БАКАЛАВРСЬКА РОБОТА

БР.ПМІ-97.00.00.000.ПЗ

Група ПМІ-20-1К

Ярема Марк

Андрійович

2022

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

Ярема Марк Андрійович
(прізвище, ім'я, по батькові)

УДК 62-52
(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка антропоморфного робота Scara
(назва роботи)

Інженерія мехатронних систем
(назва освітньої програми)

131 – Прикладна механіка
(шифр і назва спеціальності)

М.А.Ярема
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Врюкало В.В., доцент кафедри КМВ
(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 бакалавр

Спеціальність 131 – Прикладна механіка

(шифр і назва)

ЗАТВЕРДИВ
Завідувач кафедри _____

_____

«____» _____2022 року

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

Яремі Марку Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка антропоморфного робота Scara

керівник роботи Врюкало В.В., доцент кафедри КМВ

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

затверджені наказом закладу вищої освіти від "18" березня 2022 року № 137/7

2.Строки подання студентом роботи 17 червня 2022р.

3. Вихідні дані до роботи: Література з питань проектування роботів, характеристики крокових двигунів і драйверів, технічні вимоги до робота

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

1. Огляд та аналіз сучасних промислових роботів та маніпуляторів. 2. Оглад приводів та систем керування. 3. Аналіз програм керування роботами-маніпуляторами. 4. Розробка навчального проекту антропоморфного робота Scara.

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

1. Конструктивні схеми промислових маніпуляторів – 1 лист А1.

2. Конструкція та електронна схема робота Scara – 1 лист А1.

3. Пряма та обернена кінематика робота – 1 лист А1.

4. Графічний інтерфейс користувача – 1 лист А1.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
	Врюкало В.В., доцент кафедри КМВ	10.03.2022	10.03.2022

7. Дата видачі
завдання 10.03.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Термін виконання етапів роботи	Примітки
1	Аналіз літературних джерел	30.04.2022	
2	Проектування робота	31.05.2022	
3	Оформлення пояснювальної записки та графічної частини	17.06.2022	

Студент Ярема М.А.
(підпис) (прізвище та ініціали)

Керівник роботи Врюкало В.В.
(підпис) (прізвище та ініціали)

“ 10 ” 03 2022 р.

Реферат

Бакалаврська кваліфікаційна робота виконана на тему “Розробка антропоморфного робота Scara”.

Робота складається з 53 аркушів. До неї входять 26 рисунків, 3 додатки та 4 листа графічного матеріалу формату А1.

Об'єкт дослідження – робот SCARA.

Мета роботи – розробка антропоморфного робота Scara, вибір матеріалів для виготовлення робота, створення керуючої програми для виконання типових задач.

Відповідно до поставленої задачі виконано:

1) розробку 3D моделі антропоморфного робота SCARA;

2) підбір матеріалів та компонентів для розробки фізичної моделі

3) створення керуючої програми для виконання типових задач.

Ключові слова: антропоморфний робот, деталь, керуюча програма, технологічна документація.

Студент Ярема М.А.

Summary

Bachelor's thesis was performed on the topic "Development of anthropomorphic robot Scara".

The work consists of 59 sheets. It includes 26 drawings, 3 appendices 4 sheets of graphic material of A1 format.

The object of study - the robot SCARA.

The purpose of the work is to develop the anthropomorphic robot Scara, its 3D model and control program.

The purpose of the work - the development of anthropomorphic robot Scara, the choice of materials for the manufacture of the robot, the creation of a control program to perform typical tasks.

In accordance with the task performed:

1) development of a 3D model of the anthropomorphic robot SCARA;

2) selection of materials and components for the development of a physical model

3) creating a control program to perform typical tasks.

Key words: anthropomorphic robot, detail, control program, technological documentation

Student Yarema M.A

Зміст

ВСТУП

Спеціальність «Прикладна механіка» – одна з провідних у машинобудівній галузі. Перспективи розвитку сучасного машинобудування пов’язані не лише із вивченням, класифікацією та розробленням нових машин чи технологічного обладнання, але і з широким впровадженням у виробництво інформаційних технологій і комп’ютерних систем.

Освітньо-професійна програма “Інженерія мехатронних систем”, підготовки фахівців першого (бакалаврського) рівня вищої освіти за спеціальністю 131 «Прикладна механіка» галузі знань 13 «Механічна інженерія» передбачає поєднання змісту теоретичної та практичної підготовки, що дозволяє майбутнім фахівцям набути необхідних компетентностей з конструювання, виготовлення, дослідження та експлуатації мехатронних виробничих систем та комплексів. Та виконання різноманітних цілей таких як: професійна діяльність в галузі проектування, виробництво та експлуатація мехатронних систем, машин та устаткувань, робото-технічних засобів та комплексів у машинобудівельному виробництві.

Програма орієнтована на професійну підготовку, що передбачає освоєння принципів та методів створення сучасного обладнання з числовим програмним керуванням, зокрема робото технічних систем, на основі знань суміжних галузей – механіки, електроніки та програмування.

Випускник спеціальності “Прикладна механіка”, освітньо-професійної програми “Інженерія мехатронних систем” – це спеціаліст з прикладної механіки, який проводить дослідження, проєктує та випробовує автоматичні та автоматизовані механізми, системи з використанням робототехніки. Такі системи та механізми використовуються на підприємствах, які займаються машинобудуванням, видобутком, транспортуванням та переробкою нафто- та газопродуктів, на підприємствах деревообробної, легкої, харчової промисловості

тощо. Бакалавр з прикладної механіки може: брати участь у складанні та тестуванні складних мехатронних систем, досконало знати усі елементи роботизованої системи та їх призначення. Такий фахівець повинен вміти працювати з технічною документацією, володіти знаннями про прикладне програмне забезпечення, створювати програми керування.

Таким чином, сфера професійної діяльності випускників спеціальності 131 «Прикладна механіка», освітньо-професійної програми “Інженерія мехатронних систем” охоплює галузі науки і техніки, націлені на створення конкурентоздатної продукції машинобудування через застосування сучасних методів і засобів проєктування, програмування, фізичного та комп’ютерного моделювання.

Оскільки автоматизація та робототехніка є одним із пріоритетних напрямків розвитку промисловості в усіх розвинених країнах, то темою дипломного проєкту було обрано «Розробка антропоморфного робота Scara». В даній роботі буде розроблена конструкція маніпулятора, детально розглянуто електронне та програмне забезпечення робота, складено та випробовано програма для виконання певних функціональних рухів. Метою даного проєктування є можливість в подальшому використовувати робот-маніпулятор для виконання різноманітних задача як в особистих цілях так і для виконання лабораторних та практичних робіт студентами університету.

1. ТЕОРЕТИЧНА ЧАСТИНА

1.1 Основні поняття промислового робота

Промисловий робот являє собою програмовану автоматичну машину, яка складається з механічного маніпулятора, призначеного для використання в додатках промислової автоматизації, і програмований пристрій керування, який генерує керуючі дії, такі як вказівка необхідних рухів руки маніпулятора.

Маніпулятор (М) призначений для виконання рухомих функцій аналогічно тому як працює рука людини при переміщенні об'єктів в просторі, і до нього прикріплений робочий орган у вигляді хвата. Маніпулятор промислового робота містить рухомі та нерухомі ланки, з’єднання, передавальні механізми і приводи.

Приводи маніпулятора розміщені на ланках маніпулятора і призначені для реалізації руху ланок, а також об'єкта маніпулювання, закріпленого в хваті, в просторі вздовж заданої траєкторії і з заданою орієнтацією. Число приводів дорівнює числу степенів рухомості маніпулятора. Передавальні механізми використовуються для передачі руху від приводів до ланок маніпулятора.

Під промисловими роботами (автоматичними маніпуляторами з числовим програмним керуванням (ЧПК)) розуміють автоматично керовані, перепрограмовані пристрої, призначені для маніпулювання і транспортування матеріалів, деталей, інструментів або спеціальних пристроїв через різноманітні запрограмовані рухи для виконання своєрідних виробничих завдань.

Останнім часом автоматизація стала ключовою аспектом в забезпеченні конкурентоспроможності в більшості галузей промисловості, як наслідок у відповідь на загрози, пов'язані з зменшенням трудовитра у світі.

Як правило, більшість промислових виробництв інвестують в автоматизацію для отримання запланованої економії праці, але часто це не найістотніша вигода,

так як велику економію можна забезпечити за рахунок покращень, які не передбачених при розробленні проєкту.

Проте, установка промислових роботів забезпечує підвищення продуктивності за рахунок більш раціонального використання ресурсів та скорочення відходів або різноманітних процесів переробок сировини, видаленням небезпечних операцій з процесу виробництва і більш ефективного використання енергії за рахунок зменшення кількості ланок які задіяні в виробництві.

Для певної групи завдань більш раціональним є застосування промислових роботів ніж людей через якість виконуваної роботи, особливо в процесах в яких потрібно забезпечити один з перелічених параметрів на високому рівні:

- високу точність позиціонування, високу повторюваність
- відсутність відхилень зумовлених втомою
- високоточний контроль і вимірювання з використанням давачів

Також промислові роботи широко застосовуються в місцях з агресивним середовищем, в повторюваних процесах, а також для процесів в які потребують постійну високу точність.

В середовищах в яких є така можливість, промислові роботи можуть працювати цілодобово, щоб забезпечити максимальний ККД(коефіцієнт корисної дії) і економічну ліквідність.

1.1.1 Структура промислових роботів

Промислові роботи складаються з двох основних частин:

- механічна система у вигляді маніпулятора, з різною складністю і кінематичними конфігураціями;

- система керування маніпулятором і іншими пристроями.

Кожна з цих частин, в свою чергу, включає в себе ряд елементів, що взаємодіють між собою. На рис. 1.1 представлена структурна схема промислового робота.

Не залежно від фунції яка була, призначена для промислового робота (обробка, фарбування, складання, зварювання), механічна система спроектована таким чином, щоб переміщати інструмент в просторі(часто називають ефектором). Ефектор може бути просто хватом, призначеним для захоплення деталі, фарбувальним пістолетом або будь-яким іншим інструментом.

Геометричне розташування ефектора зазвичай є досить різноманітним в діапазоні досяжності, в так званому робочому просторі маніпулятора, і змінюється в часі безперервно.

Ефектор слідує траєкторії, що відповідає заданій задачі. Вона повністю описується послідовністю положень заданої точки на ефекторі і його орієнтацією навколо цієї точки. Щоб описати стан і орієнтацію ефектора в просторі, необхідно прив'язати до нього систему координат і описати положення та орієнтацію цього елементу в деякій системі відліку.

У загальній структурі механічної системи робота можна виділити п'ять основних складових:

- виконавчі механізми
- хват
- силовий привод з передавальними механізмами
- пристрої переміщення
- основа (база, несуча конструкція)

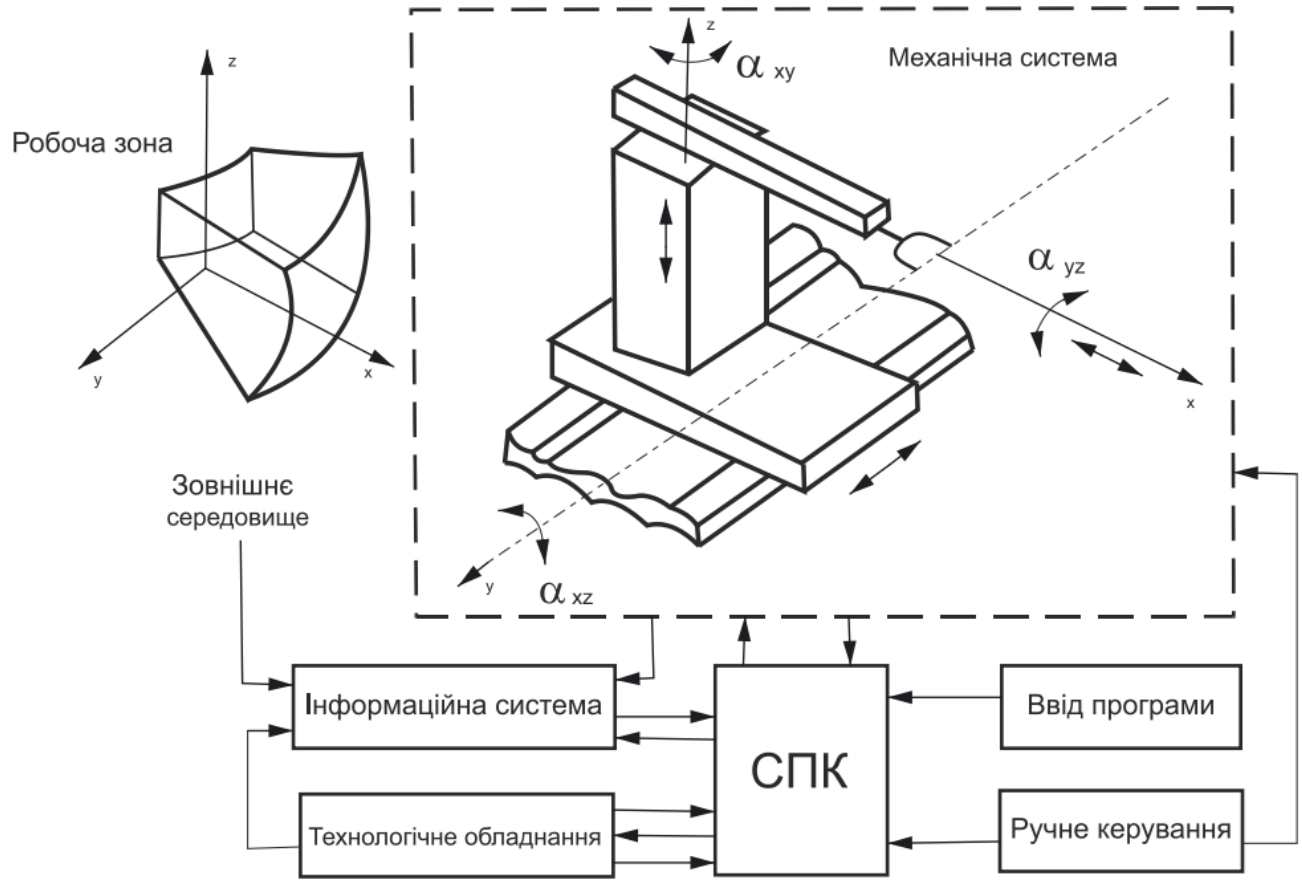


Рис. 1.1 - Структурна схема промислового робота

Виконавчий механізм являє собою жорсткі елементи або ланки, шарнірно з'єднані за допомогою механічних з'єднань. На його кінці може бути закріплений інструмент або ефектор.

Маніпулятор споряджений хватом, який є кінцевим вузлом маніпулятора, що використовується для забезпечення захоплення і утримання об'єкту маніпулювання в певному положенні.

Приводи забезпечують механічну потужність, для противодії впливу на механічну структуру сил: тяжіння, інерції і інших зовнішніх сил, а також для змін конфігурації і геометричного розташування інструменту. Приводи можуть бути електричними, гідравлічними або пневматичними.

Пристрій пересування використовується для переміщення маніпулятора або ПР в потрібне місце робочого простору і конструктивно складаються з ходової

частини і приводних пристроїв. Несучі конструкції служать для розміщення на них всіх пристроїв і агрегатів ПР, а також для забезпечення необхідної міцності і жорсткості маніпулятора.

Система керування виконує одночасно три різні ролі:

- інформаційна роль, яка полягає в збиранні та обробці інформації, що надається давачами;

- роль прийняття рішень, яка полягає в плануванні геометричного руху структури маніпулятора, починаючи з визначення завдання наданого оператором, стану системи і навколишнього середовища, що передається давачами;

- роль зв'язку, яка полягає в організації потоку інформації між блоком керування, маніпулятором і робочим середовищем.

Щоб опрацьовувати дані функції, система керування має включати в себе програмне забезпечення та різноманітні бази даних, такі як:

- модель (кінематична і/або динамічна) робота, яка виражає зв'язок між командами керування приводами і результуючим рухом конструкції;

- модель середовища, яка геометрично описує робоче середовище робота. Вона надає інформацію про наявність зон, в яких можуть виникнути зіткнення, і дозволяє відповідним чином планувати траєкторію;

- алгоритми керування, які керують рухом робота на більш низькому рівні, і відповідають за стан механічної структури і її виконавчих механізмів (за умови, що це керування становищем і швидкістю із заданими характеристиками точності і стабільності);

- протокол зв'язку, який передбачає керування повідомленнями (за формою і пріоритетом), якими обмінюються різні компоненти системи. Система керування може мати або централізовану архітектуру, і в цьому випадку один і той же процесор приймає на себе всі функції, описані вище, або ієрархічну організацію, і в

цьому випадку система організована навколо головного пристрою, який призначає кожному з підлеглих блоків деякі з виконуваних функцій.

1.1.2 Класифікація промислових роботів

Існує ряд способів, якими можуть бути класифіковані ПР. Деякі загальні признаки класифікації роботів: призначення, спеціалізація, конструкція, вантажопідйомність, тип приводу, тип системи керування, спосіб програмування, система координат, конструкція маніпулятора, мобільність. На рис. 1.2 представлена класифікація промислових роботів за різними ознаками.

Відповідно до характеру виконуваних операцій промислові роботи поділяють на три групи:

- технологічні (виробничі);
- допоміжні (підйомно-транспортні);
- універсальні.

За ступенем спеціалізації промислові роботи поділяють на:

- спеціальні;
- спеціалізовані (цільові);
- універсальні.

За типом системи координат існують промислові роботи з прямокутною, полярною, ангулярною системою координат.

За ступенем рухливості існують промислові роботи з двома, трьома, чотирма і більше чотирьох ступенями рухливості.

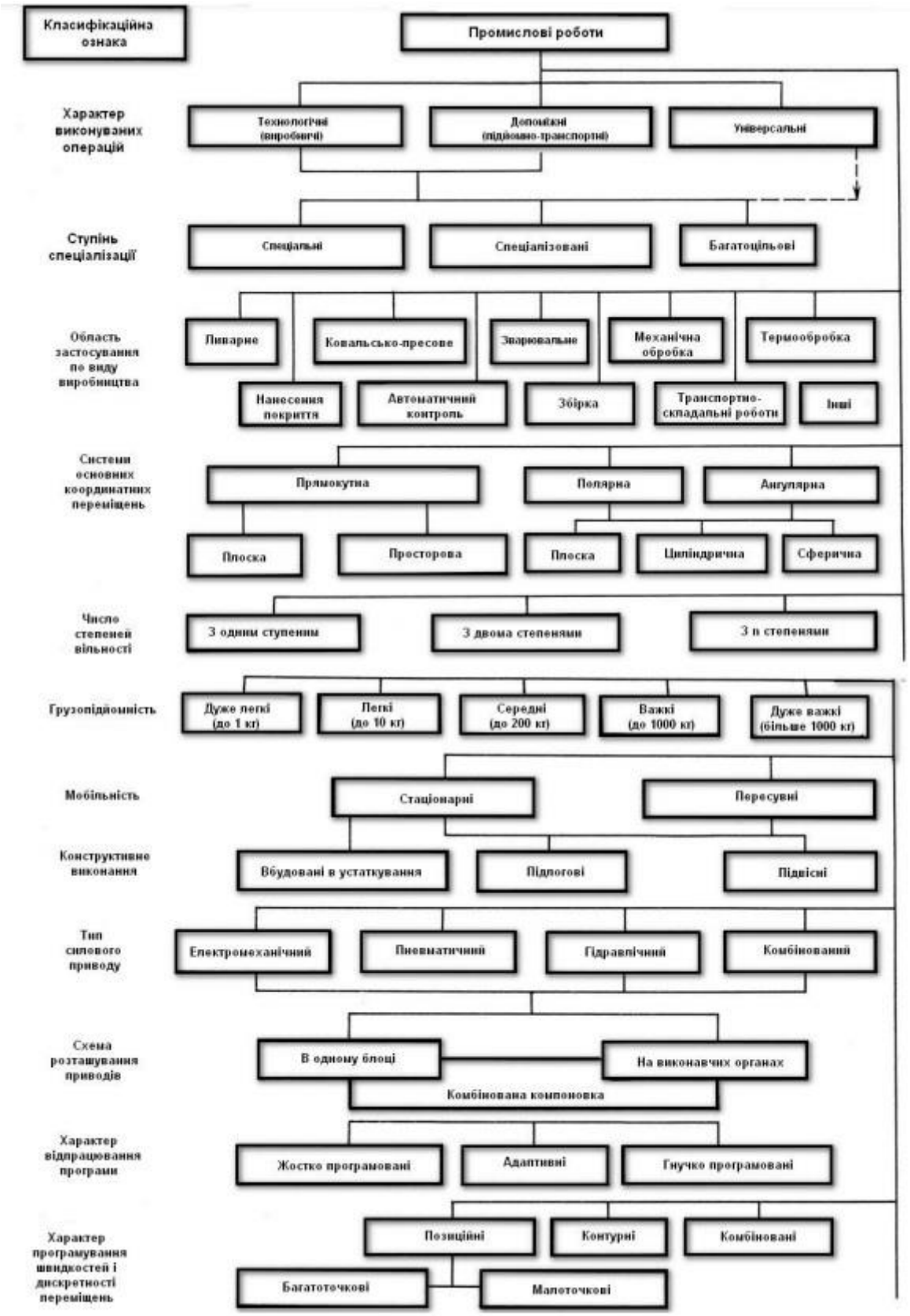


Рис. 1.2 - Класифікація промислових роботів

За вантажопідйомністю промислові роботи діляться на надлегкі (до 1 кг), легкі (від 1 до 10 кг), середні (від 10 до 200 кг), важкі (від 200 до 1000 кг) і надважкі (понад 1000 кг).

Відповідно до типу силових приводів розрізняють промислові роботи з електромеханічним, гідравлічним та пневматичним приводами.

Відповідно до типу використовуваної системи керування промислові роботи підрозділяються на ПР з програмним керуванням (позиційним, контурним і комбінованим) і з адаптивним керуванням (позиційним, контурним)

1.2 Аналіз основних видів промислових маніпуляторів за типом їх систем координат

Найважливішим завданням при проектуванні маніпулятора є вибір кінематичної структури системи, який залежить від того, які рухи, і в якій послідовності повинен виконувати маніпулятор при функціонуванні. Оскільки маніпулятор сконструйований в більшості випадків для імітації руху руки людини, то структурна схема маніпулятора повинна мати кінематичні характеристики, аналогічні характеристикам руки людини.

Механічну структуру маніпулятора ПР можна розглядати як багаторівневу систему, тобто систему тіл або ланок, теоретично жорстку, з'єднану між собою суглобами або кінематичними парами.

Для твердого тіла, вільно розташованого в просторі, можна вказати як його положення, так і орієнтацію в просторі, задавши положення трьох не колінеарних точок, прикріплених до нього. Це сукупність дев'яти параметрів, таких як координати цих точок, відстань між якими фіксована.

У загальному випадку геометричне розташування вільного твердого тіла описується шістьма параметрами або ступенями рухливості, які можуть бути класифіковані за двома категоріями:

- три незалежні параметри переміщення (П), які визначають розташування в просторі контрольної точки твердого тіла;

- три незалежні параметри обертання (О), які визначають орієнтацію в просторі твердого тіла.

Існує п'ять основних типів маніпулятора з трьома ступенями рухливості в різних системах координат і їхні робочі зони:

- маніпулятор з прямокутною системою координат (рис. 1.3) — має структуру П-|П-|П, кубічний робочий простір з об'ємом $V = L^3$.

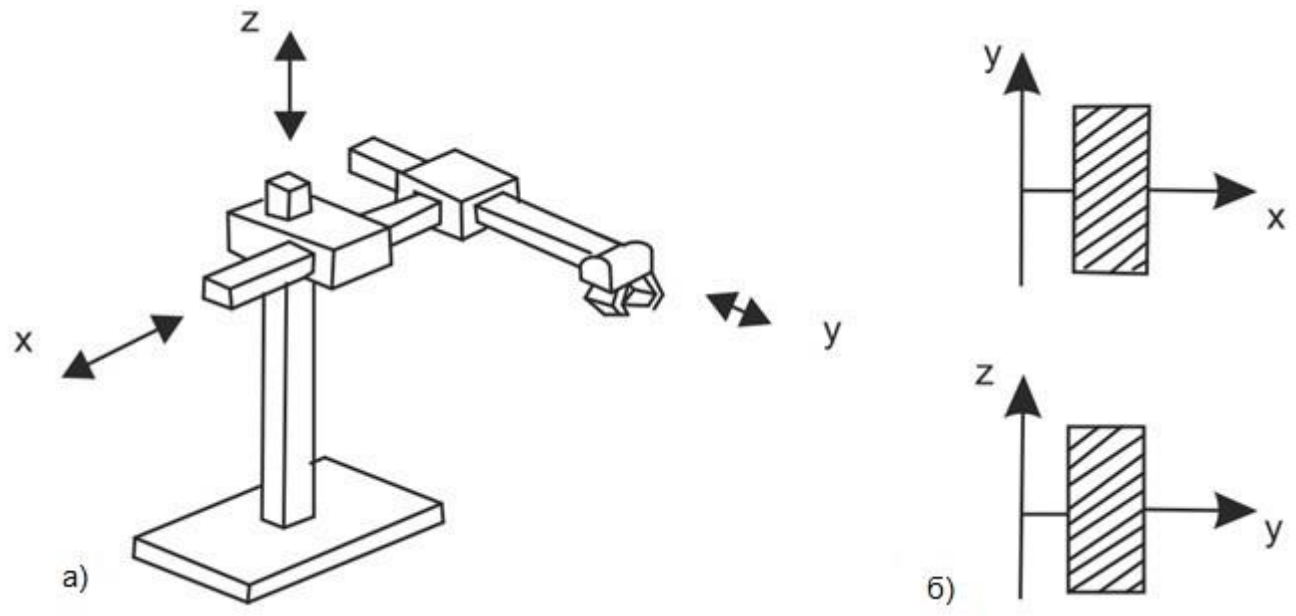


Рис. 1.3 - Маніпулятор з прямокутною системою координат (а) і його робоча зона (б)

Маніпулятор з циліндричною системою координат (рис. 1.4) — осі утворюють циліндричну систему координат. Структура О||П-|П, торовидний робочий простір квадратного перерізу з внутрішнім радіусом L, зовнішнім радіусом $2L$, з об'ємом $V = 3\pi L^3$.

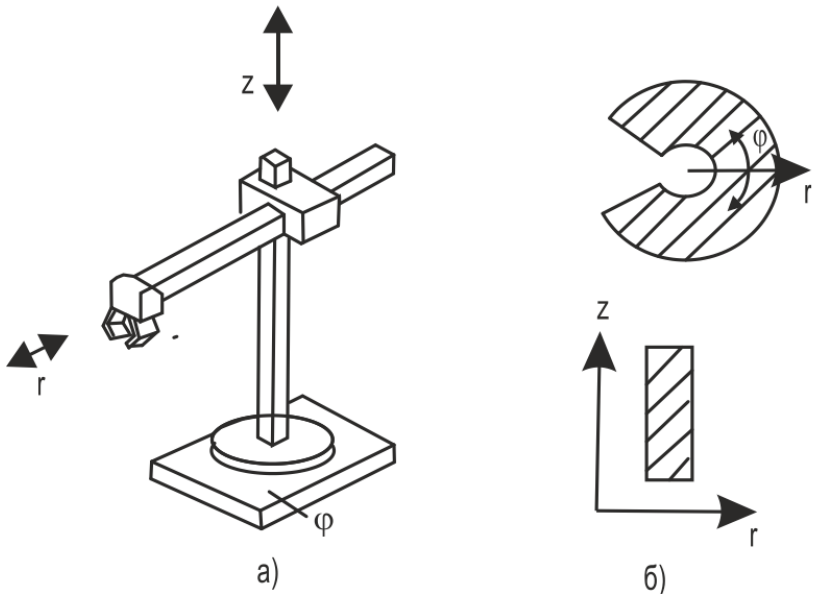


Рис. 1.4 - Маніпулятор із циліндричною системою координат (а) і його робоча зона (б)

Маніпулятор із сферичною системою координат (рис. 1.5) — структура О-|О-|П, внутрішній радіус сферичної робочої зони L, зовнішній радіус $2L$, з об'ємом $V = 28/3\pi L^3$.

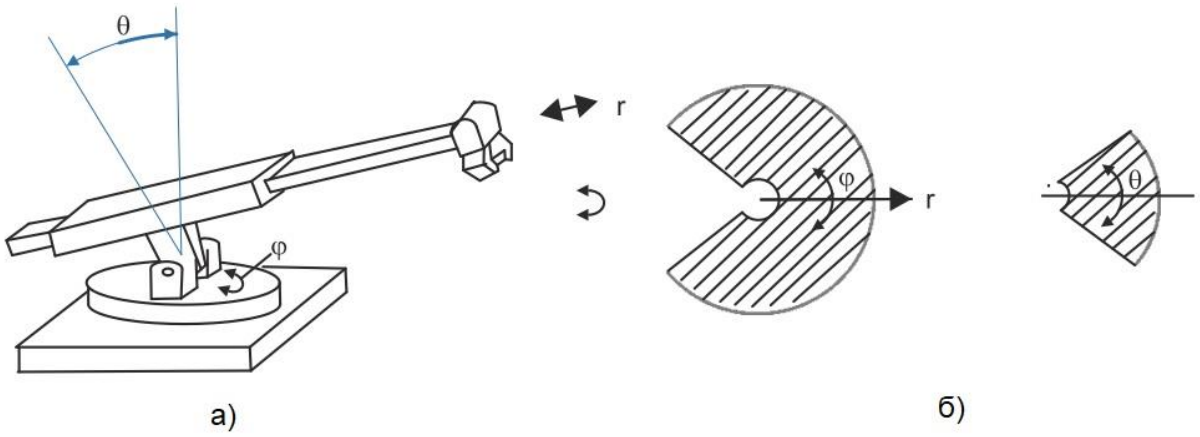


Рис. 1.5 - Маніпулятор зі сферичною системою координат (а) і його робоча зона (б)

Маніпулятор з кутовою системою координат (рис. 1.6) — структура O||O||O, радіус сферичного робочого простору $2L$, з об'ємом $V = 32/3\pi L^3$. Такі маніпулятори називають обертовими і антропоморфними.

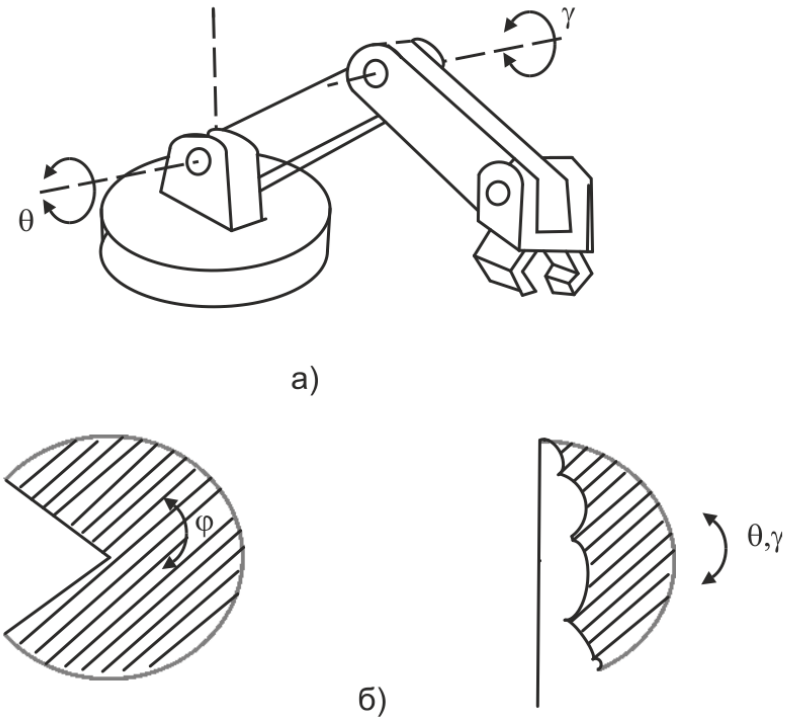


Рис. 1.6 - Маніпулятор з кутовою системою координат (а)
і його робоча зона (б).

SCARA — Selective Compliance Assembly Robot Arm (рис. 1.7) — аналогічний до конструкції маніпулятора з циліндричною системою координат, за винятком того, що плечові і ліктьові осі обертання вертикальні, що означає, що важіль дуже жорсткий у вертикальному напрямку, але сумісний в горизонтальному напрямку.

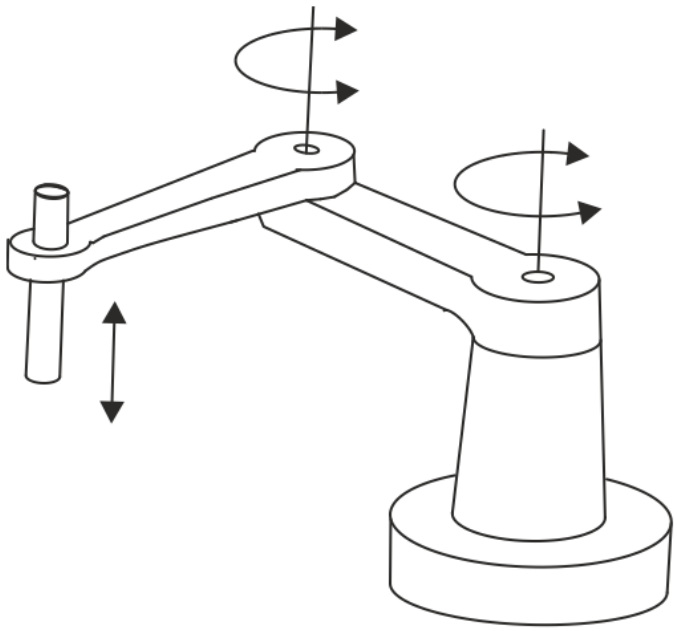


Рис. 1.7 - Маніпулятор SCARA

1.3 Аналіз основних типів приводів і систем керування маніпуляторів промислового робота.

1.3.1 Основні типи приводів маніпулятора промислового робота.

Приводи маніпулятора призначені для забезпечення руху всіх ланок маніпуляційного механізму і хвата робочого органу. Вони включають в себе: двигуни, блоки керування, передавальні механізми для з'єднання двигуна з робочим органом, гальмівні пристрої, давачі зворотного зв'язку і комунікації. Комунікації необхідні для передачі енергії до приводів і передачі сигналів керування, а також для виконання зворотного зв'язку.

Правильний вибір приводів для керування позиціями лінійних і кутових переміщень в промислових роботах вимагає врахування декількох параметрів: призначення ПР, зручність експлуатації, надійність, вартість, ліквідність, вантажопідйомність, швидкодія, точність позиціонування і вид системи керування.

При виборі типу приводу маніпулятора необхідно враховувати наступні вимоги:

• мінімально можливі габаритні розміри;

• високі енергетичні показники;

• високу швидкодію;

• високу надійність;

• забезпечення точності позиціонування;

• зручність монтажу, обслуговування і ремонту;

• можливість роботи в режимі автоматичного керування і регулювання.

Основні типи приводу (рис. 1.8), що використовують в конструкціях промислових роботів включають в себе пневматичні, гідравлічні, електричні і комбіновані приводи.

Рис. 1.8 - Структура приводу ПР

СК — система керування; МС — механічна система

В наш час, більшість існуючих маніпуляторів ПР використовують електричний привод для виконання рухів. Застосування електроприводів в конструкціях маніпулятора ПР зумовлено рядом переваг, які включають:

- високу надійність, безшумність і точність;

- легкість регулювання; простоту монтажу і налагодження;
- високий ККД (0,5 ... 0,7);
- зменшення металоємкості конструкції ПР;
- більш високу економічність.

Електропривід містить в собі підсилювачі потужності, керовані двигуни, передавальні механізми, давачі зворотного зв'язку за швидкістю і за положенням, порівняльні пристрої.

Здебільшого в електроприводах використовують синхронні, крокові двигуни та двигуни постійного струму. Асинхронні двигуни використовуються рідже ніж інші, через складність та високу вартість електронних перетворювачів для регулювання частоти обертання. Найбільш підходящим для ПР є електродвигуни постійного струму із збудженням від постійних магнітів, що мають порівняно високі показники питомої потужності.

1.3.2 Системи керування електроприводом маніпулятора промислового робота

Системи керування електроприводом маніпулятора забезпечують рух виконавчого органу за заданою просторовою траєкторією шляхом керування руху окремих ланок маніпулятора.

В сучасний час більшість систем керування ПР функціонує на основі використання зворотного зв'язку, ієрархічної структури системи керування роботом і каскадного керування.

Структура системи керування ПР містить горизонтальні рівні:

- керування загальною поведінкою ПР;

- планування рухів, тобто розрахунок необхідної траєкторії руху виконавчого органу;
- поведінки окремих приводів — на цьому рівні організовується злагоджена робота приводів, що забезпечує необхідне переміщення робочого органу;
- рівень приводу, на якому безпосередньо здійснюється керування двигуном приводу.

Системи керування електроприводом маніпулятора ПР побудовані за принципом каскадного керування. Щоб побудувати систему керування за положенням (наприклад, за кутом повороту ланки маніпулятора), система керування замикається зворотним зв'язком за положенням, а всередині системи керування за положенням функціонує система керування за швидкістю зі своїм зворотним зв'язком за швидкістю, всередині якої існує контур керування за струмом.

Системи керування руху (СКР) маніпулятора ПР поділяються на системи позиційного, циклового, й контурного керування.

Системи позиційного керування мають безліч програмованих положень точок, через які повинен пройти схват маніпулятора в процесі руху. При позиційному керуванні команди подаються таким чином, що переміщення робочого органу відбувається від точки до точки, причому положення точок задаються програмою. Швидкість переміщення між точками не задається і не контролюється.

Системи циклового керування мають мале число точок позиціонування та найчастіше вони просто перемикають рух маніпулятора за кожною приєднаною координатою від упору до упору. Дане керування програмує послідовність виконання рухів і умови початку і закінчення рухів; позиція, до якої відбувається переміщення, задаються на самому маніпуляторі механічно, а не в програмі; швидкість переміщення визначається характеристиками приводу і також не задається в програмі.

Системи контурного керування мають приводи, що стежать за кожною приведеною координатою (стежать системи зі зворотним зв'язком). Тому при спільній роботі рухомих з’єднань схват маніпулятора здійснює плавний рух за запрограмованою неперервною траєкторією і позиціонується в будь-якій точці робочої зони. При контурному керуванні рух робочого органу відбувається за заданою траєкторією із заданою швидкістю. У програмі задаються самі траєкторії і режими руху. Контурне керування використовується здебільшого в технологічних роботах.

1.3.3 Опис сервоприводів та крокових двигунів.

У конструкціях роботів-маніпуляторів використовуються, зазвичай, в якості приводів крокові двигуни сервомашинки та крокові двигуни.

Сервомашинка являє собою двигун з вбудованим редуктором і вихідним валом, положення якого можна точно контролювати за рахунок наявного внутрішнього зворотного зв’язку за кутовим положенням (рис. 1.9). Стандартні сервомашинки дозволяють задавати кутове положення вихідного вала в діапазоні від 0 до 180 градусів. У двигунах з незперервним обертанням вала можна задавати також і швидкість його обертання.

Рис. 1.9 - Зображення типового серводвигуна

для використання в моделюванні

Класична сервомашинка має в своїй конструкції вимірювач позиціонування (потенціометр), силовий агрегат (колекторний двигун) і 3-контурну керуючу систему, що відповідає за регулювання позиції, швидкісного режиму і електричної напруги. Термін «серво» означає в перекладі з латинської «помічник».

Область застосування промислових сервоприводів всьогодені є досить широкою, завдяки можливосттям винятково точного позиціонування робочого органа. Тут і механічні засувки, і клапани, і робочі органи різних інструментів і верстатів, особливо з ЧПК, включаючи автомати для заводського виготовлення друкованих плат, і різні промислові роботи, і багато інших точних приладів.

Промисловий сервопривод відрізняється від зображеного на рис. 1.9. Механізми цього типу складаються з двигуна оснащеного давачем, що відслідковує конкретний параметр, наприклад положення, швидкість або зусилля, та блока керування (рис. 1.10), завдання якого — підтримувати в автоматичному режимі необхідний параметр в процесі роботи пристрою в кожний момент часу.

Споконвічно в конструкції сервоприводів застосовувалися мотори постійного струму триполюсні колекторні, де ротор містив обмотки, а статор — постійні магніти. Крім того, був колекторно-щітковий вузол. Пізніше кількість обмоток зросла до п'яти, і крутний момент став більше, а розгін — швидше. 20

Рис. 1.10 - Промислові сервоприводи змінного струму SV-DA200

Наступна стадія вдосконалювання — обмотки розмістили зовні магнітів, так поменшала вага ротора, і скоротився час розгону, однак зросла вартість. У підсумку був зроблений ключовий крок удосконалювання - відмовилися від колектора (зокрема поширення одержали приводні мотори з постійними магнітами на роторі), і двигун вийшов безщітковим, ще більш ефективним, оскільки прискорення, швидкість, і крутний момент стали тепер ще вище.

Завдяки високій энергоефективності, можливості точного керування і відмінним робочим характеристикам, саме сервоприводи на базі безколекторних моторів усе частіше можна зустріти як в іграшках, так і в побутовій техніці та в промисловому устаткуванні.

В даний час приладо- і машинобудування засноване на автоматизації різних процесів, спрощення систем і різних вузлів за рахунок приведення механізмів до простих, але ефективних конструктивних рішень. Сервоприводи в цих процесах

зайняли своє місце, оскільки дозволяють підтримувати постійну швидкість в конструкціях промислової робототехніки і в верстатах високої точності.

1.3.4 Сервомашинки в системах керування на базі Arduino

За габаритами сервомашинки поділяються на: мікро, міні, стандарт і гігант.

Вибір сервоприводів здійснюється за двома показниками — це крутний момент на валу і швидкодія. За принципом керування розрізняють сервомашинки цифрові і аналогові.

За зовнішнім виглядом особливих відмінностей між конструкціями (рис. 1.11) цифрових і аналогових пристроїв можна і не знайти. Різниця між цими пристроями полягає в платах керування. На цифрових моделях встановлений мікропроцесор, який і аналізує сигнали, що надходять, і віддає команди керування двигуном. Функціонування цифрових та аналогових приладів серйозно не різняться між собою. Головна відмінність між ними полягає в методі переробки сигналів керування, що надходять на пристрій.

Рисунок 1.11 - Схема конструкції сервоприводу

Для керування сервомашинкою використовується метод модуляції ширини керуючого імпульсу. Ширина імпульсу, що подається на двигун з частотою 50 Гц, варіюється в певних межах і визначає кутове положення вихідного валу. Наприклад, ширина імпульсу 0,8 мс викликає кутове положення -90°, тоді як ширина імпульсу 2,2 мс задає кут повороту 90°. При ширині імпульсу 1,5 мс сервопривод знаходиться в нейтральному положенні.

Для програмування плат Arduino, які активно використовуються для керування моделями роботів-маніпуляторів створена бібліотека Servo. Ця бібліотека є основним інстументом у спрощенні написання керуючих програм для плат Arduino. У бібліотеці Servo реалізована можливість одночасного керування декількома двигунами: на більшості плат Arduino — до 12, на Arduino Mega — до 48.

У сервомашинки є три дроти: живлення, сигнальний провід і земля (Gnd). Провід живлення (зазвичай червоного кольору) повинен з'єднуватися з виводом +5V плати Arduino. Сигнальний провід (зазвичай оранжевого, жовтого або білого кольору) повинен з'єднуватися з цифровим виводом плати Arduino. Провід землі (як правило, чорний або коричневий) повинен бути приєднаний до відповідного виводу на платі Arduino. Слід пам'ятати, що серводвигуни споживають відносно великий струм, тому при необхідності керування двома або більше двигунами рекомендується живити їх від окремого джерела живлення (не використовуючи мережу + 5V Arduino). При цьому слід переконатися, що виводи землі Arduino і зовнішнього джерела живлення з'єднані разом.

1.3.5 Використання крокових двигунів в системах керування на базі Arduino

Так само як і сервоприводи, крокові двигуни є вкрай важливим елементом автоматизованих систем і робототехніки.

Кроковий двигун — це електромеханічний пристрій, що перетворює сигнали керування в кутове (або лінійне) переміщення ротора з фіксацією його в заданому положенні без пристроїв зворотного зв'язку (рис. 1.12). При проектуванні конкретних систем доводиться робити вибір між кроковим двигуном і сервомотором. Коли потрібно прецизійне позиціонування і точне керування швидкістю, а необхідний момент і швидкість не виходять за допустимі межі, то кроковий двигун є найбільш рентабильним рішенням. Як і для звичайних двигунів, для підвищення моменту можжна використати понижуючий редуктор. Однак для крокових двигунів редуктор не завжди підходить. На відміну від колекторних двигунів, у яких момент зростає із збільшенням швидкості, кроковий двигун має більший момент на низьких швидкостях. До того ж, крокові двигуни мають набагато меншу максимальну швидкість в порівнянні з колекторними двигунами, що обмежує максимальне передавальне число і, відповідно, збільшення моменту за допомогою редуктора. 23

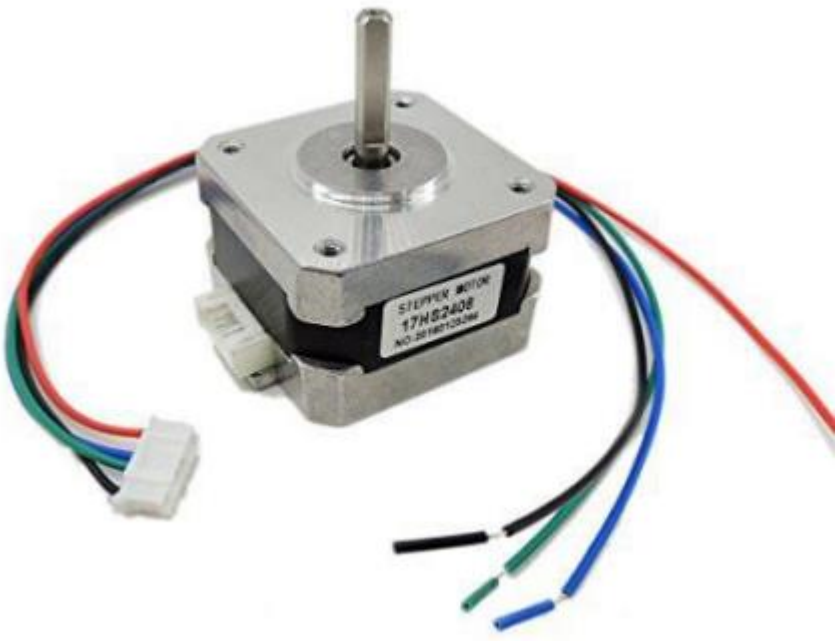


Рис. 1.12 Зображення типового крокового двигуна

Принцип дії всіх існуючих крокових двигунів заснований на дискретній зміні стану електромагнітного поля в робочому зазорі електричної машини. Це досягається імпульсним збудженням (або перемиканням) її обмоток.

Найбільше розповсюдження сьогодні мають біполярні та уніполярні крокові двигуни.

Способи керування фазами крокового двигуна: повнокроковий режим; напівкроковий режим; мікрокроковий режим.

Момент, створюваний кроковим двигуном, залежить від швидкості, струму в обмотках і схеми драйвера, а для того, щоб працювати на великій швидкості, необхідно стартувати на низькій швидкості з області старту, а потім виконувати розгін. При розгоні двигун проходить ряд швидкостей, при цьому на одній з швидкостей можна зіткнутися з неприємним явищем резонансу. Для нормального розгону бажано мати навантаження, момент інерції якого як мінімум дорівнює моменту інерції ротора. На ненавантаженому двигуні явище резонансу проявляється найбільш сильно. При здійсненні розгону або гальмування важливо правильно вибрати закон зміни швидкості і максимальне прискорення. Прискорення повинно бути тим менше, чим вище інерційність навантаження. Критерій правильного вибору режиму розгону — це здійснення розгону до потрібної швидкості для конкретного навантаження за мінімальний час. На практиці найчастіше застосовують розгін і гальмування з постійним прискоренням. Реалізація закону, за яким буде проводиться прискорення або гальмування двигуна, зазвичай проводиться програмно керуючим мікроконтролером, так як саме мікроконтролер зазвичай є джерелом тактової частоти для драйвера крокової двигуна.

Перевагами крокового двигуна є кут повороту ротора, що визначається кількістю імпульсів, які подані на обмотки двигуна. Двигун забезпечує повний момент в режимі зупинки (якщо обмотки заживлені) та дає прецизійне

позиціонування і повторюваність. Хороші крокові двигуни мають точність від 3 до 5% від величини кроку. Ця помилка не накопичується від кроку до кроку. Окрім цього, у крокового двигуна є можливість швидкого старту, зупинки та реверсування, а також висока надійність, пов'язана з відсутністю щіток, однозначна залежність положення від вхідних імпульсів забезпечує позиціонування без зворотного зв'язку, можливість отримання дуже низьких швидкостей обертання для навантаження, приєднаного безпосередньо до валу двигуна без проміжного редуктора. Термін служби крокового двигуна фактично визначається терміном служби підшипників.

Джерелом тактових імпульсів для керування двигуном є мікроконтролер керуючої плати Arduino, персональний комп’ютер або будь-який інший мікропроцесорний пристрій. Для безпосереднього живлення обмоток двигуна керуючі сигнали повинні бути підсилені. Для цієї мети використовуються спеціальні драйвери. Наприклад, це може бути пристрій на базі двох мікросхем — контролера крокового двигуна L297 і підсилювача L298N (рис. 1.13).

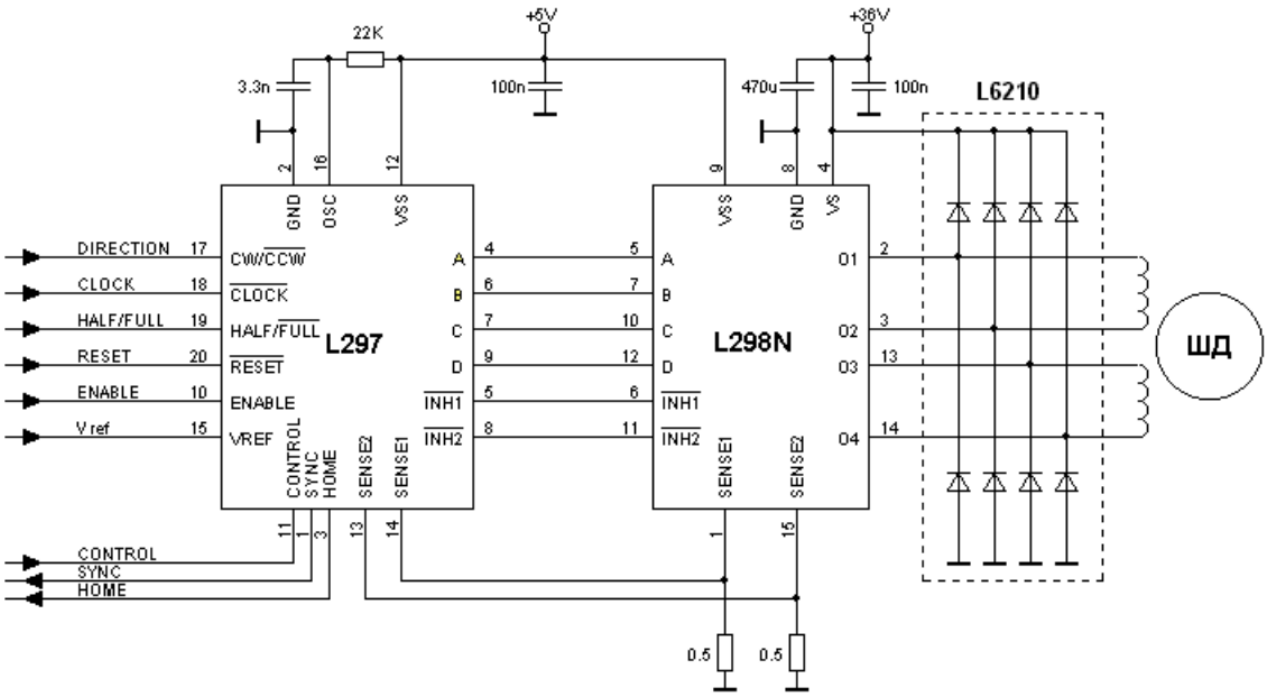


Рис. 1.13 - Схема контролера крокового двигуна

Відповідно до схеми контролер L297 виконує перетворення керуючих сигналів процесора в послідовність імпульсів, які подаються на обмотки двигуна. Всі сигнали відповідають стандарту TTL. Для роботи двигуна необхідно забезпечити передачу на його обмотки значно більших потужностей ніж може забезпечити стандарт TTL, тому необхідна присутність підсилювача L298N. Дана мікросхема забезпечує на виходах робочу напругу до 46 В і максимальний струм до 4 А.

На теперішній час в продажі є великий вибір готових до використання драйверів крокових двигунів

1.3.6 Програмне керування роботом маніпулятором

За визначенням, будь-який сучасний робот передбачає наявність програмованої системи керування. Розглянемо три найпоширеніші в наш час системи.

Програмне забезпечення Marlin

Marlin – це програмне забезпечення з відкритим кодом для пристроїв швидкого прототипіювання — відоме широкому загалу як «3D принтери». Це автономний загальнодоступний проект, який з 12 серпня 2011 реалізується через платформу GitHub. Marlin ліцензований під GPLv3 і вільний для використання в усіх застосунках.

З початку Marlin був побудований для ентузіастів RepRap, щоб бути прямим, надійним, і здатним до адаптування драйвером принтера, який "просто працює". Як запорука його якості, Marlin використовується для кількох комерційних 3D принтерів. Ultimaker, Printrbot, AlephObjects (Lulzbot) і Prusa Research — це всього навсього декілька виробників, які використовують варіант Marlin. Marlin також

придатний для керування верстатів з ЧПК, лазерних граверів і також може бути адаптований до керування настільними роботами.

Одиним з факторів який зробив програмного забезпечення (ПЗ) Marlin таким популярним — це робота недорогому 8 бітовому мікроконтролері Atmel AVR — Marlin 2.x підтримує 32-х бітові пристрої. Ця мікросхема знаходиться в центрі загальнодоступних платформ Arduino/Genuino. Зазвичай платформи для Marlin включають в себе плату Arduino Mega2560 в комплекті з платою розширення RAMPS 1,4.

Як суспільний продукт Marlin прагне до адаптованості із якомога більшою кількістю плат і конфігурацій.

Головні особливості:

• повнофункціональний G-код з більш ніж 150 командами;

• повний комплекс руху G-код, включаючи лінії, дуги і криві Безьє;

• розумна система руху з передбаченням, заснованим на перерві рухом, лінійним прискоренням

• підтримка Декартової, Delta, SCARA і Core/H-Bot кінематики;

• ПІД-контроль нагрівачів зі зворотним зв'язком, автоматичним налаштуванням, тепловим захистом;

• підтримка до 5 екструдерів плюс гарячий стіл;

• контролер рідкокристалічного дисплею з більше ніж з 30 мовними перекладами;

• друк з віддаленого ПК і SD-карти з автостартом;

• компенсація нерівностей стола;

• підтримка об 'ємного контролю подачі екструдера;

• підтримка давачів контролю обриву філамету;

• таймер роботи принтера

Програмне забезпечення Marlin працює на материнській платі 3D принтера, справляючись з усіма діями машини в реальному часі. Воно координує нагрівачі, крокові двигуни, давачі, індикатори, РК-монітор, кнопки і все інше, що задіяне в процес 3D друку.

Мова керування для Marlin — походить від G-кодів. Команди G-кодів вказують машині виконувати, прості дії, як ,наприклад, «нагріти нагрівач 1 до 180°C", або "рухатись до координати X,Y на швидкості F".

Коли Marlin отримує команду руху, він додає її до черги руху, щоб вона була виконана в порядку отримання. Переривання крокових двигунів обробляє чергу, перетворюючи лінійні руху в точно розраховані електронні імпульси для крокових двигунів. Навіть на невеликих швидкостях Marlin повинен виробляти тисячі імпульсів для крокових двигунів кожну секунду (наприклад, 80 кроків на мм * 50 мм/с = 4000 кроків в секунду!).

Нагрівачами і давачами керування відбувається іншим перериванням з низьким пріоритетом , яке виконується повільніше, поки в головному циклі здійснюється обробка команд, оновлення показів дисплея і повідомлень контролера. З міркувань безпеки Marlin виконує перезавантаження, якщо центральний процесор буде перевантажений, щоб прочитати покази давачів.

Роботою ПЗ Marlin можна керувати повністю з ПК або в автономному режимі з SD-карти. Автономна робота з SD може також бути розпочата з керуючого ПК.

Програмне забезпечення для керуючого пристрою доступне для декількох платформ, включаючи настільні системи, Raspberry Pi і планшети на базі Android. Будь-який пристрій з USB-портом і послідовним терміналом може бути використаний як керуючий. Для 3D принтерів використовується спеціально призначене програмне забезпечення, зокрема:

• Pronterface — відкрите ПЗ від Kliment;

• Repetier Host — закрите ПЗ Repetier Software;

• OctoPrint — відкрите ПЗ Raspberry Pi від Gina Häußge ;

• Cura — відкрите ПЗ від Ultimaker.

Програмне забезпечення Mach3

Mach3 — програмне забезпечення яке призначене для керування верстатом з ЧПК. Встановлюється на ПК під ОС Windows. Дане програмне забезпечення являється оптимальним співвідношенням ціна/якість.

Функції і характеристики Mach3:

Переконвертація стандартного ПК в повнофункціональну станцію керування 6-осьовим верстатом з ЧПК

Імпорт DXF, BMP, JPG, і HPGL файлів за допомогою вмонтованої програми LazyCam

- Графічна візуалізація G-кодів
- Генерування G-кодів в програмі LazyCam або в Wizard
- Інтерфейс із можливістю повного переналаштування
- Створення користувацьких M-кодів та макросів на основі VB-скриптів
- Управління частотою обертання шпинделя
- Багаторівневе релейне регулювання
- Застосування ручних генераторів імпульсів (MPG)
- Відеоспостереження за ходом обробки
- Сумісність сенсорного дисплею

- Повноекранний користувацький інтерфейс

Mach3 успішно застосовується для управління з наступними типами устаткування:

- Токарні верстати
- Фрезерні верстати
- Швидкохідні деревообробні фрезерні верстати
- Лазерні верстати
- Плазморізи
- Гравірувальне обладнання

Теоретично, Mach3 може бути налаштований для керування роботом. Але закритий вихідний код ускладнює цю задачу.

.

Програмне забезпечення LinuxCNC

LinuxCNC (раніше «Enhanced Machine Controller» або «EMC2») — вільна відкрита операційна система на базі ядра GNU/Linux для персональних комп'ютерів загального призначення, яка реалізує числове програмне керування верстатами, роботами тощо. Дана система розробляється спільнотою linuxcnc.org і 28 надається, як правило, як ISO-образ з модифікованою версією 32-бітної Ubuntu Linux з ядром реального часу.

LinuxCNC — це програмна система для числового програмного керування верстатів та машин, таких як токарні, фрезерні, фрезерування деревини, плазмового різання, розкрою, гексаподів, промислових роботів та інших декартових координатних роботів. Як вхідні дані система використовує G-код (RS-

274NGC) і може контролювати до 9 осей або приводів верстатів з крокovими двигунами або сервомашинками. Система має декілька графічних інтерфейсів користувача відповідно до конкретних видів використання (інтерактивне керування, сенсорний екран).

Система не забезпечує функцій креслення (CAD — Computer Aided Design) та генерації G-коду з креслення (CAM — Computer Automated Manufacturing).

Системне програмне забезпечення EMC було розроблене Національним інститутом стандартів і технології (NIST) та надане у суспільне надбання. Програмне забезпечення та стандарт на мову G-code інтерпретації інструкцій керування рухом інструмента або рушія в режимі реального часу викликало інтерес аматорів та професійних користувачів верстатів. Приблизно в червні 2000 року NIST переніс вихідний код на sourceforge.net з тим, щоб добровільні розробники могли надалі самостійно змінювати та розвивати проект. У 2003 році були переписані деякі частини системи, реорганізовані і спрощені інші частини, оновлений проект отримав назву EMC2. На даний момент EMC2 ліцензований під GNU General Public License та активно розвивається. EMC2 включає новий шар управління, відомий як HAL (Шар апаратних абстракцій), введений для забезпечення незалежності функцій керування від апаратного забезпечення без зміни коду або перекомпіляції.

EMC2 також включає в себе механізм поділу траєкторії і планування руху, що робить його більш легким для створення програм та керування верстатами. HAL включає віртуальний осцилограф для дослідження сигналів в режимі реального часу та модель релейної логіки, адаптовану для налаштування складних допоміжних пристроїв, таких як пристрої автоматичної зміни інструменту. Близько 2011 року за наполяганням корпорації EMC і згоди керівників проект отримав назву LinuxCNC. LinuxCNC також включає в себе програмований логічний контролер (PLC), який зазвичай використовується в великих проектах (наприклад,

складних обробних центрах). Програмне забезпечення PLC базується на вихідному відкритому проекті Classicladder і працює в режимі реального часу.

2. ПРАКТИЧНА ЧАСТИНА

2.1 Опис навчального проекту робота SCARA

В якості об’єкта для досліджень було обрано робота SCARA. Даний робот має 4 ступені свободи і приводиться в рух 4 кроковими двигунами NEMA 17. Крім того, він має невеликий сервомотор для керування кінцевим ефектором або захватом робота. Мозком цього робота SCARA є плата Arduino UNO, яка поєднана з чотирма кроковими драйверами A4988 для керування кроковими двигунами .

Механічна конструкція робота надрукована з використанням 3D технології та керується апаратним і програмним забезпеченням Arduino.

Однією з проблем закладів освіти як в Україні є висока вартість матеріалів і обладнання, які студенти повинні використовувати в процесі свого навчання. У цьому сенсі роботизована рука з відкритим кодом, модифікована та відтворена за низькою вартістю студентами, може забезпечити декілька існуючих навчальних дисциплін: механічне конструювання, автоматику, програмування мікропроцесорних систем і систем з числовим програмним керуванням тощо.

Отже, робот SCARA дозволяє навчальним закладам користуватися легко доступним обладнанням, що настроюється і модифікується, за ціною, далекою від ціни промислового обладнання, але з достатніми перевагами для навчальних цілей.

Таким чином робот SCARA, як проект для бакалаврської роботи ми обрали з кількох причин, серед основних:

- відкритий проект (як відкрите обладнання, так і відкрите програмне забезпечення);

- керуюча електроніка сумісна з Arduino;

- наявність CAD-файлів для перепроектування та вдосконалення деяких частин маніпулятора;

- всі деталі маніпулятора можна виготовити за допомогою 3D-принтера;

- стандартна механіка (підшипники, муфти, гвинти тощо можна отримати в більшості апаратних магазинів, універмагів та інтернет-магазинів).

2.2 Підбір матеріалів для робота SCARA

2.2.1 Електронні частини робота SCARA

На рисунку 2.1 зображена плата Arduino CNC Shield. Дана плата є повністю сумісна з платами Arduino UNO.

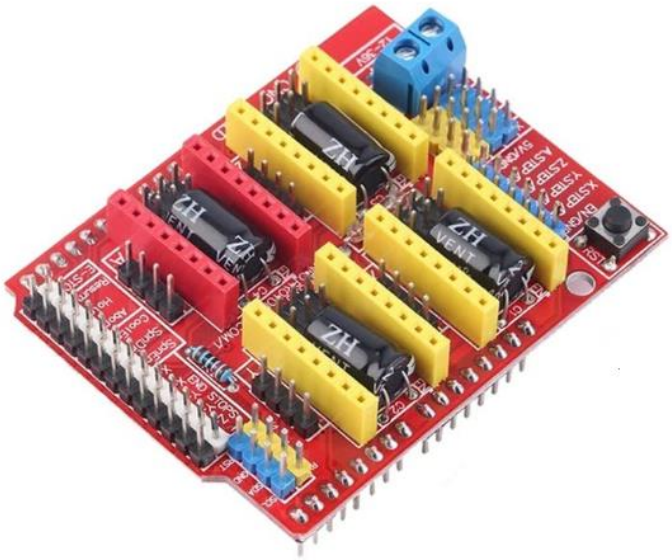


Рис 2.1 - Плата Arduino CNC Shield

Дана плата зазвичай використовується для створення на основі Arduino UNO (та інших контролерів які підходять по типу) машин з числовим програмним управлінням (гравірувальна, фрезерна машина), 3D принтерів та інше.

Плата працює за допомогою програмного забезпечення Arduino GRBL, яке потрібно завантажити (додається посилання на ПЗ).

Програмне забезпечення обмінюється інформацією з процесором за домпомогою G-кодів.

CNC Shield має чотири слоти, по 16 контактів кожен. Для підключення чотирьох крокових двигунів, біля кожного слоту є колодка, яка складається з чотирьох контактів GND, STEP, DIR, VCC. Кожен слот відповідає за свою вісь. Слоти, позначені жовтим, відповідають за вісі X, Y, Z, червоний за вісь A. Вісь A може дублювати одну з осей X, Y, Z.

Вона може бути використана для окремих випадків щоб передавати дані на екструдер (в 3D принтерах).

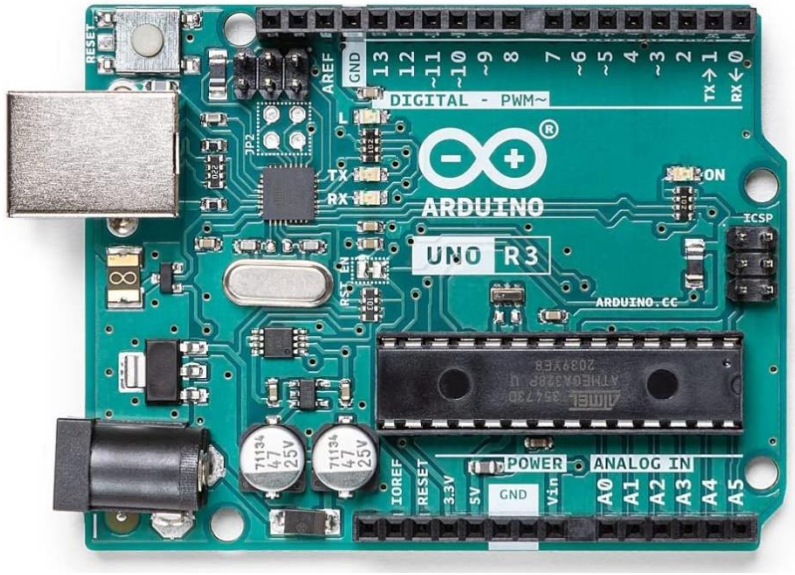


Рис. 2.2 – Плата Arduino UNO

На Рис. 2.2 зображена плата Arduino UNO. Вона являє собою плату мікроконтролера, заснованого на ATmega328. Вона має чотирнадцять цифрових вхідних / вихідних контактів (з яких шість можуть бути використані в якості PWM виходів), шість аналогових входів, шістнадцять керамічних резонаторів mhz, usb-з'єднання, роз'єм живлення, заголовок ICSP, I кнопка скидання.

Вона містить все необхідне для підтримки мікроконтролера; Для розпочання роботи з даною платою потрібно просто підключіть її до комп'ютера за допомогою usb-кабелю або включити її в адапторо AC-to-DC або батареєю, щоб почати роботу.

Рис. 2.3 – Сервопривід

Сервопривід - це пристрій в системах автоматичного регулювання або дистанційного керування, що за рахунок енергії допоміжного джерела здійснює механічне переміщення регулюючого органу відповідно до отримуваних від системи керування сигналів. Тобто, міняється положення регулюючого органа (важеля, кнопки, перемикача) — потік матеріалу або енергії, що поступає на об'єкт дії, міняється і в результаті виконується дія на робочі машини або механізми, змінюється стан робочого об'єкта.

Сервопривід працює від імпульсів змінної тривалості, які отримує через сигнальний дріт. Коли тривалість імпульсів становить близько 1,5 мілісекунди, то сервопривід перебуває в нейтральному положенні (тобто у нього однаковий потенціал обертання в обидва напрями). Кут повороту сервоприводу залежить від тривалості імпульсу. Чим триваліший імпульс, тим швидше працює двигун. Головні частини сервоприводу — це його двигун, елементи керування і передача. Крім того, в ньому є також дрібніші периферійні пристрої — блокування, сигналізація, система включення/виключення, елементи зворотного зв'язку. Як правило, сервоприводи можуть працювати, на відміну від систем сельсин/давач —

сельсин/приймач, тільки від зовнішніх джерел енергії, оскільки потужності внутрішніх джерел енергії недостатньо для ефективного функціонування сервоприводу (дуже вже енергоємну роботу йому доводиться виконувати).

Рисунок 2.4 – Крокoвий двигун NEMA 17

Крокoвий двигун — електричний двигун, в якому імпульсне живлення електричним струмом призводить до того, що його ротор не обертається неперервно, а виконує щоразу обертальний рух на заданий кут. Завдяки цьому, кут повороту ротора залежить від числа поданих імпульсів струму, а кутова швидкість ротора точно рівна частоті імпульсів помноженій на кут повороту ротора за один цикл роботи двигуна. Кут повороту двигуна під впливом одного імпульсу може мати різні значення, залежні від конструкції двигуна, — як правило це значення в діапазоні від декількох градусів до декілька десятків градусів. Крокові двигуни, залежно від призначення пристосовані до виконання від частки оберту на хвилину, до декількох тисяч обертів на хвилину.

2.3 Електронна реалізація системи керування

Для керування роботом була використана плата Arduino UNO та Arduino CNC Shield, з драйверами A4988. Загальна електронна схема приведена на рис. 2.5.

Ця електроніка схожа на таку, що реалізована у 3D-принтерах з відкритим кодом.

Після встановлення електроніки, залишається вибрати програмне забезпечення для керування роботом за допомогою комп'ютера. Є декілька альтернатив, відповідно до наших потреб, можливостей та знань. В якості програмного забезпечення для керування роботом може бути використане програмне забезпечення Arduino IDE.

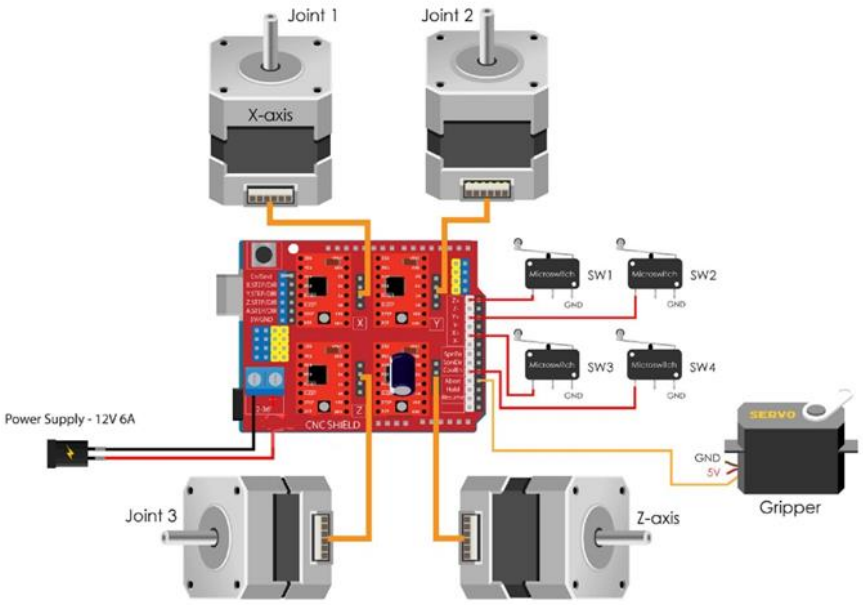


Рис. 2.5 Загальна електронна схема робота

Arduino IDE - апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки Processing/Wiring на мові програмування, що є спрощеною підмножиною C/C++.

Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: Processing, Adobe Flash, Max/MSP, Pure Data, SuperCollider).

2.4 Кінематична функціональна схема маніпулятора

Робот SCARA, об’єкт дипломного проєктування, в загальному розумінні є машиною. Машина здійснює свій робочий процес за допомогою виконання закономірних механічних рухів, носієм цих рухів є механізм. У будь-якому механізмі є одна нерухома ланка (стійка) і одна або декілька ланок, що рухаються. Робот-маніпулятор – це просторовий механізм. Просторовими називаються такі механізми, точки деталей яких описують неплоскі траєкторії чи траєкторії, що лежать у площинах, які перетинаються. Кінематична функціональна схема розробленого робота-маніпулятора подана у графічній частині проєкту. Механізм маніпулятора є системою твердих тіл. Сукупність твердих тіл (деталей), які у складі механізму рухається як одне ціле, називається ланкою. Нерухома ланка механізму – основа О, ланка 1 обертається навколо своєї осі, ланки 2,3 роблять неповний оберт в одній площині відносно точок А і В. Рухоме з'єднання двох дотичних ланок називається кінематичною парою. Кінематичні пари О , А і В - обертальні, вони обмежують п'ять незалежних рухів, це пари V класу. Клас кінематичної пари визначається числом умов в’язі, що накладаються на відносний рух ланок, які входять у цю пару.

Проаналізуємо характер рухів робочих органів маніпулятора в межах робочої зони, при необхідності переміщення об’єкта з однієї точки в іншу і визначаємо тип системи координат ПР – сферична.

Визначаємо степінь вільності за формулою Соснова -Малишева

$$W = 6n - 5p5 - 4p4 - 3p3 - 2p2 - p1,$$

де n - число рухомих ланок кінематичного ланцюга, n = 4; p5, p4, p3, p2, p1 - число кінематичних пар відповідно I, II, III, IV і V класу, p = 4, $W = 6 \cdot 4 - 5 \cdot 4 = 4$, тобто мінімальна степінь свободи маніпулятора - 4.

На рис. 2.6 зображена кінематична функціональна схема

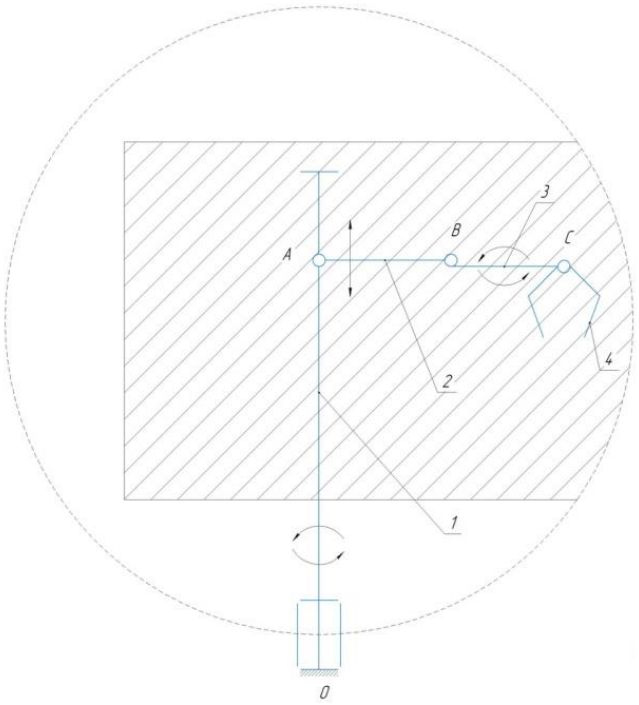


Рис. 2.6 - Кінематична функціональна схема (О - основа; 1,2,3 - ланки маніпулятора; 4 - захоплювальний пристрій; О,А,В,С - кінематичні пари)

2.5 Розробка системи керування роботом

Система керування роботом зібрана базі платформи Arduino UNO. Arduino UNO побудована на мікроконтролері ATmega2560 компанії AVR [1]. Плата має 54 цифрових входів/виходів (14 з яких можуть використовуватися як виходи ШІМ), 16 аналогових входів,4 послідовних порти UART, кварцевий генератор 16 МГц, USB коннектор, роз'єм живлення, роз'єм ICSP і кнопку скидання. Для роботи платформа підключається до комп'ютера за допомогою кабеля USB. Живлення здійснюється від основного блока живлення через понижуючий адаптер DC/DC (24 В/12 В). Плата розширення не використовується. Драйвери двигунів під’єднуються безпосередньо до виходів плати мікроконтролера.

- Коротка технічна характеристика мікроконтролера ATmega2560:
- робоча напруга 5В; вхідна напруга (рекомендована) 7-12В;
- вхідна напруга (гранична) 6-20 В;

- цифрові Входи/Виходи — 54 шт. (14 з яких можуть працювати як виходи ШІМ);
- аналогові входи — 16 шт.;
- постійний струм через вхід/вихід 40 мА; флеш-пам'ять 256 KB (з яких 8 KB використовуються для завантажувача);
- оперативна пам’ять 8 KB; енергонезалежна пам'ять 4 KB;
- тактова частота 16 МГц.

Для складання керуючої програми було використано середу розробки Arduino IDE. Середовище розробки Arduino придставляє з себе багатоплатформовий додаток на Java, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату.

Середовище розробки засноване на мові програмування Processing та спроектоване для програмування новачками, не знайомими близько з розробкою програмного забезпечення. Мова програмування аналогічна мові Wiring. Загалом, це C++, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою AVR-GCC. Програми Arduino пишуться на мові програмування C або C++.

2.5.1 Розробка керуючої програми для робота SCARA

Для розробки керуючої програми скористаємося середовищ розробки Arduino IDE та Processing IDE (Рис.2.7).

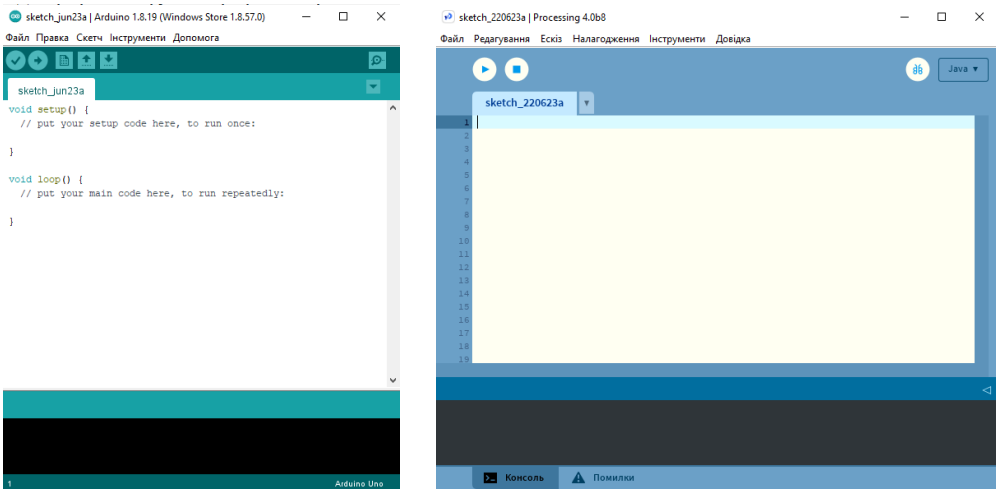


Рис. 2.7 – Середовище розробки Arduino IDE та Processing

Для початку задамо рівняння для обчислення значення X і Y кінцевого ефектора, відповідно до заданих кутів з'єднання та довжин в IDE Processing (Рис. 2.8).

```
// FORWARD KINEMATICS
void forwardKinematics() {
  float theta1F = theta1 * PI / 180;    // degrees to radians
  float theta2F = theta2 * PI / 180;
  xP = round(L1 * cos(theta1F) + L2 * cos(theta1F + theta2F));
  yP = round(L1 * sin(theta1F) + L2 * sin(theta1F + theta2F));
}
```

Рис. 2.8 - Рівняння для обчислення значення X і Y кінцевого ефектора

Наступним кром є застосування оберненої кінематики для обчислення спільних кутів, theta2 і theta1, відповідно до заданого положення або координат X і Y (Рис. 2.9).

```
/ INVERSE KINEMATICS
void inverseKinematics(float x, float y) {
  theta2 = acos((sq(x) + sq(y) - sq(L1) - sq(L2)) / (2 * L1 * L2));
  if (x < 0 & y < 0) {
    theta2 = (-1) * theta2;
  }

  theta1 = atan(x / y) - atan((L2 * sin(theta2)) / (L1 + L2 * cos(theta2)));

  theta2 = (-1) * theta2 * 180 / PI;
  theta1 = theta1 * 180 / PI;

 // Angles adjustment depending in which quadrant the final tool coordinate x,y is
  if (x >= 0 & y >= 0) {       // 1st quadrant
    theta1 = 90 - theta1;
  }
  if (x < 0 & y > 0) {       // 2nd quadrant
    theta1 = 90 - theta1;
  }
  if (x < 0 & y < 0) {       // 3d quadrant
    theta1 = 270 - theta1;
    phi = 270 - theta1 - theta2;
    phi = (-1) * phi;
  }
  if (x > 0 & y < 0) {       // 4th quadrant
    theta1 = -90 - theta1;
  }
  if (x < 0 & y == 0) {
    theta1 = 270 + theta1;
  }

  // Calculate "phi" angle so gripper is parallel to the X axis
  phi = 90 + theta1 + theta2;
  phi = (-1) * phi;

  // Angle adjustment depending in which quadrant the final tool coordinate x,y is
  if (x < 0 & y < 0) {       // 3d quadrant
    phi = 270 - theta1 - theta2;
  }
  if (abs(phi) > 165) {
    phi = 180 + phi;
  }

  theta1=round(theta1);
  theta2=round(theta2);
  phi=round(phi);

  cp5.getController("j1Slider").setValue(theta1);
  cp5.getController("j2Slider").setValue(theta2);
  cp5.getController("j3Slider").setValue(phi);
  cp5.getController("zSlider").setValue(zP);
}
```

Рис. 2.9 - Обернета кінематики для обчислення спільних кутів

Залежно від того, в якому квадранті встановлено положення, ми вносимо деякі коригування кутів з’єднання за допомогою цих операторів «якщо». Для цієї конфігурації робота ми фактично обчислюємо зворотну кінематику лише з двома ланками. Третій кут, який було називано «phi», використовується для встановлення орієнтації захвату.

Наступним кроком є створення графічного інтерфейсу користувача за допомогою бібліотеки controlP5 для IDE Processing (Рис 2.10). За допомогою цієї бібліотеки ми можемо легко створювати кнопки, повзунки, текстові поля тощо.

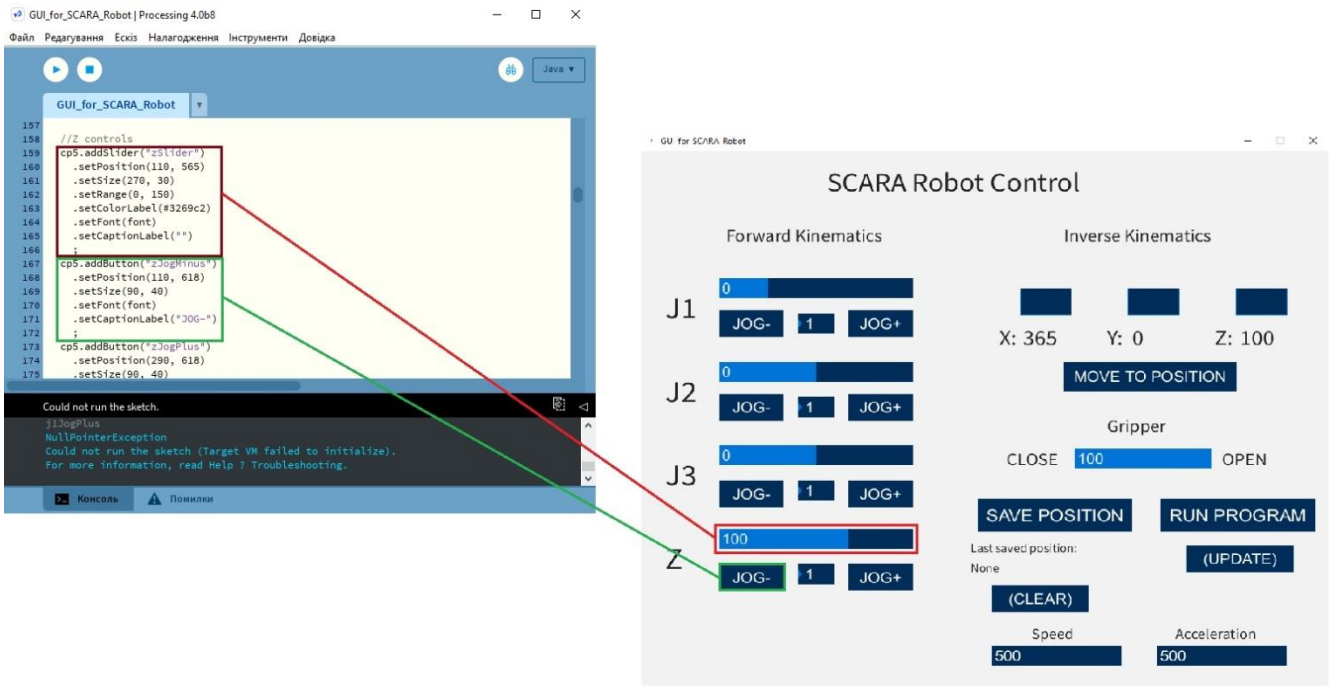


Рис. 2.10 - Графічний інтерфейс користувача

Наприклад, ми використовуємо повзунки з лівого боку, щоб керувати кутами з'єднання, і за допомогою текстових полів ми можемо ввести позицію, куди ми хочемо, щоб наш робот перейшов.

З кожною дією, яку ми виконуємо тут із програмою, ми надсилаємо дані на плату Arduino через послідовний порт (Рис. 2.11).

```
if (gripperValuePrevious != gripperValue) {
    if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Exe
      gripperAdd = round(cp5.getController("gripperValue").getValue());
      gripperValue=gripperAdd+50;
      updateData();
      println(data);
      myPort.write(data);
    }
  }
```

Рис. 2.11 - Надсилання даних на плату Arduino через послідовний порт

Ці дані включають кути з'єднання, значення захвату, значення швидкості та прискорення, а також показники, які дають змогу дізнатися, чи ми натиснули кнопку «Зберегти» чи «Виконати»(Рис. 2.12).

```
public void updateData() {
  data = str(saveStatus)
    +","+str(runStatus)
    +","+str(round(cp5.getController("j1Slider").getValue()))
    +","+str(round(cp5.getController("j2Slider").getValue()))
    +","+str(round(cp5.getController("j3Slider").getValue()))
    +","+str(round(cp5.getController("zSlider").getValue()))
    +","+str(gripperValue)
    +","+str(speedSlider)
    +","+str(accelerationSlider);
}
```

Рис. 2.12 - Фунція первірки натиснули кнопки «Зберегти» чи «Виконати»

Всі ці дані надходять у вигляді одного довгого рядка в Arduino. Отже, спочатку нам потрібно витягти дані з цього рядка і помістити їх в окремі змінні(Рис. 2.13).

```
if (Serial.available()) {
    content = Serial.readString(); // Read the incomding data from Processing
    // Extract the data from the string and put into separate integer variables (data[] arr
    for (int i = 0; i < 10; i++) {
      int index = content.indexOf(","); // Locate the first ","
      data[i] = atol(content.substring(0, index).c_str()); //Extract the number from start
      content = content.substring(index + 1); //Remove the number from the string
    }
    /*
     data[0] - SAVE button status
     data[1] - RUN button status
     data[2] - Joint 1 angle
     data[3] - Joint 2 angle
     data[4] - Joint 3 angle
     data[5] - Z position
     data[6] - Gripper value
     data[7] - Speed value
     data[8] - Acceleration value
    */
```

Рис. 2.13 – Перетворення даних та переміщення їх в окремі змінні

Тепер з цими змінними ми можемо виконувати дії з роботом. Наприклад, якщо ми натиснемо кнопку ЗБЕРЕГТИ, ми збережемо поточні значення кутів з'єднання в окремому масиві (Рис. 2.14).

```
// If SAVE button is pressed, store the data into the appropriate arrays
    if (data[0] == 1) {
      theta1Array[positionsCounter] = data[2] * theta1AngleToSteps; //store the values in
      theta2Array[positionsCounter] = data[3] * theta2AngleToSteps;
      phiArray[positionsCounter] = data[4] * phiAngleToSteps;
      zArray[positionsCounter] = data[5] * zDistanceToSteps;
      gripperArray[positionsCounter] = data[6];
      positionsCounter++;
    }
```

Рис. 2.14 – Приклад використання кнопки Save в графічному інтерфейсі

Якщо ми натиснемо кнопку RUN, ми виконаємо збережені кроки тощо.

Для керування крокoвими двигунами було використано бібліотеку AccelStepper . Хоча це чудова бібліотека для одночасного керування кількома степерами, вона має деякі обмеження, коли справа доходить до керування таким роботом. При керуванні кількома степерами бібліотека не може реалізувати прискорення та уповільнення, які важливі для більш плавної роботи робота.

Код керуючої програми в середовищах розробки Arduino IDE та Processing IDE буде поданий в додатках.

ВИСНОВОК

В процесі роботи над бакалаврською роботою було спроєктвано антропоморфного робота SCARA. Він є одним з найкращих рішень для навчального процесу, та дозволяє забезпечити студентам вивчення багатьох аспектів конструювання промислових роботів і систем керування для них, а також отримання практичних навиків в процесі їх налагодження та розробки програмного забезпечення.

В дані бакалаврській роботі було досліджено основні поняття промислових роботів та їх структура, класифікація. Був проведений аналіз основних видів промислових маніпуляторів за типом їх систем та аналіз основних типів приводів і систем керування маніпуляторів промислових роботів. Також був вивчений опис сервоприводів та крокових двигунів. Була розглянута сервомашинка в системах керування на базі Arduino. Досліджено використання крокових двигунів в системах керування на базі Arduino, а також було розглянуто програмне керування робота маніпулятора.

Проведено підбір матеріалів та електронних частин для робота SCARA, та досліджено електронна реалізація системи керування робота. Також був проведений розрахунок кінематично функціональної схеми маніпулятора та проведена розробка керуючої програми для антропоморфного робота SCARA.

Навички, які були здобуті в процесі написання бакалаврської роботи, є незамінними для моєї реалізації як спеціаліста в сфері “Інженерії мехатронних систем” в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abdellatif Baba, Aimn Mohamed Ahmed Ghiet, Robot Arm Control With Arduino. — Spring, 2017. — 42 p. — Електронний ресурс. Режим доступу: https://www.researchgate.net/profile/Abdellatif_Baba/publication/317277584_ROBOT_ARM_CONTROL_WITH_ARDUINO/links/592fd2b045851553b67ed7a4/ROBOT-ARM-CONTROL-WITH-ARDUINO.pdf
2. Abu Qassem Mohammed, Abuhadrous I.M., Elaydi H.. Modeling and Simulation of 5 DOF educational robot arm / 2nd International Conference on Advanced Computer Control (ICACC 2010) (27-29 March 2010). — IEEE, 2010. — Volume 5. — DOI: 10.1109/ICACC.2010.5487136 . — Електронний ресурс. Режим доступу: https://www.researchgate.net/publication/224146402_Modeling_and_Simulation_of_5_DOF_educational_robot_arm
3. Cao Yang. Learning Robotics through Developing A Virtual Robot Simulator in Matlab — American Society for Engineering Education, 2011. — 13 pages. — Електронний ресурс. Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwiciLfU6bPoAhWNvosKHQlJC6EQFjAAegQIBRAB&url=https%3A%2F%2Fpeer.asee.org%2Flearning-robotics-through-developing-a-virtualrobot-simulator-in-matlab.pdf&usg=AOvVaw3B3F8sFlgyGeWvoMrvFV_A
4. Clothier Kurt E., Shang Ying. A Geometric Approach for Robotic Arm Kinematics with Hardware Design, Electrical Design, and Implementation / Journal of Robotics — Hindawi Publishing Corporation, 2010. Volume 2010, Article ID 984823, 10 pages. doi:10.1155/2010/984823. — Електронний ресурс. Режим доступу: https://www.researchgate.net/publication/47697116_A_Geometric_Approach_for_Robotic_Arm_Kinematics_with_Hardware_Design_Electrical_Design_and_Implementation/fulltext/0ffc83a70cf255165fc9326d/A-Geometric-Approachfor-Robotic-Arm-Kinematics-with-Hardware-Design-Electrical-Design-andImplementation.pdf
5. Duicu Simona Sofia, Popa Luminita Modeling and simulation of the 5-axis robot arm trajectory / Advances in Applied Information Science.— 2012. — Pp. 154–159. — Електронний ресурс. Режим доступу: https://pdfs.semanticscholar.org/9d22/aaa3c8ad417bb20cfbf1c75731eb55fdee14.pdf?_ga=2.251518662.1152542547.1585078014-464915163.1585078014

6. Frankovskýa P., Hroncováa D., Delyováa I., Hudákb P. Inverse and forward dynamic analysis of two link manipulator. / Procedia Engineering — Published by Elsevier Ltd.Selection, 2012. — №48. — P. 158 – 163. — Електронний ресурс. Режим доступу: https://www.sciencedirect.com/science/article/pii/S1877705812045638
7. Hemmelder Floris, Vinel Remi, Opris Andrei, Mosseng Joachim. 3D Printed articulated arm robot with IoT capabilities. — Universitatea politehnica din 74 Bucureşti, 2018. — Електронний ресурс. Режим доступу: https://upb.ro/wpcontent/uploads/2017/11/Final-report-robotic-arm.pdf
8. Houcque David. Introduction to Matlab for Engineering Students. — August 2005. — 74 p. — Електронний ресурс. Режим доступу: https://www.mccormick.northwestern.edu/documents/students/undergraduate/iintroductio-to-matlab.pdf
9. Jazar N. Reza. Theory of Applied Robotics: Kinematics, Dynamics, and Control. —New York: Springer Science+Business Media, LLC. — 2007. — 688 p. — Електронний ресурс. Режим доступу: https://eleccompengineering.files.wordpress.com/2015/03/reza_n-_jazar_theory_of_applied_robotics.pdf
10. Moe Thu Zar, Wai Phyo Ei. Point to Point Trajectory Control of 3R Planar Robot Arm / International Journal of Science, Engineering and Technology Research (IJSETR) — November 2017, Volume 6, Issue 11, ISSN: 2278 - 7798. — 1435-1441 p. — Електронний ресурс. Режим доступу: http://ijsetr.org/wp-content/uploads/2017/11/IJSETR-VOL-6-ISSUE-11-1435- 1441.pdf
11. Moore Holly. MATLAB for Engineers. — Pearson, 2011. — 732 p. — Електронний ресурс. Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=2ahUKEwjO1u-4-rPoAhXF4aYKHXvnCNgQFjABegQIAhAB&url=https%3A%2F%2Fwww.researchgate.net%2Fprofile%2FMohamed_Mourad_Lafifi%2Fpost%2FCan_anyone_help_me_to_write_mfile_for_optimization_of_Heat_Exchanger_with_GA%2Fattachment%2F59d6445c79197b807799fa82%2FAS%3A448306506670080%401483896008389%2Fdownload%2FMATLAB%2Bfor%2BEngineers.pdf&usg=AOvVaw2DbDnXoq_TZXnCwzIuJzEh
12. Murray M. Richard, Li Zexiang, Sastry S. Shankar. A Mathematical Introduction to Robotic Manipulation. — CRC Press, 1994. 474 p. — Електронний ресурс. Режим доступу: https://www.cds.caltech.edu/~murray/books/MLS/pdf/mls94-complete.pdf

13. Ruthber Rodríguez Serrezuela, Adrián Fernando Chávarro Chavarro, Miguel Angel Tovar Cardozo, Alejandro Leiva Toquica, Luis Fernando Ortiz Martinez. Kinematic modelling of a robotic arm manipulator using MATLAB. / Journal of Engineering and Applied Sciences — Asian Research Publishing Network (ARPN). — APRIL 2017. — VOL. 12, NO. 7, 2037–2045 p. . — Електронний ресурс. Режим доступу: https://www.researchgate.net/profile/Ruthber_Rodriguez/publication/316663146_Kinematic_modelling_of_a_robotic_arm_manipulator_using_MATLAB/links/590aad6e458515ebb4a542ad/Kinematic-modelling-of-a-robotic-armmanipulator-using-MATLAB.pdf
14. Spong Mark W., Hutchinson Seth, Vidyasagar M. Robot Dynamics and Control. — 2004. — 303 p. — Електронний ресурс. Режим доступу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=2ahUKEwi--8GK8bPoAhUhlIsKHRsSB9kQFjADegQIAhAB&url=https%3A%2F%2Fwww.researchgate.net%2Fprofile%2FMohamed_Mourad_Lafifi%2Fpost%2FWhat_simple_mechanism_will_make_a_mechanical_arm_system_have_a_dual_axis_movement_X_Y%2Fattachment%2F5b6f5ae6cfe4a7f7ca59de40%2FAS%253A658560312696832%2540153402442226 6%2Fdownload%2FRobot%2BDynamics%2Band%2BControl.pdf&usg=AOvVaw2JpJXdGPnuu3yCRiQempRE 75
15. William J. Palm III. Introduction to MATLAB for Engineers. — McGraw-Hill, 2011. — 577 p. — Електронний ресурс. Режим доступу: http://www.cse.cuhk.edu.hk/~cslui/CSCI1050/book.pdf
16. Введение в мехатронику: Уч. пособие / Грабченко А.И., Клепиков В.Б., Доброскок В.Л. и др. — Х.: НТУ "ХПИ", 2014. — 274 с. — Електронний ресурс. Режим доступу: https://fileskachat.com/getfile/27923_11551bf13daf484df4415a402f4e3410
17. Зенкевич С.Л., Ющенко А.С. Основы управления манипуляционными роботами: Учебник для вузов. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2004. — 480 с. — Електронний ресурс. Режим доступу: https://docplayer.ru/27591206-Osnovy-upravleniya-manipulyacionnymirobotami.html#download_tab_content
18. Климчик А.С., Гомолицкий Р.И., Фурман Ф.В., Сѐмкин К.И. Разработка управляющих программ промышленных роботов. Курс лекций. — Минск, 2008. — 131 с. — Електронний ресурс. Режим доступу: https://docplayer.ru/101149-Razrabotka-upravlyayushchih-programmpromyshlennyh-robotov.html#download_tab_content

19. Фу К., Гонсалес Р., Ли К. Робототехника. — М.: Мир, 1989. — 624 с. Електронний ресурс. Режим доступу: https://www.twirpx.com/file/2817815/
20. Шаньгин Е.С. Управление роботами и робототехническими системами. Конспект лекций. — Уфа, 2005. — Електронний ресурс. Режим доступу: https://scicenter.online/avtomatizatsiya-knigi-scicenter/upravlenie-robotamirobototehnicheskimi.html
21. Шахинпур М. Курс робототехники. — М.: Мир, 1990. — 527 с. — Електронний ресурс. Режим доступу: https://sheba.spb.ru/delo/kursrobotehniki-1990.djvu

Додаток А

```
#include <AccelStepper.h>
#include <Servo.h>
#include <math.h>

#define limitSwitch1 11
#define limitSwitch2 10
#define limitSwitch3 9
#define limitSwitch4 A3

// Define the stepper motors and the pins the will use
AccelStepper stepper1(1, 2, 5); // (Type:driver, STEP, DIR)
AccelStepper stepper2(1, 3, 6);
AccelStepper stepper3(1, 4, 7);
AccelStepper stepper4(1, 12, 13);

Servo gripperServo;  // create servo object to control a servo


double x = 10.0;
double y = 10.0;
double L1 = 228; // L1 = 228mm
double L2 = 136.5; // L2 = 136.5mm
double theta1, theta2, phi, z;

int stepper1Position, stepper2Position, stepper3Position, stepper4Position;

const float theta1AngleToSteps = 44.444444;
const float theta2AngleToSteps = 35.555555;
const float phiAngleToSteps = 10;
const float zDistanceToSteps = 100;

byte inputValue[5];
int k = 0;

String content = "";
```

```
int data[10];

int theta1Array[100];
int theta2Array[100];
int phiArray[100];
int zArray[100];
int gripperArray[100];
int positionsCounter = 0;

void setup() {
  Serial.begin(115200);

  pinMode(limitSwitch1, INPUT_PULLUP);
  pinMode(limitSwitch2, INPUT_PULLUP);
  pinMode(limitSwitch3, INPUT_PULLUP);
  pinMode(limitSwitch4, INPUT_PULLUP);

  // Stepper motors max speed
  stepper1.setMaxSpeed(4000);
  stepper1.setAcceleration(2000);
  stepper2.setMaxSpeed(4000);
  stepper2.setAcceleration(2000);
  stepper3.setMaxSpeed(4000);
  stepper3.setAcceleration(2000);
  stepper4.setMaxSpeed(4000);
  stepper4.setAcceleration(2000);

  gripperServo.attach(A0, 600, 2500);
  // initial servo value - open gripper
  data[6] = 180;
  gripperServo.write(data[6]);
  delay(1000);
  data[5] = 100;
  homing();
}

void loop() {
```

```
 if (Serial.available()) {
  content = Serial.readString(); // Read the incomding data from Processing
  // Extract the data from the string and put into separate integer variables (data[] array)
  for (int i = 0; i < 10; i++) {
   int index = content.indexOf(","); // locate the first ","
   data[i] = atol(content.substring(0, index).c_str()); //Extract the number from start to the ","
   content = content.substring(index + 1); //Remove the number from the string
  }
  /*
   data[0] - SAVE button status
   data[1] - RUN button status
   data[2] - Joint 1 angle
   data[3] - Joint 2 angle
   data[4] - Joint 3 angle
   data[5] - Z position
   data[6] - Gripper value
   data[7] - Speed value
   data[8] - Acceleration value
  */
  // If SAVE button is pressed, store the data into the appropriate arrays
  if (data[0] == 1) {
   theta1Array[positionsCounter] = data[2] * theta1AngleToSteps; //store the values in steps = angles * angleToSteps variable
   theta2Array[positionsCounter] = data[3] * theta2AngleToSteps;
   phiArray[positionsCounter] = data[4] * phiAngleToSteps;
   zArray[positionsCounter] = data[5] * zDistanceToSteps;
   gripperArray[positionsCounter] = data[6];
   positionsCounter++;
  }
  // clear data
  if (data[0] == 2) {
   // Clear the array data to 0
   memset(theta1Array, 0, sizeof(theta1Array));
   memset(theta2Array, 0, sizeof(theta2Array));
   memset(phiArray, 0, sizeof(phiArray));
   memset(zArray, 0, sizeof(zArray));
```

```
      memset(gripperArray, 0, sizeof(gripperArray));
      positionsCounter = 0;
    }
  }
  // If RUN button is pressed
  while (data[1] == 1) {
    stepper1.setSpeed(data[7]);
    stepper2.setSpeed(data[7]);
    stepper3.setSpeed(data[7]);
    stepper4.setSpeed(data[7]);
    stepper1.setAcceleration(data[8]);
    stepper2.setAcceleration(data[8]);
    stepper3.setAcceleration(data[8]);
    stepper4.setAcceleration(data[8]);

    // execute the stored steps
    for (int i = 0; i <= positionsCounter - 1; i++) {
      if (data[1] == 0) {
        break;
      }
      stepper1.moveTo(theta1Array[i]);
      stepper2.moveTo(theta2Array[i]);
      stepper3.moveTo(phiArray[i]);
      stepper4.moveTo(zArray[i]);
      while (stepper1.currentPosition() != theta1Array[i] || stepper2.currentPosition()
!= theta2Array[i] || stepper3.currentPosition() != phiArray[i] ||
stepper4.currentPosition() != zArray[i]) {
        stepper1.run();
        stepper2.run();
        stepper3.run();
        stepper4.run();
      }
      if (i == 0) {
        gripperServo.write(gripperArray[i]);
      }
      else if (gripperArray[i] != gripperArray[i - 1]) {
        gripperServo.write(gripperArray[i]);
        delay(800); // wait 0.8s for the servo to grab or drop - the servo is slow
```

```
    }

    //check for change in speed and acceleration or program stop
    if (Serial.available()) {
      content = Serial.readString(); // Read the incomding data from Processing
      // Extract the data from the string and put into separate integer variables (data[] array)
      for (int i = 0; i < 10; i++) {
        int index = content.indexOf(","); // locate the first ","
        data[i] = atol(content.substring(0, index).c_str()); //Extract the number from start to the ","
        content = content.substring(index + 1); //Remove the number from the string
      }

      if (data[1] == 0) {
        break;
      }
      // change speed and acceleration while running the program
      stepper1.setSpeed(data[7]);
      stepper2.setSpeed(data[7]);
      stepper3.setSpeed(data[7]);
      stepper4.setSpeed(data[7]);
      stepper1.setAcceleration(data[8]);
      stepper2.setAcceleration(data[8]);
      stepper3.setAcceleration(data[8]);
      stepper4.setAcceleration(data[8]);
    }
  }
  /*
    // execute the stored steps in reverse
    for (int i = positionsCounter - 2; i >= 0; i--) {
    if (data[1] == 0) {
      break;
    }
    stepper1.moveTo(theta1Array[i]);
    stepper2.moveTo(theta2Array[i]);
    stepper3.moveTo(phiArray[i]);
    stepper4.moveTo(zArray[i]);
```

```
    while (stepper1.currentPosition() != theta1Array[i] || stepper2.currentPosition() != theta2Array[i] || stepper3.currentPosition() != phiArray[i] || stepper4.currentPosition() != zArray[i]) {
      stepper1.run();
      stepper2.run();
      stepper3.run();
      stepper4.run();
    }
    gripperServo.write(gripperArray[i]);

    if (Serial.available()) {
      content = Serial.readString(); // Read the incomding data from Processing
      // Extract the data from the string and put into separate integer variables (data[] array)
      for (int i = 0; i < 10; i++) {
        int index = content.indexOf(","); // locate the first ","
        data[i] = atol(content.substring(0, index).c_str()); //Extract the number from start to the ","
        content = content.substring(index + 1); //Remove the number from the string
      }
      if (data[1] == 0) {
        break;
      }
    }
    }
  */
 }

  stepper1Position = data[2] * theta1AngleToSteps;
  stepper2Position = data[3] * theta2AngleToSteps;
  stepper3Position = data[4] * phiAngleToSteps;
  stepper4Position = data[5] * zDistanceToSteps;

  stepper1.setSpeed(data[7]);
  stepper2.setSpeed(data[7]);
  stepper3.setSpeed(data[7]);
  stepper4.setSpeed(data[7]);
```

```
  stepper1.setAcceleration(data[8]);
  stepper2.setAcceleration(data[8]);
  stepper3.setAcceleration(data[8]);
  stepper4.setAcceleration(data[8]);

  stepper1.moveTo(stepper1Position);
  stepper2.moveTo(stepper2Position);
  stepper3.moveTo(stepper3Position);
  stepper4.moveTo(stepper4Position);

  while (stepper1.currentPosition() != stepper1Position || stepper2.currentPosition() != stepper2Position || stepper3.currentPosition() != stepper3Position || stepper4.currentPosition() != stepper4Position) {
    stepper1.run();
    stepper2.run();
    stepper3.run();
    stepper4.run();
  }
  delay(100);
  gripperServo.write(data[6]);
  delay(300);
}

void serialFlush() {
  while (Serial.available() > 0) {  //while there are characters in the serial buffer, because Serial.available is >0
    Serial.read();        // get one character
  }
}

void homing() {
  // Homing Stepper4
  while (digitalRead(limitSwitch4) != 1) {
    stepper4.setSpeed(1500);
    stepper4.runSpeed();
    stepper4.setCurrentPosition(17000); // When limit switch pressed set position to 0 steps
  }
```

```
  delay(20);
  stepper4.moveTo(10000);
  while (stepper4.currentPosition() != 10000) {
    stepper4.run();
  }

  // Homing Stepper3
  while (digitalRead(limitSwitch3) != 1) {
    stepper3.setSpeed(-1100);
    stepper3.runSpeed();
    stepper3.setCurrentPosition(-1662); // When limit switch pressed set position to
0 steps
  }
  delay(20);

  stepper3.moveTo(0);
  while (stepper3.currentPosition() != 0) {
    stepper3.run();
  }

  // Homing Stepper2
  while (digitalRead(limitSwitch2) != 1) {
    stepper2.setSpeed(-1300);
    stepper2.runSpeed();
    stepper2.setCurrentPosition(-5420); // When limit switch pressed set position to
-5440 steps
  }
  delay(20);

  stepper2.moveTo(0);
  while (stepper2.currentPosition() != 0) {
    stepper2.run();
  }

  // Homing Stepper1
  while (digitalRead(limitSwitch1) != 1) {
    stepper1.setSpeed(-1200);
    stepper1.runSpeed();
```

```
    stepper1.setCurrentPosition(-3955); // When limit switch pressed set position to
0 steps
  }
  delay(20);
  stepper1.moveTo(0);
  while (stepper1.currentPosition() != 0) {
    stepper1.run();
  }
}
```

Додаток Б

```
import processing.serial.*;
import controlP5.*;
import static processing.core.PApplet.*;

Serial myPort;
ControlP5 cp5; // controlP5 object

int j1Slider = 0;
int j2Slider = 0;
int j3Slider = 0;
int zSlider = 100;
int j1JogValue = 0;
int j2JogValue = 0;
int j3JogValue = 0;
int zJogValue = 0;
int speedSlider = 500;
int accelerationSlider = 500;
int gripperValue = 180;
int gripperAdd=180;
int positionsCounter = 0;


int saveStatus = 0;
int runStatus = 0;

int slider1Previous = 0;
```

```
int slider2Previous = 0;
int slider3Previous = 0;
int sliderzPrevious = 100;
int speedSliderPrevious = 500;
int accelerationSliderPrevious = 500;
int gripperValuePrevious = 100;

boolean activeIK = false;

int xP=365;
int yP=0;
int zP=100;
float L1 = 228; // L1 = 228mm
float L2 = 136.5; // L2 = 136.5mm
float theta1, theta2, phi, z;

String[] positions = new String[100];

String data;

void setup() {

  size(960, 800);
  //myPort = new Serial(this, "COM3", 115200);

  cp5 = new ControlP5(this);
```

```
PFont pfont = createFont("Arial", 25, true); // use true/false for smooth/no-smooth
ControlFont font = new ControlFont(pfont, 22);
ControlFont font2 = new ControlFont(pfont, 25);


//J1 controls
cp5.addSlider("j1Slider")
  .setPosition(110, 190)
  .setSize(270, 30)
  .setRange(-90, 266) // Slider range, corresponds to Joint 1 or theta1 angle that the robot can move to
  .setColorLabel(#3269c2)
  .setFont(font)
  .setCaptionLabel("")
  ;
cp5.addButton("j1JogMinus")
  .setPosition(110, 238)
  .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG-")
  ;
cp5.addButton("j1JogPlus")
  .setPosition(290, 238)
  .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG+")
  ;
cp5.addNumberbox("j1JogValue")
  .setPosition(220, 243)
```

```
            .setSize(50, 30)
            .setRange(0, 20)
            .setFont(font)
            .setMultiplier(0.1)
            .setValue(1)
            .setDirection(Controller.HORIZONTAL) // change the control direction to
left/right
            .setCaptionLabel("")
            ;

          //J2 controls
          cp5.addSlider("j2Slider")
            .setPosition(110, 315)
            .setSize(270, 30)
            .setRange(-150, 150)
            .setColorLabel(#3269c2)
            .setFont(font)
            .setCaptionLabel("")
            ;
          cp5.addButton("j2JogMinus")
            .setPosition(110, 363)
            .setSize(90, 40)
            .setFont(font)
            .setCaptionLabel("JOG-")
            ;
          cp5.addButton("j2JogPlus")
            .setPosition(290, 363)
            .setSize(90, 40)
```

```
            .setFont(font)
            .setCaptionLabel("JOG+")
            ;
          cp5.addNumberbox("j2JogValue")
            .setPosition(220, 368)
            .setSize(50, 30)
            .setRange(0, 20)
            .setFont(font)
            .setMultiplier(0.1)
            .setValue(1)
            .setDirection(Controller.HORIZONTAL) // change the control direction to
left/right
            .setCaptionLabel("")
            ;
          //J3 controls
          cp5.addSlider("j3Slider")
            .setPosition(110, 440)
            .setSize(270, 30)
            .setRange(-162, 162)
            .setColorLabel(#3269c2)
            .setFont(font)
            .setCaptionLabel("")
            ;
          cp5.addButton("j3JogMinus")
            .setPosition(110, 493)
            .setSize(90, 40)
            .setFont(font)
            .setCaptionLabel("JOG-")
```

```
            ;
          cp5.addButton("j3JogPlus")
            .setPosition(290, 493)
            .setSize(90, 40)
            .setFont(font)
            .setCaptionLabel("JOG+")
            ;
          cp5.addNumberbox("j3JogValue")
            .setPosition(220, 493)
            .setSize(50, 30)
            .setRange(0, 20)
            .setFont(font)
            .setMultiplier(0.1)
            .setValue(1)
            .setDirection(Controller.HORIZONTAL) // change the control direction to
left/right
            .setCaptionLabel("")
            ;


          //Z controls
          cp5.addSlider("zSlider")
            .setPosition(110, 565)
            .setSize(270, 30)
            .setRange(0, 150)
            .setColorLabel(#3269c2)
            .setFont(font)
            .setCaptionLabel("")
            ;
```

```
        cp5.addButton("zJogMinus")
          .setPosition(110, 618)
          .setSize(90, 40)
          .setFont(font)
          .setCaptionLabel("JOG-")
          ;
        cp5.addButton("zJogPlus")
          .setPosition(290, 618)
          .setSize(90, 40)
          .setFont(font)
          .setCaptionLabel("JOG+")
          ;
        cp5.addNumberbox("zJogValue")
          .setPosition(220, 618)
          .setSize(50, 30)
          .setRange(0, 20)
          .setFont(font)
          .setMultiplier(0.1)
          .setValue(1)
          .setDirection(Controller.HORIZONTAL) // change the control direction to
left/right
          .setCaptionLabel("")
          ;



        cp5.addTextfield("xTextfield")
          .setPosition(530, 205)
          .setSize(70, 40)
```

```
  .setFont(font)
  .setColor(255)
  .setCaptionLabel("")
  ;
cp5.addTextfield("yTextfield")
  .setPosition(680, 205)
  .setSize(70, 40)
  .setFont(font)
  .setColor(255)
  .setCaptionLabel("")
  ;
cp5.addTextfield("zTextfield")
  .setPosition(830, 205)
  .setSize(70, 40)
  .setFont(font)
  .setColor(255)
  .setCaptionLabel("")
  ;

cp5.addButton("move")
  .setPosition(590, 315)
  .setSize(240, 45)
  .setFont(font)
  .setCaptionLabel("MOVE TO POSITION")
  ;

cp5.addButton("savePosition")
  .setPosition(470, 520)
```

```
  .setSize(215, 50)
  .setFont(font2)
  .setCaptionLabel("SAVE POSITION")
  ;

cp5.addButton("run")
  .setPosition(725, 520)
  .setSize(215, 50)
  .setFont(font2)
  .setCaptionLabel("RUN PROGRAM")
  ;

cp5.addButton("updateSA")
  .setPosition(760, 590)
  .setSize(150, 40)
  .setFont(font)
  .setCaptionLabel("(Update)")
  ;

cp5.addButton("clearSteps")
  .setPosition(490, 650)
  .setSize(135, 40)
  .setFont(font)
  .setCaptionLabel("(CLEAR)")
  ;

//Z controls
cp5.addSlider("speedSlider")
```

```
    .setPosition(490, 740)
    .setSize(180, 30)
    .setRange(500, 4000)
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("")
    ;

  cp5.addSlider("accelerationSlider")
    .setPosition(720, 740)
    .setSize(180, 30)
    .setRange(500, 4000)
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("")
    ;
  cp5.addSlider("gripperValue")
    .setPosition(605, 445)
    .setSize(190, 30)
    .setRange(0, 100)
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("")
    ;
}

void draw() {
  background(#F2F2F2); // background black
```

```
textSize(26);
fill(33);
text("Forward Kinematics", 120, 135);
text("Inverse Kinematics", 590, 135);
textSize(40);
text("SCARA Robot Control", 260, 60);
textSize(45);
text("J1", 35, 250);
text("J2", 35, 375);
text("J3", 35, 500);
text("Z", 35, 625);
textSize(22);
text("Speed", 545, 730);
text("Acceleration", 745, 730);

//println("PREV: "+accelerationSlider);
fill(speedSlider);
fill(accelerationSlider);
fill(j1Slider);
fill(j2Slider);
fill(j3Slider);
fill(zSlider);
fill(j1JogValue);
fill(j2JogValue);
fill(j3JogValue);
fill(zJogValue);
fill(gripperValue);
```

```
updateData();
//println(data);


saveStatus=0; // keep savePosition variable 0 or false. See, when button SAVE pressed it makes the value 1, which indicates to store the value in the arduino code


// If slider moved, calculate new position of X,Y and Z with forward kinematics
if (slider1Previous != j1Slider) {
  if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre Forward kinematics only if inverseKinematics mode is off or false
    theta1 = round(cp5.getController("j1Slider").getValue()); // get the value from the slider1
    theta2 = round(cp5.getController("j2Slider").getValue());
    forwardKinematics();
    myPort.write(data);
  }
}
slider1Previous = j1Slider;


if (slider2Previous != j2Slider) {
  if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre Forward kinematics only if inverseKinematics mode is off or false
    theta1 = round(cp5.getController("j1Slider").getValue()); // get the value from the slider1
    theta2 = round(cp5.getController("j2Slider").getValue());
    forwardKinematics();
    myPort.write(data);
  }
}
```

```
slider2Previous = j2Slider;


if (slider3Previous != j3Slider) {
  if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre Forward kinematics only if inverseKinematics mode is off or false
    theta1 = round(cp5.getController("j1Slider").getValue()); // get the value from the slider1
    theta2 = round(cp5.getController("j2Slider").getValue());
    forwardKinematics();
    myPort.write(data);
  }
}
slider3Previous = j3Slider;


if (sliderzPrevious != zSlider) {
  if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre Forward kinematics only if inverseKinematics mode is off or false
    zP = round(cp5.getController("zSlider").getValue());
    myPort.write(data);
  }
}
sliderzPrevious = zSlider;


if (gripperValuePrevious != gripperValue) {
  if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre Forward kinematics only if inverseKinematics mode is off or false
    gripperAdd = round(cp5.getController("gripperValue").getValue());
    gripperValue=gripperAdd+50;
    updateData();
```

```
    println(data);
    myPort.write(data);
   }
  }
  gripperValuePrevious = gripperValue;
  activeIK = false; // deactivate inverseKinematics so the above if statements can be executed the next interation


  fill(33);
  textSize(32);
  text("X: ", 500, 290);
  text(xP, 533, 290);
  text("Y: ", 650, 290);
  text(yP, 685, 290);
  text("Z: ", 800, 290);
  text(zP, 835, 290);
  textSize(26);
  text("Gripper", 650, 420);
  text("CLOSE", 510, 470);
  text("OPEN", 810, 470);
  textSize(18);

  if (positionsCounter >0 ) {
   text(positions[positionsCounter-1], 460, 630);
   text("Last saved position: No."+(positionsCounter-1), 460, 600);
  } else {
   text("Last saved position:", 460, 600);
   text("None", 460, 630);
```

```
  }
}

 // FORWARD KINEMATICS
void forwardKinematics() {
  float theta1F = theta1 * PI / 180;   // degrees to radians
  float theta2F = theta2 * PI / 180;
  xP = round(L1 * cos(theta1F) + L2 * cos(theta1F + theta2F));
  yP = round(L1 * sin(theta1F) + L2 * sin(theta1F + theta2F));
}

 // INVERSE KINEMATICS
void inverseKinematics(float x, float y) {
  theta2 = acos((sq(x) + sq(y) - sq(L1) - sq(L2)) / (2 * L1 * L2));
  if (x < 0 & y < 0) {
    theta2 = (-1) * theta2;
  }

  theta1 = atan(x / y) - atan((L2 * sin(theta2)) / (L1 + L2 * cos(theta2)));

  theta2 = (-1) * theta2 * 180 / PI;
  theta1 = theta1 * 180 / PI;

 // Angles adjustment depending in which quadrant the final tool coordinate x,y is
  if (x >= 0 & y >= 0) {       // 1st quadrant
    theta1 = 90 - theta1;
  }
  if (x < 0 & y > 0) {       // 2nd quadrant
```

```
  theta1 = 90 - theta1;
}
if (x < 0 & y < 0) {       // 3d quadrant
  theta1 = 270 - theta1;
  phi = 270 - theta1 - theta2;
  phi = (-1) * phi;
}
if (x > 0 & y < 0) {       // 4th quadrant
  theta1 = -90 - theta1;
}
if (x < 0 & y == 0) {
  theta1 = 270 + theta1;
}

// Calculate "phi" angle so gripper is parallel to the X axis
phi = 90 + theta1 + theta2;
phi = (-1) * phi;

// Angle adjustment depending in which quadrant the final tool coordinate x,y is
if (x < 0 & y < 0) {       // 3d quadrant
  phi = 270 - theta1 - theta2;
}
if (abs(phi) > 165) {
  phi = 180 + phi;
}

theta1=round(theta1);
theta2=round(theta2);
```

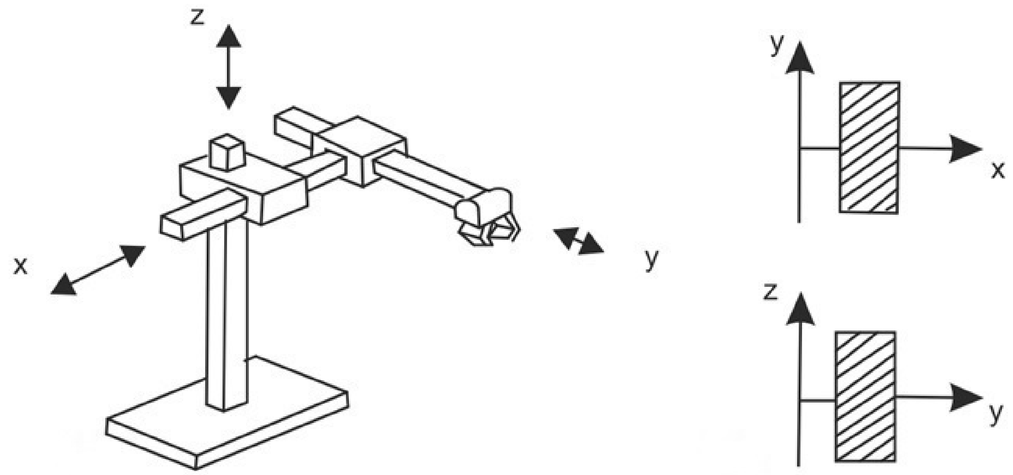
```
  phi=round(phi);


  cp5.getController("j1Slider").setValue(theta1);
  cp5.getController("j2Slider").setValue(theta2);
  cp5.getController("j3Slider").setValue(phi);
  cp5.getController("zSlider").setValue(zP);
 }

 void controlEvent(ControlEvent theEvent) {


  if (theEvent.isController()) {
   println(theEvent.getController().getName());
  }
 }


 public void xTextfield(String theText) {
  //If we enter a value into the Textfield, read the value, convert to integer, set the inverseKinematics mode active
  xP=Integer.parseInt(theText);
  activeIK = true;
  inverseKinematics(xP, yP); // Use inverse kinematics to calculate the J1(theta1), J2(theta2), and J3(phi) positions
  //activeIK = false;
  println("Test; theta1: "+theta1+" theta2: "+theta2);
 }
 public void yTextfield(String theText) {
  yP=Integer.parseInt(theText);
  activeIK = true;
  inverseKinematics(xP, yP);
```

```
  //activeIK = false;
}
public void zTextfield(String theText) {
  zP=Integer.parseInt(theText);
  activeIK = true;
  inverseKinematics(xP, yP);
}

public void j1JogMinus() {
  int a = round(cp5.getController("j1Slider").getValue());
  a=a-j1JogValue;
  cp5.getController("j1Slider").setValue(a);
}
//J1 control
public void j1JogPlus() {
  int a = round(cp5.getController("j1Slider").getValue());
  a=a+j1JogValue;
  cp5.getController("j1Slider").setValue(a);
}
//J2 control
public void j2JogMinus() {
  int a = round(cp5.getController("j2Slider").getValue());
  a=a-j2JogValue;
  cp5.getController("j2Slider").setValue(a);
}
public void j2JogPlus() {
  int a = round(cp5.getController("j2Slider").getValue());
  a=a+j2JogValue;
```

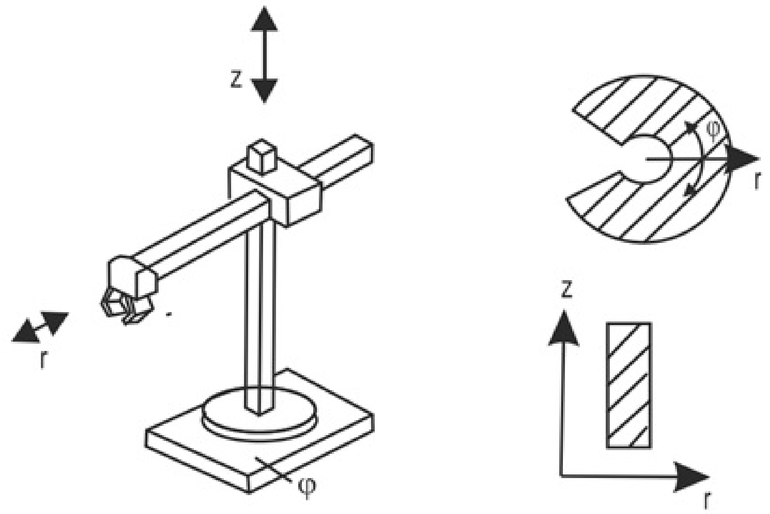
```
  cp5.getController("j2Slider").setValue(a);
}
//J3 control
public void j3JogMinus() {
  int a = round(cp5.getController("j3Slider").getValue());
  a=a-j3JogValue;
  cp5.getController("j3Slider").setValue(a);
}
public void j3JogPlus() {
  int a = round(cp5.getController("j3Slider").getValue());
  a=a+j3JogValue;
  cp5.getController("j3Slider").setValue(a);
}
//J3 control
public void zJogMinus() {
  int a = round(cp5.getController("zSlider").getValue());
  a=a-zJogValue;
  cp5.getController("zSlider").setValue(a);
}
public void zJogPlus() {
  int a = round(cp5.getController("zSlider").getValue());
  a=a+zJogValue;
  ;
  cp5.getController("zSlider").setValue(a);
}

public void move() {
```

```
  myPort.write(data);
  println(data);
 }

 public void savePosition() {
  // Save the J1, J2, J3 and Z position in the array

positions[positionsCounter]="J1="+str(round(cp5.getController("j1Slider").getValue()))
   +"; J2=" + str(round(cp5.getController("j2Slider").getValue()))
   +"; J3="+str(round(cp5.getController("j3Slider").getValue()))
   +"; Z="+str(round(cp5.getController("zSlider").getValue()));
  positionsCounter++;
  saveStatus = 1;
  updateData();
  myPort.write(data);
  saveStatus=0;
 }

 public void run() {

  if (runStatus == 0) {
   cp5.getController("run").setCaptionLabel("STOP");
   cp5.getController("run").setColorLabel(#e74c3c);

   runStatus = 1;
  } else if (runStatus == 1) {
   runStatus = 0;
   cp5.getController("run").setCaptionLabel("RUN PROGRAM");
```

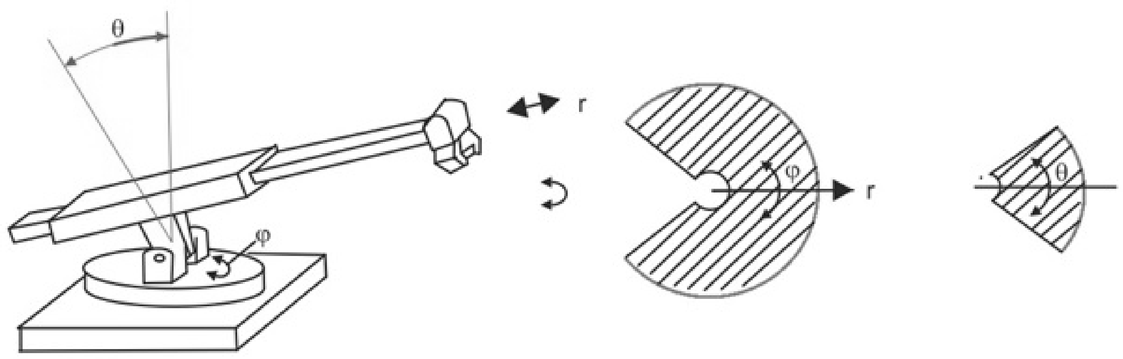
```
    cp5.getController("run").setColorLabel(255);
   }
   updateData();
   myPort.write(data);
  }
  public void updateSA() {
   myPort.write(data);
  }
  public void clearSteps() {
   saveStatus = 2; // clear all steps / program
   updateData();
   myPort.write(data);
   println("Clear: "+data);
   positionsCounter=0;
   saveStatus = 0;
  }

  public void updateData() {
   data = str(saveStatus)
     +","+str(runStatus)
     +","+str(round(cp5.getController("j1Slider").getValue()))
     +","+str(round(cp5.getController("j2Slider").getValue()))
     +","+str(round(cp5.getController("j3Slider").getValue()))
     +","+str(round(cp5.getController("zSlider").getValue()))
     +","+str(gripperValue)
     +","+str(speedSlider)
     +","+str(accelerationSlider);
}
```



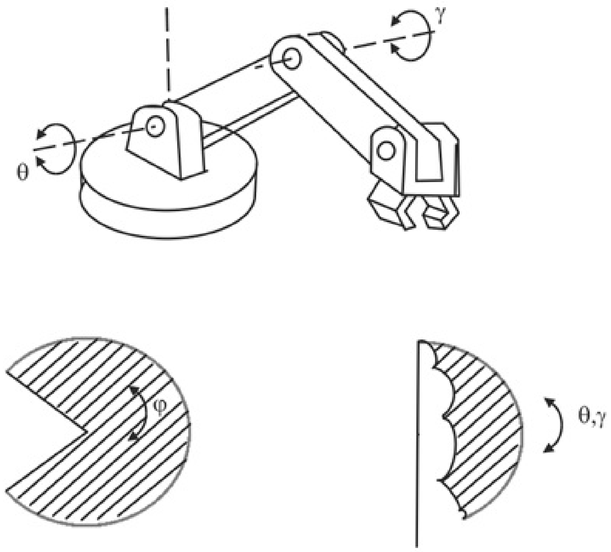
Маніпулятор з прямокутною системою координат і його робоча зона



Маніпулятор із циліндричною системою координат та його робоча зона



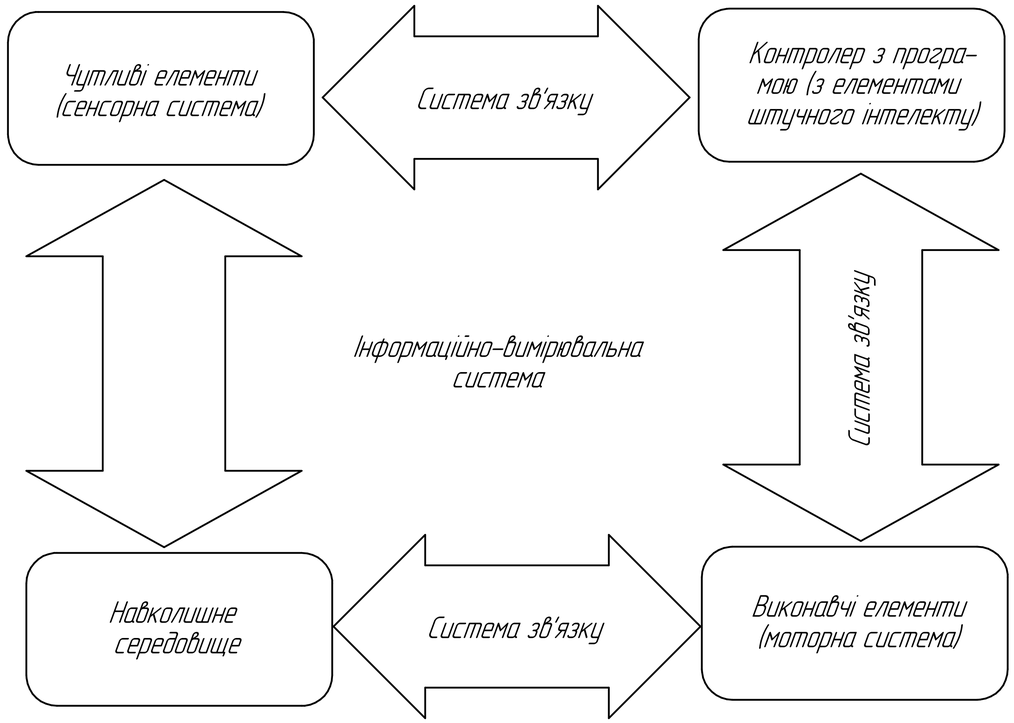
Маніпулятор зі сферичною системою координат і його робоча зона

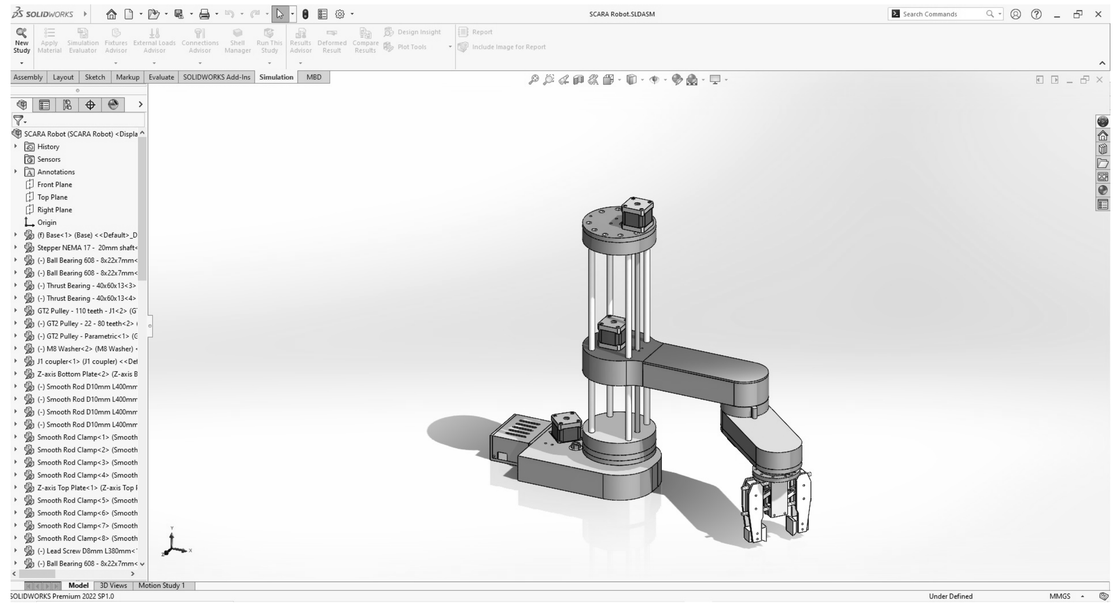


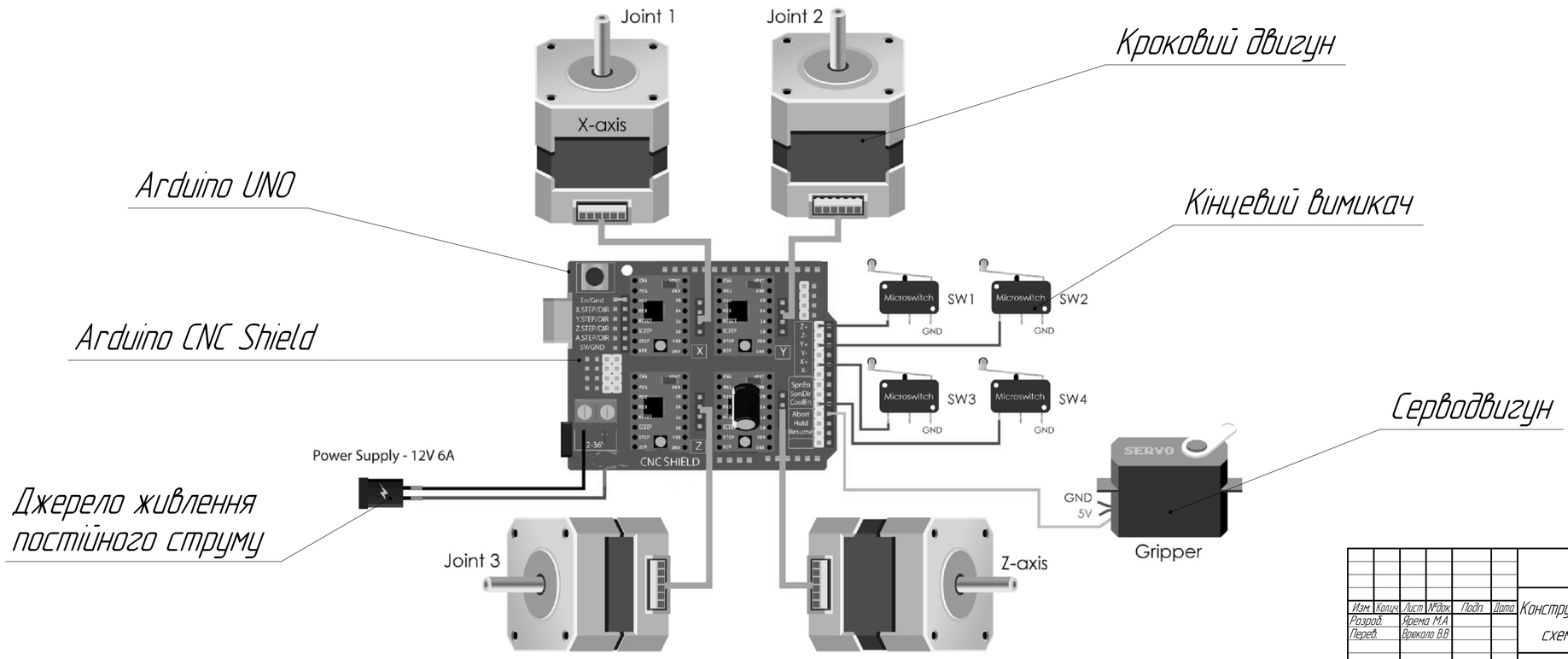
Маніпулятор з кутовою системою координат і його робоча зона

						БР.ПМІ-97.00.001	
Изм.	Колич.	Лист	№док.	Подп.	Дата	Конструктивні схеми промислових маніпуляторів	Стадия / Масса / Масштаб
Розроб.		Ярема М.А					
Перев.		Врикало В.В					Лист / Листов
Н.контр.		Врикало В.В					ІФНТУНГ ПМІ-20-1К
Затв.		Панчук В.Г					

Формат А1







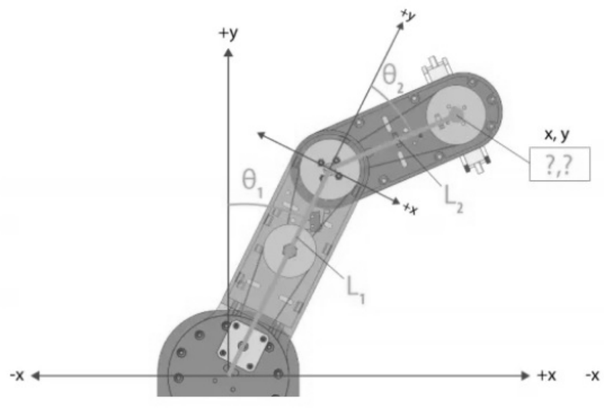
								БР.ПМІ-97.00.002	
Изм.	Колич.	Лист	№док.	Подп.	Дата	Конструкція та електронна схема робота Scara	Стадия	Масса	Масштаб
Разраб.		Ярема М.А.							
Перев.		Врикало В.В.					Лист	Листов	
Н.контр.		Врикало В.В.					ІФНТУНГ ПМІ-20-1К		
Затв.		Панчук В.Г.							

Формат А1

Рівняння прямої кінематики

$$x = L_1 \times \sin(\theta_1) + L_2 \times \sin(\theta_1 + \theta_2)$$

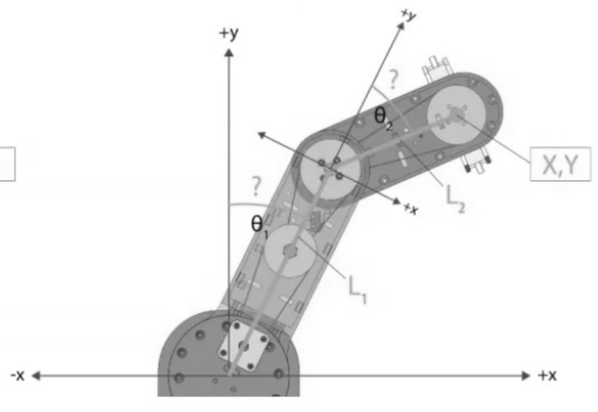
$$y = L_1 \times \cos(\theta_1) + L_2 \times \cos(\theta_1 + \theta_2)$$



Рівняння обернутої кінематики

$$\theta_2 = \arccos((x^2 + y^2 - L_1^2 - L_2^2) / 2 \times L_1 \times L_2)$$

$$\theta_1 = \arctan(x / y) - \arctan((L_2 \times \sin(\theta_2)) / (L_1 + L_2 \times \cos(\theta_2)))$$



Рівняння прямої та обернутої кінематики в вигляді програмного коду

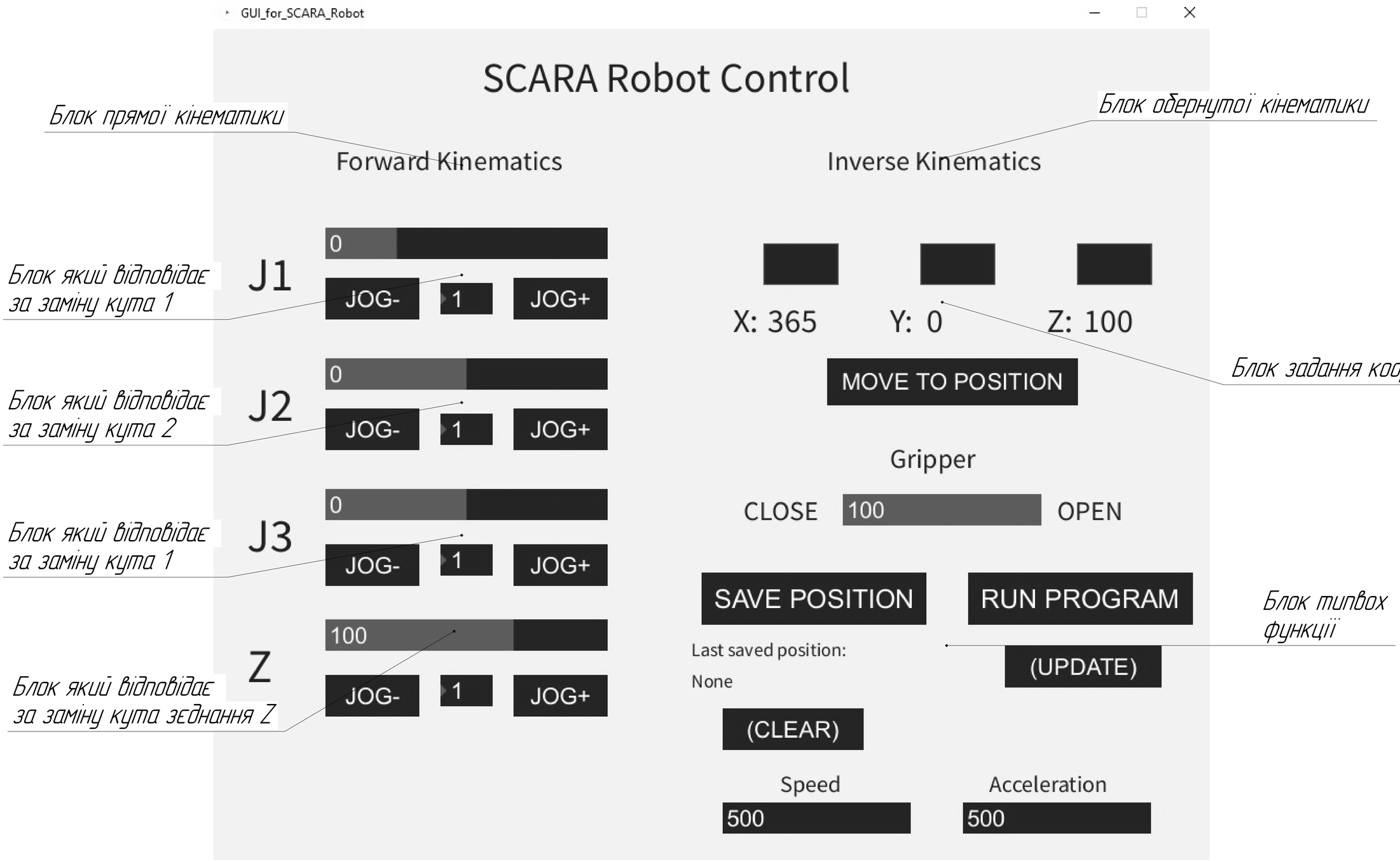
```
// FORWARD KINEMATICS
void forwardKinematics() {
  float theta1F = theta1 * PI / 180;   // degrees to radians
  float theta2F = theta2 * PI / 180;
  xP = round(L1 * cos(theta1F) + L2 * cos(theta1F + theta2F));
  yP = round(L1 * sin(theta1F) + L2 * sin(theta1F + theta2F));
}
```

Програмне вирішення рівння прямої та обернутої кінематик

```
/ INVERSE KINEMATICS
void inverseKinematics(float x, float y) {
  theta2 = acos((sq(x) + sq(y) - sq(L1) - sq(L2)) / (2 * L1 * L2));
  if (x < 0 & y < 0) {
    theta2 = (-1) * theta2;
  }

  theta1 = atan(x / y) - atan((L2 * sin(theta2)) / (L1 + L2 * cos(theta2)));

  theta2 = (-1) * theta2 * 180 / PI;
  theta1 = theta1 * 180 / PI;

 // Angles adjustment depending in which quadrant the final tool coordinate x,y is
  if (x >= 0 & y >= 0) {       // 1st quadrant
    theta1 = 90 - theta1;
  }
  if (x < 0 & y > 0) {       // 2nd quadrant
    theta1 = 90 - theta1;
  }
  if (x < 0 & y < 0) {       // 3d quadrant
    theta1 = 270 - theta1;
    phi = 270 - theta1 - theta2;
    phi = (-1) * phi;
  }
  if (x > 0 & y < 0) {       // 4th quadrant
    theta1 = -90 - theta1;
  }
  if (x < 0 & y == 0) {
    theta1 = 270 + theta1;
  }

  // Calculate "phi" angle so gripper is parallel to the X axis
  phi = 90 + theta1 + theta2;
  phi = (-1) * phi;

  // Angle adjustment depending in which quadrant the final tool coordinate x,y is
  if (x < 0 & y < 0) {       // 3d quadrant
    phi = 270 - theta1 - theta2;
  }
  if (abs(phi) > 165) {
    phi = 180 + phi;
  }

  theta1=round(theta1);
  theta2=round(theta2);
  phi=round(phi);

  cp5.getController("j1Slider").setValue(theta1);
  cp5.getController("j2Slider").setValue(theta2);
  cp5.getController("j3Slider").setValue(phi);
  cp5.getController("zSlider").setValue(zP);
}
```

							БР.ПМІ-97.00.003		
Изм.	Колич.	Лист	№док.	Подп.	Дата	Пряма та обернена кінематика робота	Стадия	Масса	Масштаб
Разраб.		Ярема М.А.							
Перев.		Врикало В.В.					Лист	Листов	
Н.контр.		Врикало В.В.					ІФНТУНГ ПМІ-20-1К		
Затв.		Панчук В.Г.							

Формат А1



						БР.ПМІ-97.00.004	
Изм.	Колич.	Лист	№док.	Подп.	Дата	Графічний інтерфейс користувача	Стадия / Масса / Масштаб
Розроб.		Ярема М.А.					
Перев.		Врикало В.В.					Лист / Листов
Н.контр.		Врикало В.В.				ІФНТУНГ ПМІ-20-1К	
Затв.		Панчук В.Г.					Формат А1