0)
      time.sleep(t)
      stop()

def LF(t):
      pin8.write(0)
      pin9.write(1)
      time.sleep(t)
      stop()

def LB(t):
      pin8.write(1)
      pin9.write(0)
      time.sleep(t)
      stop()

def F(t):
      RF(0.1, False)
      LF(t)
      stop()
```

```
def B(t):
      RB(0.1)
      LB(t)
      stop()
```

Масив ознак «x», для машинного навчання, за якими розпізнається об’єкт шириною та висотою 10см.:

```
x=np.array([
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
])
```

Масив класів «y». Класу 1 відповідає ситуація – об’єкт наявний, класу 0 – об’єкт відсутній:

```
y=np.array([1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0])
```

Процедура машинного навчання, з перехресною перевіркою якості моделі. За допомогою функції «fit» виконується навчання :

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1)
model=GradientBoostingClassifier(n_estimators=10, learning_rate=0.1,
max_depth=3)
model.fit(x_train, y_train) # виконати навчання
print y_test # фактичні тестові класи
print model.predict(x_test) # прогнозовані тестові класи
print model.score(x_test, y_test) # правильність класифікатора

from sklearn.model_selection import cross_val_score
```

Функція «cross_val_score» перехресна перевірка якості моделі:

```
s=cross_val_score(model, x, y, cv=9)
print s, s.mean() # правильність класифікатора на кожній ітерації і її середнє
значення
print model.predict([[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]])
```

Головний цикл програми. Виконується сканування, результати заносяться в масиви «H», «V», «D», масив «D» використовується для прогнозу моделі машинного навчання. Функція «predict» повертає прогнозований клас. Якщо об’єкт ідентифікований – робот штовхає його на певну відстань. Ця частина програми, може бути зміненою в залежності від потреб:

```
      while True:
            H,V,D=scan3D()
            p=model.predict(np.array(D).reshape(1,-1))[0] # клас: 1 - є об'єкт,
      0 - немає
            if p: # якщо об'єкт ідентифіковано
                  F(5) # штовхати!
      else:
            # інакше рухатись випадково
            direction=random.choice([F,LF,RF])
            direction(random.random())
```

Кінець програми.

```
board.exit()
```

Опис версія програми для тестування прототипу в майбутньому змаганні, грі змагання роботів «Program_V2». Програма відрізняється від «Program_V1», підключенням інших бібліотек, зокрема «urllib2», що дозволяє звертатися до web сервера:

```
from pyfirmata import Arduino, util
import numpy as np
import urllib2
import random
```

Основний цикл програми відрізняється тим, що робот не використовує алгоритми машинного навчання. Робот звертається до web сервера який повідомляє йому координати кольорових об'єктів, за допомоги машинного зору.

Функція «getXY» звертається до web сервера, Отримує відповідь, в якій знаходяться координати двох кольорових об’єктів. До прикладу координати роботів, або ж шуканих об’єктів взаємодії:

```
def getXY():
      response = urllib2.urlopen('http://192.168.0.101/')
      data=response.read()
      response.close()
      if not data:
            time.sleep(2)
            return None
      x1,y1,x2,y2=[int(x) for x in data.split()]
      return x1,y1,x2,y2
```

Приклад алгоритму взаємодії з шуканим об'єктом. Ця частина програми повинна змінюватися учасником змагання.

Функцією «inCercle» робот отримує інформацію, чи знаходиться в межах певнорго круга заданим радіусом:

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

За допомогою функції «dist» розраховуємо відстань між точками координат робота і об’єкта:

```
def dist(x1,y1,x2,y2):
      return ((x2 - x1)**2 +(y2 - y1)**2)**0.5

while True:
      data=getXY()
      if not data: continue
      x1,y1,x2,y2=data
      d1=dist(x1,y1,x2,y2)
```

Алгоритм взаємодії з шуканим об'єктом. Робот рухається вперед, отримує координати свого місця знаходження, визначається відстань до шуканого об'єкта, якщо відстань більша ніж попередня, робот рухається в право, інакше прямо.

```
      F(1)
      data=getXY()
      if not data: continue
      x1,y1,x2,y2=data
      d2=dist(x1,y1,x2,y2)
      if d2>d1:
            R(1)
      else:
            F(1)
      time.sleep(2)
```

Кінець програми.

```
board.exit()
```

В свою чергу «CV2_Detect_Color_Web», є веб сервером, який повідомляє клієнтам, до прикладу «Program_V2» координати кольорових об’єктів, які визначає за допомого відеокамери і бібліотеки «openCV».

Початок програми:

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

Імпорт модулей:

```
from bottle import route, run, request, WSGIRefServer
import numpy as np
import cv2
```

Функція «detectColor» накладає на зображення маску, для його бінарізації і знаходження областей з заданим кольором. Визначаються координати центру області:

```
def detectColor(h1, s1, v1, h2, s2, v2):
      h_min = np.array((h1, s1, v1), np.uint8)
      h_max = np.array((h2, s2, v2), np.uint8)
      RealTimeMask = cv2.inRange(frame_hsv, h_min, h_max)
      moments = cv2.moments(RealTimeMask, 1)
      dM01 = moments['m01']
      dM10 = moments['m10']
      Area = moments['m00']
      #print Area
      if Area:
            x = int(dM10 / Area)
            y = int(dM01 / Area)
            cv2.circle(frame, (x, y), 10, (0,0,255),-1)
            return x, y, RealTimeMask
      else:
            return None, None, RealTimeMask
```

Доступ вебкамери:

```
cap = cv2.VideoCapture(0)
frame=None
frame_hsv=None
```

Функція «defXY» викликається два рази, для визначення різних кольорів, повертає координати різних кольорових об’єктів:

```
def getXY():
      global frame, frame_hsv
```

Отримуємо поточний кадр з вебкамери:

```
      ret, frame = cap.read()
      frame_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV )
      x1, y1, RealTimeMask=detectColor(159, 39, 103, 180, 255, 255)
      x2, y2, RealTimeMask=detectColor(25, 70, 159, 55, 162, 255)
      if all([x1,y1,x2,y2]): return "%i %i %i %i"%(x1,y1,x2,y2)
      return ""

@route('/') # http://localhost:80
```

Функція «index» працює в момент, коли клієнт звертається до web сервера, повертає значення функції «getXY»:

```
def index():
      return getXY()
```

Запуск сервера:

```
run(server=WSGIRefServer, host='192.168.0.101', port=80)
```

Завершуємо роботу з камерою:

```
cap.release()
cv2.destroyAllWindows()
```

2.5 Тестування прототипу в грі змагання роботів

Тестування прототипу робота і налагодження алгоритму проводиться в змаганні, на платформі для тестування (рис 2.23). Веб камера для ідентифікації об’єктів закріплена над платформою, в формі круга. Керуюча Python програма запущена на ПК, з’єднання з прототипом робота реалізовано по Bluetooth (рис 2.24). Кожен робот і об’єкт взаємодії позначені різними кольорами.

За правилами змагання прототип робота повинен виштовхувати об’єкти білого кольору за межу і не дотикатися до об’єктів чорного кольору. За кожен успішно виштовханий об’єкт присвоюється бал.

Також існують різні рівні складності завдань, які ставляться перед учасниками. Найлегший рівень 1 - це коли програми керування роботами отримують від веб сервера координати усіх об’єктів. Ці координати веб сервер (на основі мікро-вебфреймворку bottle) обчислює за допомогою алгоритмів машинного зору (наприклад OpenCV-функції inRange). На вищому рівні 2 ці програми отримують від веб сервера тільки координати усіх роботів. Найскладніший рівень 3 не передбачає передачі будь-яких даних веб сервером. На рівнях 2 та 3 програмам керування слід використовувати власні сенсори і більш складні алгоритми для ідентифікації об’єкта. Зокрема можуть бути використані моделі машинного навчання на основі градієнтного бустінга для задачі класифікації з Python-пакету scikit-learn. 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.

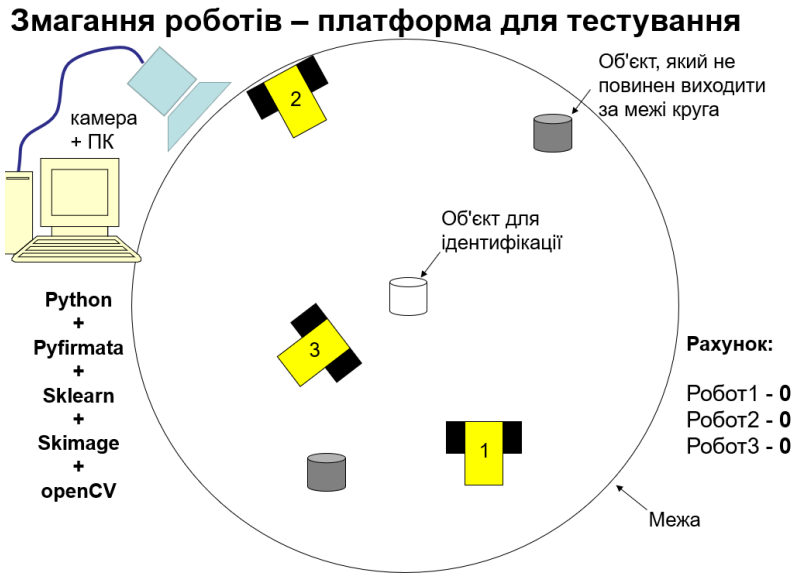


Рисунок 2.23 – Платформа тестування роботів

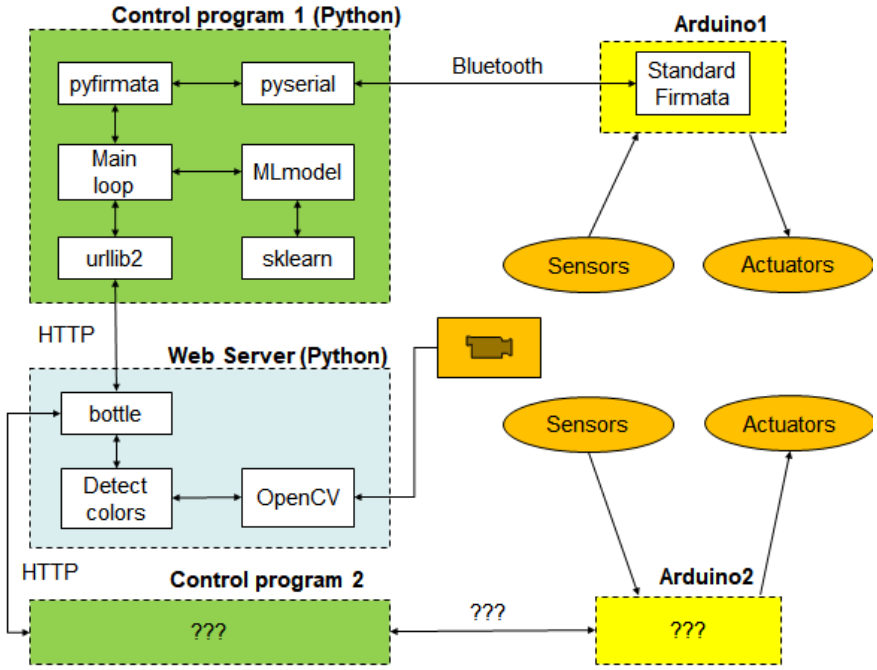


Рисунок 2.24 – Схема програми платформи тестування роботів

2.6 Проектування повноцінної моделі робота, за допомогою ПП SolidWorks

Почати проектування моделей, для повноцінної моделі робота слід з деталей, що будуть виготовлятися, тобто не є стандартизованими. Друкуватись

на 3D принтері з PETG пластика та вирізатись лазерним різаком з акрилових листів.

Створювати моделі для друку на 3D принтері з PETG слід таким чином, конструкцією, щоб їх було легко надрукувати, з як найменшою кількістю підпорок.

Статичний тримач для ультразвукового датчика відстані HC-SR04 повинен мати відсік 45х25мм і кріпитися до акрилової плити гвинтом М4. Готовий тримач зображено на рисунку 2.25.

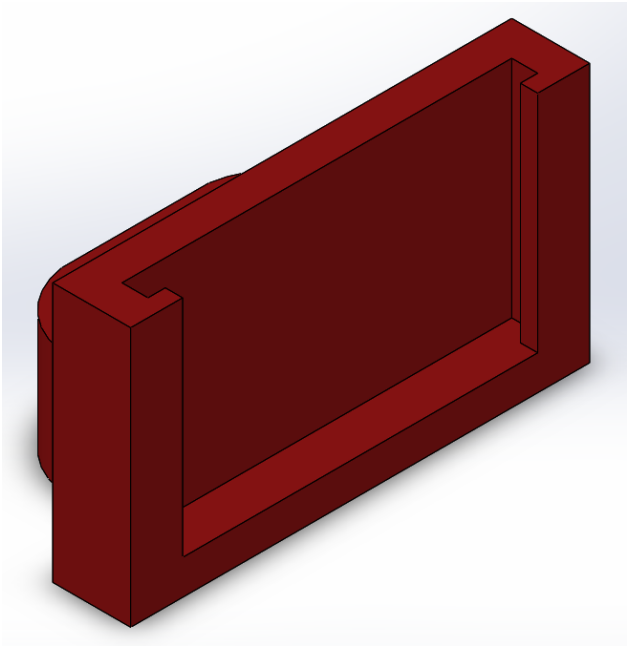


Рисунок 2.25 – Статичний тримач для ультразвукового датчика відстані HC-SR04

Дерево побудови зображено на рисунку 2.26. Операції побудови моделі зображено на рисунку 2.27.

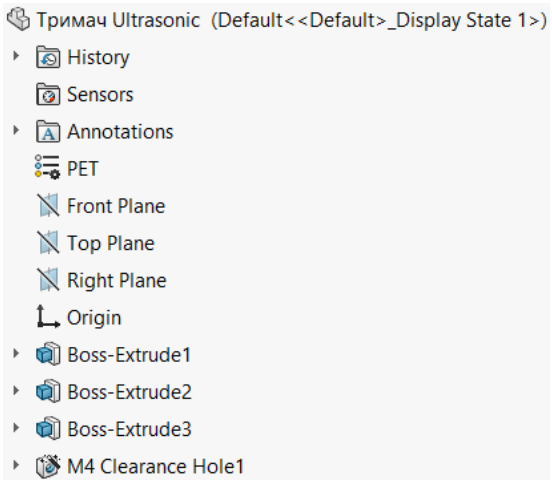
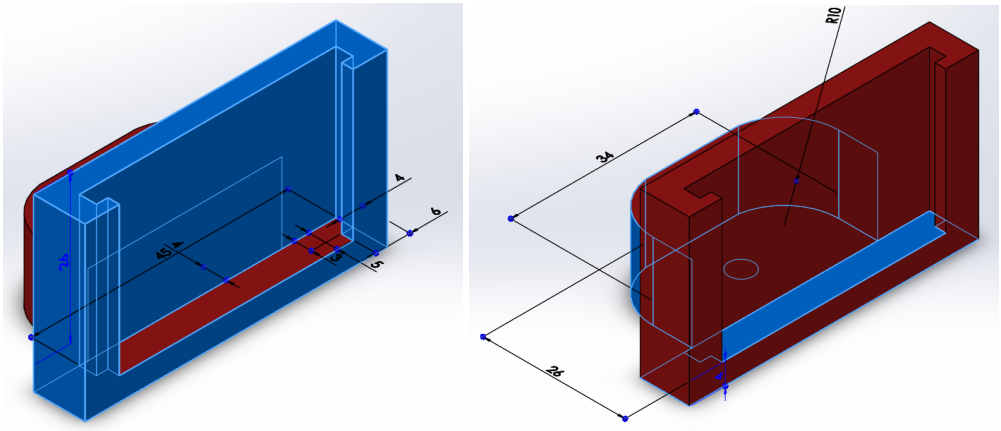
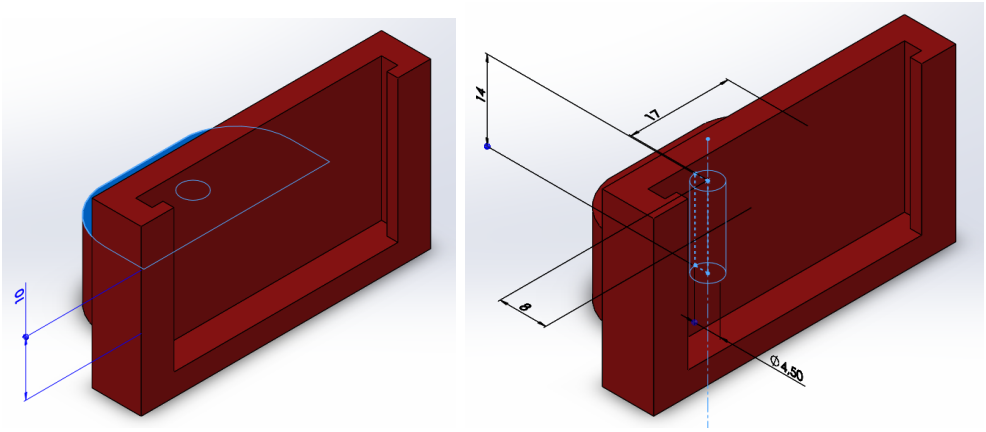


Рисунок 2.26 – Дерево побудови моделі статичного тримача для ультразвукового датчика відстані HC-SR04



1) 2)



3) 4)

Рисунок 2.27 – Операції побудови моделі статичного тримача для ультразвукового датчика відстані HC-SR04

де: 1) Операція 1; 2) Операція 2; 3) Операція 3; 4) Операція 4.

Рухомий тримач для ультразвукового датчика відстані HC-SR04 повинен мати відсік 45х25мм і кріпитися до механізму FPV, з пластмасових деталей і сервомоторів SG90. Готовий тримач зображено на рисунку 2.28.

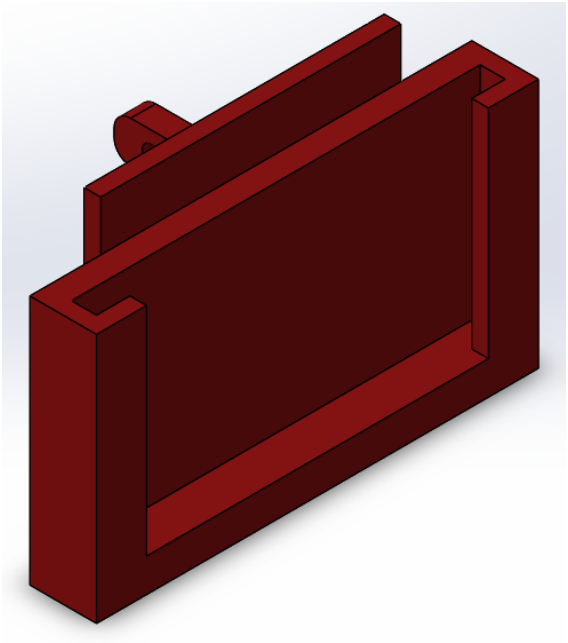


Рисунок 2.28 – Рухомий тримач для ультразвукового датчика відстані HC-SR04

Дерево побудови зображено на рисунку 2.29. Операції побудови моделі зображено на рисунку 2.30.

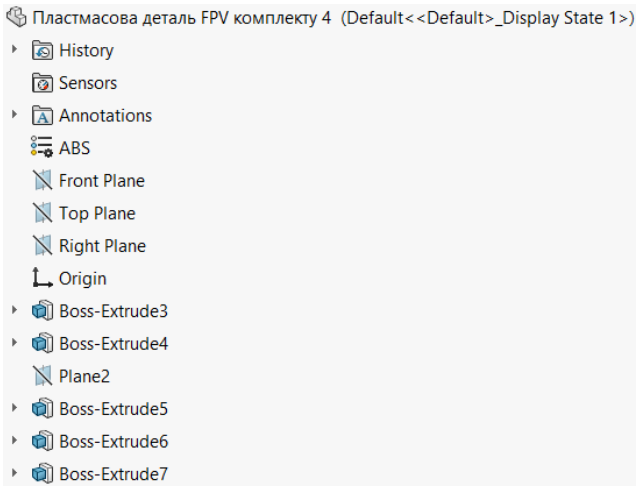
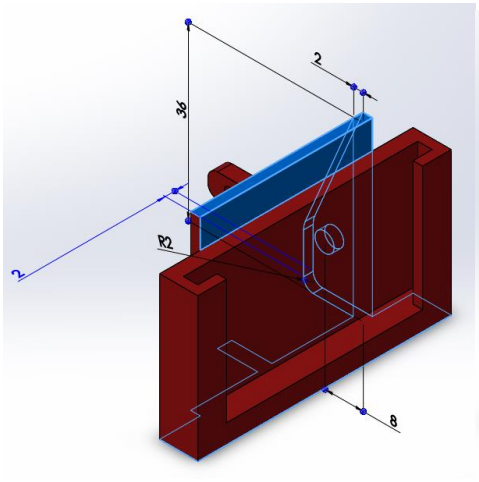
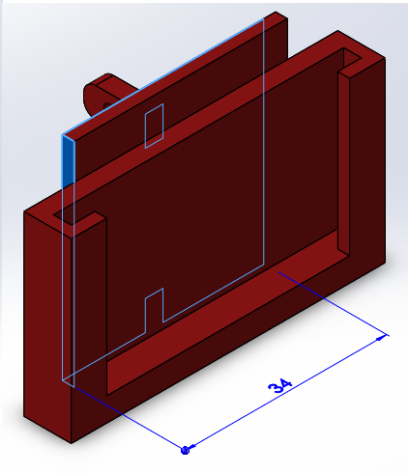


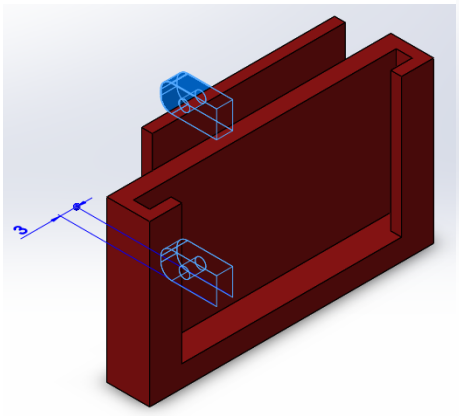
Рисунок 2.29 – Дерево побудови моделі рухомого тримача для ультразвукового датчика відстані HC-SR04



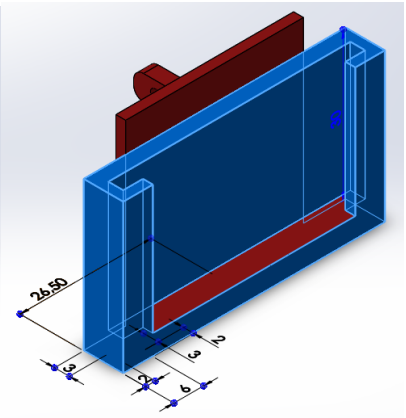
1)



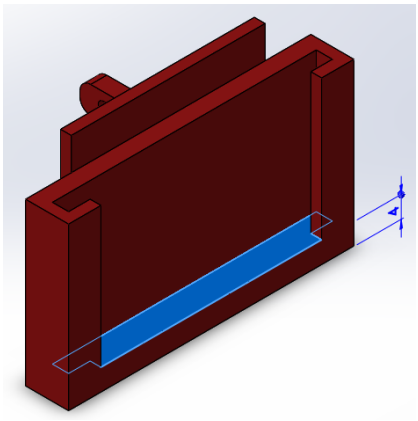
2)



3)



4)



5)

Рисунок 2.30 – Операції побудови моделі рухомого тримача для ультразвукового датчика відстані HC-SR04

де: 1) Операція 1; 2) Операція 2; 3) Операція 3; 4) Операція 4;

5) Операція 5.

Робот буде обладнаний захоплювачем, який також надрукований на 3D принтері з PETG пластика (рис. 2.31).

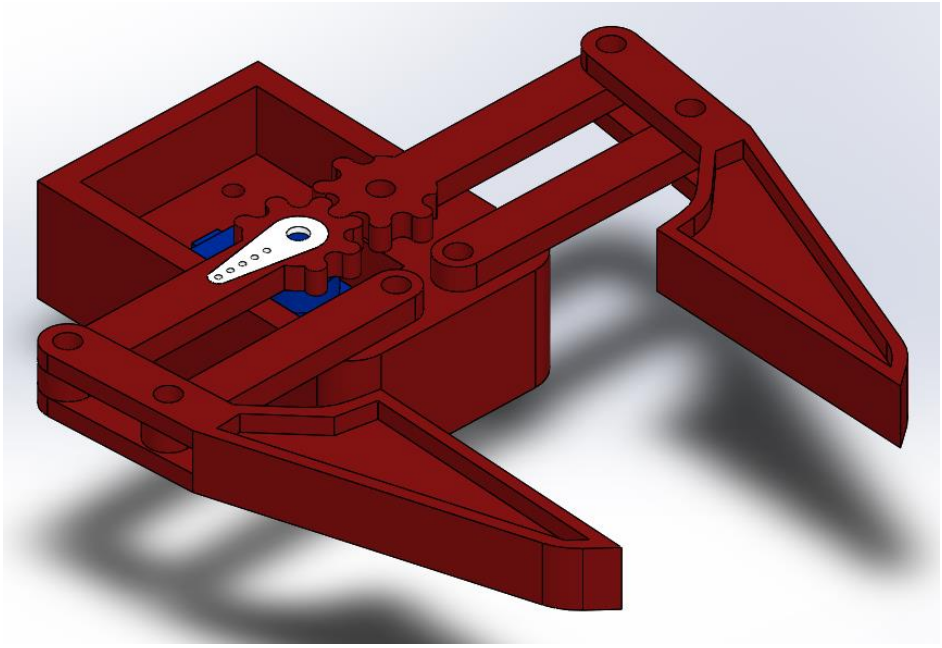


Рисунок 2.31 – Модель захоплювача в зборі

Захоплювач працює завдяки сервомотора SG90, налічує 5 різних деталей, та комплектну з сервомотором SG90 пластмасову деталь, для передачі руху (рис. 2.32).

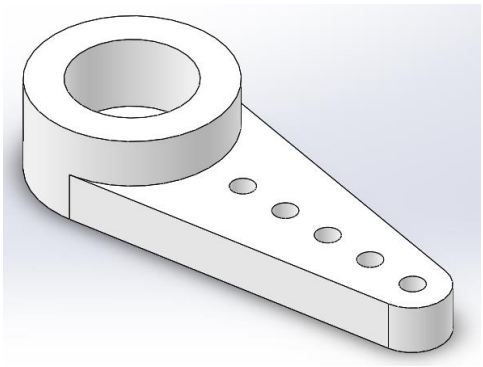


Рисунок 2.32 – Модель пластмасової деталі передачі руху

Основа захоплювача зображена на рисунку 2.33. Дерево побудови зображено на рисунку 2.34. Операції побудови на рисунку 2.35.

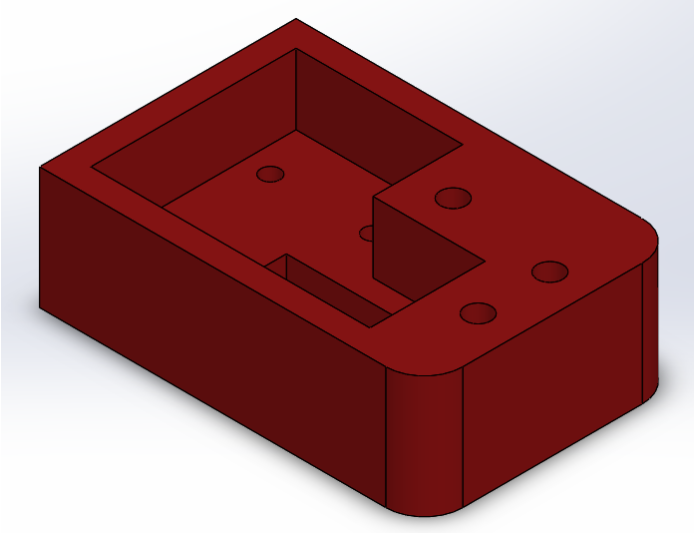


Рисунок 2.33 – Основа захоплювача

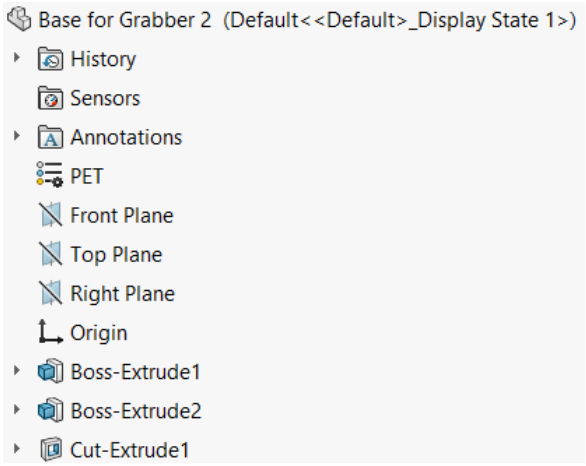
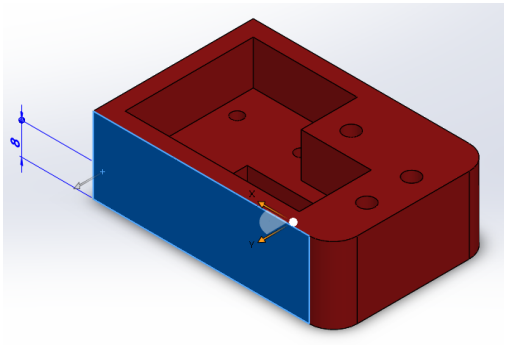
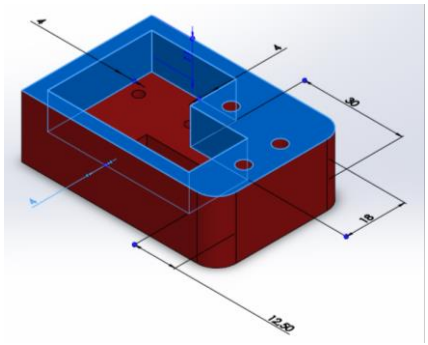


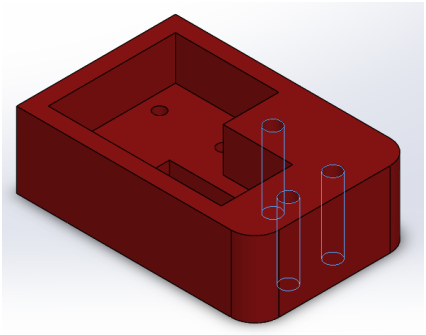
Рисунок 2.34 – Дерево побудови моделі основи захоплювача



1)



2)



3)

Рисунок 2.35 – Операції побудови моделі захоплювача

де: 1) Операція 1; 2) Операція 2; 3) Операція 3.

Лівий шестерний з’єднувач (рис. 2.36) та звичайний з’єднувач, що використовується два рази (рис. 2.37) виконані однією операцією, з ескізу.

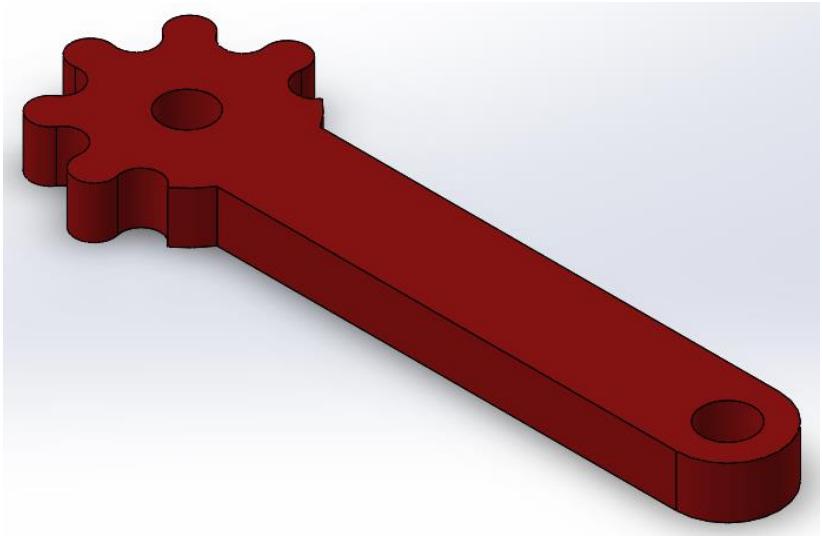


Рисунок 2.36 – Лівий шестерний з’єднувач

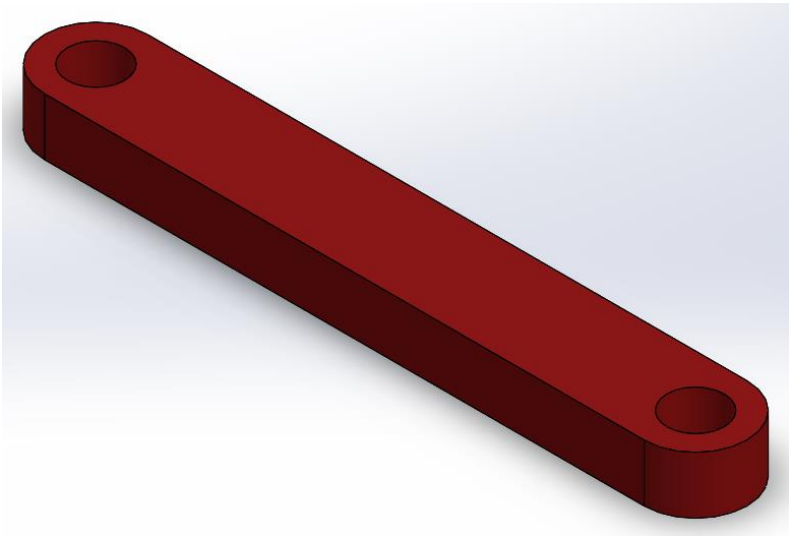


Рисунок 2.37 – З’єднувач

Правий шестерний з’єднувач (рис. 2.38) має роз’єм для пластмасової деталі, для передачі руху з серводвигуна SG90 (рис. 2.32).

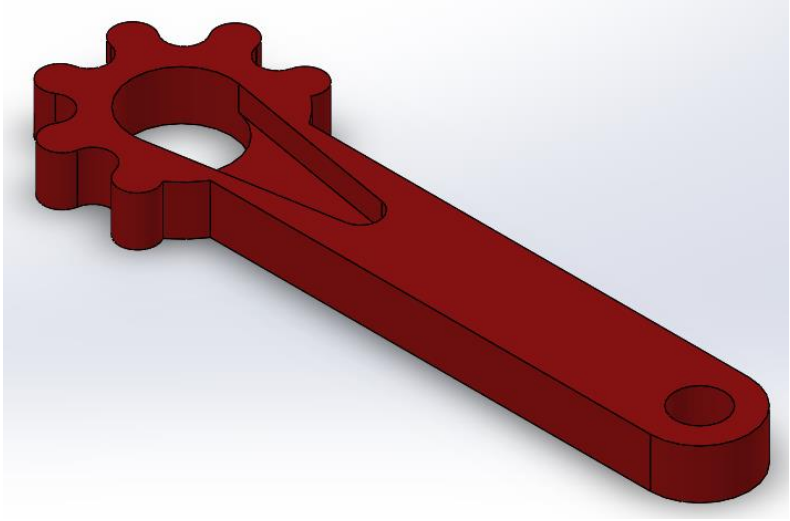


Рисунок 2.36 – Правий шестерний з’єднувач

Дерево побудови зображено на рисунку 2.37. Операції побудови моделі зображено на рисунку 2.38.

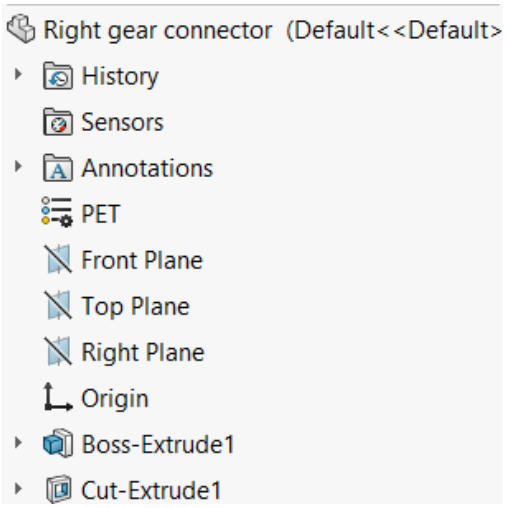
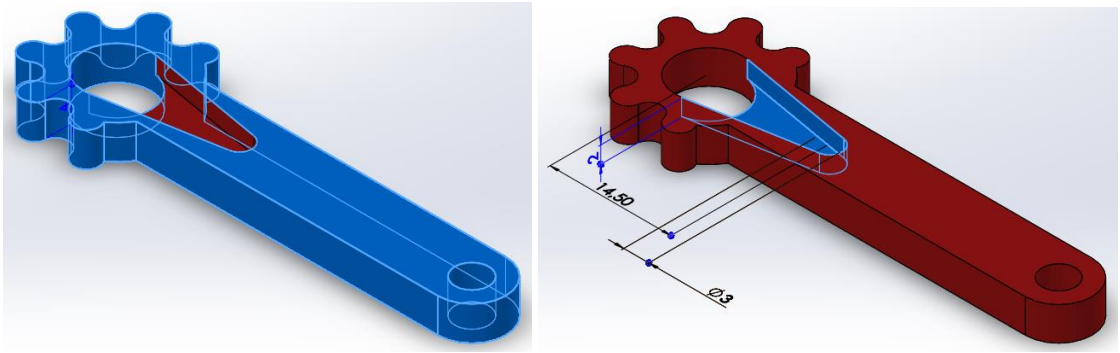


Рисунок 2.37 – Дерево побудови моделі правого шестерного з’єднувача



1) 2)

Рисунок 2.38 – Операції побудови моделі правого шестерного з’єднувача

де: 1) Операція 1; 2) Операція 2.

Деталь «клешня» (рис. 2.39), що контактує з об’єктом взаємодії і кріпиться до з’єднувачів, використовується два рази.

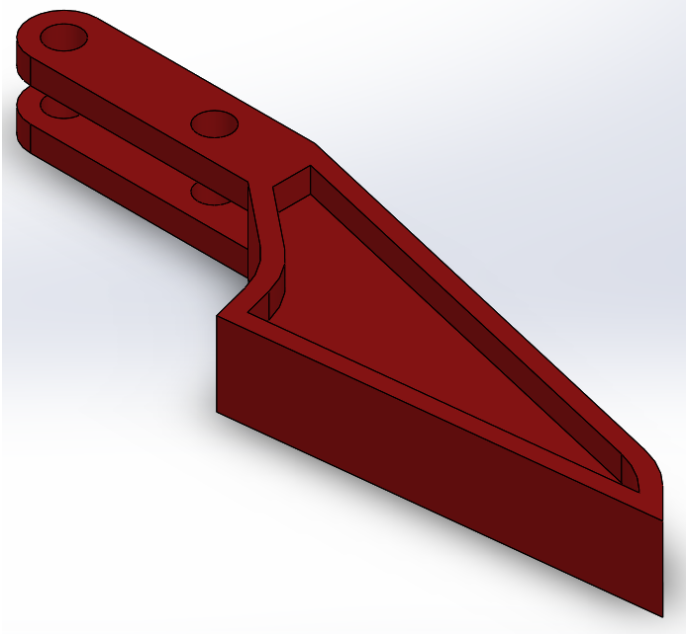


Рисунок 2.39 – Деталь «клешня»

Дерево побудови зображено на рисунку 2.40. Операції побудови моделі зображено на рисунку 2.41.

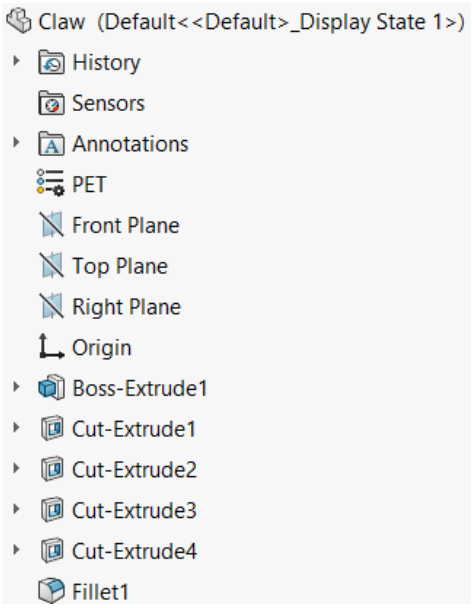
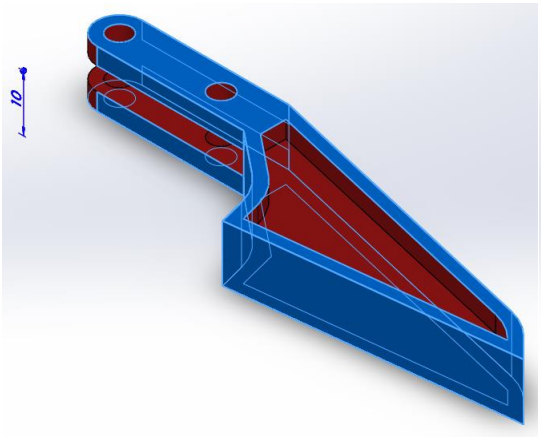
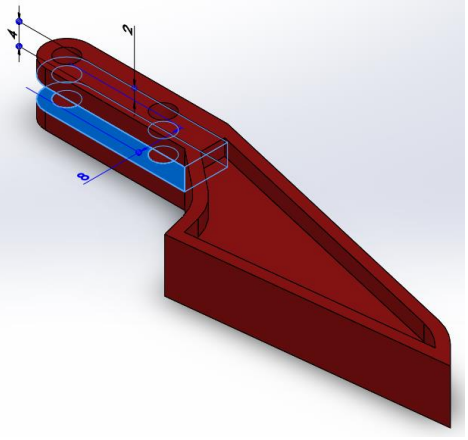


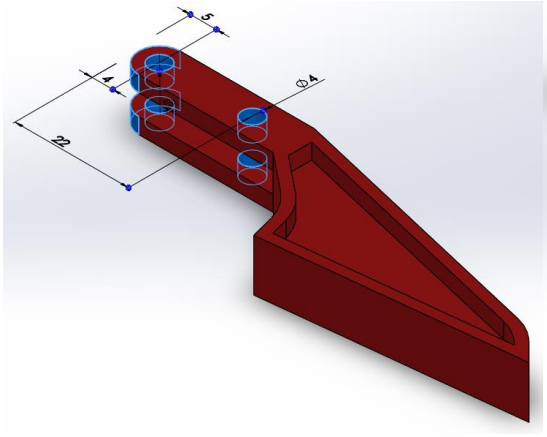
Рисунок 2.40 – Дерево побудови моделі деталі «клешня»



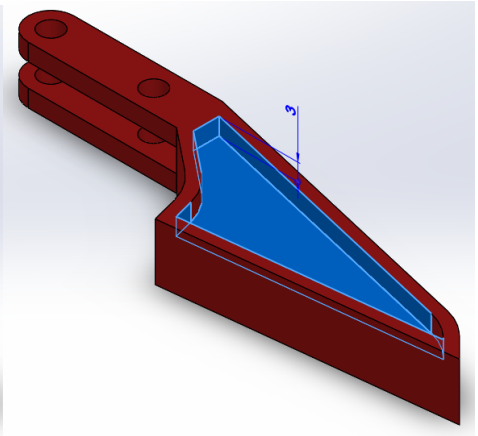
1)



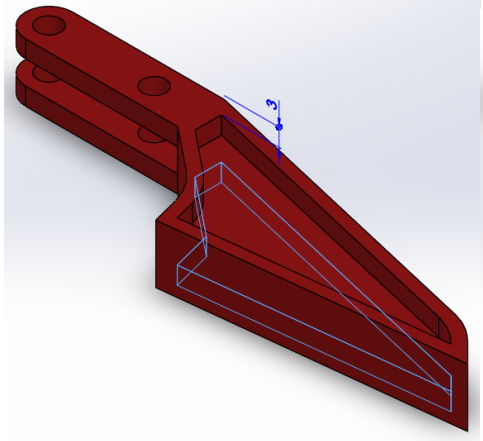
2)



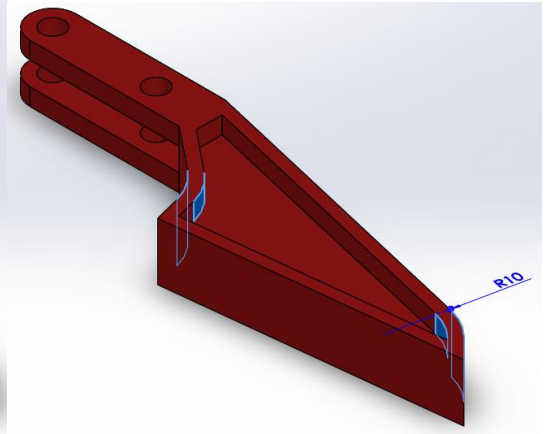
3)



4)



5)



6)

Рисунок 2.41 – Операції побудови моделі деталі «клешня»

де: 1) Операція 1; 2) Операція 2; 3) Операція 3; 4) Операція 4; 5) Операція 5; 6) Операція 6.

Основа робота, що вирізається з акрилового листа товщиною 10мм (рис. 2.42) являє собою плиту розміром 300х168мм з групами наскрізних отворів, для закріплення елементів на ній, за допомогою гвинтів. А також прямокутного, наскрізного отвору для сервомотора SG90.

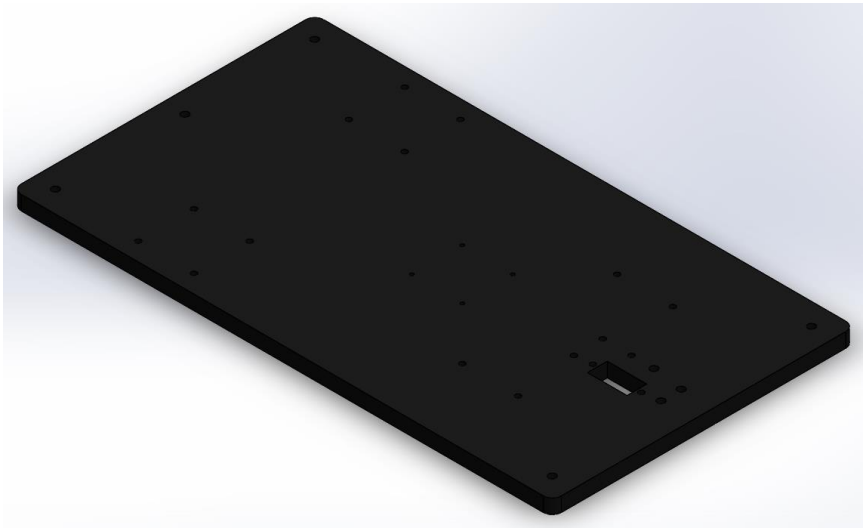


Рисунок 2.42 – Основа робота

Дерево побудови основи робота зображено на рисунку 2.43. Воно налічує одну операцію витягування (рис. 2.44) та п’ять операцій вирізання, груп отворів (рис. 2.45).

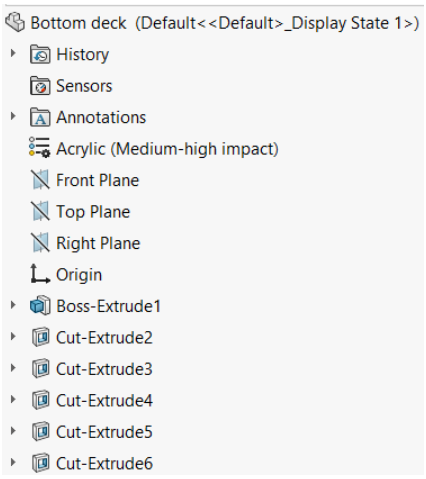


Рисунок 2.43 – Дерево побудови моделі основи робота

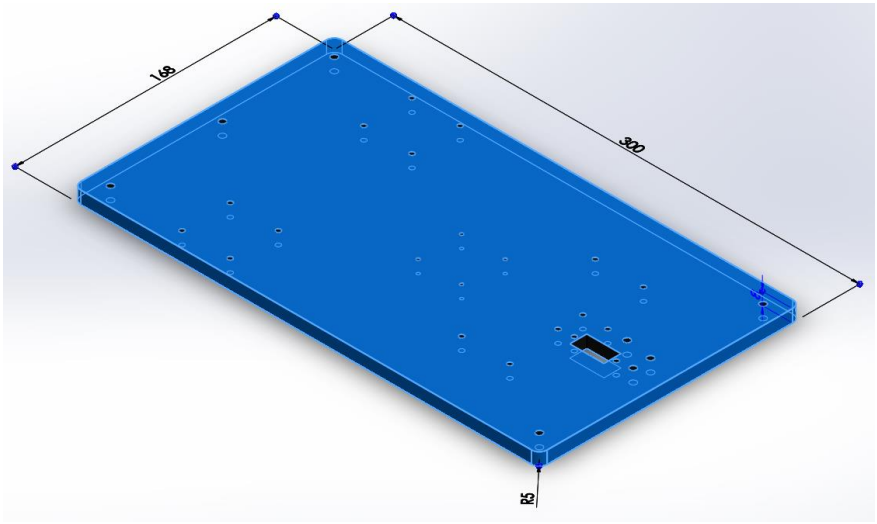


Рисунок 2.44 – Операція витягування

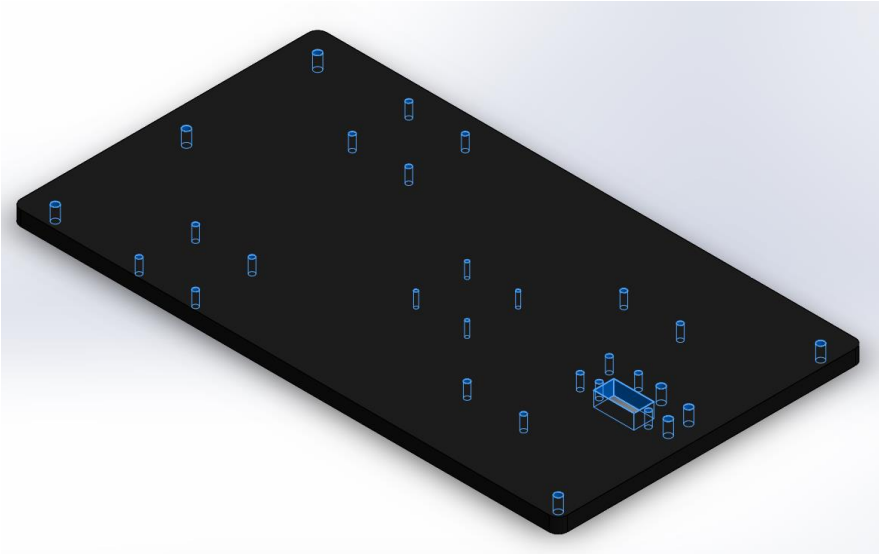


Рисунок 2.45 – Операції вирізання

В повноцінної моделі робота двигунами виступають двигуни типу NEMA 17 моделі MINEBEA 05701607. Рух на передні колеса передаватиметься за допомогою пасової передачі, паса і зубчастих роликів. Двигуни кріпляться до основи робота за допомогою стандартизованих кутників, з отворами збільшеного діаметру, фрезерною операцією. Задні колеса до валів двигунів кріпитимуться за допомогою власних муфт. Передні колеса за допомогою власних муфт кріпяться на гвинт М5, до якого кріпиться підшипник, що щільно вставлений в отвір кутника.

Ближче до центру основи встановлений рухомий механізм, що складається з пластмасових деталей FPV, двох серводвигунів та раніше спроектованого тримача ультразвукового датчика відстані HC-SR04, відтворено окремою 3D збіркою, для спрощення прив’язок між елементами 3D збірної моделі цілого робота.

Моделі стандартизованих деталей буде спроектовано зі спрощеною геометрією, але відповідністю розмірів їх реальних прототипів, за для полегшення майбутньої 3D збірки робота, в плані затрат системних потреб, за для її створення та відтворення.

Отже слід створити моделі наступних, стандартизованих деталей: кроковий двигун NEMA 17 MINEBEA 05701607 (рис. 2.46), ремінь замкнутий 280мм 280-2gt-6 (рис. 2.47), зубчастий ролик, шків ременя GT2 5мм 12311 (рис. 2.48), кутник (рис. 2.49), пластмасова деталь FPV комплекту 1 (рис. 2.50), пластмасова деталь FPV комплекту 2 (рис. 2.51), пластмасова деталь FPV комплекту 3 (рис. 2.52), пластмасова деталь для передачі руху серводвигуна 1 (рис. 2.32), пластмасова деталь для передачі руху серводвигуна 2 (рис. 2.53), серводвигун SG90 (рис. 2.54), колесо з муфтою (рис. 2.55) відтворено окремою 3D збіркою (рис. 2.55).

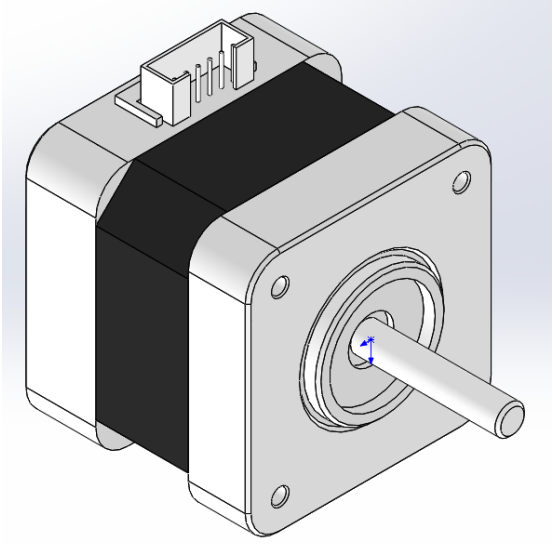


Рисунок 2.46 – Модель NEMA 17 MINEBEA 05701607

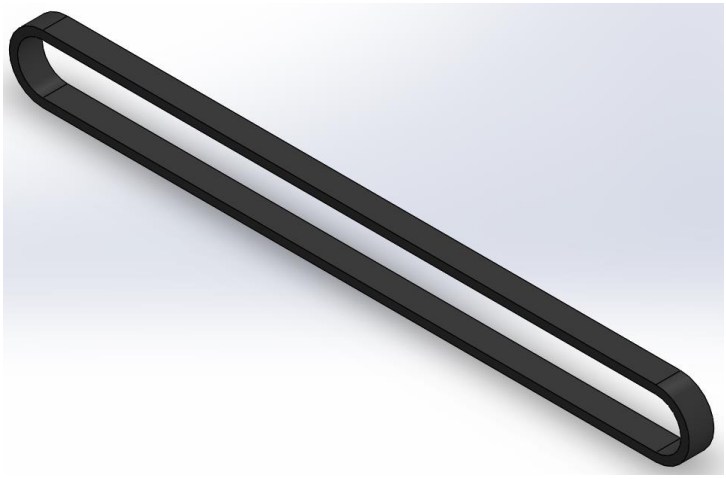


Рисунок 2.47 – Модель реміня замкнутого 280мм 280-2gt-6

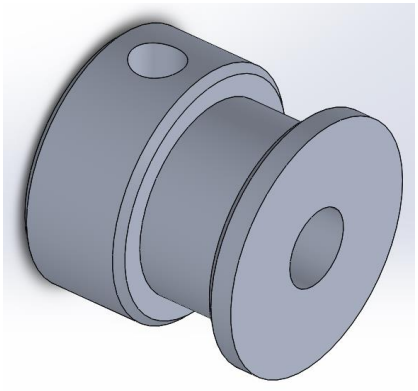


Рисунок 2.48 – Модель зубчастого ролика, шківа ременя GT2 5мм 12311

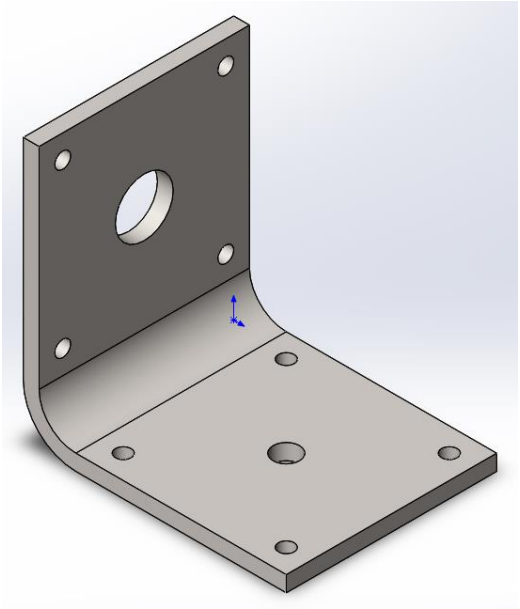


Рисунок 2.49 – Модель кутника

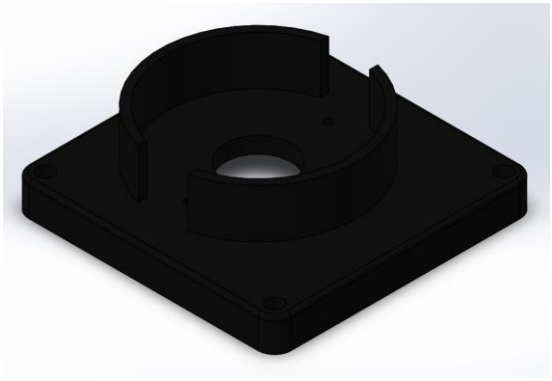


Рисунок 2.50 – Модель пластмасової деталі FPV комплекту 1

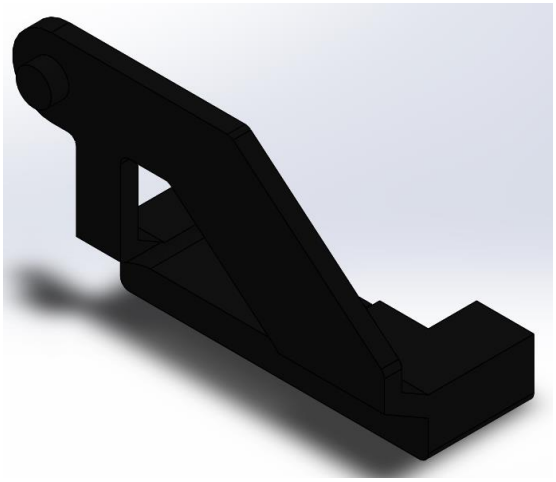


Рисунок 2.51 – Модель пластмасової деталі FPV комплекту 2

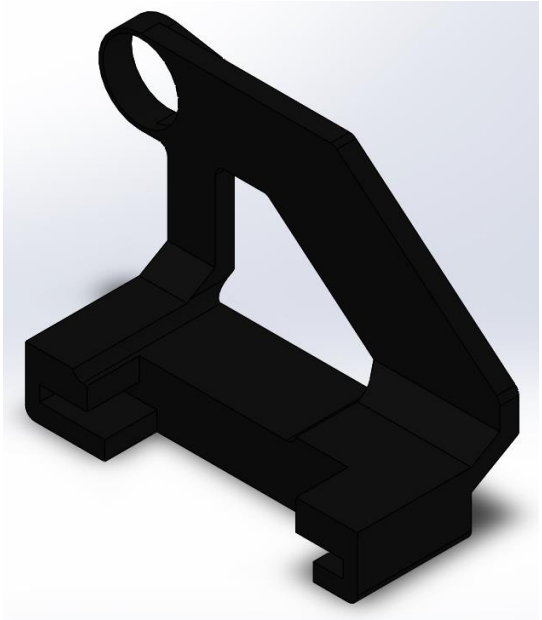


Рисунок 2.52 – Модель пластмасової деталі FPV комплекту 3

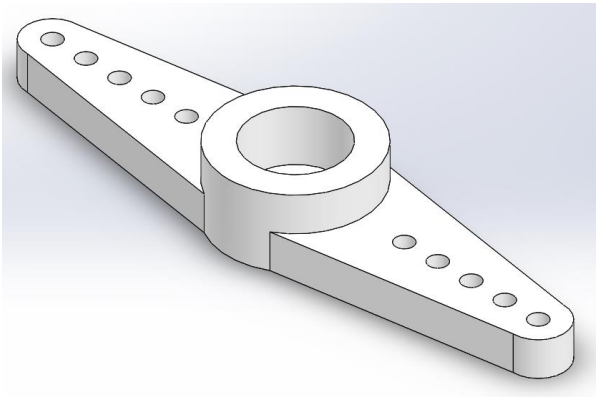


Рисунок 2.53 – Модель пластмасова деталь для передачі руху серводвигуна 2

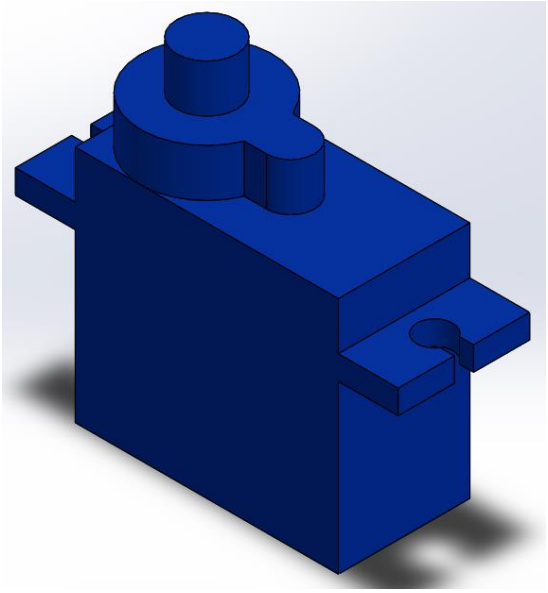


Рисунок 2.54 – Модель серводвигуна SG90

Рисунок 2.55 – Фото колеса з муфтою

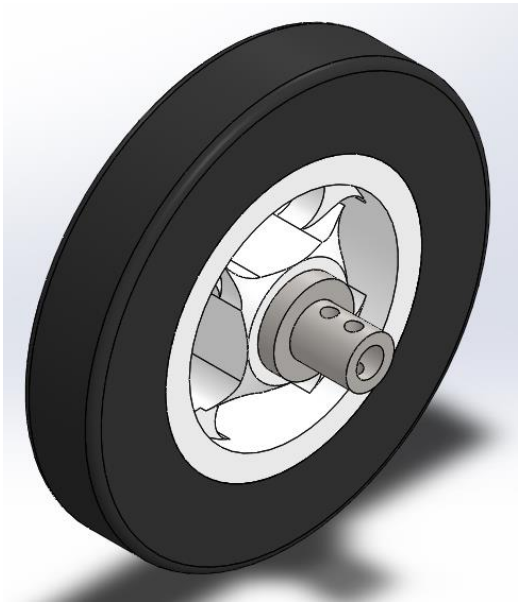


Рисунок 2.56 – Модель колеса з муфтою

Підшипники та кріпильні вироби можна отримати готовими в «Toolbox» ПП SolidWorks (рис 2.58).

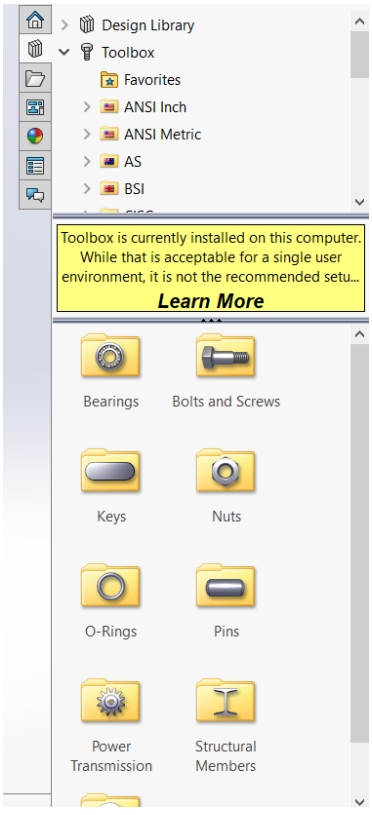


Рисунок 2.58 – «Toolbox» ПП SolidWorks

Готова модель, 3D збірка повноцінного робота, на базі двигунів NEMA 17 MINEBEA 05701607 з захоплювачем для об’єктів взаємодії, тримачами для шести ультразвукових датчиків відстані та пасовою передачею, що дозволить йому змінювати напрямок руху обертаючись навкруг своєї осі зображено на рисунку 2.59.

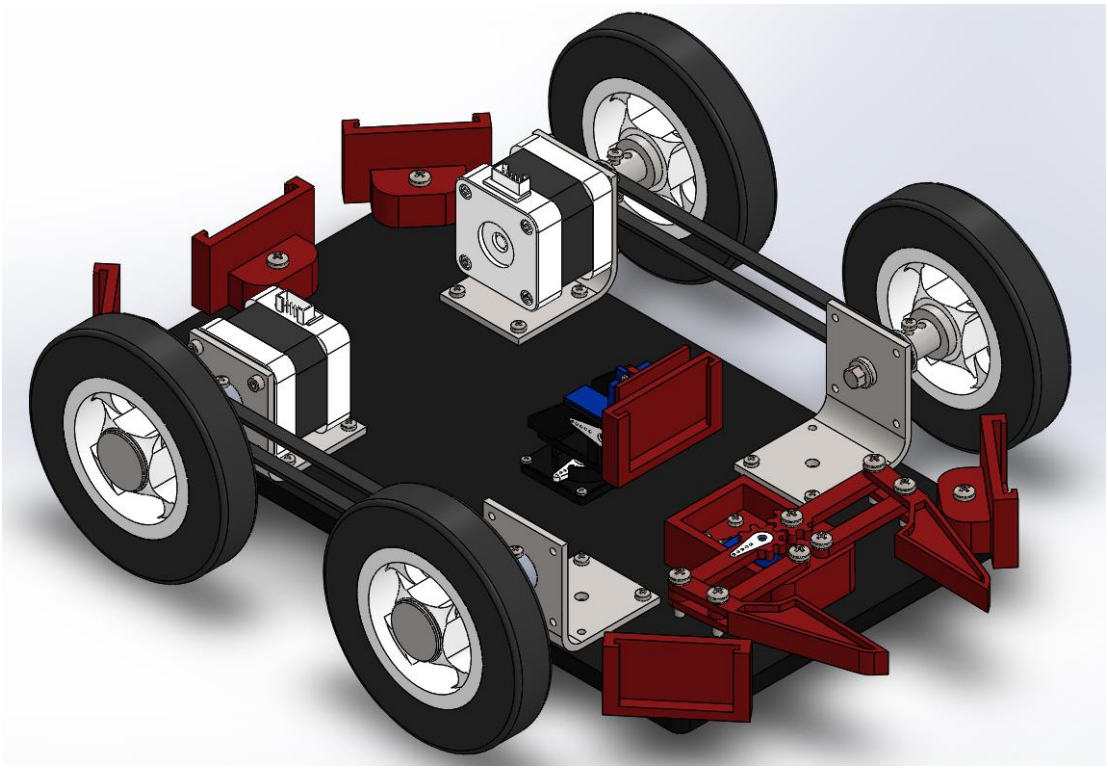


Рисунок 2.59 – 3D збірка повноцінної моделі робота

В майбутньому, дана модель робота може буде вдосконалена, заміною кутників на стандартизовані кронштейни для двигунів типу NEMA 17. А двигуни замінені на модель аналогічного типу, яка буде більш доступна.

Висновки

1. Запропонований спосіб створення прототипу складського робота характеризується простотою реалізації, низькою вартістю компонентів та широкими можливостями завдяки мові Python та її пакетам. Прототип є гнучким і варіативним в налаштуванні. Може застосовуватися для налагодження алгоритмів роботи керуючої програми, ідентифікації об’єктів. У разі застосування більш надійного обладнання, він має перспективи в допомозі проектування повноцінної моделі.
2. Було спроектовано повноцінну модель робота, що обладнана шістьма ультразвуковими датчиками відстані HC-SR04, один з яких є рухомим. А також захоплювачем об’єктів, для складських цілей. Акрилова плита, основи робота передбачає варіативні модифікації в майбутньому. Конструкція шасі на основі двох крокових двигунів та пасових передач, дозволяє реалізувати повороти на місці, що підвищує точність пересування, яка відіграє важливу роль в характеристиках складських роботів.
3. Спроектована модель є розрахованою для використання в навчальних цілях та проведенні експериментів, написання різноманітних керуючих програм, напрацювання яких можуть бути використані в промислових складських роботах.
4. Було складено кошторис, згідно якого виготовлення такого робота коштуватиме близько 9 тис. гривень.
5. Складські роботи є перспективними в майбутньому підприємств та заводів з серійним виробництвом. З їхньою допомогою можливе підвищення продуктивності та покращення дохідності. Виробництва сьогодення напряму залежить від використання промислових роботів, що взаємодіють між собою.

Список використаних джерел

1. Конструирование роботов: Пер. с франц./Андре П., К 65 Кофман Ж. М., Лот Ф., Тайар Ж. П. М.: Мир, 1986. 360 с, ил.
2. ZN.UA, «Азія знову попереду: скільки промислових роботів було встановлено у 2021 році» [електронний ресурс]. https://zn.ua/ukr/TECHNOLOGIES/azija-znovu-poperedu-skilki-promislovikh-robotiv-bulo-vstanovleno-u-2021-rotsi.html
3. WareTeka. [електронний ресурс]. https://wareteka.com.ua/uk/blog/roboti-na-skladah-prikladi-avtomatizaciyi/
4. Robot Grippers. Textbook/G. J. Monkman, S. Hesse, R. Steinmann, H. Schunk]. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim. 2007. 30 с.
5. Robots.com. [електронний ресурс]. https://www.robots.com/articles/grippers-for-robots
6. Бучинський М. Я., Горик О. В., Чернявський А. М., Яхін С. В. Основи творення машин / [За редакцією О. В. Горика, доктора технічних наук, професора, заслуженого працівника народної освіти України]. Харків: Вид-во «НТМТ», 2017. 448 с. : 52 іл.
7. ArduinoUA. [електронний ресурс]. https://doc.arduino.ua/
8. Arduino Robotics. Textbook/[John-David Warren, Josh Adams, Harald Molle]. Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. 2011. 601 с.].
9. [Изучаем Python, 4-е издание. Пер. с англ. СПб.: Символ-Плюс, 2011. 1280 с, ил.].
10. Arduino.cc Docs [електронний ресурс]. https://docs.arduino.cc/hacking/software/FirmataLibrary
11. GitHub, firmata/Arduino [електронний ресурс]. https://github.com/firmata/arduino#firmata-client-libraries

12. Реєстр Python-пакунків (PyPI). [електронний ресурс]. https://pypi.org/project/pyFirmata/#files
13. Wikipedia, OpenCV [електронний ресурс]. https://uk.wikipedia.org/wiki/OpenCV
14. GitHub, opencv/open [електронний ресурс]. https://github.com/opencv/opencv/tree/master/data
15. Wikipedia, Scikit-learn [електронний ресурс]. https://uk.wikipedia.org/wiki/Scikit-learn
16. Python. [електронний ресурс]. https://www.python.org/downloads/release/python-272/
17. Pyzo. [електронний ресурс] https://pyzo.org/
18. SolidWorks.com. [електронний ресурс]. https://www.solidworks.com/domain/design-engineering
19. Compu Phase.com [електронний ресурс]. https://www.compuphase.com/software_termite.htm]
20. ДСТУ 2879-94 Маніпулятори, автооператори, роботи промислові та системи виробничі гнучкі. Терміни та визначення.
21. ГОСТ 30097-93 Роботы промышленные. Системы координат и направления движений.
22. ГОСТ 25204-82 Роботы промышленные. Ряд номинальной грузоподъемности.
23. ГОСТ 8032-84 Предпочтительные числа и ряды предпочтительных чисел.
24. ГОСТ 25685-83 Роботы промышленные. Классификация.
25. Компьютерное зрение на Python R. Первые шаги / Э. Д. Шакирьянов. Электрон. изд. М. : Лаборатория знаний, 2021. 163 с. Ч (Школа юного инженера).

26. Проць Я.І. Захоплювальні пристрої промислових роботів: Навчальний посібник./ Я.І. Проць. Тернопіль: Тернопільський державний технічний університет ім. І. Пулюя, 2008. 232 с.
27. Робототехніка. Підручник / [В. І. Костюк, Г. О. Спину, Л. С. Ямпольський, М. М. Ткач.] К.: Вища школа. 1994. 447 с.
28. Воротников С. А. Информационные устройства робототехнических систем: учебное пособие. М.: Изд-во МГТУ им. Н. Э. Баумана, 2005. 384 с.
29. Промисловий робот [електронний ресурс]. http://uk.wikipedia.org/wiki/Промисловий_робот

Додатки

Додаток А – Керуюча програма робота

Program_V1

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

from pyfirmata import Arduino, util
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
import random

time=util.time
board = Arduino('COM25')
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')

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

def scan():
      angle=0
      X=[]
      Y=[]
      while angle<=130:
            servo1.write(angle)
            time.sleep(1)
            dist=ping(3)
            X.append(angle)
            Y.append(dist)
            print board.analog[0].read()
            angle+=13
      return X,Y

def scan3D():
      H,V,D=[],[],[]
      for v in [110, 60]:
            h=0

            while h<=130:
                  servo2.write(v)
                  time.sleep(0.01)
                  servo1.write(h)
                  time.sleep(1)
                  d=ping(3)
                  H.append(h)
                  V.append(v)
                  D.append(int(d<40 and d!=0))
                  h+=13
                  return H,V,D
```

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

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

def RB(t):
      pin10.write(1)
      pin11.write(0)
      time.sleep(t)
      stop()

def LF(t):
      pin8.write(0)
      pin9.write(1)
      time.sleep(t)
      stop()

def LB(t):
      pin8.write(1)
      pin9.write(0)
      time.sleep(t)
      stop()

def F(t):
      RF(0.1, False)
      LF(t)
      stop()

def B(t):
      RB(0.1)
      LB(t)
      stop()


x=np.array([
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1],
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
])

y=np.array([1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0])

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1)
model=GradientBoostingClassifier(n_estimators=10, learning_rate=0.1,
max_depth=3)
model.fit(x_train, y_train)
print y_test
print model.predict(x_test)
print model.score(x_test, y_test)

from sklearn.model_selection import cross_val_score
s=cross_val_score(model, x, y, cv=9)
print s, s.mean()
print model.predict([[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0]])

      while True:
            H,V,D=scan3D()
            p=model.predict(np.array(D).reshape(1,-1))[0]
            if p:
                  F(5)
      else:
            direction=random.choice([F,LF,RF])
            direction(random.random())

board.exit()
```

Program_V2

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

from pyfirmata import Arduino, util
import numpy as np
import urllib2
import random

time=util.time
board = Arduino('COM25')
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')

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

def scan():
      angle=0
      X=[]
      Y=[]
      while angle<=130:
            servo1.write(angle)
            time.sleep(1)
            dist=ping(3)
            X.append(angle)
            Y.append(dist)
            print board.analog[0].read()
            angle+=13
      return X,Y

def scan3D():
      H,V,D=[],[],[]
      for v in [110, 60]:
            h=0

            while h<=130:
                  servo2.write(v)
                  time.sleep(0.01)
                  servo1.write(h)
                  time.sleep(1)
                  d=ping(3)
                  H.append(h)
                  V.append(v)
                  D.append(int(d<40 and d!=0))
                  h+=13
                  return H,V,D
```

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


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

def RB(t):
      pin10.write(1)
      pin11.write(0)
      time.sleep(t)
      stop()

def LF(t):
      pin8.write(0)
      pin9.write(1)
      time.sleep(t)
      stop()

def LB(t):
      pin8.write(1)
      pin9.write(0)
      time.sleep(t)
      stop()

def F(t):
      RF(0.1, False)
      LF(t)
      stop()

def B(t):
      RB(0.1)
      LB(t)
      stop()

def getXY():
      response = urllib2.urlopen('http://192.168.0.101/')
      data=response.read()
      response.close()
      if not data:
            time.sleep(2)
            return None
      x1,y1,x2,y2=[int(x) for x in data.split()]
      return x1,y1,x2,y2

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

def dist(x1,y1,x2,y2):
      return ((x2 - x1)**2 +(y2 - y1)**2)**0.5
```

```
while True:
      data=getXY()
      if not data: continue
      x1,y1,x2,y2=data
      d1=dist(x1,y1,x2,y2)
      F(1)
      data=getXY()
      if not data: continue
      x1,y1,x2,y2=data
      d2=dist(x1,y1,x2,y2)
      if d2>d1:
            R(1)
      else:
            F(1)
      time.sleep(2)

board.exit()
```

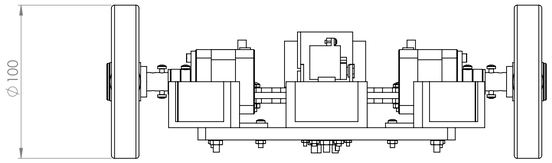
CV2_Detect_Color_Web

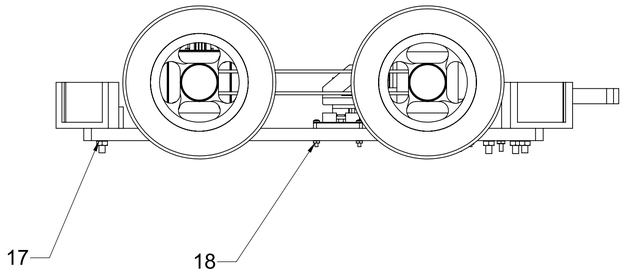
```
# -*- coding: utf-8 -*-
from bottle import route, run, request, WSGIRefServer
import numpy as np
import cv2

def detectColor(h1, s1, v1, h2, s2, v2):
      h_min = np.array((h1, s1, v1), np.uint8)
      h_max = np.array((h2, s2, v2), np.uint8)
      RealTimeMask = cv2.inRange(frame_hsv, h_min, h_max)
      moments = cv2.moments(RealTimeMask, 1)
      dM01 = moments['m01']
      dM10 = moments['m10']
      Area = moments['m00']
      #print Area
      if Area:
            x = int(dM10 / Area)
            y = int(dM01 / Area)
            cv2.circle(frame, (x, y), 10, (0,0,255),-1)
            return x, y, RealTimeMask
      else:
            return None, None, RealTimeMask

cap = cv2.VideoCapture(0)
frame=None
frame_hsv=None

def getXY():
      global frame, frame_hsv
      ret, frame = cap.read()
      frame_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV )
      x1, y1, RealTimeMask=detectColor(159, 39, 103, 180, 255, 255)
      x2, y2, RealTimeMask=detectColor(25, 70, 159, 55, 162, 255)
      if all([x1,y1,x2,y2]): return "%i %i %i %i"%(x1,y1,x2,y2)
      return ""

@route('/') # http://localhost:80
def index():
      return getXY()

run(server=WSGIRefServer, host='192.168.0.101', port=80)
cap.release()
cv2.destroyAllWindows()
```

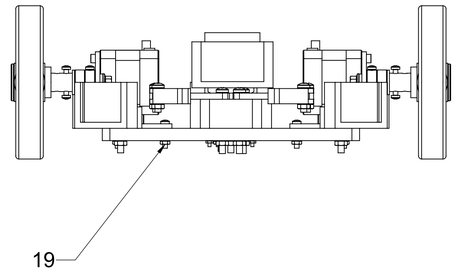
Додаток Б – Кошторис

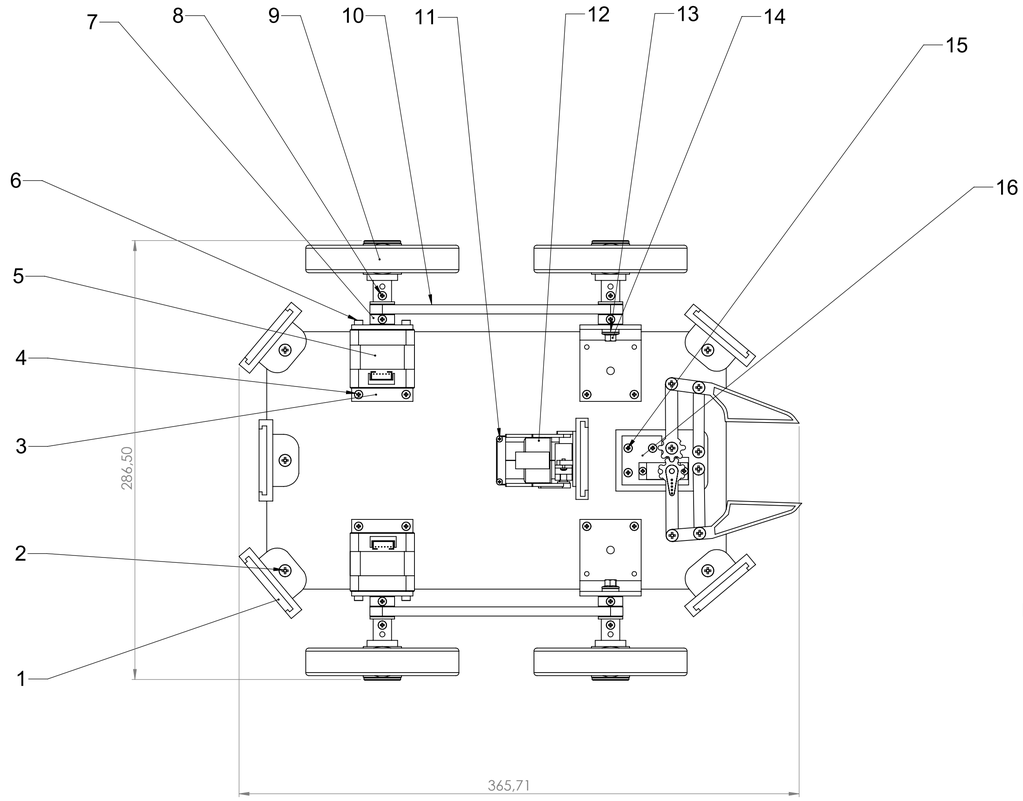
№	Назва	Кількість	Ціна
1	Лист акрилу 500х600мм, 4мм	3	1881
2	Шлейфова перекладка KLS1-SBJW04-FF-300MM-22AWG 40пін, M-M 30см	1	60,50
3	Шлейфова перекладка KLS1-SBJW04-MF-300MM-22AWG 40пін, M-F 30см	1	65
4	Поворотне кріплення для FPV камери сервомотору	1	67
5	Датчик відстані ультразвуковий HC-SR04	6	297
6	Пластик для 3D друку, нитка PETG. Прозоро-червона (1.75 мм/0.5 кг)	1	340
7	Колесо 100мм, з муфтою	4	1220
8	Підшипник 5х10х4мм (105zz)	2	45
9	Алюмінієвий кутник 2028 20 * 28мм	4	95
10	Ремінь замкнутий 280мм (280-2gt-6)	2	76
11	Зубчастий ролик шків ременя GT2 5мм 20-3D зубів принтера (12311)	4	144
12	Крокoвий двигун NEMA 17 17HS4401	2	1050
13	Кронштейн для КД Nema23 (Bracket Nema 23)	4	198
14	Гвинт М3х10	8	36
15	Гвинт М3х20	28	184,80
16	Гвинт М3х30	1	7
17	Гвинт М4х18	8	26,40
18	Гвинт М4х40	2	5
19	Гвинт М5х12	8	24
20	Гвинт М5х40	2	27
21	Гайка М4	10	20
22	Гайка М3	29	26,10
23	Гайка М5	10	20
24	Сервомотор Tower Pro MG90S	4	520
25	Драйвер крокового двигуна DRV8825	2	286

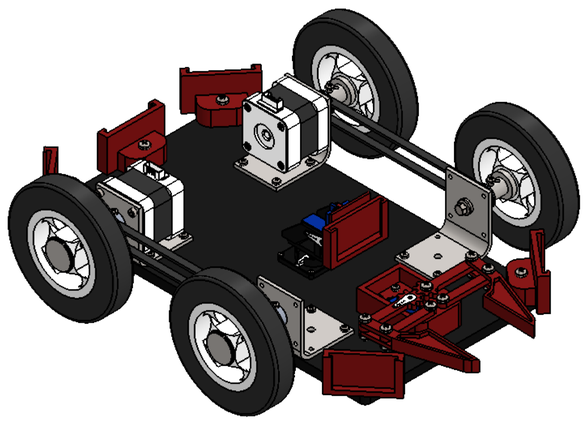
26	Плата розширення портів для Arduino UNO V5 Shield	1	84
27	Мікроконтролер Arduino Uno - R3+Кабель	1	369
28	Bluetooth модуль HC-06 на друкованій платі	1	141
29	Акумулятор LiIon 18650 EVE 3.7V 3500mAh 3C (18815)	6	1200
30	Відсік для батарейок 3*18650	2	140
31	Контролер захисту заряду розряду BMS 6*Li-Ion 12A (12926)	1	321
Всього			8975,80









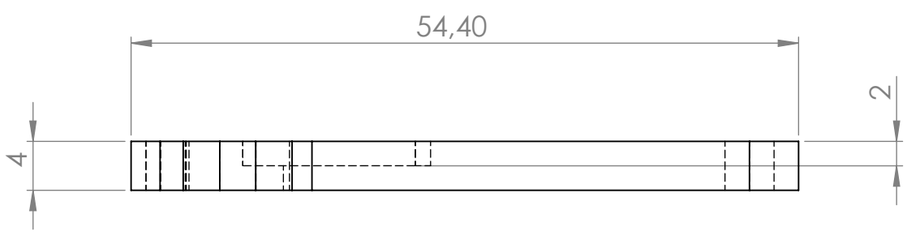


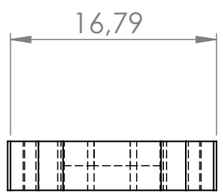
Позиція	Назва	Кількість
1	Тримач для ультразвукового датчика відстані статичнйи	5
2	Гвинт М4х30	5
3	Кутник	4
4	Гвинт М3х16	12
5	Двигун NEMA 17 **MINEBEA 05701607**	2
6	Гвинт М3х8	8
7	Зубчастий ролик, шків ремня GT2 5мм 20-3D	4
8	Гвинт М3х6	16
9	Колесо	4
10	Ремінь замкнутий 280мм (280-2gt-6)	2
11	Гвинт М2х16	4
12	Механізм рухомого тримача ультразвукового датчика відстані в зборі	1
13	Підшипник 5х10х4мм (105zz)	2
14	Гвинт М5х35х16	2
15	Гвинт М3х20	3
16	Захоплювач в зборі	1
17	Гайка М4	5
18	Гайка М2	4
19	Гайка М3	15

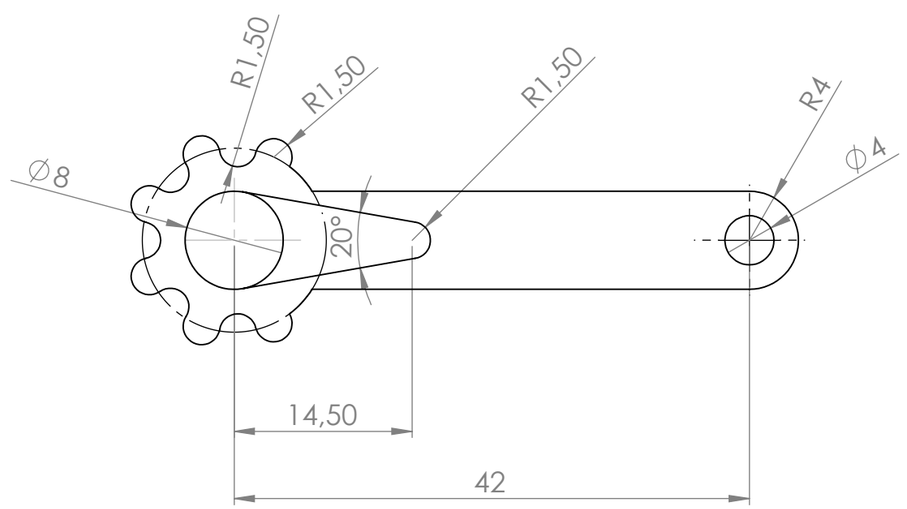
					МР.ПМКм-40.00.01.000 СК			
Зм.	Арк.	№ докум.	Підпис	Дата	Складальне креслення робота	Літ.	Маса	Масштаб
Розроб		Пронюк І.В.				Н		1:2
Перев.		Копей В.Б.						
Т. контр.						Аркуш 1	Аркушів 13	
Н. контр.						ІФНТУНГ ПМКм-21-1		
Затв.								

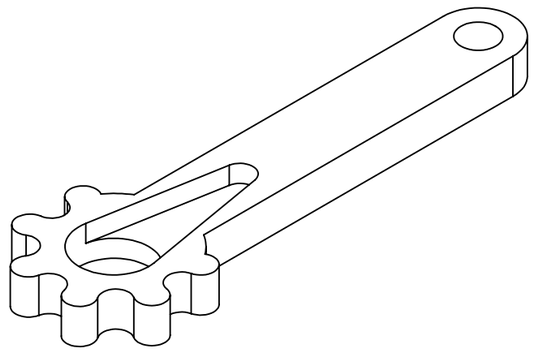
Формат А1

МР.ПМКм-40.00.02.004









					МР.ПМКм-40.00.02.004		
					Правий шестерний з'єднувач	Літ.	Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата		Н	2:1
Розроб		Пронюк І.В.				Аркуш 10	Аркушів 13
Перев.		Копей В.Б.					
Т. контр.					PETG пластик	ІФНТУНГ ПМКм-21-1	
Н. контр.							
Затв.							

Формат А3

МР.ПМКм-40.00.02.005

					МР.ПМКм-40.00.02.005		
Зм.	Арк.	№ докум.	Підпис	Дата	З'єднувач	Літ.	Маса / Масштаб
Розроб		Пронюк І.В.				Н	1:1
Перев.		Копей В.Б.				Аркуш 11	Аркушів 13
Т. контр.							
Н. контр.					PETG пластик	ІФНТУНГ ПМКм-21-1	
Затв.							

Формат А4

Керуюча програма робота "Program_V1"

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

from pyfirmata import Arduino, util
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
import random
```

```
time=util.time
board = Arduino('COM25')
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')
```

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

def scan():
    angle=0
    X=[]
    Y=[]
    while angle<=130:
        servo1.write(angle)
        time.sleep(1)
        dist=ping(3)
        X.append(angle)
        Y.append(dist)
        print board.analog[0].read()
        angle+=13
    return X,Y

def scan3D():
    H,V,D=[],[],[]
    for v in [110, 60]:
        h=0
        while h<=130:
            servo2.write(v)
            time.sleep(0.01)
            servo1.write(h)
            time.sleep(1)
            d=ping(3)
            H.append(h)
            V.append(v)
            D.append(int(d<40 and d!=0))
            h+=13
    return H,V,D
```

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

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

def RB(t):
    pin10.write(1)
    pin11.write(0)
    time.sleep(t)
    stop()

def LF(t):
    pin8.write(0)
    pin9.write(1)
    time.sleep(t)
    stop()

def LB(t):
    pin8.write(1)
    pin9.write(0)
    time.sleep(t)
    stop()

def F(t):
    RF(0.1, False)
    LF(t)
    stop()

def B(t):
    RB(0.1)
    LB(t)
    stop()
```

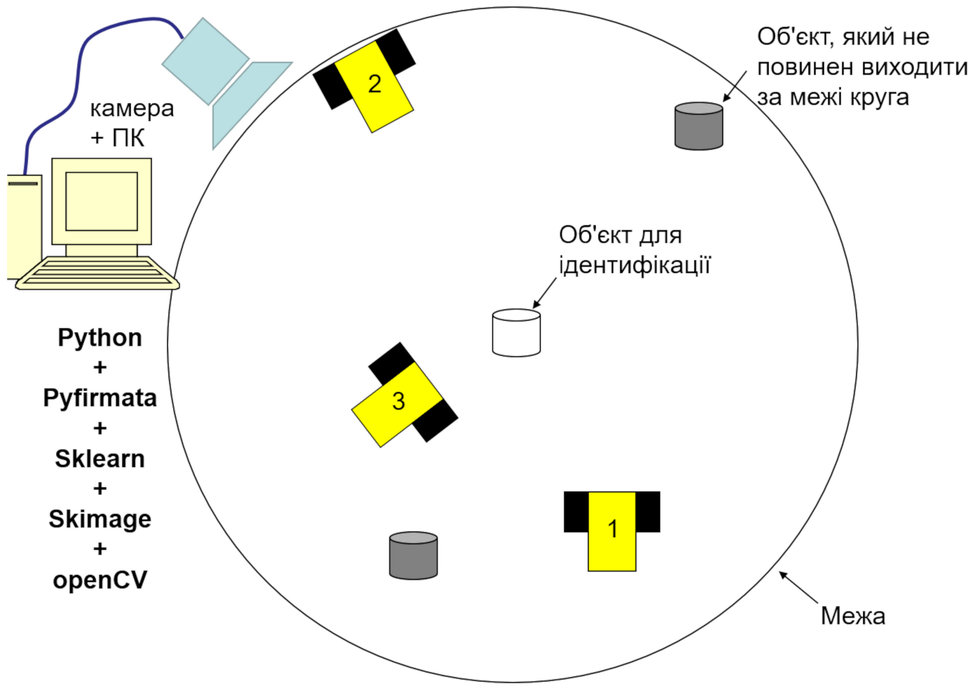
```
x=np.array([
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,  1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,  0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,  0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,  0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,  0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,  0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,  0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,  0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,  0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
])

y=np.array([1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0])
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1)
model=GradientBoostingClassifier(n_estimators=10, learning_rate=0.1, max_depth=3)
model.fit(x_train, y_train)
print y_test
print model.predict(x_test)
print model.score(x_test, y_test)

from sklearn.model_selection import cross_val_score
s=cross_val_score(model, x, y, cv=9)
print s, s.mean()
print model.predict([[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

```
while True:
    H,V,D=scan3D()
    p=model.predict(np.array(D).reshape(1,-1))[0]
    if p:
        F(5)
    else:
        direction=random.choice([F,LF,RF])
        direction(random.random())

board.exit()
```

						МР.ПМКм-40.00.04.000	
Зм.	Арк.	№ докум.	Підпис	Дата	Керуюча програма робота	Літ.	Маса / Масштаб
Розроб.		Пронюк І.В.				Н	
Перев.		Копей В.Б.				Аркуш 12	Аркушів 13
Т.контр.						ІФНТУНГ ПМКм-21-1	
Н.контр.							
Затв.							

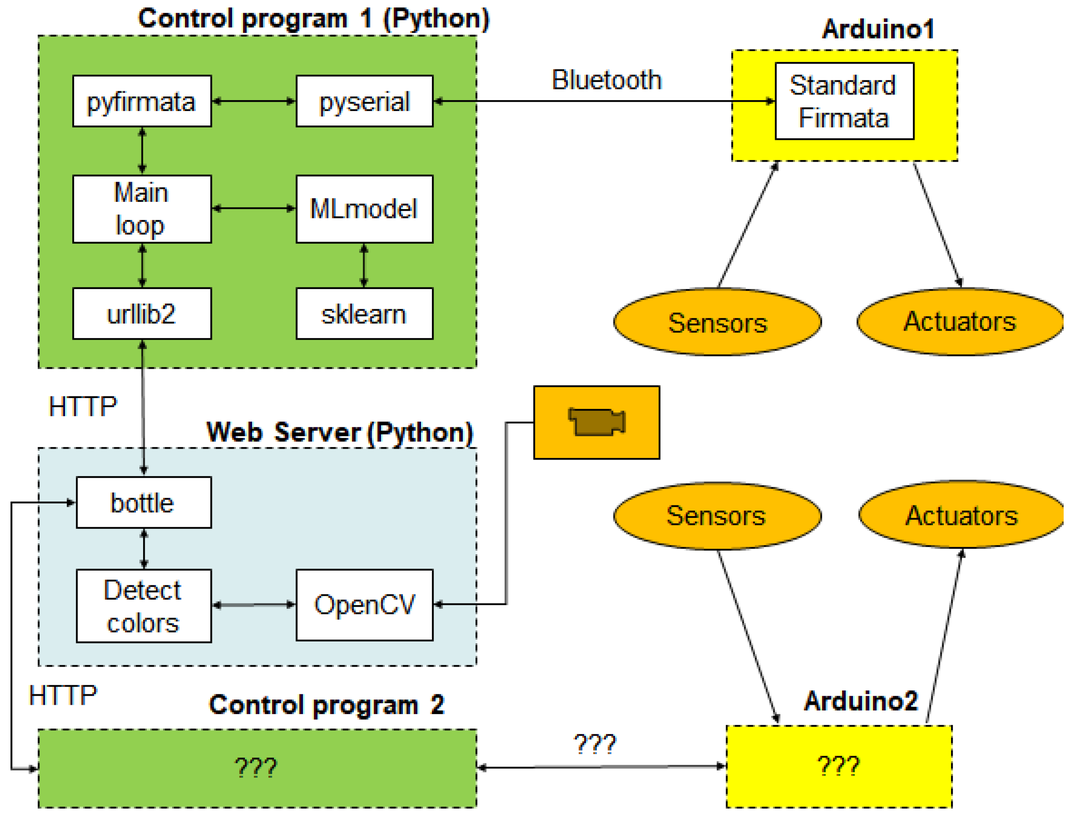
Змагання роботів - платформа для тестування



Рахунок:

Робот1 - **0**
Робот2 - **0**
Робот3 - **0**

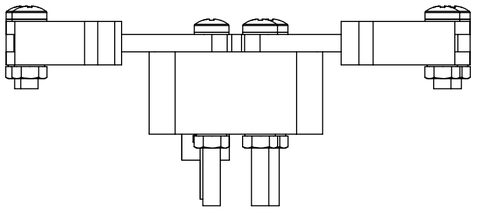
Схема платформи для тестування роботів

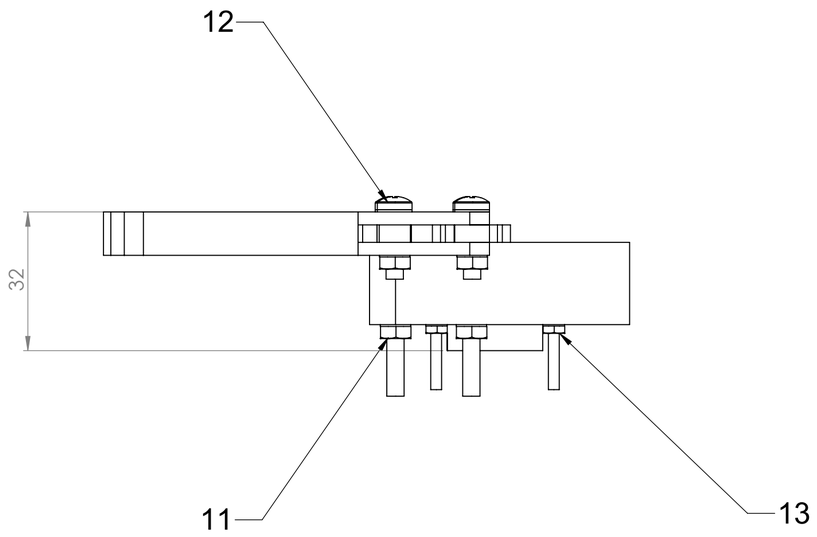


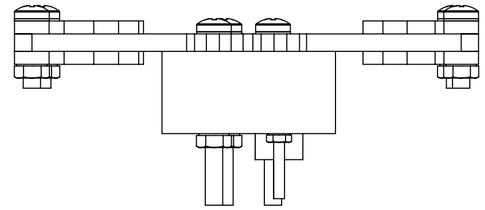
Рівні складності:
1. Програми керування роботами отримують від веб сервера координати усіх об'єктів. Ці координати веб сервер (на основі мікро-вебфреймворку bottle) обчислює за допомогою алгоритмів машинного зору (наприклад OpenCV-функції inRange).
2. Програми отримують від веб сервера тільки координати усіх роботів.
3. Не передбачає передачі будь-яких даних веб сервером.
На рівнях 2 та 3 програмам керування слід використовувати власні сенсори і більш складні алгоритми для ідентифікації об'єкта.

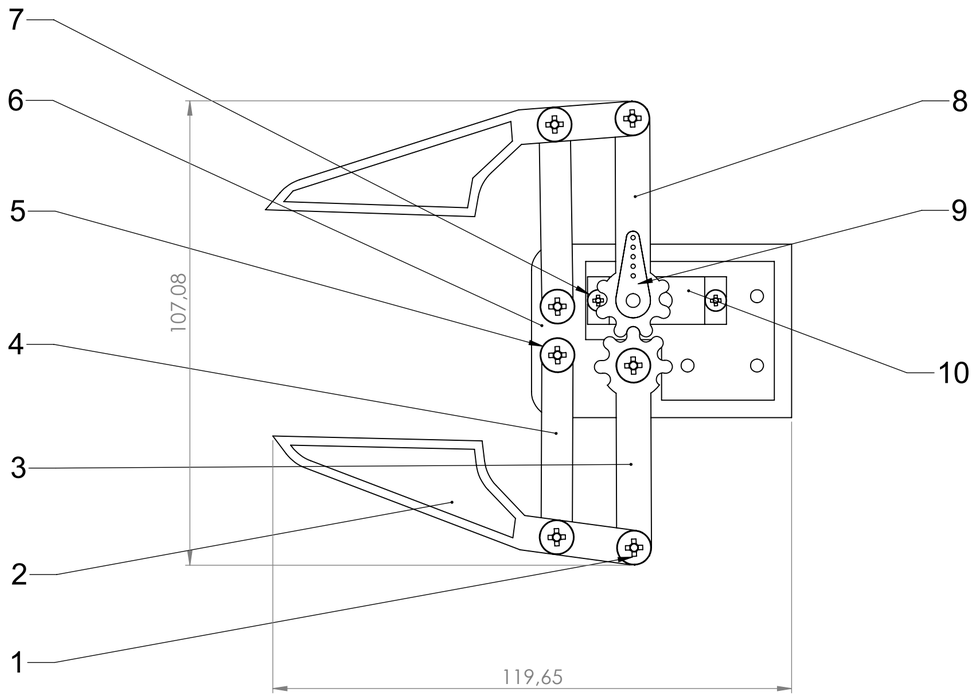
					МР.ПМКм-40.00.05.000	
Зм.	Арк.	№ докум.	Підпис	Дата	Платформа тестування роботів	Літ. / Маса / Масштаб
Розроб.		Пронюк І.В.				Н
Перев.		Копей В.Б.				Аркуш 13 / Аркушів 13
Т.контр.						ІФНТУНГ ПМКм-21-1
Н.контр.						
Затв.						

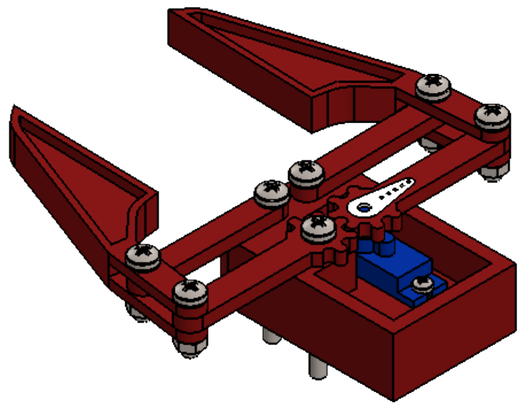
МР.ПМКм-40.00.02.000 СК







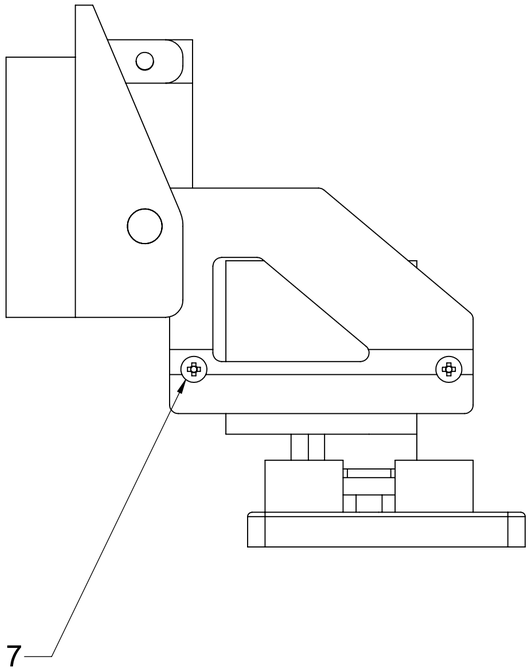


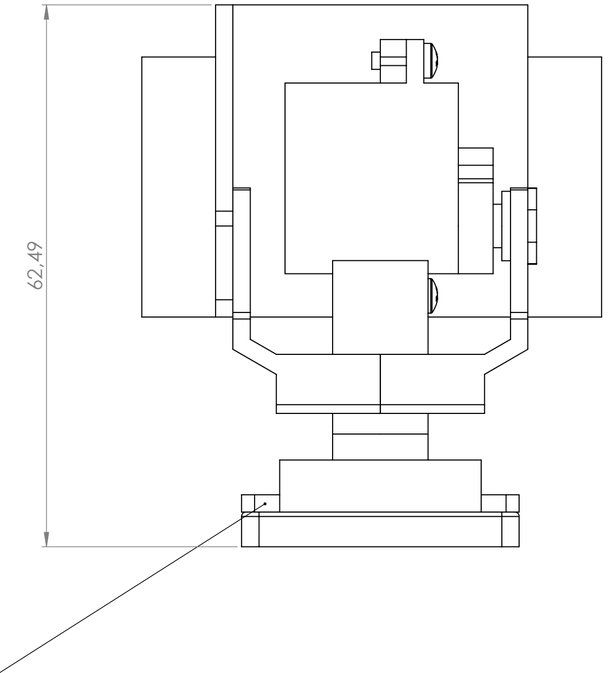


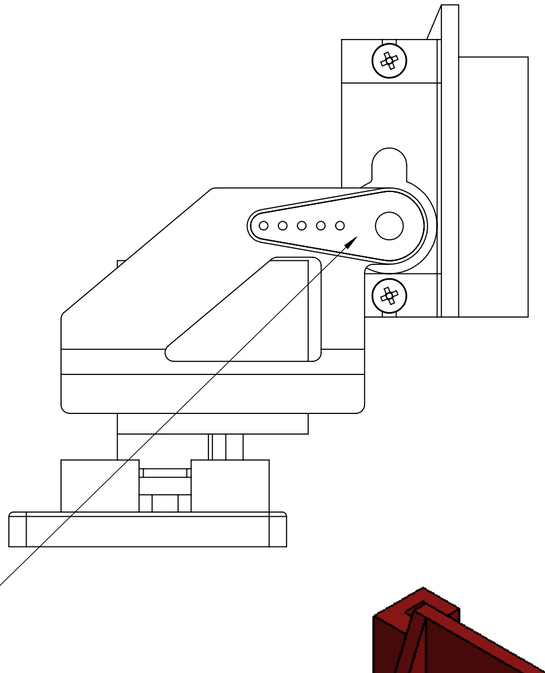
Позиція	Назва	Кількість
1	Гвинт М4х16	4
2	"Клешня"	2
3	Лівий шестерний з'єднувач	1
4	З'єднувач	2
5	Гвинт М4х40	3
6	Основа захоплювача	1
7	Гвинт 2.5х25	2
8	Правий шестерний з'єднувач	1
9	Пластмасова деталь передачі руху (мала)	1
10	Скрводвигун SG90	1
11	Гайка М4	7
12	Шайба М4	7
13	Гайка М2.5	2

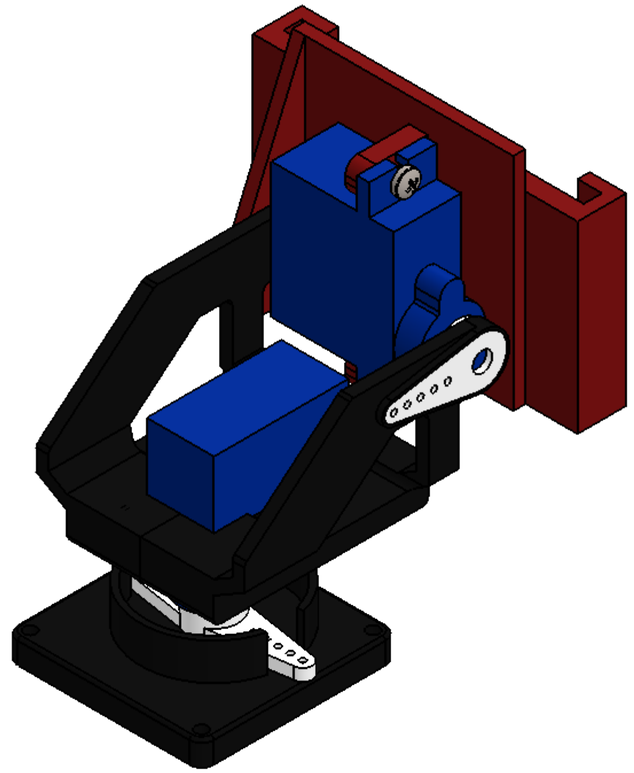
					МР.ПМКм-40.00.02.000 СК	
Зм.	Арк.	№ докум.	Підпис	Дата	Складальне креслення захоплювача	Літ. / Маса / Масштаб
Розроб		Пронюк І.В.				Н / 1:1
Перев.		Копей В.Б.				Аркуш 2 / Аркушів 13
Т. контр.						
Н. контр.						ІФНТУНГ ПМКм-21-1
Затв.						

Формат А2

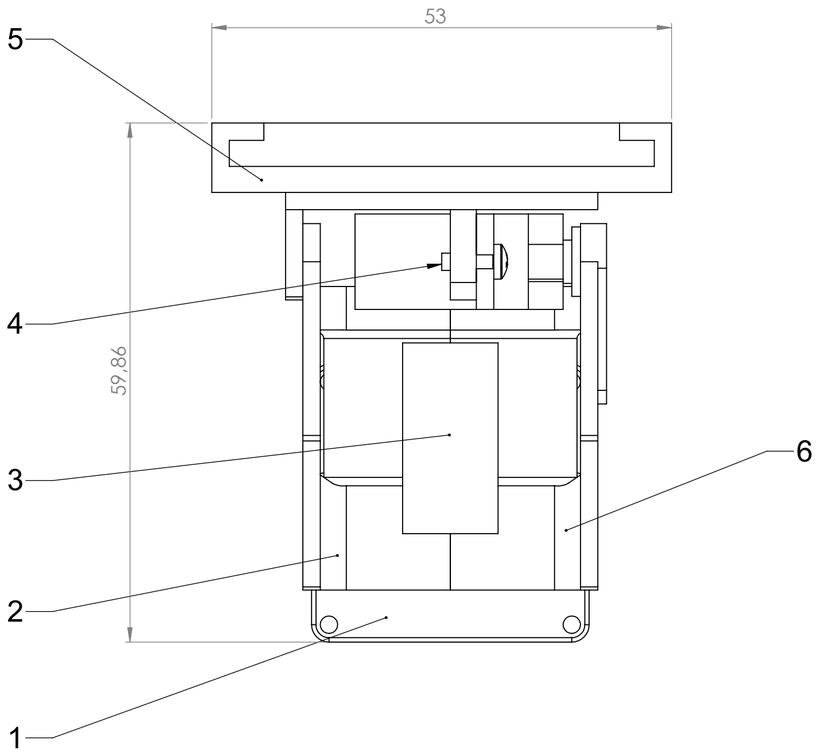








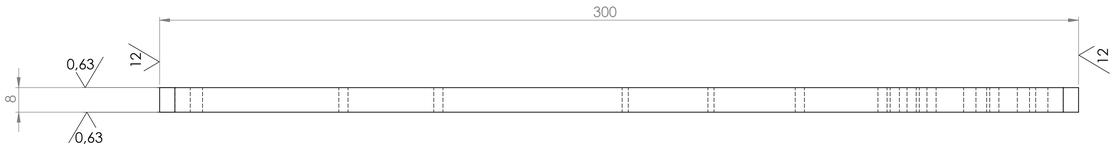
Позиція	Назва	Кількість
1	Пластмасова деталь механізму рухомого тримача ультразвукового датчика відстані 1	1
2	Пластмасова деталь механізму рухомого тримача ультразвукового датчика відстані 2	1
3	Серводвигун SG90	2
4	Гвинт М2х6	1
5	Пластмасова деталь механізму рухомого тримача ультразвукового датчика відстані 4	1
6	Пластмасова деталь механізму рухомого тримача ультразвукового датчика відстані 3	1
7	Гвинт 1.6х8	2
8	Пластмасова деталь для передачі руху (велика)	1
9	Пластмасова деталь для передачі руху (мала)	1

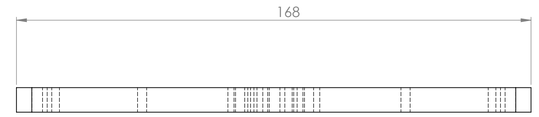


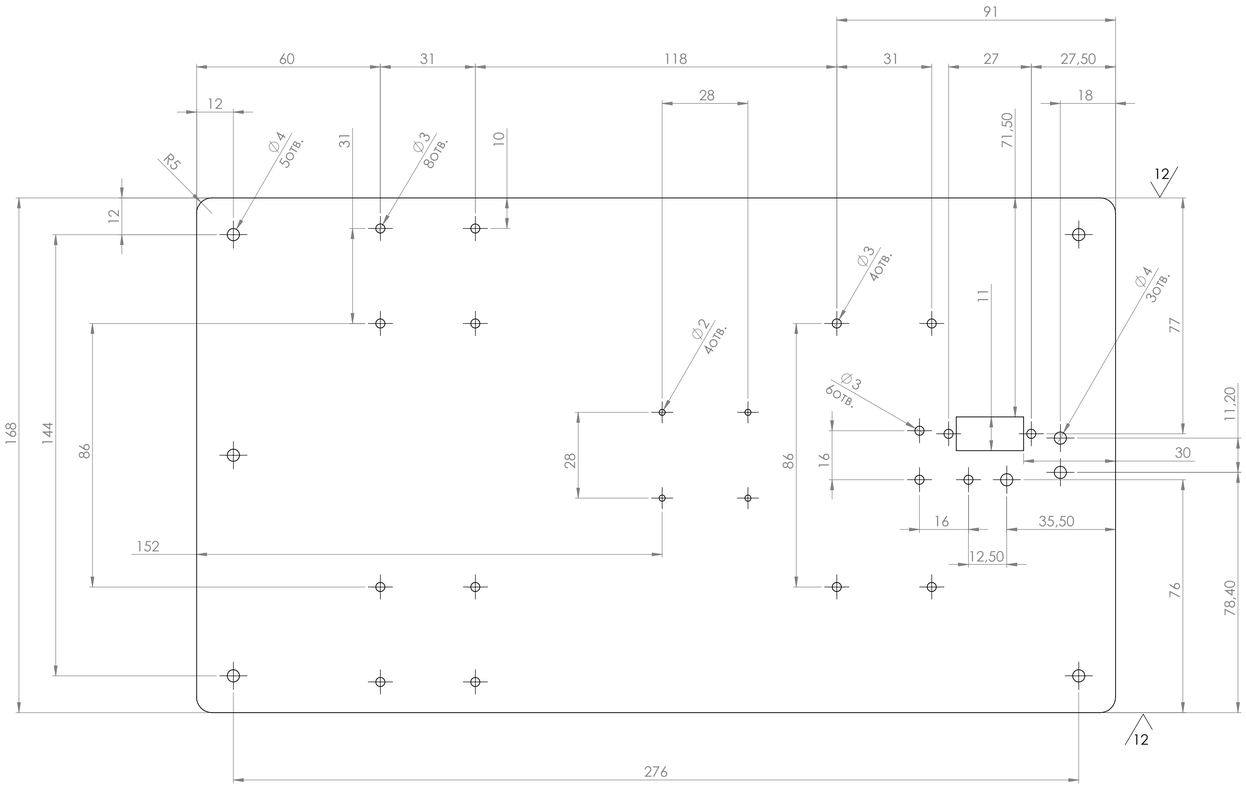
					МР.ПМКм-40.00.03.000 СК
Зм.	Арк.	№ докум.	Підпис	Дата	Складальне креслення механізму рухомого тримача ультразвукового датчика відстані HC-SR 04
Розроб		Пронюк І.В.			Літ. Н; Маса; Масштаб 2:1
Перев.		Копей В.Б.			Аркуш 3 Аркушів 13
Т. контр.					
Н. контр.					ІФНТУНГ ПМКм-21-1
Затв.					

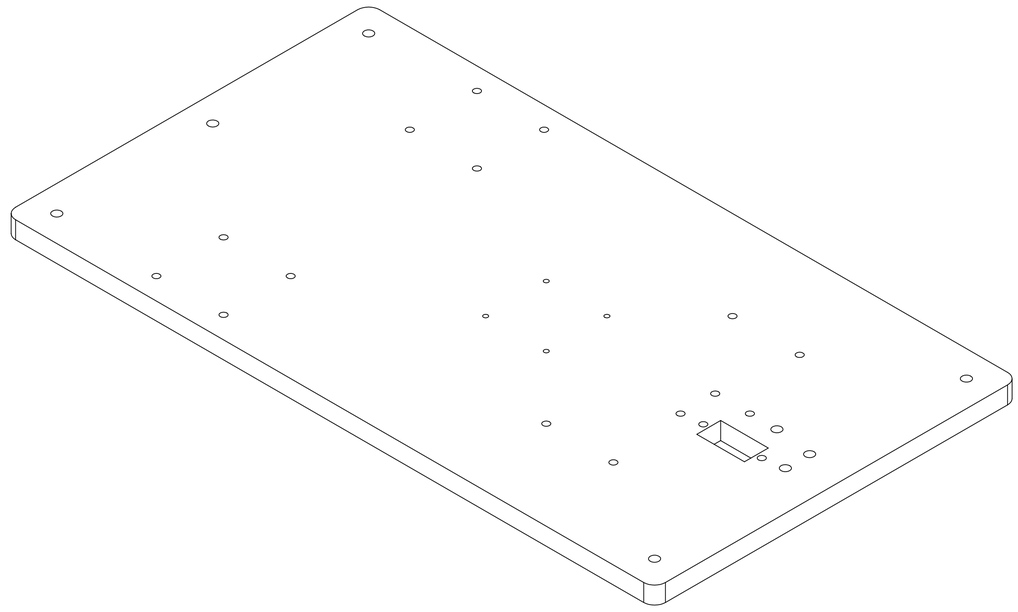
Формат А2

6,3



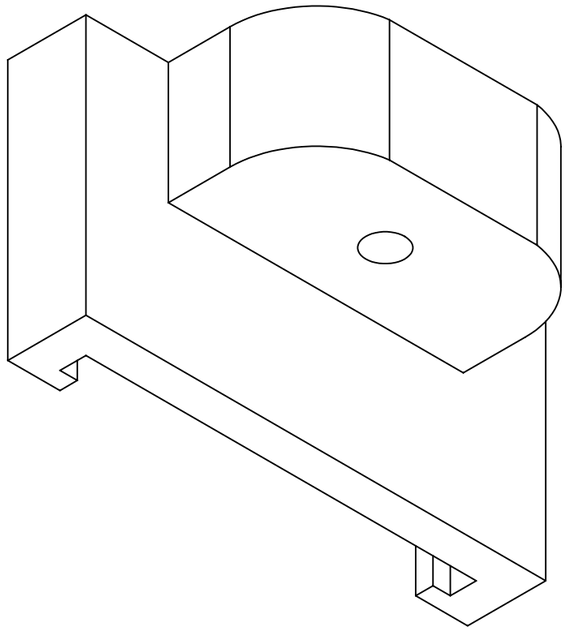


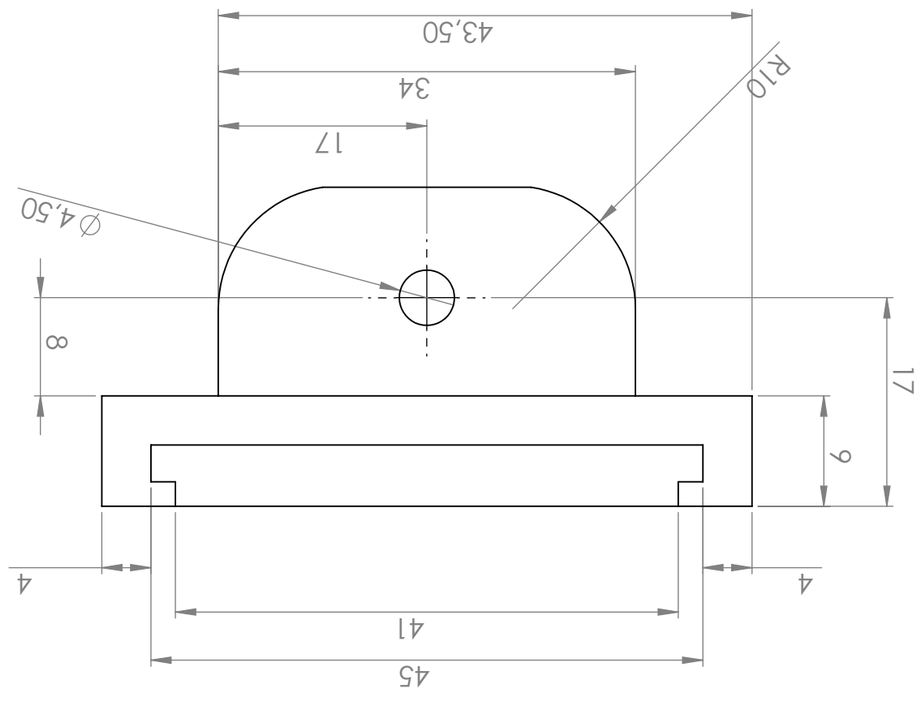


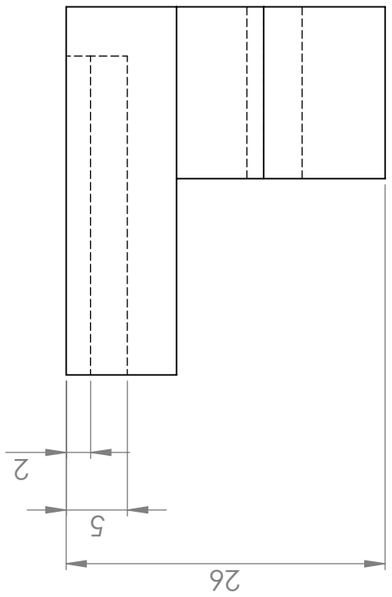


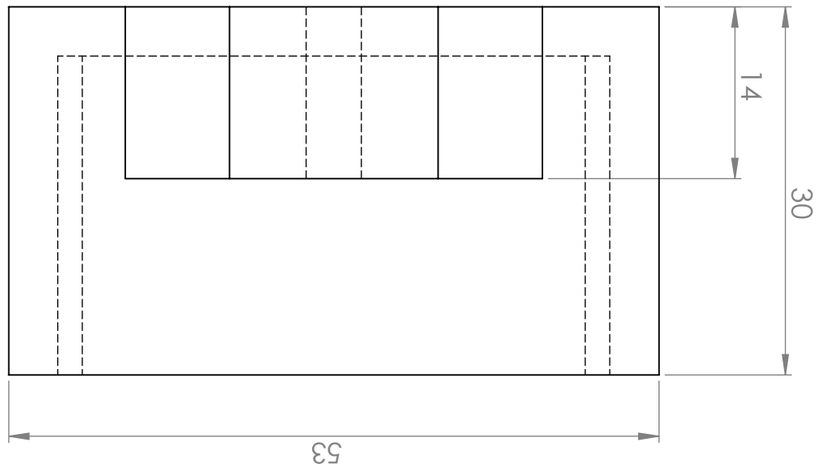
					МР.ПМКм-40.00.01.001	
					Основа	Літ. Маса Масштаб: Н, 1:1
Зм.	Арк.	№ докум.	Підпис	Дата		Аркуш 4 Аркушів 13
Розроб		Пронюк І.В.				
Перев.		Копей В.Б.				
Т. контр.						
Н. контр.					Акрил лист	ІФНТУНГ ПМКм-21-1
Затв.						

Формат А1







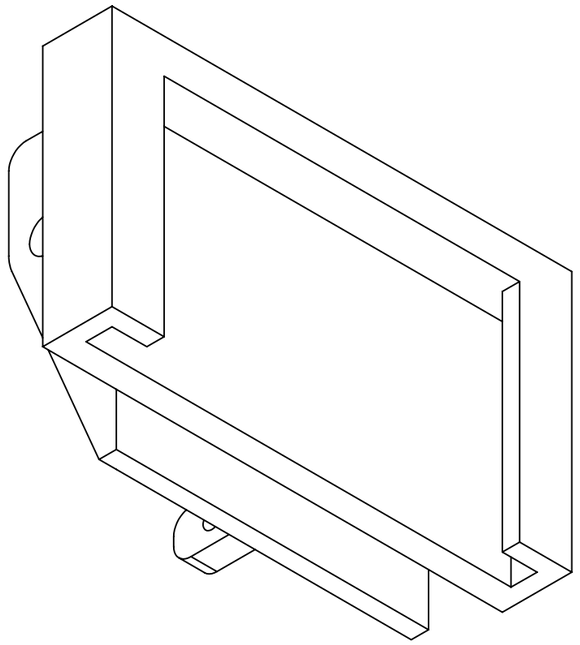


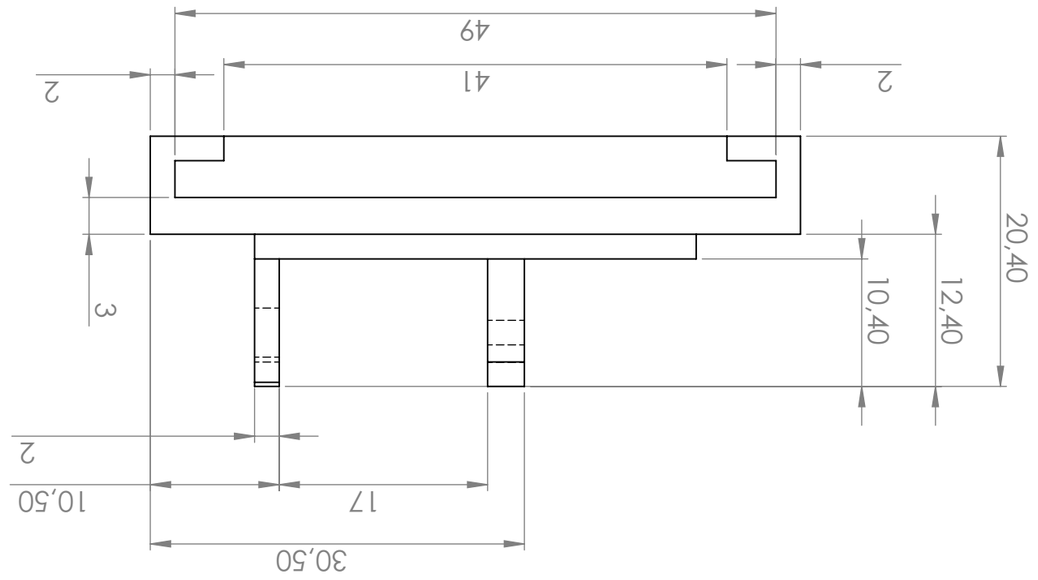
Зм.	Арк.	№ докум.	Підпис	Дата	Тримач ультразвукового датчика відстані НС-SR 04 нерухомий				
Розроб		Пронюк І.В.				Літ.	Маса	Масштаб	
Перев.		Конєй В.Б.				Н		2:1	
Т. контр.						Аркуш 5	Аркушів 13		
Н. контр.					РЕТG пластик	ІФНТУНГ ПМКм-21-1			
Затв.									

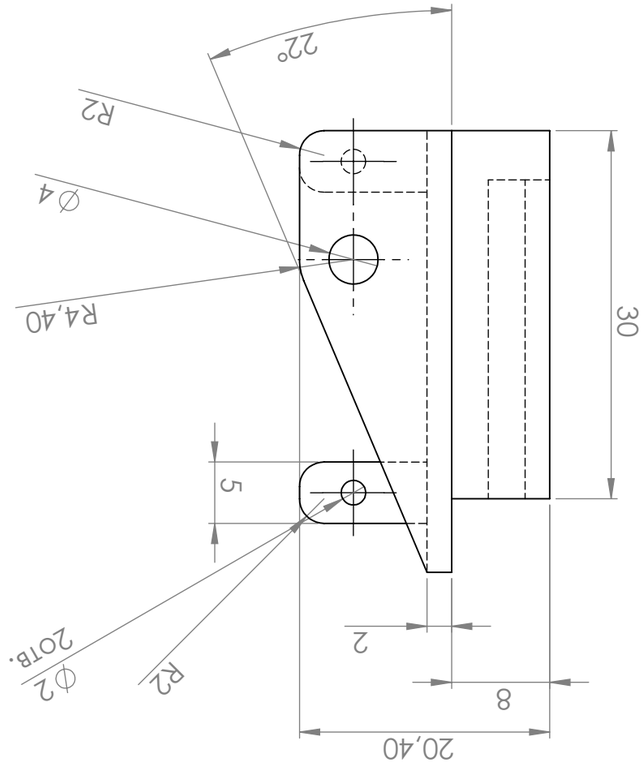
МР.ПМКм-40.00.03.001

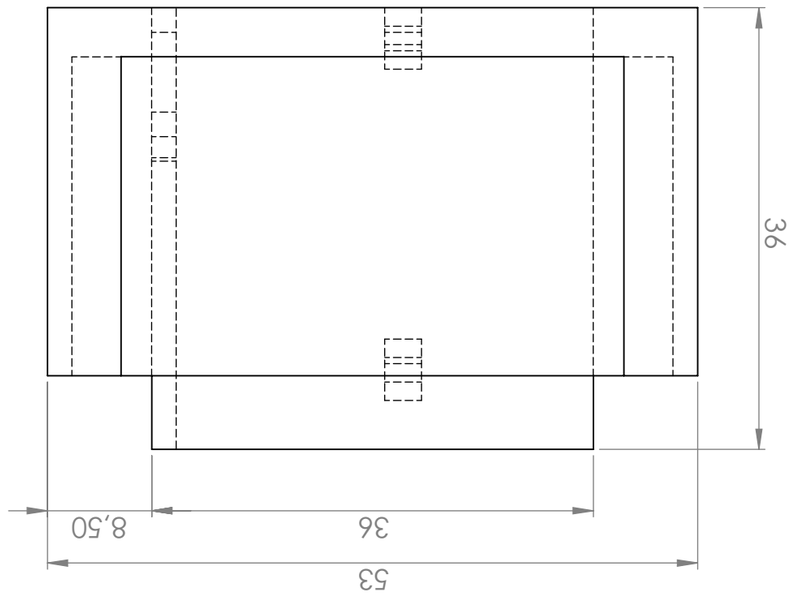
МР.ПМКм-40.00.03.001

Формат А3



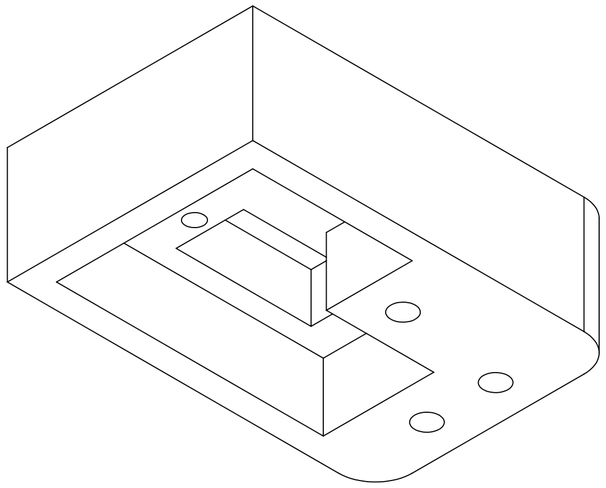


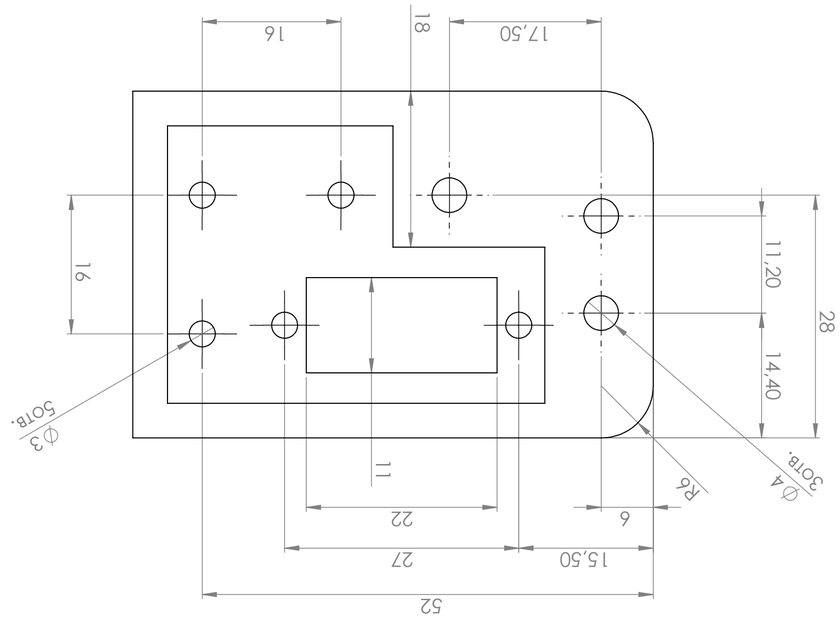


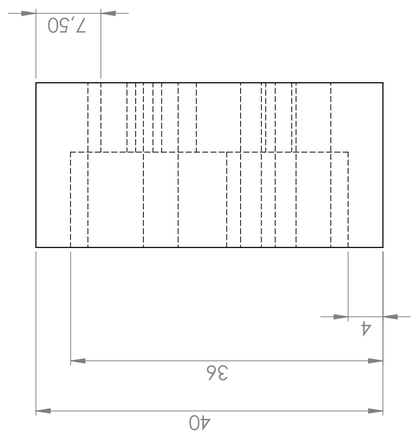


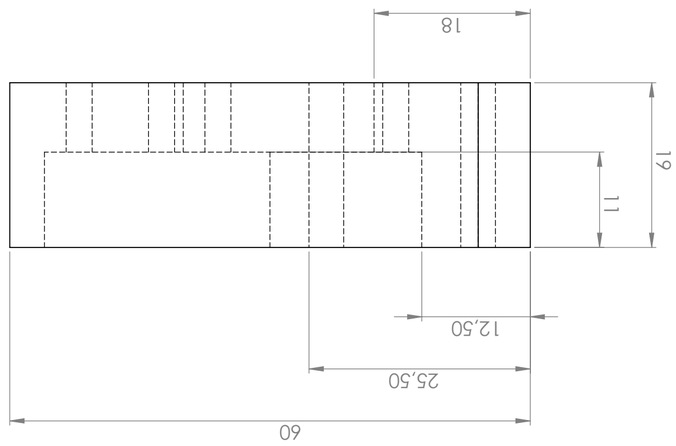
Зм.	Арк.	№ докум.	Підпис	Дата	МР.ПМКм-40.00.03.002			
Розроб		Пронюк І.В.			Тримач універсального датчика відстані HC-SR 04 рухомий	Літ.	Маса	Масштаб
Перев.		Конєв В.Б.				Н		2:1
Т. контр.						Аркуш 6	Аркушів 13	
Н. контр.					РЕТG пластик	ІФНТУНГ ПМКм-21-1		
Затв.								

Формат А3







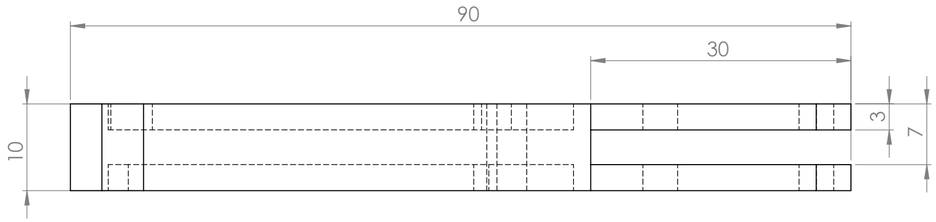


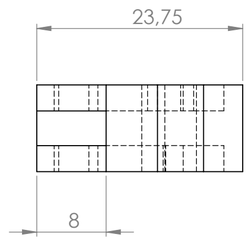
Зм.	Арк.	№ докум.	Підпис	Дата	MP.ПМКм-40.00.02.001	
Розроб		Пронік І.В.				Літ. Маса Масштаб
Перев.		Конєв В.Б.			Основа захоплювача	H 2:1
Т. контр.						Аркуш 7 Аркушів 13
Н. контр.						
Затв.						

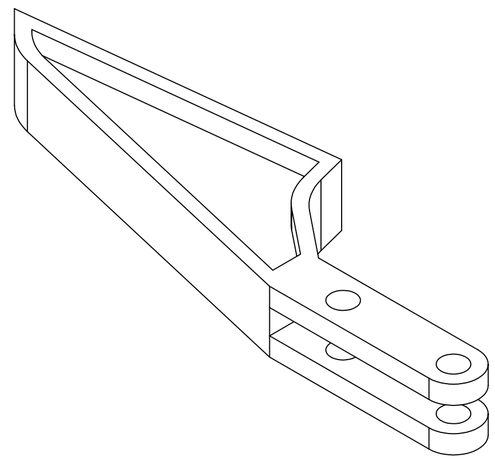
PETG пластик	ІФНТУНГ ПМКм-21-1

Формат А2

МР.ПМКм-40.00.02.002

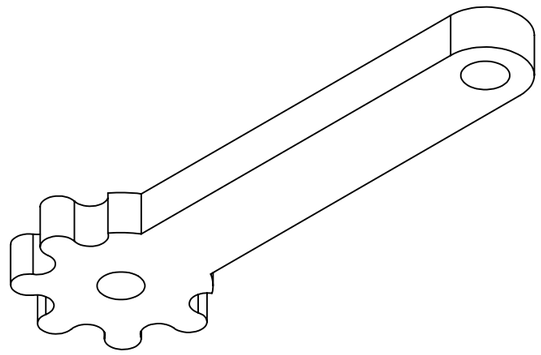


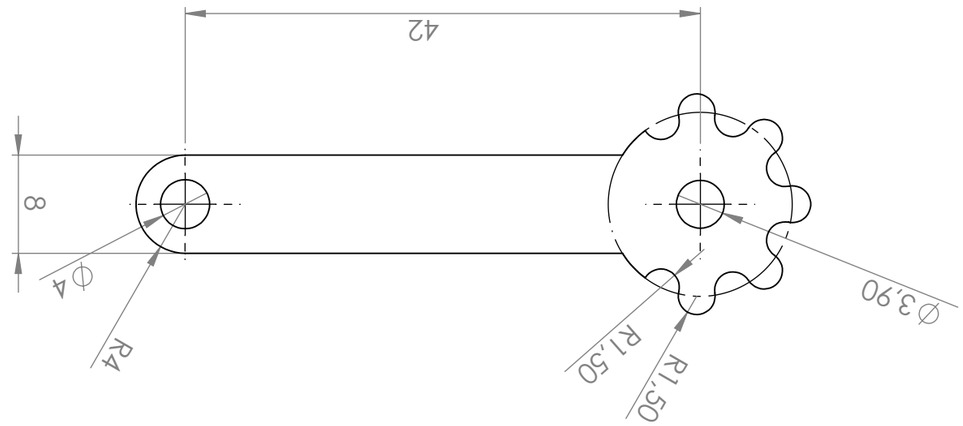


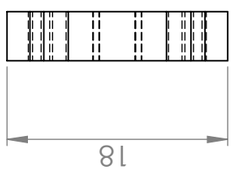


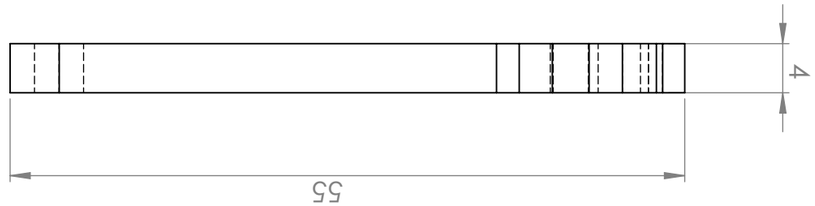
					МР.ПМКм-40.00.02.002			
Зм.	Арк.	№ докум.	Підпис	Дата	"Клешня"	Літ.	Маса	Масштаб
Розроб		Пронюк І.В.				Н		2:1
Перев.		Копей В.Б.				Аркуш 8	Аркушів 13	
Т. контр.								
Н. контр.					PETG пластик	ІФНТУНГ ПМКм-21-1		
Затв.								

Формат А2









Зм.	Арк.	№ докум.	Підпис	Дата	МР.ПМКм-40.00.02.003				
Розроб.		Пронюк І.В.				Літ.	Маса	Масштаб	
Перев.		Конєв В.Б.			Длівий шестерний з'єднувач			2:1	
Т. контр.							Н		
						Аркуш 9		Аркушів 13	
Н. контр.					РЕТG пластик	ІФНТУНГ ПМКм-21-1			
Затв.									

Формат А3

МР.ПМКм-40.00.02.003