Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеню вищої освіти – магістр
за освітньо-професійною програмою
«Кіберфізичні системи в промисловості, бізнесі та транспорті»

зі спеціальності
174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка

тема роботи:

«*Керування рухом маніпулятора SCARA-робота з автоматичним пошуком траси при наявності фізичних перешкод*»

Виконав ст. гр. АКІТР-23-1м.	_____	Кам'яний Д. О.
Керівник	_____	Савицький О. І.
Нормоконтроль	_____	Маринич І. А.
Завідувач кафедри	_____	Рубан С. А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Магістр*

Спеціальність: *174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка*

ЗАТВЕРДЖУЮ

Зав. кафедри: *к.т.н. Рубан С.А.*

_____

« *5* » *липня* 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи АКІТР-23-1м. *Кам'яному Данііілу Олександровичу*

1. **Тема кваліфікаційної роботи:** *«Керування рухом маніпулятора SCARA-робота з автоматичним пошуком траси при наявності фізичних перешкод»*

затверджено наказом по університету № *595с* від *04.07.2024 р.*

2. Термін здачі кваліфікаційної роботи: *01.12.2024 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 85с., додатки, презентація у Microsoft PowerPoint (15 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-3 доц. Савицький О. І.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.07.24*
2	*Розділ 1*	*15.07.24*
3	*Розділ 2*	*18.08.24*
4	*Розділ 3*	*19.09.24*
5	*Висновки*	*15.10.24*
6	*Оформлення кваліфікаційної роботи*	*20.11.24*
7	*Підготовка презентації та графічного матеріалу*	*28.11.24*
8	*Підготовка доповіді до захисту*	*01.12.24*

6. Дата видачі завдання: *28.06.2024р.*

Керівник _____ / Савицький О. І./

7. Запевнення: *Я, Кам'яний Даніїл Олександрович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Здобувач _____ / Кам'яний Д. О./

АНОТАЦІЯ

Кам’яний Д.О. Керування рухом маніпулятора SCARA-робота з автоматичним пошуком траси при наявності фізичних перешкод.

Кваліфікаційна робота на здобуття ступеню вищої освіти магістр за освітньо-професійною програмою «Кіберфізичні системи в промисловості, бізнесі та транспорті» зі спеціальності 174 – Автоматизація, комп’ютерно – інтегровані технології та робототехніка– Криворізький національний університет, Кривий Ріг, 2024.

Об’єктом проектування є система керування рухом маніпулятора SCARA-робота з автоматичним пошуком траси при наявності фізичних перешкод.

Метою проекту є розробка системи автоматизованого керування руху ланок маніпулятора з автоматичним корегуванням руху ланок маніпулятора за наявності фізичних перешкод на шляху. Модернізація лабораторного стенду з метою розробки системи керування маніпулятором.

У результаті розробки системи керування промисловим маніпулятором з автоматичним пошуком траси при наявності фізичних перешкод, було отримано систему, що може сканувати робочу область маніпулятору лабораторного стенду, виявляти наявність та положення фізичних перешкод у цій зоні та має змогу відправляти отримані дані лабораторному стенду для обробки.

Ключові слова:

МАНІПУЛЯТОР, УЛЬТРАЗВУКОВИЙ ДАТЧИК, СЕРВОПРИВІД, КОРИГУВАННЯ РУХУ МАНІПУЛЯТОРА, КЕРУВАННЯ ЛАНКАМИ МАНІПУЛЯТОРА, TIA PORTAL, NODE RED, PROCESSING, ARDUINO.

ANNOTATION

Kamiany D.O. Controlling the movement of the SCARA-robot manipulator with automatic route search in the presence of physical obstacles.

Graduation master`s work for obtaining an educational degree «Master» for the educational and professional program « Cyber-physical systems in industry, business and transport » in specialty 174 – «Automation, computer-integrated technologies, and robotics». – Kryvyi Rih National University, Kryvyi Rih, 2024

The object of the design is the motion control system of the SCARA-robot manipulator with automatic route search in the presence of physical obstacles.

The goal of the project is to develop a system of automated control of the movement of the manipulator links with automatic correction of the movement of the manipulator links in the presence of physical obstacles on the way. Modernization of the laboratory stand for the purpose of developing a manipulator control system.

As a result of the development of a control system for an industrial manipulator with automatic route search in the presence of physical obstacles, a system was obtained that can scan the working area of the manipulator of the laboratory stand, detect the presence and position of physical obstacles in this area, and is able to send the received data to the laboratory stand for processing.

Keywords:

MANIPULATOR, ULTRASONIC SENSOR, SERVO DRIVE, CORRECTION OF MANIPULATOR MOVEMENT, CONTROL OF MANIPULATOR LINKS, TIA PORTAL, NODE RED, PROCESSING, ARDUINO.

ЗМІСТ

ВСТУП

Метою кваліфікаційної роботи є розробка системи автоматичного керування рухом маніпулятора SCARA-робота з використанням ультразвукових датчиків для автоматичного пошуку траси в умовах наявності фізичних перешкод. У сучасних умовах індустріалізації та автоматизації виробничих процесів маніпулятори SCARA знаходять широке застосування в різних галузях, зокрема в складальних роботах, упаковці та обробці матеріалів. Однак одним з основних викликів є необхідність адаптації рухів маніпулятора до зміни умов навколишнього середовища, таких як поява фізичних перешкод у робочому просторі.

Розв'язання цієї проблеми дозволить значно підвищити ефективність використання роботизованих систем, зменшити час на переналаштування маніпулятора та знизити ризик аварійних ситуацій, спричинених перешкодами. Використання ультразвукових датчиків для виявлення перешкод і автоматичного коригування траєкторії руху є важливим кроком до забезпечення більшої гнучкості та безпеки роботи маніпулятора.

Основними завданнями цієї роботи є:

- аналіз існуючих підходів до управління маніпуляторами SCARA;
- розробка системи автоматичного пошуку траси з використанням ультразвукових датчиків;
- дослідження та вибір апаратного забезпечення для реалізації системи управління;
- розробка програмного забезпечення для керування рухами маніпулятора;
- створення принципових, макетних та структурних схем системи управління маніпулятором.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСУ

1.1 Загальні відомості систем управління роботів-маніпуляторів

Промисловий робот — це автоматичний механічний пристрій, що складається з маніпулятора, як виконавчого пристрою, що має кілька ступенів рухливості, і пристрою з функцією перепрограмування для виконання переміщення об’єктів або обробних функцій у виробничому процесі[1].

Кожен промисловий робот складається з трьох основних систем: інформаційний, керуючий, виконавчий;

Інформаційна система(сенсорна система) призначена для збору всієї необхідної інформації у системі. Для цих цілей використовують різного виду датчики. Використовують електромагнітні, механічні, теплові, тактильні, оптичні, акустичні та інші датчики.

Керуюча система використовується для опису законів управління двигунів виконавчих органів на основі попередньо заданих значень та даних, зібраних датчиками.

Виконавча система надає можливість обробки керуючих сигналів. Зазвичай ця частина системи реалізується за допомогою маніпулятора.

Маніпулятор — це механічний пристрій, що призначений для переміщення у просторі об’єкта маніпулювання за допомогою утримання його захоплюючим органом або виконання інших механічних операцій у заданому просторі. (рис. 1.1)

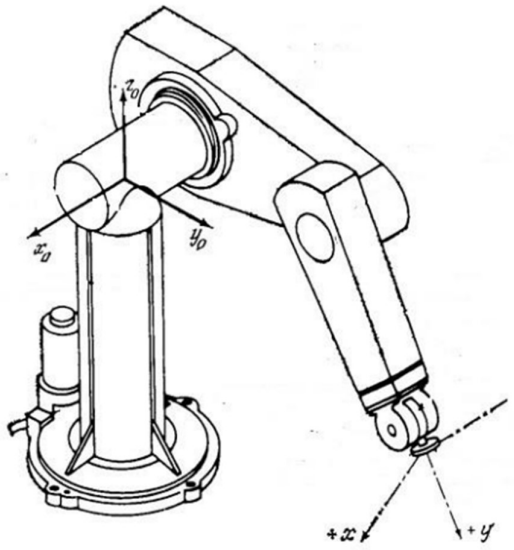

Рисунок 1.1 – Загальна схема робота-маніпулятором

Усі промислові роботи-маніпулятори можна розділити по ступеню складності на роботів 1, 2 та 3-го поколінь.

Роботи 1-го покоління також називають програмними роботами, оскільки у своїй основі вони мають «жорстку» програму і потребують точного позиціювання рухомих елементів та розміщення об’єктів маніпулювання. Більшість промислових роботів-маніпуляторів, що наразі використовуються у промислових процесах можна віднести до роботів 1-го покоління[2].

Зазвичай роботи першого покоління складаються з маніпулятору та програмних блоків. До програмних блоків можна віднести наступні елементи: пристрої зчитування, пристрої вводу та збереження програми та блок генератору машинного часу.

Для роботи таких промислових роботів необхідний оператор, який задаватиме режим роботи через панель управління. Режимів роботи може бути декілька і до них можна віднести ручний режим роботи, автоматичний, навчання або циклічне виконання програми[3].

Для роботи промислового робота-маніпулятора спочатку потрібно завантажити програму. Існують три основні методи навчання: за допомогою налаштування механічних елементів робота(навчання по точкам); за допомогою

панелі управління роботом, як на станках з ЧПК(числове програмне керування); за допомогою оператора, що також називають навчанням показом.

Роботи 2-го покоління також називають адаптивними роботами, оскільки вони вже не потребують точного позиціювання рухомих елементів та розміщення об’єктів маніпулювання через наявність можливості частково адаптуватися до навколишньої середи. Ключовою відмінністю є адаптивність, яка реалізується за допомогою імплементації необхідних датчиків та блоків корекції вхідних сигналів.

Роботи 3-го покоління також називають інтелектуальними, оскільки може повністю адаптувати свої дії та алгоритми відносно отриманих з датчиків даних про навколишню середу, виконуваний процес та його елементи.

Також у даному випадку людина виступає вже не в ролі оператора, а у ролі диспетчера, тобто задає певну задачу і отримує дані про її виконання від робота-маніпулятора.

Промислові роботи-маніпулятори вже довгий час використовують у промисловості, де вони і здобули найбільшого поширення. Особливо широке використання вони здобули у машинобудуванні. За цей час роботи-маніпулятори показали наскільки вони можуть бути корисними і продуктивними у рамках промислового виробництва.

Так, наприклад, використання роботів-маніпуляторів підвищують якість виготовленої продукції, оскільки при їх використанні значно покращується дотримання технологічного процесу. Таким чином зменшується відсоток браку продукції через помилки оператора.

Також використання роботів-маніпуляторів може покращити кількість зекономленого матеріалу. Підвищується безпека праці на виробництві, оскільки всю небезпечну або несприятливу для здоров’я людини працю може виконувати робот- маніпулятор[4].

1.2 Аналіз системи управління маніпулятором лабораторного стенду SIEMENS

Дослідження та модернізацію промислового робота-маніпулятора було проведено у аудиторії КНУ №367, на основі лабораторного стенду SIEMENS.

Основними елементами схеми є наступні елементи: Комунікаційний модуль CSM 1277, ПЛК SIMATIC S7-1200 з CPU 1215C, SIWAREX WP231 ваговий модуль ETHERNET, SIMATIC IOT2000, модуль введення/виводу, блок живлення PM1207, стабілізований блок живлення, SINAMICS G120, силовий модуль PM240-2, керуючий модуль CU240E-2, інтелектуальна панель оператора IOP-2, маніпулятор у складі драйверів шагового двигуна TB6560 V2 та шагових двигунів 28BYJ-48.

Основною складовою стенду є ПЛК SIMATIC S7-1200.

SIMATIC S7-1200 (рис. 1.2) – мікроконтролер компанії Siemens , що надає широкий спектр застосувань, ефективне вирішення задач автоматизації низького та середнього рівня складності, дає можливість залучення всіх центральних процесорів в автономному режимі, у складі мережевих структур та опцій розподіленого введення-виводу.

Рисунок 1.2 – ПЛК SIMATIC S7-1200

ПЛК SIMATIC S7-1200 підтримує комунікаційний обмін даними за допомогою мереж Industrial Ethernet/PROFINET та Point-to-Point з'єднання. Програмовані логічні контролери S7-1200 дають можливість їх монтажу на профільну шину DIN стандарту 35 мм або монтажну плату. Також контролер має змогу працювати у діапазоні температур 0 - +50 °C. Корпус контролера має ступень захисту IP20. ПЛК S7-1200 має можливість обслуговувати 10-284 дискретних та 2-51 аналогового каналу введення-виводу.

Центральний процесор S7-1200 CPU 1215C DC/DC/DC оснащений вбудованим інтерфейсом Ethernet. Для одного процесорного модуля є можливість налаштування обміну даними через 16 різних з'єднань. Для організації обміну даними є можливість використовувати транспортні протоколи ISO на TCP,TCP/IP і S7 функції зв'язку (S7-сервер або ж S7-клієнт).

За необхідності у складі ПЛК може використовуватися 4-канальний комутатор Industrial Ethernet. На лабораторному стенді такий комутатор представлений у вигляді модуля S7-1200, а саме комунікаційний модуль CSM 1277.

Контролер має підтримку мов програмування LAD і FBD. Час логічних операцій не перевищує 0.1 мкс. Наявна вбудована пам'ять об'ємом до 2 Мбайт і наявна можливість її розширення за допомогою картки пам'яті з ємністю до 24 Мбайт. Наявна підтримка функцій ПІД-регулювання та функцій керування переміщенням згідно вимог PLCopen.

Комунікаційний модуль СМ 1277 (рис. 1.3) надає можливість встановити PtP(Point-to-Point) з'єднання між ПЛК S7-1200 та промисловими контролерами інших виробників, сканерами, принтерами, модемами, тощо. Даний модуль має два варіанти з інтегрованим послідовним інтерфейсом RS 485 або RS 232. Обидва модулі надають підтримку протоколів Modbus RTU і ASCII(ведений або провідний пристрій). Окрім цього, комунікаційний модуль із інтерфейсом RS 485 надає підтримку USS протоколу. Всі команди керування обміном даних вбудовані у систему команд ПЛК. Комунікаційний модуль СМ 1277 встановлений ліворуч від контролера та підключаються до внутрішньої шини ПЛК через вбудовані у

модуль з'єднувач. Комунікаційний модуль CSM 1277, що використовується у лабораторному стенді являє собою 4-канальний некерований комутатор та має 4X10/100 Мбит порта RJ45. Живлення комунікаційного модуля забезпечується подачею 24В DC.

Рисунок 1.3 – Комунікаційний модуль CSM 1277

У досліджуваному стенді підключення частотного перетворювача здійснюється за допомогою його підключення до комунікаційного модуля через PROFINET порт. Сам же комунікаційний модуль підключений до процесорного модуля та підключаються до його внутрішньої шини[5].

Також одним з основних елементів стенду є перетворювачем частоти SINAMICS G120.

SINAMICS G120 – це промисловий перетворювач частоти. Перетворювач використовується у багатьох промислових галузях. Так прикладом таких галузей можуть бути: харчова, металургія, гірничозбагачувальна, конвеєрні системи та машинобудування. Використовуються такі перетворювачі для керування двигунами насосів, вентиляторів, компресорів, конвеєрів, змішувачів і т.п.. SINAMICS G120 має широкий діапазон потужності 0,55 - 250 кВт, що дозволяє зібрати оптимальний для виконання задачі перетворювач. Данна серія перетворювачів має 3 варіанти напруги для підключення до мереж: 200, 400 та 690 В. Також включає комплексний пакет безпеки. Для введення функцій безпеки

в експлуатацію наявні відповідні інструменти у вигляді SINAMICS Startdrive або STARTER.

SINAMICS G120 має модульну конструкцію і складається з силового модуля, блоку управління та панелі оператора. Усі модулі можуть бути замінені відповідно до потреб та сумісні між собою. Блок управління забезпечує керування і контроль силового модуля та підключеного двигуна за допомогою різних типів керування із зворотним зв'язком. Блок підтримує зв'язок з пристроями моніторингу та локальним або центральним ПЛК. Силовий модуль слугує для подачі живлення на двигун у діапазоні потужності 0,37 - 250 кВт. У модулі використовується технологія IGBT із широтно-імпульсною модуляцією, що надає високий рівень надійності та підвищує гнучкості роботи двигуна. Панель оператора являє собою опціональний елемент, що створений для локального управління перетворювачем частоти оператором. Комплексні функції захисту забезпечують високий рівень захисту силового модуля та двигуна.

SIEMENS пропонує широкий асортимент силових модулів для різних вимог до приводів, такі як: силові модулі PM250, силові модулі PM240-2 для наскрізного монтажу, силові модулі PM240-2.

Для SINAMICS G120 поставляється кілька блоків керування з різними характеристиками, такі як: блок управління CU230P-2(для насосів, вентиляторів та компресорів), блок управління CU250S-2(для використання з високими робочими характеристиками), блок управління CU240E-2(для використання з стандартними робочими характеристиками).

SINAMICS надає можливість розширення експлуатаційних можливостей та керуючих функцій перетворювачів частоти SINAMICS. Прикладом можуть бути такі модулі як: SINAMICS Smart Access модуль, інтелектуальна панель оператора SINAMICS IOP-2, базова панель оператора SINAMICS BOP-2.

У досліджуваних стендах використовуються наступні модулі частотного перетворювача SINAMICS G120.

У стенді SINAMICS G120 базується на силовому модулі PM240-2 (рис. 1.4). Дана серія силових модулів зазвичай використовується для стандартного застосування

в машинобудуванні. Створений для виконання завдань у роботі з різними двигунами. Силовий модуль PM240-2 оснащений гальмівним переривником (має можливість роботи в чотирьох квадрантах) та підходять для широкого застосування в загальному машинобудуванні.

Рисунок 1.4 – Силовий модуль PM240-2

До основних особливостей силового модуля PM240-2 можна віднести: широкий діапазон налаштувань для вирішення задач виробничих процесів, вбудований гальмівний переривник, високий ККД, підключення модуля до мереж з різною системою заземлення, можливість використання в безпечно орієнтованих програмах, широкий вибір додаткового обладнання;

У якості блоку управління у досліджуваному стенді використовується блок управління CU240E-2 (рис. 1.5).

Стандартний блок керування (CU240E-2) зі збільшеною кількістю вводів-виводів ідеально підходить для застосування з вимогами безпеки без енкодера. CU240E-2 являє собою серію зі стандартним набором входів/виходів та вбудованою технікою безпеки. Наявна інтегрована функція безпеки "Безпечно вимкнений момент" (STO)[6].

Рисунок 1.5 – Блок керування CU240E-2

У якості панелі оператора у стенді використовується інтелектуальна панель оператора IOP-2 (рис. 1.6).

Інтелектуальна панель оператора IOP-2 забезпечує досить швидке налаштування, надає можливість проведення діагностики несправностей та має інтуїтивно зрозумілий інтерфейс керування перетворювачем частоти SINAMICS G120 у локальному режимі. Окрім цього, спрощує коригування параметрів при виконанні роботи. Багатофункціональна панель оператора має централізоване управління та зрозумілий інтерфейс, що полегшує роботу за рахунок встановлених налаштувань.

Рисунок 1.6 – Інтелектуальна панель оператора IOP-2

IOP-2 постачається як опціональний модуль для перетворювачів частоти SINAMICS. До основних переваг даної інтелектуальної панелі можна віднести: висококонтрастний дисплей з різноманітними варіантами індикації, досить швидке введення в експлуатацію, проста конфігурація інтерфейсу промислової шини на базі Ethernet, надає легкий доступ до технічної інформації системи приводу, надає швидкий контакт до клієнтської підтримки через SIEMENS Industry Online Support App, просте та швидке з'єднання з мобільними пристроями шляхом використання двомірного коду (DataMatrix/QR-код).

Також однією з основних частин лабораторного стенду є маніпулятор.
У нашому випадку маніпулятор стенду являє собою робота-маніпулятора з 3 осями дії. Для його реалізації було використано крокові двигуни та драйвери крокових двигунів, для кожної осі відповідно.

Розглянемо використовувані крокові двигуни 28BJ48 (рис. 1.7).

Рисунок 1.7 – Кроковий двигун 28BJ48

Даний кроковий двигун являє собою двигун малої потужності. Кроковий двигун 28BJ48 має дві обмотки(кожна з обмоток має відведення з середини). Таким чином маємо 4 фази, тобто пристрій являє собою 4-фазний кроковий двигун (рис. 1.8). Відведення обмоток з’єднані між собою та підключені до червоного проводу. Являє собою однополярним кроковим двигуном. Живлення подається на червоний провід. Переміщення валу здійснюється за допомогою електричного імпульсу[7].

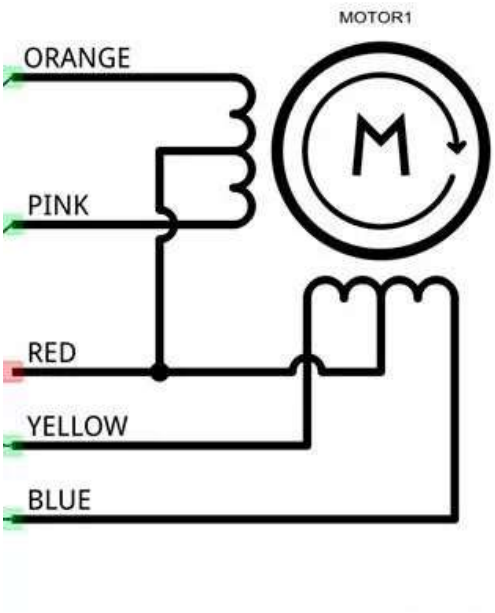

Рисунок 1.8 – Принципова схема крокового двигуна 28BJ48

Обертання ротору двигуна виконується за допомогою пошагової комутації фаз. Для здійснення повороту на певний кут або ж для здійснення певної кількості обертів на фази двигуна подається серія імпульсів (табл. 1.1).

Кроковий двигун має два режими роботи: повнокроковий режим(4 ступені імпульсів на 1 крок) та напівкроковий режим(8 ступенів імпульсів на 1 крок).

Таблиця 1.1 Характеристики крокового двигуна 28BJ48

Параметр крокового двигуна	Значення параметру
Напруга живлення, В	5
Кількість фаз	4
Коефіцієнт редукції	1/63.68395
Кількість кроків ротора	64
Номінальна швидкість обертання, об/хв	15
Обертаючий момент, г*см	450
Розміри(діаметр, висота), мм	25x18
Вага, гр	40

Розглянемо використовувані драйвери крокових двигунів TB6560 V2 (рис. 1.9).

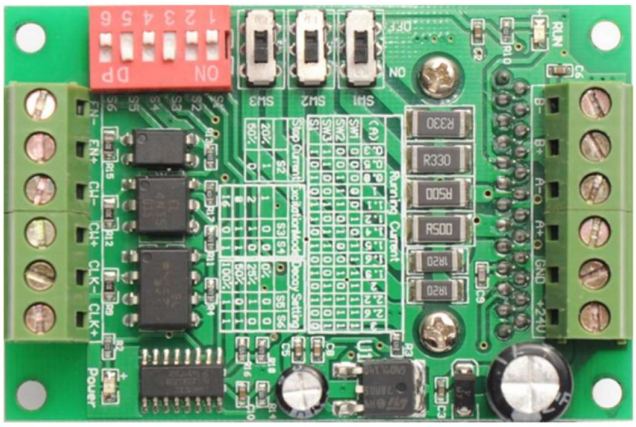

Рисунок 1.9 – Драйвер крокових двигунів TB6560 V2

Драйвер крокових двигунів TB6560 V2 виконаний на спеціалізованій платі Toshiba TB6560AHQ з живленням від 10 до 35 В постійного струму [8-9]. Даний драйвер призначений для роботи з кроковими двигунами типу NEMA17 – NEMA23 з максимальним током фази до 3А (табл. 1.2).

Таблиця 1.2 Характеристики драйвера крокового двигуна TB6560 V2

Параметр драйверу	Значення параметру
Напруга живлення, В	10-35
Вихідний струм, А	0.3 ~ 3
Температура експлуатації, °С	-10 ~ 45
Регулювання струму	14 ступенів
Мікрокрок	1 .. 2 .. 8 .. 16
Розмір, мм	75*50*35
Вага, г	73

Для стенду розроблено програму для управління стендом у ручному та автоматичному режимі. Вікно оператора (рис. 1.10) виглядає наступним чином:

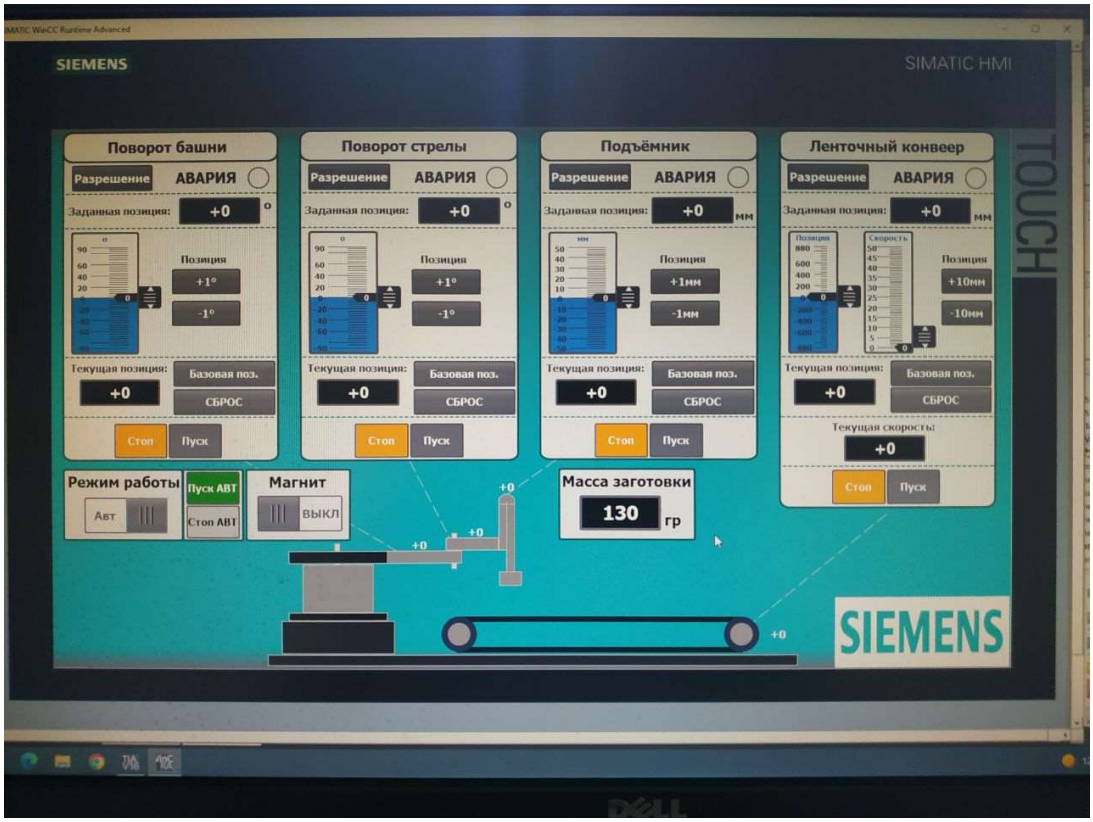

Рисунок 1.10 – Вікно оператора

Дослідимо роботу стенду у ручному режимі.

Вікно оператора надає можливість управління стендом. У вікні присутні 4 блоки управління для керування баштою, стрілою, підйомником та конвеєром, відповідно. Є можливість перемикання режимів роботи з автоматичного у ручний і навпаки.

Є можливість включення/виключення магніту підйомника відповідною кнопкою. Наявне схематичне зображення стенду з виводом стану відповідних елементів та їх позицій. Також виводиться вага вантажу.

У головних блоках є можливість управління окремими елементами стенду у ручному режимі. Для початку роботи потрібно натиснути кнопку «Разрешение». Лампа «Аварія» повідомляє про наявність несправностей[10].

Поле «Заданная позиция» показує задану нами позицію відповідного елемента стенду. За допомогою повзунка або кнопок «+1» та «-1» можна обрати позицію,

на яку ми хочемо перемістити елемент. При натисканні кнопки «Пуск» елемент перейде у задану позицію.

Поле «Текущая позиция» показує поточну позицію елемента відповідно його базової позиції. Встановити базову позицію позицію можна натиснувши кнопку «Базовая поз.», після чого поточна позиція стане базовою і переміщення елемента буде проводитись відносно цієї позиції.

Позиції башта та стріли керуються по горизонтальній площині, а підйомника – по вертикалі. Для конвеєру, окрім позиції, є також можливість встановити його швидкість.

Дослідимо роботу автоматичного режиму роботи стенда.

Для переходу у автоматичний режим потрібно перетягнути повзунок «Режим работы» у автоматичний режим і для пуску натиснути кнопку «Пуск АВТ». У будь-який момент є можливість зупинити роботу стенду за допомогою кнопки «Стоп АВТ». Робота автоматичного режиму базується на програмі, що була попередньо завантажена у ПЛК S7-1200. Дослідимо роботу цієї програми.

При виконанні програми автоматичного режиму покроково виконуються певні дії.

Спочатку опускається підйомник та включається магніт, що підбирає вантаж. Після чого підйомник піднімається і стрілка повертається вліво на 90 градусів. Далі башта повертається вправо на 90 градусів. Після чого на 90 градусів вправо повертається і стріла та опускається підйомник. Потім вимикається магніт і кладе вантаж на конвеєрну стрічку.

Далі на стрічці спрацьовує датчик, що запускає конвеєрну стрічку і везе вантаж до другого датчика. В цей час маніпулятор піднімає підйомник та повертає стрілу на 90 градусі вправо. Коли вантаж доходить до кінця конвеєру, спрацьовує другий датчик. Датчик зупиняє конвеєр.

Також при спрацюванні другого датчика маніпулятор повертає башту на 90 градусів вліво і так само повертає стрілку після руху башти. Далі маніпулятор опускає підйомник на базове значення та на цьому робота програми завершується.

1.3. Аналіз існуючих систем керування ланок маніпулятора

Розвиток сучасних виробничих процесів потребує впровадження новітніх технологій, що забезпечують високий рівень автоматизації та інтеграції управлінських систем. Інтегровані автоматизовані системи управління виробництвом є ключовими компонентами таких технологій, оскільки вони дозволяють ефективно керувати всіма аспектами виробничого процесу, від планування до контролю якості продукції.

Технологічний процес використання роботизованих маніпуляторів з дистанційним керуванням включає кілька основних етапів: планування маршруту, виявлення перешкод, корекцію траєкторії та виконання операцій. Кожен з цих етапів вимагає високої точності та надійності, що забезпечується завдяки використанню сучасних сенсорних технологій та алгоритмів обробки даних. Ультразвукові датчики, які використовуються для сканування простору робочої зони маніпулятора, є одним з найбільш ефективних інструментів для виявлення перешкод. Вони дозволяють отримати точну інформацію про положення об'єктів у просторі, що критично важливо для забезпечення безперервного і безпечного руху маніпулятора(рис. 1.11).

Існуючі рішення в галузі ІАСУВ базуються на використанні різних типів датчиків, включаючи ультразвукові, інфрачервоні та лазерні. Кожен з цих типів має свої переваги і недоліки.

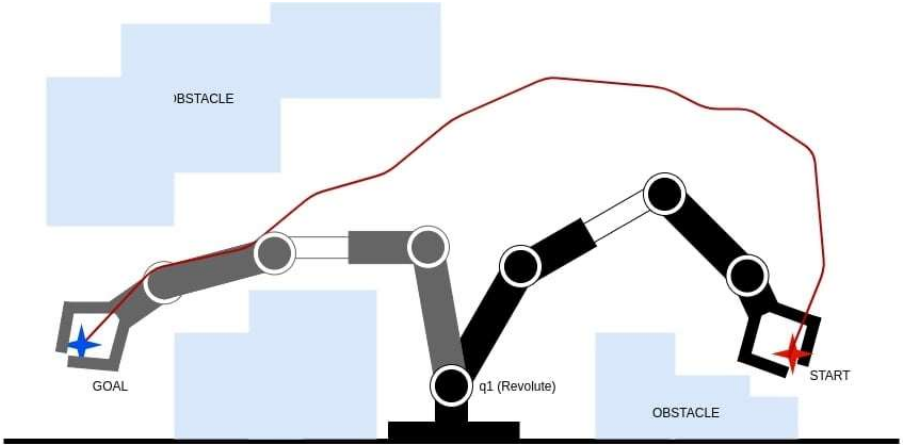

Рисунок 1.11 – Планування руху маніпулятора без зіткнень

Ультразвукові датчики є відносно недорогими і простими у використанні, але їх точність може бути обмеженою в умовах високої щільності об'єктів або шуму. Інфрачервоні датчики забезпечують високу точність на коротких відстанях, але можуть мати проблеми з виявленням прозорих або сильно відбивних поверхонь. Лазерні датчики, навпаки, забезпечують високу точність і надійність в широкому діапазоні умов, що робить їх оптимальним вибором для багатьох завдань у робототехніці[11].

На сьогоднішній день існують різноманітні комерційні рішення, які інтегрують різні датчики у системи управління маніпуляторами. Такі системи часто використовують алгоритми машинного навчання та штучного інтелекту для аналізу даних від сенсорів і прийняття рішень у режимі реального часу. Наприклад, компанія FANUC пропонує інтегровані рішення для роботів-маніпуляторів, що включають лазерні сенсори для уникнення зіткнень.

Спосіб і пристрій планування беззіткненого руху маніпулятора

Спосіб планування руху без зіткнень першого маніпулятора від початкової точки до цільової точки. Спосіб містить етапи: повторне визначення цільового шляху першого маніпулятора до цільової точки за допомогою глобального планувальника; безперервне визначення руху першого маніпулятора з локальним планувальником на основі поточного цільового шляху глобального планувальника; і виконання руху першим маніпулятором паралельно визначенню глобальним і локальним планувальником(рис. 1.12).

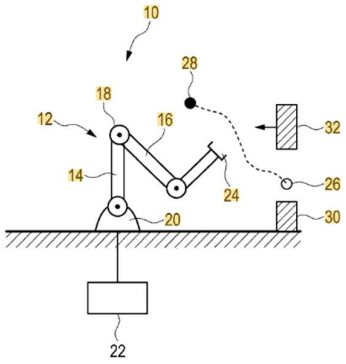

Рисунок 1.12. Спосіб і пристрій планування беззіткненого руху маніпулятора

Таким чином, метою цього винаходу є забезпечення оптимізованого планування руху для робота в мінливому середовищі, яке дозволяє уникнути вищевказаних недоліків. Зокрема, це завдання вказати планування руху, яке враховує динаміку робота та забезпечує безперервне перепланування без зупинки чи зниження швидкості[12].

Відповідно до одного з аспектів цього винаходу, ця мета досягається за допомогою способу планування руху без зіткнень першого маніпулятора від початкової точки до цільової точки з наступними етапами:

багаторазове визначення цільового шляху першого маніпулятора до цільової точки за допомогою глобального планувальника;

безперервне визначення руху першого маніпулятора з локальним планувальником на основі поточного цільового шляху глобального планувальника;

Виконання руху першим маніпулятором паралельно визначенню глобальним і локальним планувальником[13].

Таким чином, ідея цього винаходу полягає в тому, щоб поєднати повторюване глобальне планування з безперервним локальним плануванням при плануванні руху робота. У цьому контексті безперервний означає, що перепланування відбувається безперервно, бажано через певний інтервал. Повторне, з іншого боку, означає, що глобальний планувальник перепланує принаймні один раз. Повторне визначення цільового шляху особливо запитується невмотивовано, тобто спочатку виконується нове визначення цільового шляху незалежно від того, чи поточний цільовий шлях все ще є дійсним або все ще виконується. На відміну від відомих методів, нове планування здійснюється навмисно, навіть якщо середовище явно не змінилося для місцевого планувальника. Перепланування, здійснене глобальним планувальником, не обов’язково має бути «запущене» місцевим планувальником. Таким чином глобальне планування може знайти коротший цільовий шлях, якщо, наприклад, перешкода зникає з попередньо запланованого цільового шляху.

Відповідно до винаходу, таким чином, передбачено два різних планувальники, локально і глобально, які виконуються паралельно, доповнюють один одного і, зокрема, враховують один одного у відповідному плануванні. У цьому контексті паралельний означає паралельний у часі, тобто планувальники обробляються обчислювально паралельно, наприклад, у потоках[14].

Завдяки безперервному плануванню з місцевим планувальником і додатковому перепланування з глобальним планувальником планування руху можна оптимально адаптувати до мінливого та складного середовища, оскільки недоліки однієї процедури компенсуються перевагами іншої. Зокрема, динаміка робота може бути адекватно врахована при плануванні, що дає можливість перепланування без зупинки і зниження швидкості робота, а також плавну адаптацію руху в зміненому середовищі. Це дає змогу прогнозовано обходити рухомі перешкоди та використовувати коротші шляхи, якщо перешкоди видалено. Таким чином, завдання, згадане на початку, повністю виконано.

Метод і система для прогнозування та уникнення зіткнень маніпулятора з людиною.

Цей винахід належить до сфери обчислювальної техніки. Його метою є підвищення безпеки персоналу, який працює поблизу промислового маніпулятора. Досягнення технічного результату забезпечується шляхом: створення віртуального графа прохідності маніпулятора; отримання RGB-зображення та карти глибини з камери, що спостерігає за робочою зоною; сегментації RGB-зображення з використанням глибокої згорткової нейронної мережі для ідентифікації пікселів, які належать людині; формування хмари точок людини шляхом перетворення відповідних пікселів та їх глибини; обчислення відстані від кожної тривимірної точки хмари до вершин віртуального графа прохідності; визначення відстані між людиною та робочими точками маніпулятора; порівняння цих відстаней із заданим пороговим значенням. (рис. 1.13).

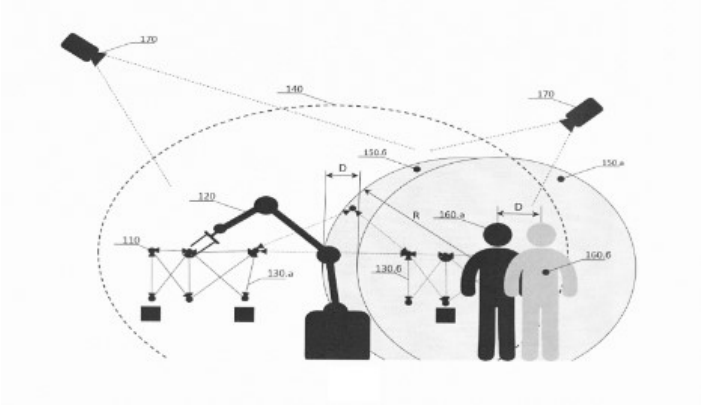

Рисунок 1.13. Спосіб та система предиктивного уникнення зіткнення маніпулятора з людиною

Маніпулятор та спосіб його керування визначають ступінь свободи (DOF), що використовується для уникнення зіткнення з перешкодами, враховуючи порядок пріоритетів DOF для виконання операцій і запобігання зіткненням. Система оцінює можливість зіткнення маніпулятора з перешкодою та перевіряє, чи є серед доступних DOF хоча б одна, здатна уникнути зіткнення. Якщо ризик зіткнення підтверджується, маніпулятор визначає найменш пріоритетну DOF, яка може забезпечити уникнення перешкоди, і використовує її для запобігання зіткненню під час виконання операції. Такий підхід гарантує безпечну роботу маніпулятора за умови наявності робочої DOF, здатної уникнути зіткнення. (рис. 1.14).

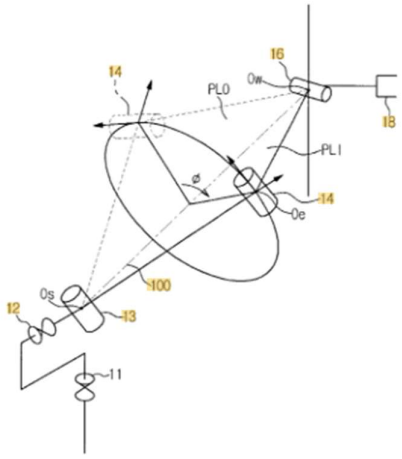

Рисунок 1.14. Маніпулятор і спосіб керування ним

Одним із ключових аспектів цих варіантів реалізації є забезпечення маніпулятора, здатного визначати ступінь свободи (DOF), що використовується для уникнення зіткнення з перешкодами, з урахуванням пріоритетів DOF для виконання операцій та їх ефективності в запобіганні зіткненням, а також спосіб керування цим процесом.

Додаткові аспекти будуть частково розкриті у подальшому описі, частково стануть очевидними з представлених матеріалів, або можуть бути зрозумілими під час практичного застосування винаходу. [15].

Один із аспектів методу керування маніпулятором, який має кілька робочих ступенів свободи (DOF) для виконання операцій, полягає в наступному. Метод передбачає оцінку ризику зіткнення маніпулятора з перешкодою, перевірку наявності хоча б одного робочого DOF, здатного уникнути зіткнення серед доступних DOF, у разі виявлення ризику, та уникнення зіткнення за допомогою DOF із найнижчим пріоритетом. Це відбувається під час виконання операції після підтвердження, що щонайменше один DOF може забезпечити уникнення зіткнення з перешкодою.

Маніпулятор може мати принаймні один надлишковий ступінь свободи (DOF), додатковий до робочих DOF. Метод також може включати перевірку, чи здатен маніпулятор уникнути зіткнення з перешкодою, використовуючи лише надлишковий DOF, після виявлення ризику зіткнення. Якщо визначено, що маніпулятор може уникнути зіткнення за допомогою одного чи кількох надлишкових DOF, зіткнення запобігається, використовуючи лише ці надлишкові DOF.

Роботи-фарбувальники, які обробляють деталь, що рухається на конвеєрі, синхронізуються шляхом створення для кожного з роботів головної послідовності інструкцій комп’ютерної програми для безконфліктного руху роботів уздовж пов’язаних траєкторій головної послідовності відносно рухомої частини, кожного з шляхів головної послідовності. включаючи позиції пов’язаного робота та конвеєра в попередньо визначених точках синхронізації та запуск кожної головної послідовності на контролері, підключеному до пов’язаного робота, для

переміщення пов’язаного робота та порівняння поточного шляху пов’язаного робота та конвеєра з основний шлях послідовності. Спосіб додатково включає роботу контролерів для коригування поточних шляхів на основі порівняння між основним шляхом послідовності та поточним шляхом, а також керування контролерами для запиту утримання руху конвеєра, якщо це необхідно для сприяння синхронізації між рухом роботів і конвеєром(рис. 1.15).

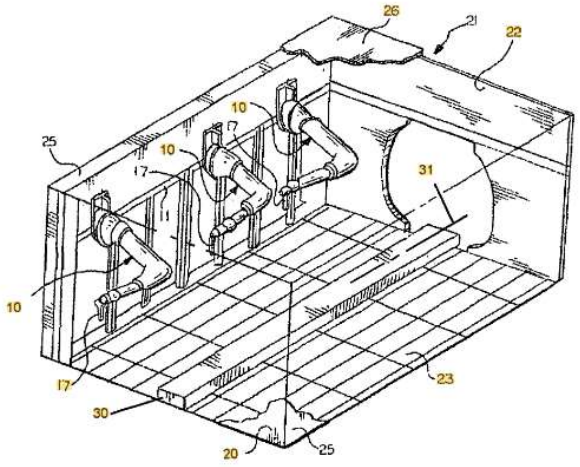

Рисунок 1.15. Синхронізація процесу фарбування багаторукового робота

Спосіб згідно з винаходом синхронізації фарбування декількома роботизованими маніпуляторами, які обробляють об’єкт, що рухається вздовж конвеєра, містить етапи: запис головної послідовності, в якій записується оптимальний шлях без зіткнень, включаючи позиції робота-маніпулятора та конвеєра разом; головна послідовність складається з роботизованого маніпулятора та позицій конвеєра в попередньо визначених точках синхронізації; програмне забезпечення під час виконання порівнює поточний рух шляху виробництва з головною послідовністю; програмне забезпечення під час виконання, яке впливає на планування руху, базується на порівнянні між головною послідовністю та поточним шляхом; програмне забезпечення під час виконання запитує утримання конвеєра, якщо це необхідно для полегшення синхронізації.

1.4 Опис прийнятої системи управління

Розглянуті системи управління для маніпулятора SCARA-робота включають декілька сучасних підходів, які можуть бути застосовані залежно від конкретних вимог до точності, вартості та зручності реалізації. Однак під час аналізу було прийнято рішення зосередитися на розробці системи управління, яка використовує ультразвукові датчики для автоматичного пошуку траси при наявності фізичних перешкод.

Система управління на основі лазерних датчиків була відхилена через її високу вартість і складність інтеграції. Використання лазерних сенсорів вимагає розробки складних алгоритмів обробки даних, зокрема фільтрації шумів та розпізнавання перешкод у реальному часі, що робить таку систему менш підходящою для застосування в умовах обмеженого бюджету.

Система з використанням камер та алгоритмів машинного зору також розглядалася як альтернатива. Вона забезпечує високу точність та адаптивність, але є технічно складною в реалізації. Такі системи потребують значних обчислювальних ресурсів, додаткових етапів калібрування та можуть бути схильними до впливу змін зовнішніх умов, таких як освітлення.

Ультразвукові датчики були обрані як основний інструмент для побудови системи через їхню простоту, надійність і відносно низьку вартість. Ці датчики дозволяють ефективно визначати перешкоди та будувати трасу маніпулятора в реальному часі без необхідності складної обробки даних. Основні недоліки, такі як можливість появи мертвих зон або обмеження за дальністю, будуть враховані шляхом оптимального розташування сенсорів і розробки відповідного алгоритму для аналізу отриманих даних.

Таким чином, прийнята система управління забезпечує баланс між технічною складністю, вартістю та функціональністю, що робить її оптимальним вибором для розв'язання поставленого завдання.

Висновки за розділом

Було розглянуто таке поняття як маніпулятор, його принципи роботи, види маніпуляторів та його основні завдання. Досліджено маніпулятор лабораторного стенду та описано його ключові складові. Розглянуто основні елементи стенду, на якому проводилось розробка системи керування маніпулятором. Розроблено схему системи координат і зони дії елементів маніпулятору.

Аналіз існуючих рішень систем керування ланок промислових роботів-маніпуляторів показав, що системи з використанням ультразвукових датчиків для автоматичного пошуку траси маніпулятора забезпечує баланс між точністю, зручністю реалізації та витратами. Це робить розроблену систему управління ефективним та практичним рішенням для вирішення поставлених завдань.

Окрім обраної системи було розглянуто такі системи: управління за допомогою камер, лазерних датчиків та машинного зору. Усі інші розглянуті системи виявилися недоцільними варіантами для розробки системи навчання на їх базі, оскільки кожна з систем мала низку своїх недоліків, що робить розробку цих систем більш складною та вимагає більших грошових затрат.

РОЗДІЛ 2
ІДЕНТИФІКАЦІЯ МАТЕМАТИЧНОЇ МОДЕЛІ ТА СИНТЕЗ КЕРУВАННЯ ПРОЦЕСОМ

2.1 Огляд апаратної частини для модернізації системи

Розробка системи керування рухом маніпулятора SCARA-робота з автоматичним пошуком траси при наявності фізичних перешкод буде проводитись на лабораторному стенді SIEMENS, що оснований на промисловому контролері SIMATIC S7-1200. Для розробки системи керування рухом маніпулятора з автоматичним пошуком траси на стенді Siemens знадобиться певна модернізація, шляхом додання додаткових елементів обладнання. Для реалізації системи було використано ультразвуковий датчик HC-SR04, серводвигун SG90 та плату Arduino UNO. Ультразвуковий датчик використовується для знаходження перешкоди на шляху руху ланок маніпулятора. Серводвигун буде використовується для повертання ультразвукового датчику для подальшого зчитування робочого простору маніпулятора у конусі 150 градусів перед собою. Всі датчики та інші елементи системи сканування робочого простору маніпулятора підключені до плати Arduino UNO, що містить у собі програму для роботи системи сканування та надає живлення(5V) усім підключеним елементам.

Для виконання поставленої задачі було обрано саме ці елементи, оскільки вони мали ряд переваг перед іншими опціями. Дані елементи є досить простими у конструкції і надають відносно непогану точність. Також використовувані елементи є простими для монтажу і експлуатації. Дані елементи не є дорогими, а також були наявними у елементній базі кафедри[16].

Для реалізації системи необхідно зчитувати робочій простір ланок маніпулятора для подальшого коригування руху маніпулятора. Для цього буде використовуватися ультразвуковий датчик HC-SR04 (рис. 2.1).

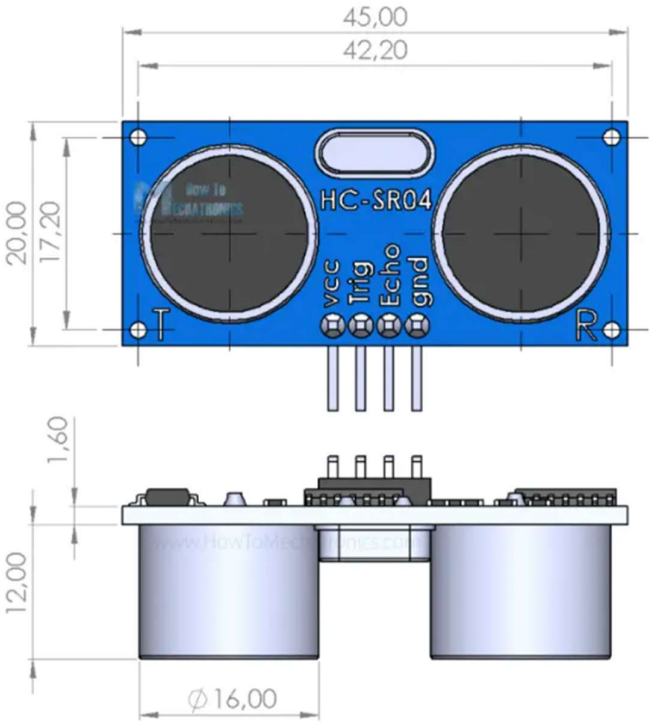

Рисунок 2.1 – Зображення ультразвукового датчику HC-SR04

HC-SR04 являє собою датчик відстані, що використовується для безконтактного вимірювання відстані шляхом випромінення ультразвуковий імпульс і вимірюючи час, потрібний для повернення імпульсу, щоб обчислити відстань до об’єкта.

Датчик складається з двох ультразвукових перетворювачів. Один являє собою передавач, який видає ультразвукові звукові імпульси, а інший є приймачем, який приймає відбиті хвилі ультразвукового імпульсу.

Він випромінює ультразвук із частотою 40 000 Гц, який поширюється в просторі, і при наявності на його шляху є предмету або перешкоди, він повертається до модуля.

Розглянемо характеристики обраного ультразвукового датчику (табл. 2.1).

Таблиця 2.1 – Характеристики ультразвукового датчику

Параметр	Значення параметру
Напруга живлення, В	3.8 … 5.5
Робочий струм, мА	8
Частота, кГц	40
Розміри модуля, мм	37x20x15
Максимальна дистанція, мм	1500
Мінімальна дистанція, мм	0
Ширина імпульсів, мкс	10
Кут	15
Точність, мм	3

Датчик має 4 контакти. VCC і GND підходять до контактів 5V і GND, а Trig і Echo підключаються до будь-якого цифрового контакту. За допомогою штифта Trig ми посилаємо ультразвукову хвилю від передавача, а за допомогою штифта Echo ми приймаємо відбитий сигнал(рис. 2.2).

Розглянемо входи та виходи обраного ультразвукового датчику(табл. 2.2).

Таблиця 2.2 – входи/виходи ультразвукового датчику HC-SR04 [11]

Вхід/вихід	Значення входу/виходу
Trig (Trigger)	контакт для запуску імпульсу
Echo	контакт для прийняття імпульсу
VCC (+)	напруга живлення
GND	«земля»(контакт GND)

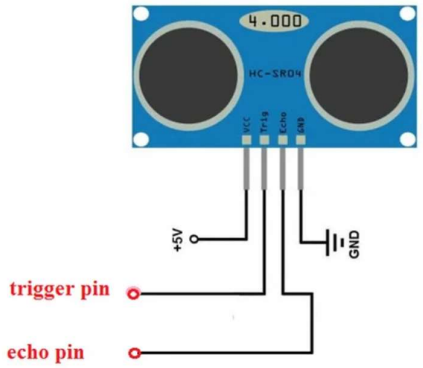

Рисунок 2.2 – Зображення пінів ультразвукового датчику

Для сенсорних елементів використовуються п'єзоелектричні кристали. П'єзоелектричні кристали коливаються на високих частотах коли до них подається електричний сигнал. П'єзоелектричні кристали генеруватимуть електричний сигнал, коли ультразвукова хвиля потрапила на поверхню датчика в зворотному напрямку(рис. 2.3).

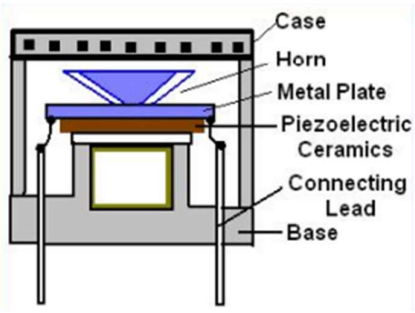

Рисунок 2.3 – Зображення конструкції ультразвукового датчику

SG90 являє собою якісний серводвигун з пусковим моментом 2 кг/див.

Усередині корпусу знаходиться невеликий модуль керування, який під дією вхідного сигналу подає живлення електродвигун. Вхідний сигнал керування має дані про необхідне положення валу. Для визначення положення валу на даний момент часу редуктор з'єднаний із двигуном змінного резистора. Електроніка Tower Pro SG90 визначає різницю поточного положення редуктора та необхідного положення. Модуль управління використовуючи опір змінного резистора надає живлення відповідної полярності на двигун для повороту редуктора, що приводить у відповідність положення передане сигналом управління[17].

Інформація про необхідне положення валу міститься у шпаруватості імпульсів керованого сигналу. Частота керованого сигналу повинна бути постійна і складати 50 Гц (рис. 2.4).

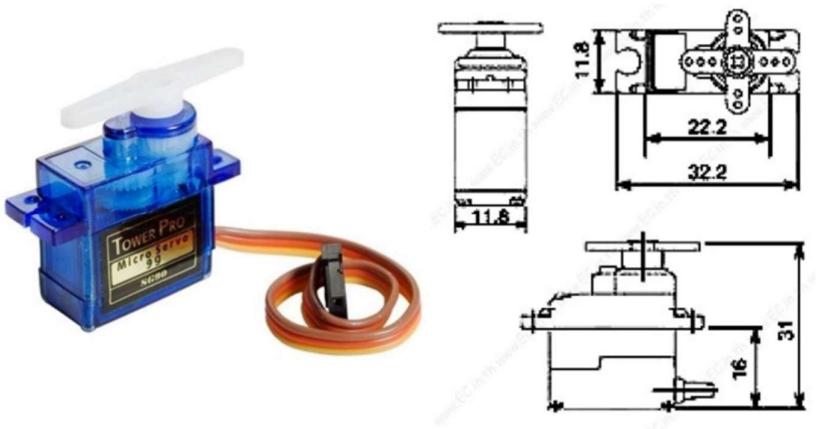

Рисунок 2.4 – Зображення сервоприводу SG90

Розглянемо основні характеристики сервоприводу SG90 (табл. 2.3).

Таблиця 2.3 – Характеристика сервоприводу SG90 [11]

Параметр	Значення параметру
Напруга живлення, В	4.8
Крутний момент зупинки, кгс·см	1.8
Швидкість роботи	0,1 с/60 градусів
Розміри модуля, мм	22.2 x 11.8 x 31
Вага, г	9
Макс.струм у русі, мА	50-80
Макс. струм в утриманні, мА	5-10
Кут повороту, º	180

Датчик має 3 контакти (рис. 2.5).. VCC і GND підходять до контактів 5V і GND. PWM (Pulse Width Modulation) передає сигнал, де робочий цикл імпульсу визначає кут важеля сервоприводу (табл. 2.4).

Таблиця 2.4 – входи/виходи ультразвукового датчику HC-SR04 [11]

Вхід/вихід	Значення входу/виходу
PWM	контакт керування важеля
VCC (+)	напруга живлення
GND	«земля»(контакт GND)

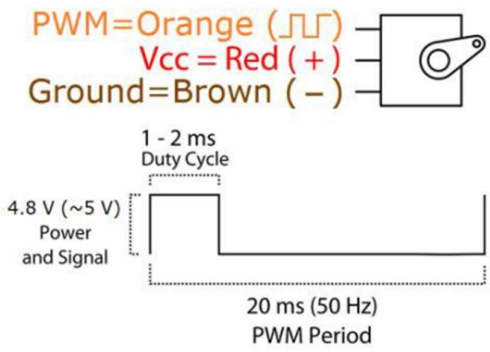

Рисунок 2.5 – Зображення входів сервоприводу SG90

Arduino Uno – це плата для створення прототипів у сфері електроніки та програмування. Вона базується на мікроконтролері ATmega328P, що забезпечує гарну обчислювальну потужність для багатьох проектів. Arduino Uno має 14 цифрових виводів (6 з яких можуть працювати в режимі PWM) і 6 аналогових входів, що дозволяють зчитувати значення з різноманітних сенсорів або взаємодіяти з іншими пристроями.

Плата працює на частоті 16 МГц і оснащена 32 КБ флеш-пам’яті для зберігання програми, з яких 0,5 КБ зайняті завантажувачем. Для тимчасового зберігання даних доступно 2 КБ оперативної пам’яті SRAM і 1 КБ EEPROM для збереження даних, які залишаються навіть після вимкнення живлення.

Arduino Uno підтримує живлення через USB-порт або зовнішній адаптер постійного струму з напругою 7–12 В. Вбудований стабілізатор напруги забезпечує оптимальні умови для роботи плати та підключених пристроїв. Для підключення до комп’ютера використовується USB-інтерфейс на базі мікросхеми ATmega16U2, що дозволяє програмувати плату без додаткових програматорів (рис. 2.6).

На платі також є кнопка для перезавантаження, а також кілька світлодіодів: живлення (PWR) і індикатори активності цифрових пінів (TX і RX). Інтуїтивно зрозуміле програмне середовище Arduino IDE дозволяє писати, компілювати та завантажувати код на плату за допомогою мови програмування, що базується на C/C++[18].

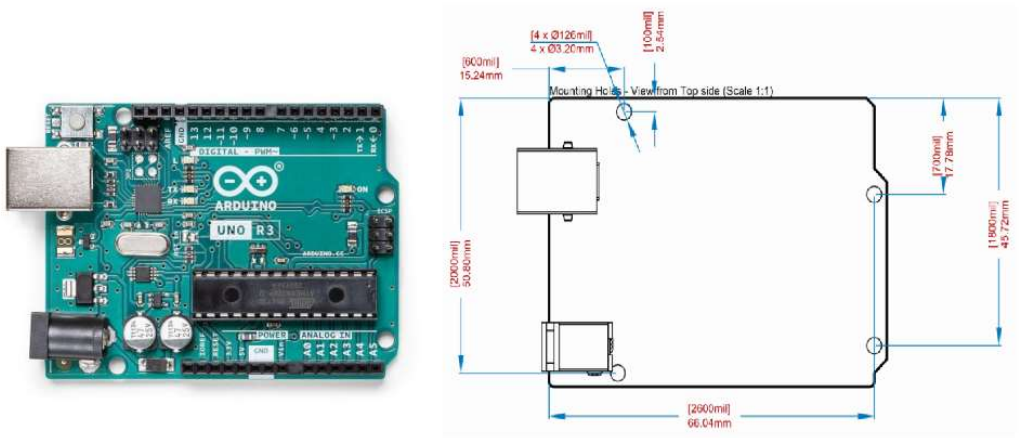

Рисунок 2.6 – Зображення Arduino Uno

Розглянемо характеристики обраного плати Arduino Uno (табл. 2.5).

Таблиця 2.5 – Характеристика Arduino Uno [11]

Параметр	Значення параметру
Напруга живлення, В	7…12
Цифрові входи/виходи	14
Аналогові входи	6
Розміри модуля, мм	660 x 420 x 12.5
Макс. струм одного виводу, мА	40
Макс. струм виводу 3.3V, мА	50
Flash-пам'ять, КБ	32
Тактова частота, МГц	16

Arduino Uno є велика кількість переваг порівняно з іншими аналогами. Плата потребує мінімальну кількість зовнішніх компонентів для початку роботи. Відносно низька вартість порівняно з іншими платами. Наявний широкий вибір шилдів для додавання нових функцій (модулі GSM, Wi-Fi, Bluetooth, дисплеї тощо). Arduino має активну базу користувачів, що надає багато ресурсів, бібліотек і прикладів.

Arduino Uno є універсальним рішенням для реалізації різноманітних проектів, від простих схем зі світлодіодами до складних систем, таких як робототехніка, автоматизація, управління пристроями чи моніторинг датчиків[19].

2.2 Опис системи координат і зони дії елементів маніпулятора

Дослідження було проведено на основі стенду лабораторного стенду SIEMENS. (рис. 1.7, рис. 1.8)

Рисунок 2.7 – Загальний вигляд лабораторного стенду SIEMENS №1.

Рисунок 2.8 – Загальний вигляд лабораторного стенду SIEMENS №1.

Стенд складається з маніпулятору, конвеєрної стрічки, двигуна, контролера SIMANTIC S7-1200, перетворювач частоти SINAMICS G120(силовий модуль PM240-2, блок управління CU240E-2, панель оператора IOP-2).

Маніпулятор стенду складається з наступних ключових частин: башта, стрілка, підйомник та магніт.

Дослідимо елементи маніпулятору та, отримавши їх розмірності, розробимо схему зони дії маніпулятора за допомогою середовища розробки Eplan Education (рис. 2.9-2.11).

Схема зони дії башти та стріли маніпулятора лабораторного стенду(вид зверху):

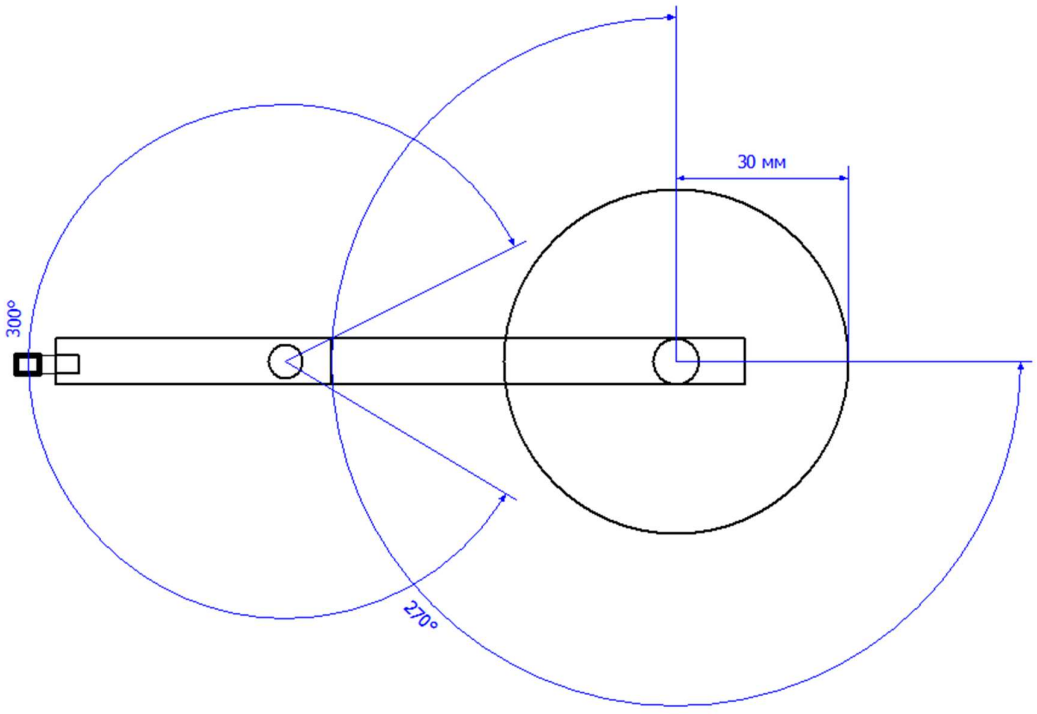

Рисунок 2.9 – Зона дії маніпулятора

Схема зони дії башти та стріли маніпулятора лабораторного стенду(вид збоку):

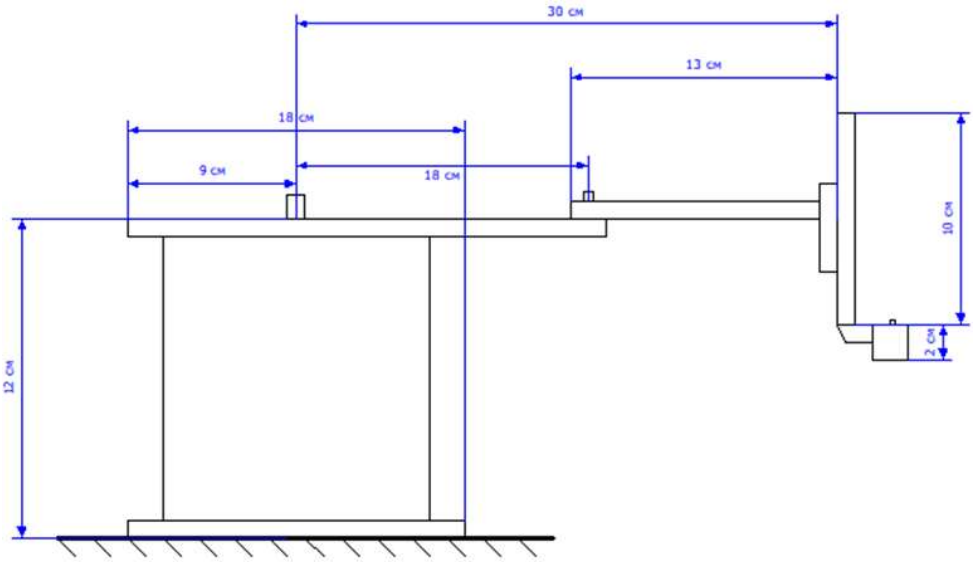

Рисунок 2.10 – Зона дії маніпулятора

Схема зони дії підйомника маніпулятора лабораторного стенду:

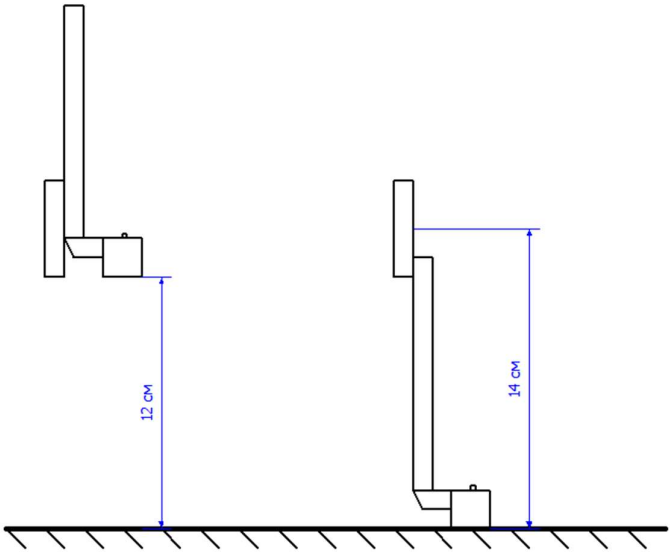

Рисунок 2.11 – Зона дії маніпулятора

Розглянемо макетну схему проекту (рис. 2.12, рис. 2.13). На макетна схема складається з двох сторінок. На першій сторінці зображено вид зверху, а на другій – вид збоку. На схемах представлені усі розмірності основних елементів стенду та їх розмірності відносно один одного. Також на схему додано та наведено розмірності елементів, що були додані з цілю модернізації стенду.

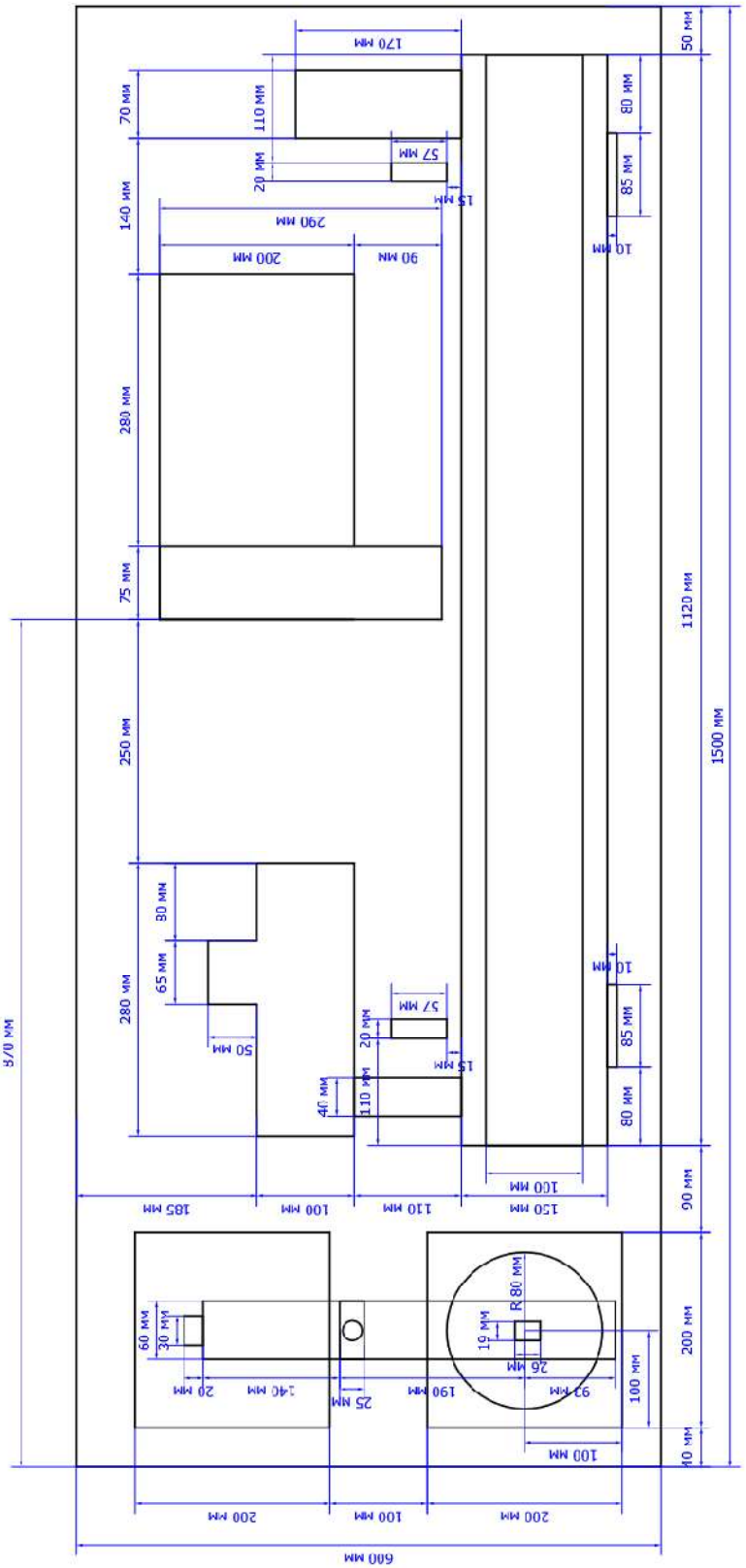

Рисунок 2.12 – Макетна схема(вид зверху)

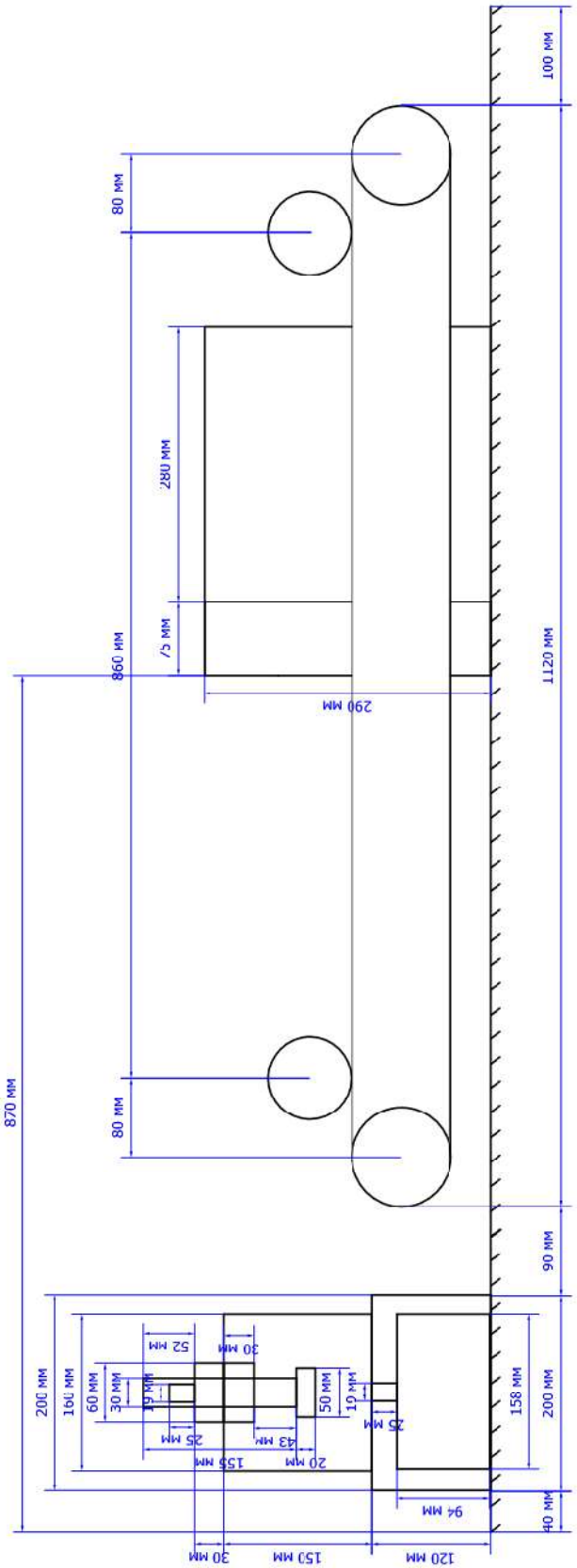

Рисунок 2.13 – Макетна схема(вид збоку)

2.3 Математичне забезпечення

Математична модель системи керування маніпулятором SCARA з ультразвуковими датчиками складається з кількох компонентів, які відображають кінематику маніпулятора, обробку даних від датчиків та прийняття рішень для уникнення перешкод. Нижче описані основні елементи цієї моделі

Кінематична модель маніпулятора SCARA

Маніпулятор SCARA має кілька ступенів свободи, зазвичай 3 або 4, і його кінематика визначається рівняннями, що описують положення кінцевої ефекторної ланки (руху маніпулятора) у просторі.

Пряма кінематика:

Пряма кінематика використовується для визначення кінцевого положення маніпулятора на основі кутів повороту його ланок:

$$x = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2) \quad (2.1)$$

$$y = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2) \quad (2.2)$$

де: x,y – координати кінцевої точки ефектора, l_1,l_2 – довжини першої та другої ланок маніпулятора, θ_1, θ_2 – кути повороту ланок.

Обернена кінематика:

Обернена кінематика дозволяє визначити необхідні кути повороту ланок на основі бажаного кінцевого положення маніпулятора:

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 + l_1^2 + l_2^2}{2 l_1 l_2}\right) \quad (2.3)$$

$$\theta_1 = \arccos\left(\frac{y}{x}\right) - \arctan\left(\frac{l_2 \sin(\theta_2)}{l_1 + l_2 \cos(\theta_2)}\right) \quad (2.4)$$

Ці рівняння використовуються для планування руху маніпулятора.

Модель ультразвукових датчиків

Ультразвукові датчики забезпечують виявлення перешкод на шляху маніпулятора. Дані з кожного датчика можна моделювати у вигляді змінної di, що

відображає відстань до найближчої перешкоди для кожного із ступенів свободи маніпулятора (башта, стріла, підйомник).

Математична модель ультразвукового датчика:

$$d_i = \frac{v_{зв} \times t_i}{2} \quad (2.5)$$

де: di – виміряна відстань до перешкоди (м), vзв– швидкість звуку в повітрі (~343 м/с при кімнатній температурі), ti – час, необхідний для повернення відбитого сигналу до датчика (с).

У разі виявлення перешкоди, коли di менше критичного значення dкр, система активує алгоритм обходу.

Алгоритм обходу перешкод.

Алгоритм обходу перешкод включає в себе перевірку стану кожного датчика та прийняття рішення щодо зміни траєкторії руху. Основний підхід — це перехід у безпечну зону, обхід перешкоди та повернення на початкову траєкторію після її подолання.

Якщо датчик di виявляє перешкоду (di<dкр), то вводиться корекція для кутів повороту маніпулятора:

$$\Delta\theta_1 = f(d_i) \quad (2.6)$$

$$\Delta\theta_2 = f(d_i) \quad (2.7)$$

де f(di) — функція корекції траєкторії, яка базується на відстані до перешкоди і забезпечує маніпулятору безпечний обхід.

Модель системи керування.

Система керування працює на основі даних з ультразвукових датчиків та моделі кінематики маніпулятора. Основна задача системи — забезпечити, щоб маніпулятор досягав кінцевої точки, обминаючи перешкоди. Вона базується на використанні методу управління зворотним зв'язком.

Управління маніпулятором виконується за допомогою під-регулятора, який контролює положення кінцевого ефектора, порівнюючи його з бажаною траєкторією:

$$u(t) = K_p \times e(t) + K_i \times \int e(t)dt + K_d \times \frac{de(t)}{dt} \quad (2.8)$$

де: e(t) – похибка між бажаним і реальним положенням, Kp,Ki,Kd – коефіцієнти ПІД-регулятора.

Визначення бажаних динамічних характеристик та критеріїв оптимальності
Основна задача синтезу системи керування маніпулятором SCARA полягає у досягненні оптимальних динамічних характеристик. Це означає, що необхідно мінімізувати час перехідного процесу, забезпечити високу точність траєкторії та мінімізувати енерговитрати. Як критерій оптимальності використовується інтегральний квадратичний критерій, що описує баланс між похибкою системи та витратами на керування. Формула критерію має вигляд:

$$J = \int_0^\infty (e(t)^2 + pu(t)^2)dt \quad (2.9)$$

У даній системі критерій оптимальності орієнтований на мінімізацію похибки у траєкторії маніпулятора при виявленні перешкод і коригуванні шляху. Важливо, щоб система зберігала високу точність, забезпечуючи швидке та ефективне обходження перешкод.

Моделювання і обмеження системи.

Перед тим, як синтезувати систему керування, необхідно визначити її математичну модель, яка буде основою для подальшої оптимізації. Модель можна подати у вигляді передавальної функції, що відображає динамічні властивості системи. Для спрощення розрахунків передаточна функція маніпулятора SCARA може бути подана у стандартному вигляді:

$$W(s) = \frac{K}{Ts + 1} \quad (2.10)$$

Також слід врахувати обмеження на фізичні параметри системи, такі як:

Максимальна швидкість та прискорення маніпулятора: обмежується механічними властивостями і забезпечує безпечну роботу системи.

Максимальна потужність приводів: обмежує можливість використання сильних перевантажень.

Затримка у роботі ультразвукових датчиків: вимагає врахування часових затримок у системі при обробці даних про перешкоди.

Обмеження можна виразити математично у вигляді нерівностей:

$$|u(t)| \leq u_{max}, |x(t)| \leq u_{max}, |x(t)| \leq a_{max} \tag{2.11}$$

Апроксимація оптимальних характеристик

Оскільки фізичні обмеження зазвичай не дозволяють досягти ідеальних характеристик, на другому етапі проводиться апроксимація. Для цього використовуються регулятори, які дозволяють наблизити систему до бажаних характеристик.

Одним з найбільш поширених підходів для управління такими системами є використання PID-регулятора, передаточна функція якого записується як:

$$W_{PID}(s) = K_p + \frac{K_i}{s} + K_{dS} \tag{2.12}$$

Налаштування цих параметрів дозволяє досягти необхідних динамічних характеристик. Наприклад, збільшення Kp забезпечує більш швидку реакцію системи, тоді як збільшення Kd допомагає зменшити перерегулювання. Інтегральна складова, Ki, відповідає за усунення сталої помилки.

Для забезпечення оптимальної роботи системи керування з урахуванням фізичних обмежень використовуються додаткові корекційні пристрої. Наприклад, фільтри низьких частот можуть бути використані для обробки сигналів з ультразвукових датчиків, щоб уникнути помилкових спрацювань через шум.

Крім того, може бути застосована корекція на основі спостерігачів стану. Спостерігач стану дозволяє оцінити внутрішні стани системи (наприклад, швидкість і положення), які неможливо виміряти безпосередньо. Його динаміка може бути описана наступним рівнянням:

$$x(t) = Ax(t) + Bu(t) + L\big(y(t) - Cx(t)\big) \tag{2.13}$$

Аналіз отриманої системи керування

Завершальний етап синтезу включає в себе аналіз і перевірку отриманої системи. Це можна зробити як через моделювання, так і через експериментальні дослідження.

Для оцінки якості керування можуть бути використані наступні показники:

Максимальне динамічне відхилення (Md): максимальне відхилення фактичної траєкторії від бажаної у перехідному процесі.

Час регулювання (tрег): час, необхідний для стабілізації системи після виявлення перешкоди.

Інтегральний квадратичний критерій (J): загальна величина похибки системи протягом часу.

Перевірка може здійснюватися за допомогою комп'ютерного моделювання, в якому враховуються всі фізичні параметри маніпулятора і датчиків. Якщо система не відповідає пред'явленим вимогам, налаштовуються параметри PID-регулятора або корегувальних пристроїв для досягнення необхідних показників якості.

2.4 Кінематика маніпулятору

Однією з основних задач системи є переміщення ланок маніпулятора. Для виконання різноманітних робочих функцій нам необхідно знати де знаходяться ланки маніпулятора та як саме їх необхідно переміщати у просторі аби досягти необхідного результату у вигляді переміщення робочого органу у необхідну для

нас позицію. Окрім цього це необхідно робити приймаючи до уваги робочу область ланок маніпулятору.

Для вирішення цих задач ми опишемо наш об’єкт керування у вигляді маніпулятора та розв’яжемо обернену задачу кінематики. Для вирішення задачі необхідно розв’язати спочатку пряму, а потім обернену задачу кінематики.

Рішення прямої задачі говорить нам про координати робочого органу маніпулятору по значенню узагальненим координатам та по кінематичній схемі маніпулятора.

Рішення оберненої задачі дає нам величини узагальнених координат по координатам робочого органу та по кінематичній схемі маніпулятора.

Тобто рішення прямої задачі дає нам положення робочого органу маніпулятора при певному положенні його ланок, а оберненої задачі – як маніпулятору потрібно буде рухатись, щоб робочий орган встав у задане положення.

Таким чином побудуємо кінематичну схему маніпулятора, що знадобиться для рішення задач кінематики у вигляді знаходження параметрів маніпулятора (рис. 2.14).

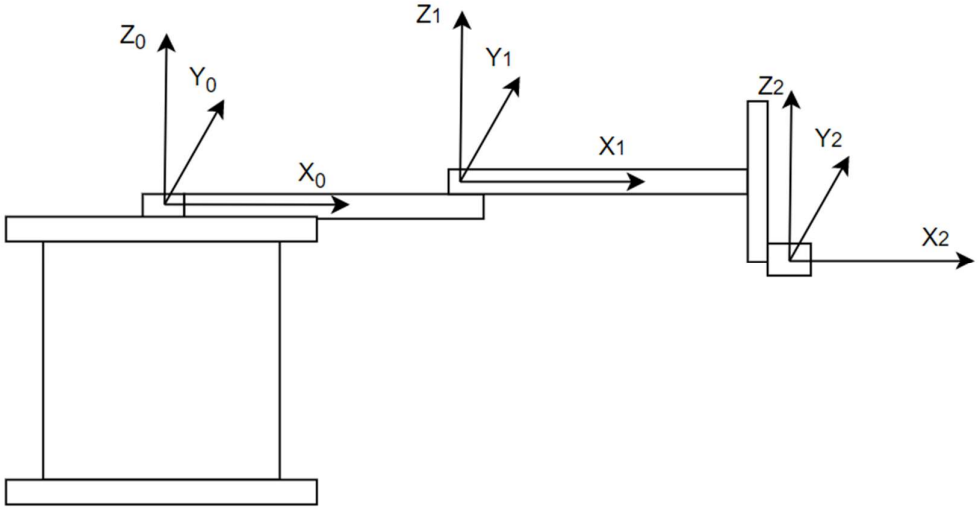

Рисунок 2.14 – Кінематична схема маніпулятора

У нашому випадку у маніпулятора є 3 осі свободи. Башта являє собою першу ось, стріла є другою і підйомник – третю. Також основуючись на конструкції маніпулятора проставимо початкові координати і-их систем.

Основуючись на конструкційних особливостях використовуваного маніпулятора можна сказати, що ми маємо 3 кінематичні пари: дві обертальні (башта та стріла) та одну поступальну (підйомник).

За кінематичною схемою і заданою положенням робочого органу розрахуємо змінні параметри маніпулятора. Досягнемо поставленої задачі шляхом вирішення оберненої задачі кінематики, використовуючи метод матриць.

Положення робочого органу:

$$X = 50;\ Y = 100;\ Z = 30;$$

Спочатку визначимо параметри маніпулятора і занесемо результати у таблицю (табл. 2.6).

Таблиця 2.6 – Параметри маніпулятора

№ кінематичної ланки	Значення параметрів маніпулятора			
	θ	α	r	d
1	θ_1	0	a_1	0
2	θ_2	0	a_2	$-d_2$

Почнемо вирішення прямої задачі кінематики.

Для розв’язання цієї задачі необхідно скласти матриці A_i. Це матриці, що зв’язують між собою системи і-1 та і-ту. У нашому випадку маємо матриці A_1 та A_2 (матриці повороту).

Розрахуємо розширені матриці переходу для кінематичної системи згідно таблиці параметрів маніпулятора.

$$A_1 = \begin{bmatrix} cos\theta_1 & -sin\theta_1 \times cos0 & sin\theta_1 \times sin0 & a_1 \times cos\theta_1 \\ sin\theta_1 & cos\theta_1 \times sin0 & -cos\theta_1 \times sin0 & a_1 \times sin\theta_1 \\ 0 & sin0 & cos0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & a_1 cos\theta_1 \\ sin\theta_1 & cos\theta_1 & 0 & a_1 sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

$$A_2 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 \times cos0 & sin\theta_2 \times sin0 & a_2 \times cos\theta_2 \\ sin\theta_2 & cos\theta_2 \times sin0 & -cos\theta_2 \times sin0 & a_2 \times sin\theta_2 \\ 0 & sin0 & cos0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & a_2 cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2 sin\theta_2 \\ 0 & 0 & 1 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

Далі знайдемо матрицю T_2, що встановить положення і орієнтацію робочого органу. Для знаходження матриці T_2 необхідно перемножити матриці A_1 та A_2.

$$T_2 = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & a_1 cos\theta_1 \\ sin\theta_1 & cos\theta_1 & 0 & a_1 sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & a_2 cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2 sin\theta_2 \\ 0 & 0 & 1 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} cos(\theta_1 + \theta_2) & -sin(\theta_1 + \theta_2) & 0 & cos(\theta_1 + \theta_2)a_2 + cos(\theta_1)a_1 \\ sin(\theta_1 + \theta_2) & cos(\theta_1 + \theta_2) & 0 & sin(\theta_1 + \theta_2)a_2 + a_1 sin(\theta_1) \\ 0 & 0 & 1 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

Таким чином координати робочого органу, що зв'язаний з нульовою ланкою маніпулятора можна отримати за наступними формулами:

$$X_2^0 = a_1 cos(\theta_1) + a_2 cos(\theta_1 + \theta_2) \quad (2.4)$$

$$Y_2^0 = a_1 sin(\theta_1) + a_2 sin(\theta_1 + \theta_2) \quad (2.5)$$

$$Z_2^0 = -d_2 \quad (2.6)$$

Тепер, знаючи рішення прямої задачі, розв'яжемо обернену задачу кінематики.

Беремо три перші елементи четвертого стовпця отриманої матриці T_2 та одержуємо систему трьох рівнянь:

$$\begin{cases} X_2^0 = a_1 cos(\theta_1) + a_2 cos(\theta_1 + \theta_2) \\ a_1 sin(\theta_1) + a_2 sin(\theta_1 + \theta_2) \\ Z_2^0 = -d_2 \end{cases} \quad \begin{cases} 50 = a_1 cos(\theta_1) + a_2 cos(\theta_1 + \theta_2) \\ 100 = a_1 sin(\theta_1) + a_2 sin(\theta_1 + \theta_2) \\ 30 = -d_2 \end{cases} \tag{2.7}$$

$$\begin{cases} 50 = (cos(\theta_1) + cos(\theta_1 + \theta_2)) \times (a_1 + a_2) \\ 100 = a_1 sin(\theta_1) + a_2 sin(\theta_1 + \theta_2) \\ 30 = -d_2 \end{cases} \tag{2.8}$$

За теоремою Піфагора визначимо a_1 та a_2:

$$(a_1 + a_2) = \pm\sqrt{(50)^2 + (100)^2} = \pm\sqrt{2500 + 10^4} = \pm 111{,}8 \tag{2.9}$$

$$50 = cos(\theta_1) + cos(\theta_1 + \theta_2) \times 111{,}8 \tag{2.10}$$

$$cos(\theta_1) + cos(\theta_1 + \theta_2) = 50 \div 111{,}8 = 0{,}4472 \tag{2.11}$$

$$cos(\theta_1) + cos(\theta_1 + \theta_2) = \arccos(0{,}4472) = 1{,}107 \text{ рад} = 63°26' \tag{2.12}$$

2.5 Моделювання системи керування в Matlab

Розв'яжемо пряму та обернену задачу кінематики у середовищі Matlab.

Для цього спочатку визначимо довжину ланок, кути та кінцеве розташування ланок маніпулятора у якості символьних змінних (рис. 2.15).

```
>> syms L_1 L_2 theta_1 theta_2 XB YB
>> L1 = 190;
L2 = 140;
```

Рисунок 2.15 – Визначення довжини ланок

Далі задаємо координати X та Y для робочого органу маніпулятора у якості функції кутів з'єднання θ_1 та θ_2 (рис. 2.16).

```
>> XE_RHS = L_1*cos(theta_1) + L_2*cos(theta_1+theta_2)

XE_RHS =

L_2*cos(theta_1 + theta_2) + L_1*cos(theta_1)

>> YE_RHS = L_1*sin(theta_1) + L_2*sin(theta_1+theta_2)

YE_RHS =

L_2*sin(theta_1 + theta_2) + L_1*sin(theta_1)
```

Рисунок 2.16 – Визначення координат X та Y для робочого органу

Далі враховуючи значення кута ланок, можемо використати пряму задачу кінематики для знаходження положення робочого органу.

Спочатку задаємо вхідні значення для кутів ланок як $-180° < \theta_1 < 90°$ та $-150° < \theta_2 < 150°$ (рис. 2.17).

```
>> t1_degs_row = linspace(-180,90,100);
t2_degs_row = linspace(-150,150,100);
[tt1_degs,tt2_degs] = meshgrid(t1_degs_row,t2_degs_row);
```

Рисунок 2.17 – Визначення вхідних значень для кута ланок

Після цього необхідно конвертувати значення кутів у радіани (рис. 2.18).

```
>> tt1_rads = deg2rad(tt1_degs);
tt2_rads = deg2rad(tt2_degs);
```

Рисунок 2.18 – Переведення значення кутів у радіани

Далі обчислимо координати X та Y за допомогою функцій у Matlab XE_MLF та YE_MLF (рис. 2.19).

```
>> X_mat = XE_MLF(L1,L2,tt1_rads,tt2_rads);
Y_mat = YE_MLF(L1,L2,tt1_rads,tt2_rads);
```

Рисунок 2.19 – Обчислення координат за допомогою Matlab функцій

Тепер, знаючи рішення прямої задачі, розв’яжемо обернену задачу кінематики.

Задаємо рівняння прямої кінематики для знаходження оберненої кінематики та розв’яжемо задачу для θ_1 та θ_2 (рис. 2.20).

```
>> XE_EQ = XE == XE_RHS;
YE_EQ = YE == YE_RHS;
>> S = solve([XE_EQ YE_EQ], [theta_1 theta_2])

S =

  struct with fields:

    theta_1: [2×1 sym]
    theta_2: [2×1 sym]
```

Рисунок 2.20 – Задання рівнянь прямої кінематики та рішення оберненої кінематики

Отримана структура S являє собою структуру з декількома рішеннями для θ_1 та θ_2. Переглянемо пару рішень для θ_1

2.6 Аналіз математичної моделі ультразвукового датчику

Математична модель ультразвукового датчика HC-SR04 може бути описана через основні фізичні принципи поширення звукових хвиль у середовищі. Розглянемо ключові аспекти моделювання ультразвукового датчику.

HC-SR04 працює за принципом вимірювання часу, за який ультразвуковий імпульс проходить до об'єкта й повертається назад [20]. Ультразвуковий датчик передає звукові хвилі на частоті 40 кГц. Ці звукові хвилі потрапляють на об’єкт і повертаються, а приймач HC-SR04 дає вам загальний час, який знадобився звуковим хвилям від передавача, щоб досягти приймача (рис. 2.21).

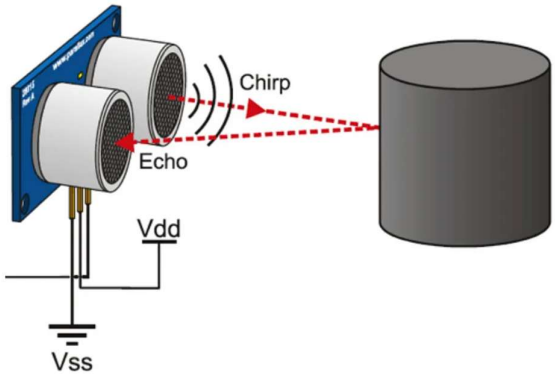

Рисунок 2.21 – Відправка ультразвукового імпульсу

Щоб відправити ультразвуковий імпульс, нам потрібно встановити висновок Trig у високий стан на 10 мкс. Це призведе до імпульсу ультразвуку тривалістю 8 циклів, який поширюватиметься зі швидкістю звуку. Після надсилання 8-циклового ультразвукового випромінювання сигнал Echo переходить у високий стан, і він починає чекати, поки ця хвиля буде повернеться від об’єкта [21]. Якщо немає жодного об’єкта або відбитого імпульсу, контакт Echo вимкнеться через 38 мс і повернеться до низького рівня (рис. 2.22).

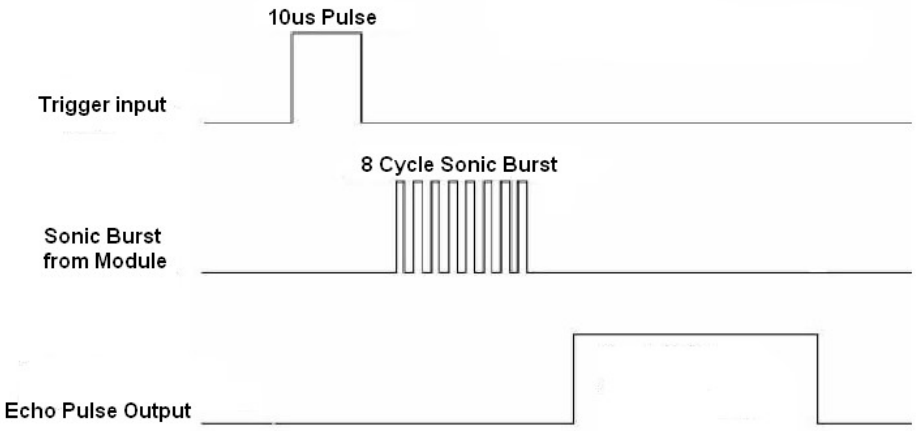

Рисунок 2.22 – Часова діаграма роботи датчика

Таким чином для того шоб виміряти дистанцію до об’єкта нам потрібно знайти дві сутності; швидкість і час. Тому розберемо обчислення для вимірювання відстані в сантиметрах.

$$d = v \times t \tag{2.13}$$

У нас уже є загальний витрачений час, що ми отримали при зчитуванні ультразвукового імпульсу приймачем. Таким чином нам необхідно розділити його на 2, оскільки це загальний час, витрачений звуковими хвилями від передавача до об’єкта та назад до приймача HC-SR04.

Також для вимірювання потрібно означити швидкість ультразвукових хвиль. Швидкість звуку в повітрі залежить від температури T(градуси в Цельсія). Формула розрахунку швидкості буде мати наступний вигляд:

$$v = 331.4 + 0.6 \cdot T \quad (2.14)$$

Швидкість звуку в сухому повітрі при 20 °C становить 343 метри/секунду.

HC-SR04 дасть нам загальний час t у мікросекундах. Тепер для обчислення відстані у сантиметрах, нам необхідно зробити два перетворення: перевести швидкість звуку в сантиметри за секунду та секунди в мікросекунди, оскільки отримане значення виражене в мікросекундах. Швидкість звуку в сухому повітрі буде становити 0,0343 сантиметрів/мікросекунди.

Таким чином формула для знаходження дистанції до об'єкта буде мати наступний вигляд:

$$d = (0.0343 \times t)/2 \quad (2.15)$$

Ще одним з ключових факторів вимірювання дистанції до об'єктів є наявність кута огляду у ультразвукового датчику. HC-SR04 має конусоподібний кут огляду ~15°. Геометрично, зона покриття є частиною кулі:

$$r = d \times \tan\left(\frac{\theta}{2}\right) \quad (2.16)$$

де θ – кут огляду, d – відстань до об'єкта.

Також у ультразвукового датчика HC-SR04 є різноманітні похибки, які необхідно вичитувати при роботі з ним. До фізичних похибок можна віднести температурну похибку. Якщо температура змінюється, але модель використовує постійне v=343 м/с, що призводить до систематичних відхилень. Таким чином можна уникнути цього використовуючи формулу 2.14 для поправки. Також вологість повітря може дещо збільшити швидкість звуку, а також хвилі можуть розсіюватися, якщо поверхня об'єкта неоднорідна [22].

Окрім цього необхідно приділяти увагу часу реакції апаратної частини датчика. Так, наприклад, електроніка датчика має затримку у вимірюванні $T_{trigger}$. Це додає фіксовану похибку Δt [23].

$$t_{corr} = t_{echo} - \Delta t \quad (2.17)$$

Може бути присутня кутова похибка при якій відбиття хвиль під непрямими кутами може спотворити реальну відстань. Відбитий сигнал може містити шум через сторонні джерела звуку [24].

2.7 Обґрунтування вибору програмного забезпечення для реалізації системи.

Під час вибору програмного забезпечення для реалізації автоматизованої системи пошуку траси з перешкодами для руху маніпулятора з дистанційним керуванням були враховані декілька ключових факторів, що визначили вибір конкретних інструментів. Даний вибір має на меті забезпечити оптимальну функціональність, надійність та інтегрованість системи.

Tia Portal був обраний для програмування контролерів Siemens, що використовуються у системі керування маніпулятором. Це рішення обумовлене його високою сумісністю із апаратною частиною Siemens, що забезпечує ефективне управління мехатронічними пристроями. Tia Portal включає у себе інтегровані засоби для програмування, налагодження та моніторингу контролерів, що значно спрощує розробку і підтримку системи.

Основні переваги використання Tia Portal включають:

Інтегроване середовище розробки: Tia Portal забезпечує єдине середовище для програмування, налаштування та діагностики контролерів Siemens, що спрощує розробку та підтримку системи.

Підтримка промислових стандартів: Відповідність Tia Portal стандартам Industry 4.0 забезпечує сумісність і інтеграцію з іншими цифровими технологіями.

Ефективне керування проектами: Можливість організації проектів і управління конфігураціями дозволяє забезпечити структурованість і зручність у розробці.

WinCC було обрано для розробки людсько-машинного інтерфейсу (HMI), що використовується для візуалізації даних і керування системою. Основні переваги використання WinCC включають:

Гнучкість і налаштування: WinCC надає можливості для створення графічного інтерфейсу згідно з вимогами проекту, включаючи різноманітні елементи управління та відображення стану системи.

Підтримка мультиплатформенності: WinCC дозволяє розробляти інтерфейс, який може бути доступний на різних пристроях і платформах, що підвищує його універсальність.

Інтеграція з Tia Portal: Інтеграція між WinCC і Tia Portal спрощує обмін даними і забезпечує єдиною точкою доступу для відображення стану маніпулятора.

Для реалізації функціональності дистанційного керування було використано Node-Red, що є потужним інструментом для створення веб-інтерфейсів та інтеграції різноманітних систем. Використання Node-Red дозволяє швидко розробляти та впроваджувати веб-додатки для моніторингу і керування системою з будь-якого пристрою, підвищуючи доступність та зручність управління.

Основні переваги використання Node-Red включають:

Гнучкість в розробці програм: Node-Red дозволяє створювати програми з використанням візуального програмування, що спрощує створення та зміну логіки керування.

Інтеграція з IoT: Вбудовані засоби для роботи з протоколами IoT дозволяють легко інтегрувати систему з дистанційними пристроями і сервісами.

Масштабованість і розширюваність: Node-Red підтримує розширення функціональності за допомогою розширень (nodes), що дозволяє відповідати змінним вимогам проекту.

Обидва інструменти вибирались через їх здатність до інтеграції з існуючим обладнанням та системами. Це важливо для забезпечення стабільності і сумісності всієї системи, а також для зниження часу впровадження нових технологій у виробничому середовищі.

Висновки за розділом

За результатами розділу було обрано та досліджено елементи для модернізації лабораторного стенду SIEMENS для розробки системи керування рухом маніпулятора з автоматичним пошуком траси при наявності фізичних перешкод. Для цього було прийнято рішення про додання двох плати Arduino Uno, серводвигуна та ультразвукового датчику.

Було розроблено схему системи координат і зони дії елементів маніпулятору.

Було досліджено математичне забезпечення системи у вигляді рішення прямих та обернених задач кінематики для досліджуваного маніпулятору з трьома осями свободи. Рішення прямої задачі дало координати робочого органу маніпулятору по значенню узагальненим координатам та по кінематичній схемі маніпулятора. Рішення оберненої задачі дало величини узагальнених координат по координатам робочого органу та по кінематичній схемі маніпулятора. Також вирішення задач кінематики було проведено у середовищі розробки Matlab.

Досліджено математичну модель ультразвукового датчику. Виведено формули для розрахунку дистанції до об’єкта на базі отриманого часу за який ультразвуковий імпульс повернувся до датчику.

Було обґрунтовано вибір програмних пакетів для розробки системи в результаті чого було обрано Tia Portal, Node Red, Arduino IDE та Processing. Обрані програмні застосунки чудово підходять для даного проекту через свою простоту, надійність та функціональність. Вони мають великий функціонал і найкраще підходять для розробки системи на базі обраних апаратних елементів.

РОЗДІЛ 3
ПРОГРАМНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ ПРОЦЕСОМ

3.1 Структурна схема роботи системи

Розберемо структурну схему проекту (рис. 3.1). Основними елементами нашої системи для вирішення поставленої проблеми є ПЛК SIMATIC S7-1200, маніпулятор та ультразвуковий датчик, серводвигун, плата Arduino Uno та персональний комп'ютер. Маніпулятор ж складається з шести елементів у вигляді трьох двигунів та трьох драйверів шагових двигунів, що керують наведеними двигунами маніпулятора.

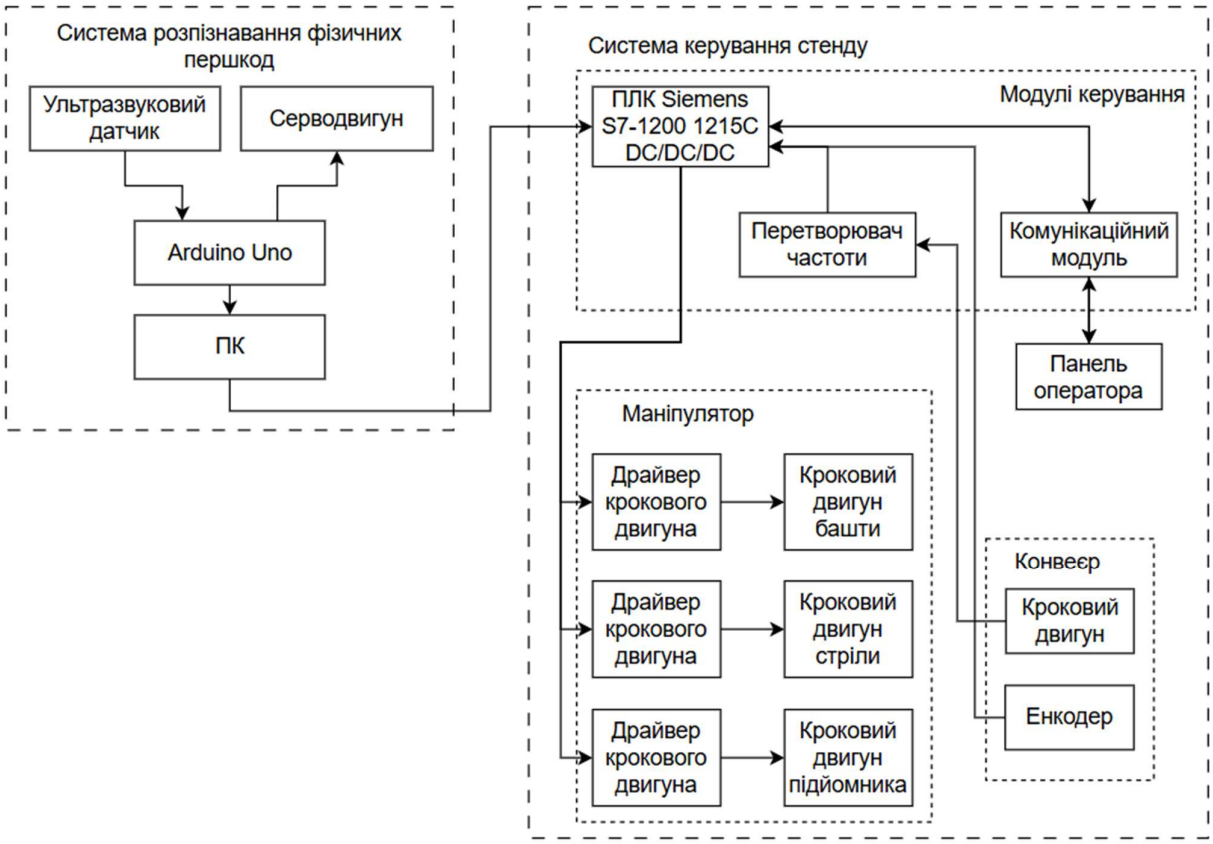

Рисунок 3.1 – Структурна схема системи

Таким чином ультразвуковий датчик та серводвигун під’єднані до плати Arduino, де ультразвуковий датчик передає и приймає сигнали з плати, а серводвигун отримує команди з плати. Плата Arduino підключена до ПК для зв’язку її з ПЛК стенду.

ПЛК S7-1200 отримує дані з ПК для обробки і може використовувати їх для корекції руху маніпулятору, шляхом посилання коригованих сигналів на драйвери крокових двигунів, що керують кроковими двигунами ланок маніпулятору. Також, за необхідності, ПЛК може коригувати інші елементи системи керування стенду.

3.2 Макетна схема системи розпізнавання перешкод

Для коректної роботи системи необхідно з’єднати усі ключові елементи. Таким чином для системи розпізнавання фізичних перешкод на шляху маніпулятора критичними елементами системи є Arduino Uno, ультразвуковий датчик та серводвигун.

Для підключення ультразвукового датчика HC-SR04 та серводвигуна SG90 до мікроконтролера Arduino Uno використовується схема, що наведена нижче(рис. 3.2).

HC-SR04 має чотири основні контакти: VCC, GND, Trig та Echo. VCC підключається до 5V на Arduino Uno для живлення датчика. GND з’єднується з GND на платі Arduino. Trig (тригерний контакт) підключається до цифрового піну Arduino D10. Цей контакт використовується для подачі імпульсу тривалістю 10 мікросекунд, який активує датчик для випромінювання ультразвукової хвилі.

Echo (контакт отримання сигналу) з'єднується з іншим цифровим піном ArduinoD11. На цьому контакті фіксується тривалість повернення відбитого сигналу, яка пропорційна відстані до об’єкта.

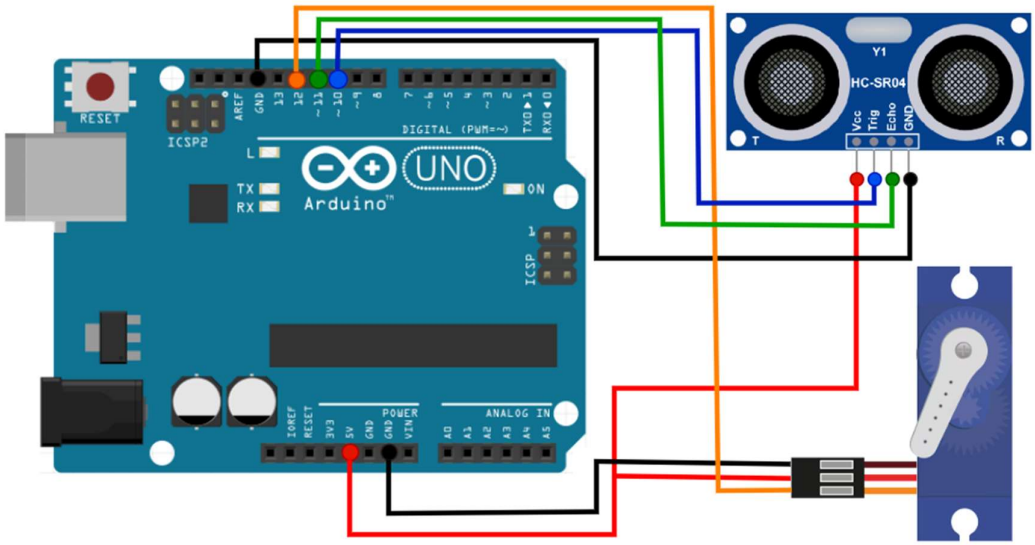

Рисунок 3.2 – Макетна схема системи

Серводвигун SG90 має три контакти: VCC, GND та Signal. VCC підключається до 5V на Arduino Uno. Серводвигун можна живити безпосередньо від Arduino, якщо він використовується для невеликих навантажень. У випадку важчого навантаження слід використовувати зовнішнє джерело живлення, щоб уникнути перевантаження Arduino.

GND з'єднується з GND на Arduino Uno (або з GND зовнішнього джерела живлення, якщо використовується окреме живлення).

Signal (сигнальний контакт) підключається до одного з цифрових виходів Arduino з підтримкою ШІМ D12. Цей контакт використовується для подачі сигналу керування, що визначає кут повороту серводвигуна.

3.3 Алгоритм роботи системи

Метою системи є знаходження оптимального шляху SCARA-робота маніпулятора для уникнення фізичних перешкод на шляху його ланок. У результаті розробки було отримано систему, що може сканувати робочу область маніпулятору лабораторного стенду, виявляти наявність та положення фізичних

перешкод у цій зоні та має змогу відправляти отримані дані лабораторному стенду для обробки. Також розроблено програму у Node-Red для оброки отриманих даних та подальшої відправки результатів до ПЛК лабораторного стенду через OPC. Розроблено програму у Tia Portal для коригування руху маніпулятора для уникнення перешкод на шляху його ланок.

Далі наведено блок-схему алгоритму керування системи навчання промислового маніпулятора (рис. 3.3). На блок-схемі у графічному вигляді детально наведено послідовність дій, що необхідні для ефективної роботи системою.

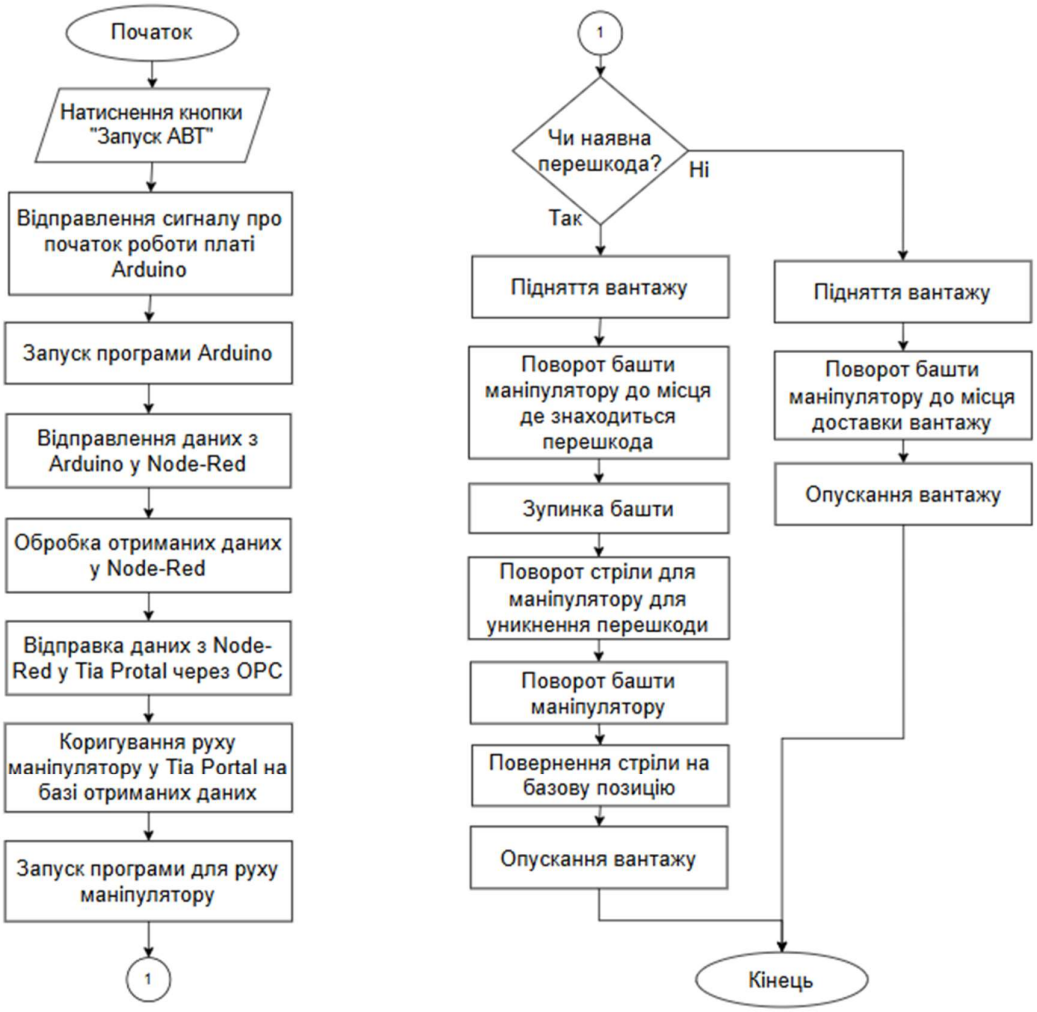

Рисунок 3.3 – Алгоритм роботи системи

При початку роботи програми програма у Tia Portal передає повідомлення у Node-Red стосовно початку роботи програми системи, яке в свою чергу передається до плати Arduiono. При надходженні цього сигналу починається робота програма плати.

Система управління маніпулятором побудована на основі взаємодії кількох ключових компонентів. Виявлення перешкод у робочому просторі здійснюється за допомогою ультразвукового датчика, встановленого на серводвигуні. Серводвигун забезпечує обертання датчика для сканування простору у 150 градусів. Отримані дані з ультразвукового датчика передаються на мікроконтролер Arduino Uno, який обробляє ці дані та визначає наявність і координати перешкод.

Далі Arduino передає інформацію на персональний комп'ютер у Node-Red, який виконує додаткову обробку отриманих даних та координує обмін інформацією між Arduino та програмованим логічним контролером (ПЛК) Siemens S7-1200. При надходженні даних з плати Arduino, Node-Red виконує подальшу обробку цих даних в результаті чого система отримує позицію, що відповідає початку перешкоди та позицію, що являє собою кінець перешкоди. Далі отримані дані передаються до ПЛК, де програма контролера за необхідності змінює траєкторію ланок маніпулятора для перенесення вантажу на інший конвеєр.

ПЛК виступає основним контролером, який приймає оброблені дані про перешкоди та формує відповідні команди для керування маніпулятором.

Для реалізації руху маніпулятора ПЛК надсилає сигнали на драйвери крокових двигунів, які, у свою чергу, керують переміщенням ланок маніпулятора. Це дозволяє виконувати необхідні маневри, враховуючи розташування перешкод у робочому просторі. Усі компоненти системи з'єднані в єдину інтегровану структуру, де кожен елемент виконує свою роль, забезпечуючи ефективне функціонування маніпулятора.

Така система дозволяє сканувати робочий простір, виявляти перешкоди, та точно керувати положенням ланок маніпулятора в режимі реального часу.

3.4 Огляд програмного забезпечення системи

3.4.1 Огляд програмного заебзпечення в Arduino

Система розпізнавання фізичних перешкод на шляху маніпулятора базується на взаємодії кількох основних компонентів. Для виявлення перешкод у робочій зоні використовується ультразвуковий датчик, закріплений на серводвигуні. Серводвигун здійснює обертання датчика, забезпечуючи сканування простору в межах 150 градусів. Дані, отримані з ультразвукового датчика, передаються на мікроконтролер Arduino Uno, який аналізує їх і визначає наявність та координати перешкод.

Для реалізації цього процесу була розроблена програма у Arduino IDE. Розберемо принцип роботи програми.

Спочатку маємо частину коду для підготовки роботи програми(рис. 3.4).

```
// Includes the Servo library
#include <Servo.h>.

// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;

Servo myServo; // Creates a servo object for controlling the servo motor

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor attached

}
```

Рисунок 3.4 – Код підготовки програми та ініціалізації змінних

У даній частині відбувається підготовки програми та ініціалізації змінних. Для роботи з сервоприводом використовується бібліотека Servo.h. Далі йде оголошення пінів для ультразвукового датчика, де 10 пін є Вихідний пін

ультразвукового датчика (Trig), а пін 11 – Вхідний пін ультразвукового датчика (Echo), який приймає відбиті ультразвукові сигнали.

Оголошуємо змінні для збереження обчисленої відстані до об’єкта та для збереження тривалості ультразвукового імпульсу (в мікросекундах), що потрібна для обчислення відстані.

Далі створюємо об’єкт для надсилання сигналів на сервопривод і управління його кутовим положенням та встановлюємо налаштування пінів. Так trigPin встановлюється як вихід для генерації ультразвукових імпульсів, а echoPin встановлюється як вхід для приймання відбитих сигналів.

В кінці встановлюємо підключення до серійний порт для надсилання даних через нього та підключаємо сервопривід до 12 піна.

Далі розглянемо основну частину програми плати Arduino (рис. 3.5).

```
void loop() {

 if (Serial.available() > 0) {
   // Read the incoming data as a string
   String TiaPortalStart = Serial.readStringUntil('\n'); // Read
   if(TiaPortalStart = "StartProgram"){
     // rotates the servo motor from 15 to 165 degrees
     for(int i=15;i<=165;i++){
       myServo.write(i);
       delay(30);
       distance = calculateDistance();// Calls a function for ca
       Serial.print(i);
       Serial.print(",");
       Serial.print(distance);
       Serial.print(".");
       }
     }
     myServo.write(90);
   }
 }
}
```

Рисунок 3.5 – Програма сканування робочого простору маніпулятора

Так як програма повинна працювати лише при початку роботи програми у Tia Protal, то через Node Red у серійний порт передається змінна для початку роботи програми плати. Для цього ми перевіряємо, чи є нові дані в серійному

буфері і зчитуємо дані у вигляді рядка до символу нового рядка (\n). Якщо зчитаний рядок дорівнює StartProgram, запускається програма управління.

Далі починається рух сервоприводу. Серводвигун обертається від 15 до 165 градусів, зупиняючись кожні 30 мс. На кожному куті викликається функція calculateDistance(рис. 3.6), яка обчислює відстань до об'єкта. Поточний кут (i) та виміряна відстань передаються через серійний порт. В кінці сервопривод повертається у центральне положення (90 градусів) після завершення циклу.

```
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound v
  distance= duration*0.034/2;
  return distance;
}
```

Рисунок 3.6 – Функція розрахунку відстані до об’єкту

Як вже було розглянуто у другому розділі, нам необхідно написати програму для того, щоб розраховувати дистанцію до об’єкту.

Перші п’ять строк функції коротко встановлює trigPin на HIGH (10 мкс) для генерації ультразвукового імпульсу.

Далі функція pulseIn вимірює тривалість сигналу на піні echoPin у мікросекундах. Тривалість пропорційна відстані до об'єкта.

Після чого ми використовуємо формулу отриману до цього для розрахунку дистанції до об’єкту. Формула перетворює тривалість сигналу в дистанцію, де Швидкість звуку ≈ 0.034 см/мкс та ділимо отримане значення на 2, оскільки імпульсу потрібен був час на проходження до об’єкту і назад.

3.4.2 Огляд програмного забезпечення в Processing

Цей код на Processing візуалізує дані, що надходять із Arduino (або іншого пристрою) через серійний порт, для створення "радарного" дисплея. Основні функції коду такі: отримання даних із серійного порту, обробка кутів та відстаней, і графічне представлення цих даних у вигляді радарного дисплея. Детально розберемо кожну частину (рис. 3.7).

```
void setup() {

 size (1280, 720); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
 smooth();
 myPort = new Serial(this,"COM3", 9600); // starts the serial communication
 myPort.bufferUntil('.'); // reads the data from the serial port up to the ch
 //orcFont = loadFont("OCRAExtended-30.vlw");
}
```

Рисунок 3.7 – Функція setup

Встановлюємо розмір вікна. Користувачеві можна змінити цей розмір відповідно до екрану. Встановлюємо з’єднання з серійним портом та читаємо дані із порту до символу «.».

Далі розглянемо функцію serialEvent, що використовується для перетворення потоку даних з плати Arduino та запису отриманого значення у відповідні змінні (рис. 3.8).

```
void serialEvent (Serial myPort) { // starts reading data from the Serial Port
  // reads the data from the Serial Port up to the character '.' and puts it into t
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the variab
  angle= data.substring(0, index1); // read the data from position "0" to position
  distance= data.substring(index1+1, data.length()); // read the data from position

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}
```

Рисунок 3.8 – Функцію для обробки даних

В програмі присутні інші функції для відображення загальних ліній радару та інших елементів інтерфейсу [25]. Розглянемо частину коду, що відповідає за відображення об’єкта, що знаходиться в зоні дії ультразвукового датчика. Вона малює червону лінію на екрані, яка вказує місце розташування об'єкта залежно від кута та відстані (рис. 3.9).

```
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}
```

Рисунок 3.9 – Відображення об’єкту на радарі

Спочатку зберігаємо поточну матрицю координат. Це потрібно для ізоляції трансформацій (переміщення та обертання). Переміщуємо систему координат так, щоб початок (0, 0) знаходився внизу екрана по центру радіолокаційного кола.

Далі необхідно перетворити відстань в пікселі для коректного відображення об’єкту на радарі. Для цього беремо виміряну ультразвуковим сенсором відстань до об’єкта в сантиметрах та перемножаємо її на максимальну висоту в пікселях, яка відповідає радіусу радіолокаційного кола та коефіцієнт масштабування, який переводить сантиметри в пікселі.

Далі можемо приступати до малювання об’єкта на радарі. Для цього спочатку перевіряємо, чи об’єкт знаходиться в межах 40 см від сенсора. Якщо ні, об'єкт не буде намальовано.

pixsDistance * cos(radians(iAngle)) та -pixsDistance * sin(radians(iAngle)) обчислюють координати початкової точки лінії на основі кута iAngle (кут обертання радіолокаційного датчика) та відстані pixsDistance.

Cos і sin переводять кут у радіани, щоб обчислити положення об'єкта в полярних координатах. Знак «-» у координаті «y» враховує перевернуту вісь «y» у системі координат Processing.

Координати кінця лінії, які відповідають максимальному радіусу радіолокаційного кола розраховуємо завдяки наступному коду (width - width * 0.505) * cos(radians(iAngle)). Після чого малюємо лінію від точки, що відповідає відстані pixsDistance, до краю радіолокаційного сектора за допомогою команди line().

Таким чином маємо вікно радару де якщо об'єкт знаходиться в далі 40 см, на радіолокаційному екрані малюється зелена лінія, яка вказує положення об'єкта залежно від кута та відстані. Лінія починається від центру радіолокаційного кола і тягнеться до місця, де знаходиться об’єкт (рис. 3.10).

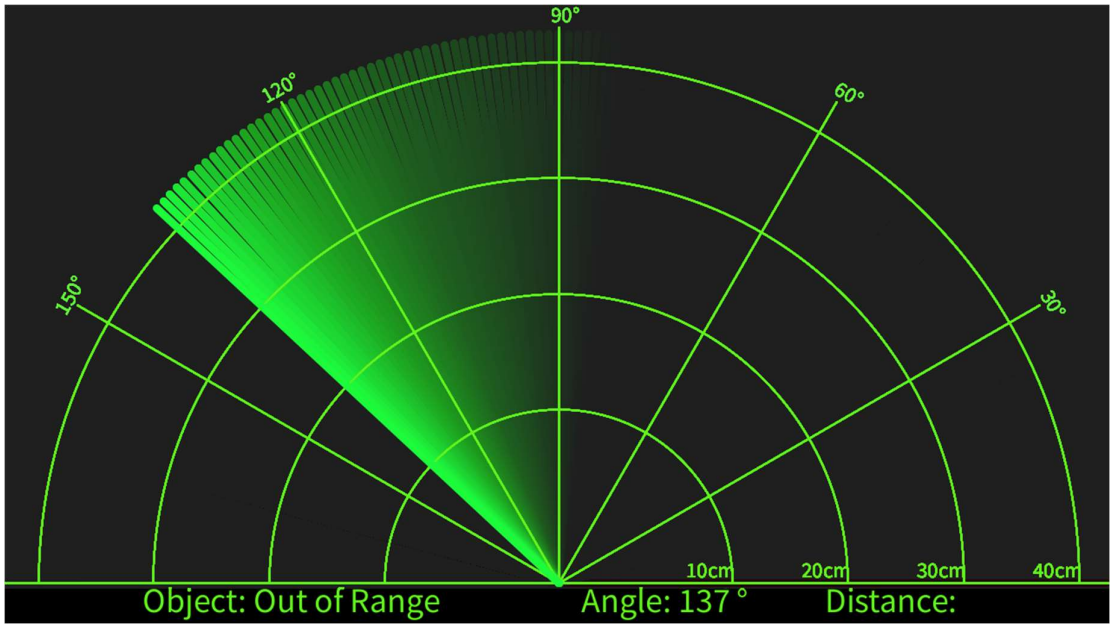

Рисунок 3.10 – Зображення роботи радару без перешкоди

Таким чином маємо вікно радару де якщо об'єкт знаходиться в межах 40 см, на радіолокаційному екрані малюється червона лінія, яка вказує положення об'єкта залежно від кута та відстані. Якщо перешкода знаходиться на деякій відстані, то на відстані, де немає перешкоди, малюється зелена лінія, а після початку перешкоди до кінця радару малюється червона лінія Лінія починається від центру радіолокаційного кола і тягнеться до місця, де знаходиться об’єкт (рис. 3.11).

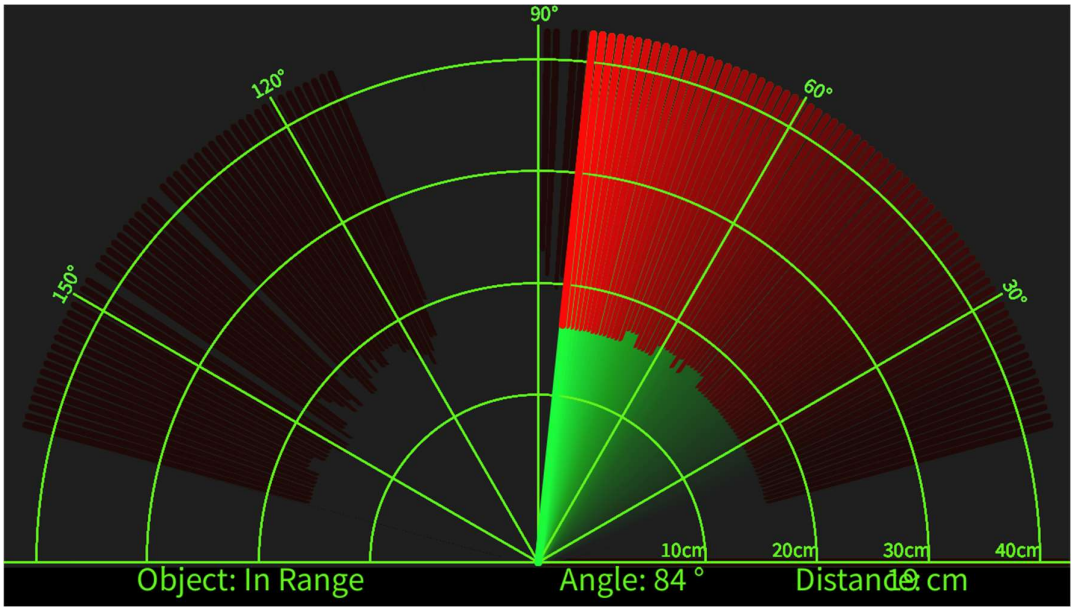

Рисунок 3.11 – Зображення роботи радару при наявній перешкоді

3.4.3 Огляд програмного забезпечення Node Red та UaExpert

Для того, щоб передати отримані з плати Arduino дані у Tia Portal для коригування руху траси маніпулятора потрібно передавати їх через Node Red та OPC server. Таким чином було розроблено код у Node Red, який вирішує дану проблему (рис. 3.12).

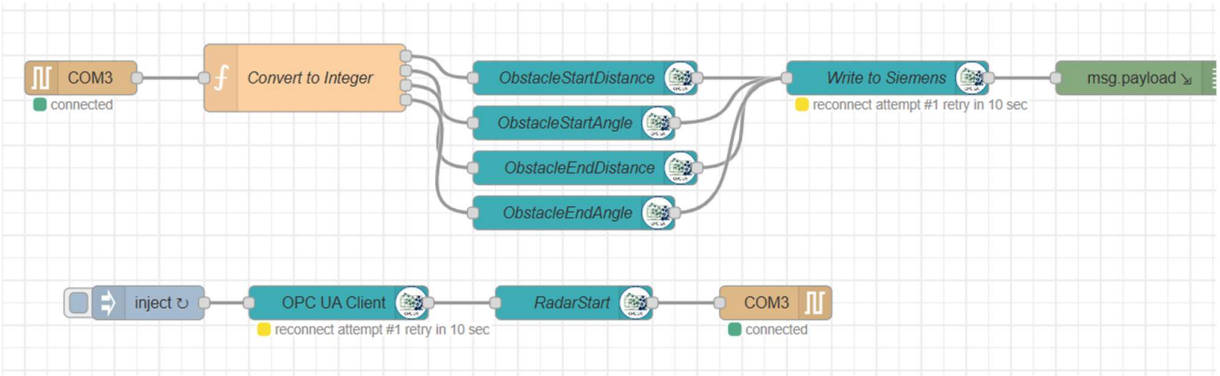

Рисунок 3.12 – Програма у Node Red

Для початку небхідно підключитись до серійного порту для того, щоб зчитувати з нього інформацію, яку туди передає плата. Для цього використовується Serial In Node. Ці дані передаються в Function node для

подальшої оброки, оскільки з серійного порту приходить поток даних, які ми не можемо використовувати в такому вигляді. Приклад частини даних з серійного порту матиме насупний вигляд:

«15.148,16.148,17.148,18.19,19.18,20.15,21.14,22.30,23.72»

Тому була розроблена програма, яка виділяє з цього потоку даних необхідну нам інформацію. Цей код у вузлі Function Node у Node-RED обробляє потік даних із радара, розраховує параметри знайденого об’єкта (кут і дистанцію на початку та в кінці), а потім надсилає ці параметри на різні виходи вузла [26].

Розглянемо детальніше роботу цієї програми (рис. 3.13).

```
1  // Отримуємо дані з msg.payload
2  let input = msg.payload.trim(); // Видаляємо зайві пробіли, якщо є
3
4  // Розділяємо дані на масив пар "кут.дистанція"
5  let dataPairs = input.split(',');
6
```

Рисунок 3.13 – Розділення потоку даних

input.trim() використовується для видалення зайвих пробілів з початку та кінця рядка. За допомогою input.split(',') рядок розділяється на окремі пари, що представляють кут і дистанцію (наприклад, "15.148") (рис. 3.14).

```
// Проходимо по всіх парах даних
dataPairs.forEach(pair => {
    let [angle, distance] = pair.split('.').map(Number); // Розбиваємо на кут та дистанцію
```

Рисунок 3.14 – Проходження всіх пар даних

Кожна пара розділяється на дві складові за допомогою split('.') — кут (angle) та дистанція (distance), які перетворюються в числа (map(Number)).

Після того як ми розділили отримані дані, ми можемо почати виділяти з нього необхідну нам інформацію у вигляді наявності перешкоди в робочій області маніпулятору (рис. 3.15).

```
19      // Перевіряємо, чи дистанція менше 20 см
20      if (distance < 20) {
21          if (!currentObject) {
22              // Починається новий об'єкт
23              currentObject = {
24                  startAngle: angle,
25                  startDistance: distance,
26                  endAngle: angle,
27                  endDistance: distance
28              };
29          } else {
30              // Продовження поточного об'єкта
31              currentObject.endAngle = angle;
32              currentObject.endDistance = distance;
33          }
34      } else {
35          if (currentObject) {
36              // Об'єкт завершено
37              ObstacleStartAngle = currentObject.startAngle;
38              ObstacleStartDistance = currentObject.startDistance;
39              ObstacleEndAngle = currentObject.endAngle;
40              ObstacleEndDistance = currentObject.endDistance;
41              currentObject = null;
42          }
43      }
44  });
```

Рисунок 3.15 – Ідентифікація об'єкта

Спочатку перевіряємо чи дистанція менша за 20 см. Якщо це перша точка нового об'єкта, фіксується початковий кут (startAngle) і початкова дистанція (startDistance). кщо це продовження поточного об'єкта, оновлюється кінцевий кут (endAngle) і кінцева дистанція (endDistance).

Якщо дистанція більша або дорівнює 20 см і є поточний об'єкт, то він вважається завершеним, і його параметри зберігаються в змінних: ObstacleStartDistance, ObstacleStartAngle, ObstacleEndDistance, ObstacleEndAngle.

Також якщо останній об'єкт не завершився, але існує, його параметри також зберігаються (рис. 3.16).

```
if (currentObject) {
    // Об'єкт завершено
    ObstacleStartAngle = currentObject.startAngle;
    ObstacleStartDistance = currentObject.startDistance;
    ObstacleEndAngle = currentObject.endAngle;
    ObstacleEndDistance = currentObject.endDistance;
    currentObject = null;
}
```

Рисунок 3.16 – Завершення об’єкта

Далі для подальшої передачі результатів роботи програми необхідно сформувати повідомлення (рис. 3.17).

```
55  // Формуємо повідомлення для кожного виходу
56  let msg1 = { payload: ObstacleStartDistance }; // Для першого виходу
57  let msg2 = { payload: ObstacleStartAngle };    // Для другого виходу
58  let msg3 = { payload: ObstacleEndDistance };   // Для третього виходу
59  let msg4 = { payload: ObstacleEndAngle };      // Для четвертого виходу
60
61  // Повертаємо масив повідомлень
62  return [msg1, msg2, msg3, msg4];
```

Рисунок 3.17 – Створення повідомлення

Створюється окреме повідомлення (msg) для кожного параметра:

msg1 містить початкову дистанцію (ObstacleStartDistance).

msg2 містить початковий кут (ObstacleStartAngle).

msg3 містить кінцеву дистанцію (ObstacleEndDistance).

msg4 містить кінцевий кут (ObstacleEndAngle).

Код повертає масив [msg1, msg2, msg3, msg4], де кожен елемент масиву направляється на відповідний вихід вузла.

Далі нам необхідно передати ці дані у UaExpert для подальшої передачі у TiaPortal. Для цього використовуються елементи OpcUa-Item node та OpcUa-Client node.

Так кожену змінну передаємо у відповідний OpcUa-Item node. Цей вузол є проміжним елементом для обробки даних, які передаються між вузлами Node-RED і сервером OPC UA. Він служить для підготовки даних до передачі через клієнтський вузол (рис. 3.18).

Рисунок 3.18 – Налаштування одного з OpcUa-Item

Далі ми передаємо дані до OpcUa-Client. Цей вузол відповідає за комунікацію з сервером OPC UA, передаючи або отримуючи дані через налаштовані OpcUa-Item вузли (рис. 3.19).

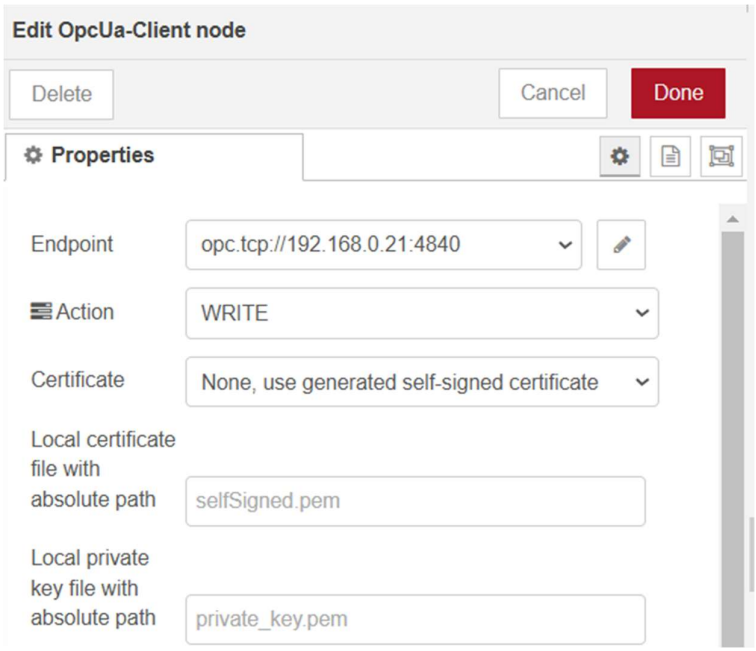

Рисунок 3.19 – Налаштування OpcUa-Client

Також схожим чином працює передача змінної, що відповідає за початок роботи системи пошуку фізичних перешкод RadarStart. Змінна передається через OpcUa-Client до OpcUa-Item, що визначає яку саму змінну ми хочемо прочитати. Після цього значення цієї змінної передається у Serial Out node, де його може прочитати Arduino.

Так для передачі до Tia Portal було створено OPC сервер у UaEpxert.

Таким чином ми додаємо сервер та вводимо вводимо Endpoint URL сервера. opc.tcp://127.0.0.21:4840. Після цього UaExpert з'єднається із сервером і завантажить його простір адрес. У вкладці Address Space можна знайти потрібні змінні, наприклад, ns=4;i=6. Також можна перетягнути їх у вкладку Data Access View для моніторингу їх значень у реальному часі (рис. 3.20).

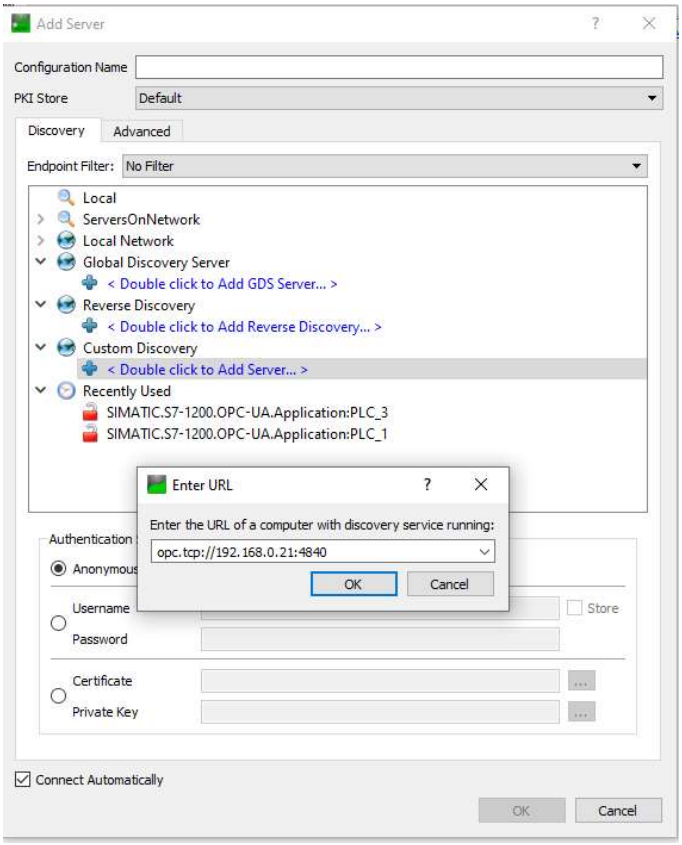

Рисунок 3.20 – Додання серверу у UaExpert

Таким чином ми можемо переглядати усі існуючі змінні. В нашому випадку це змінна початку роботи радару та змінні для опису знайденого об’єкта у робочій області маніпулятора (рис. 3.21 – рис. 3.22).

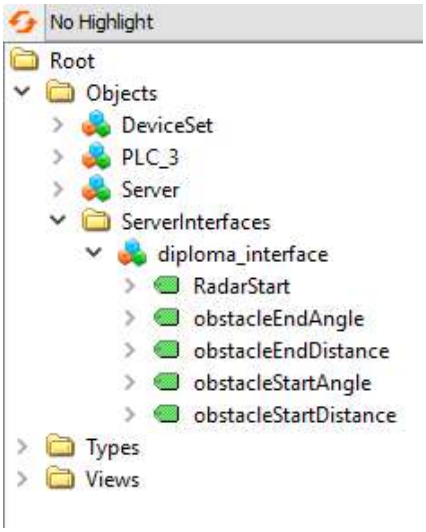

Рисунок 3.21 – Змінні у UaExpert

Attribute	Value
NodeId	ns=4;i=5
NamespaceIndex	4
IdentifierType	Numeric
Identifier	5
NodeClass	Variable
BrowseName	4, "obstacleEndAngle"
DisplayName	"", "obstacleEndAngle"
Description	BadAttributeIdInvalid (0x80350000)
Value	
SourceTimestamp	30.11.2024 13:38:10.356
SourcePicoseconds	0
ServerTimestamp	30.11.2024 13:38:10.356
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	0
DataType	Int16
NamespaceIndex	0

Рисунок 3.22 – Приклад даних змінної у UaExpert

3.4.4 Огляд програмного забезпечення в Tia Portal

Для того, щоб корегувати рух ланок маніпулятора відповідно до наявних/відсутніх перешкод необхідно написати програму у Tia Portal.

Спочатку потрібно провести всі необхідні налаштування проекту для його коректної роботи. Виставляємо необхідний нам IP адресу у налаштуваннях контролера проекту (рис. 3.23).

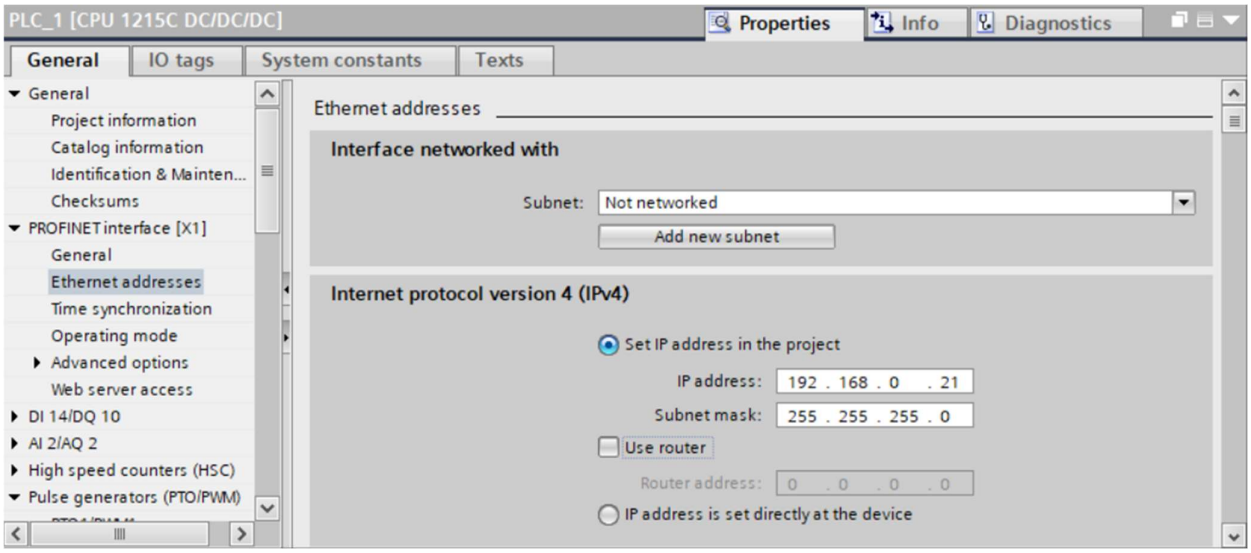

Рисунок 3.23 – Налаштування ПЛК проекту

Далі необхідно активувати OPC сервер і вибрати необхідний сертифікат для подальшого створення інтерфейсу (рис. 3.24).

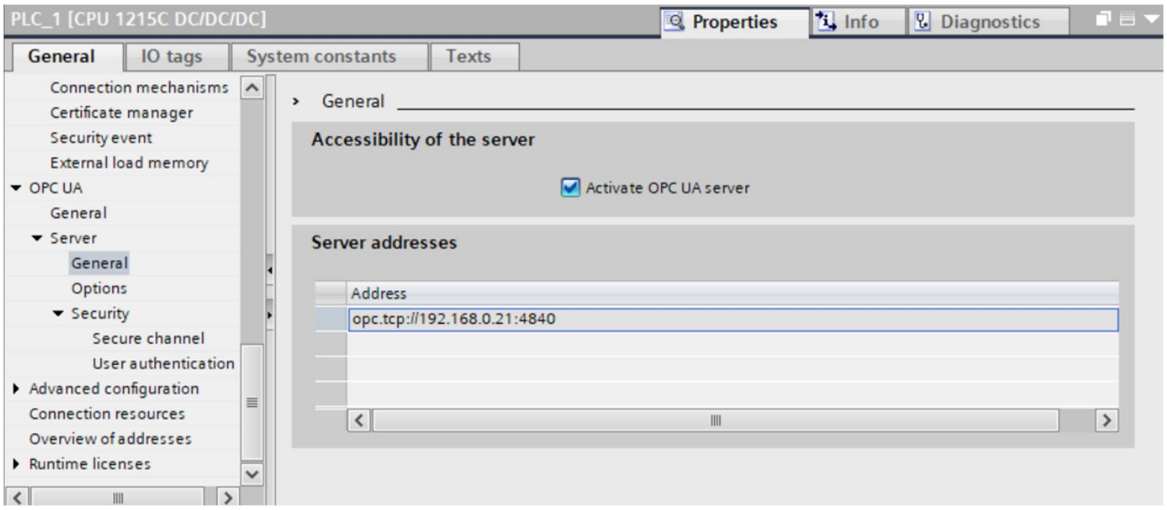

Рисунок 3.24 – Налаштування ПЛК проекту

Після створення серверного інтерфейсу ми маємо можливість додати до нього всі локальні змінні з бази даних проекту, які ми використовуємо для коригування позиції ланок маніпулятору (рис. 3.25).

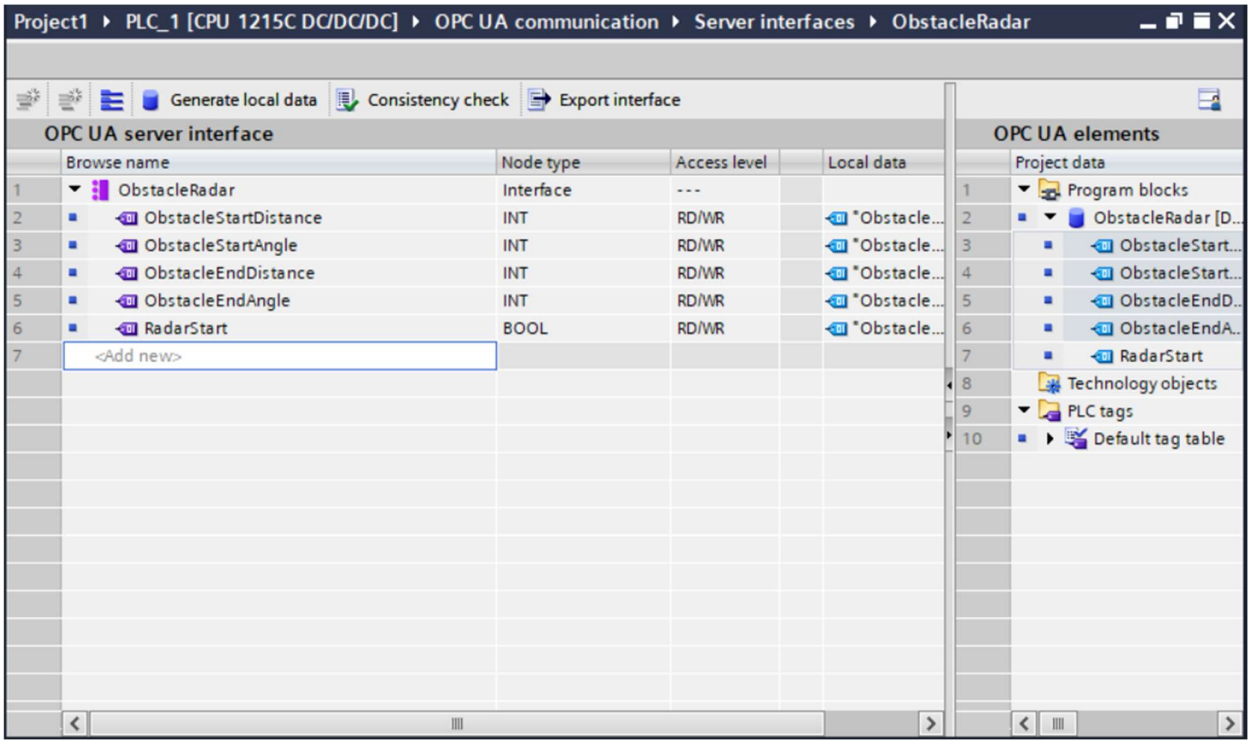

Рисунок 3.25 – Серверний інтерфейс OPC

Тепер, після закінчення всіх необхідних налаштувань та додання необхідних інтерфейсів для передачі даних у проект, маємо можливість розглянути розроблену програму для керування рухом маніпулятора SCARA-робота з автоматичним пошуком траси при наявності фізичних перешкод.

На початку роботи програми ми відправляємо сигнал платі Arduino для початку збору даних, які будуть потім використовуватись для коригування руху маніпулятору (рис. 3.26).

```
"ObstacleRadar".RadarStart := 1;
"IEC_Tim_OFF_0_DB".TON(IN := #Start_Timer_0,
                       PT := T#10s,
                       ET => #ET_TIME_ON_0,
                       Q => #Timer_0_END);
"ObstacleRadar".RadarStart := 0;
```

Рисунок 3.26 – Відправлення сигналу для Arduino

Після отримання всіх необхідних даних ми можемо скоригувати маршрут траси робота маніпулятора до початку його руху. Проект та управління маніпулятором було розроблено на основі автоматичного режиму проекту лабораторного стенду. Таким чином перевірячи наявність перешкоди ми можемо підняти вантаж та почати рух баштою маніпулятора до позиції перешкоди (рис. 3.27 – рис. 3.28).

```
IF "ObstacleRadar".ObstacleTrue THEN
    "Data_block_1"."angle D1" := "ObstacleRadar".ObstacleStartAngle - 10;
    "Data_block_1"."move D1" := 1;
    #Start_Timer_5 := 1;

    "IEC_Tim_OFF_5_DB".TON(IN := #Start_Timer_5,
                           PT := T#10s,
                           ET => #ET_TIME_ON_5,
                           Q => #Timer_5_END);

    IF #Timer_5_END THEN
        "Data_block_1"."move D1" := 0;
        "STEP_3" := 0;
        "STEP_4" := 1;
    END_IF;
END_IF;
```

Рисунок 3.27 – Код початоку руху башти маніпулятору до першкоди

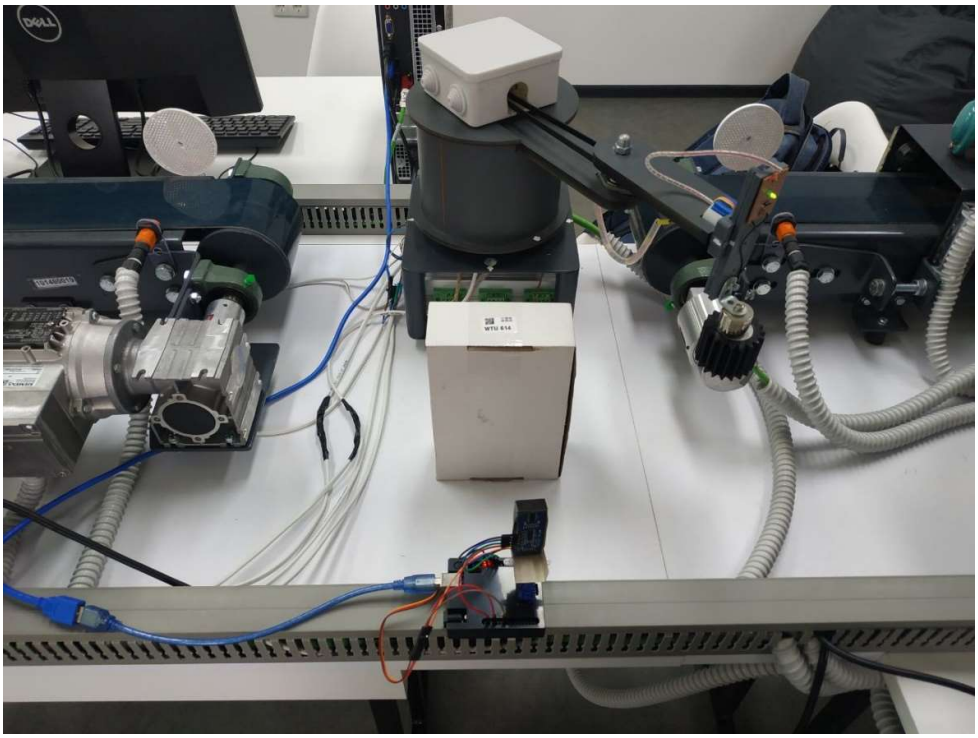

Рисунок 3.28 – Початок руху башти маніпулятору до першкоди

Після того як башта наблизилась до перешкоди необхідно прибрати стрілу маніпулятору з шляху аби подолати перешкоду. Для цього після завершення руху башти ми починаємо виконувати крок, що прибирає стрілу з шляху (рис. 3.29).

```
IF "STEP_4" THEN
    "Data_block_1"."angle D2" := -70;
    "Data_block_1"."move D2" := 1;
    #Start_Timer_4 := 1;

    "IEC_Tim_OFF_4_DB".TON(IN := #Start_Timer_4,
                           PT := T#10s,
                           ET => #ET_TIME_ON_4,
                           Q => #Timer_4_END);

    IF #Timer_4_END THEN
        "Data_block_1"."move D2" := 0;
        "STEP_2" := 0;
        "STEP_3" := 1;
    END_IF;
END_IF;
```

Рисунок 3.29 – Код переміщення стріли маніпулятору

Після чого башта знову почне рухатись і маніпулятор зможе подолати поставлену фізичну перешкоду. Подальші кроки не відрізняються від автоматичного режиму роботи проекту та маніпулятор переносить вантаж на конвеєр та повертається на початкову позицію. Інші рухи ланок відбувається аналогічним чином до наведених частин коду програми (рис. 3.30).

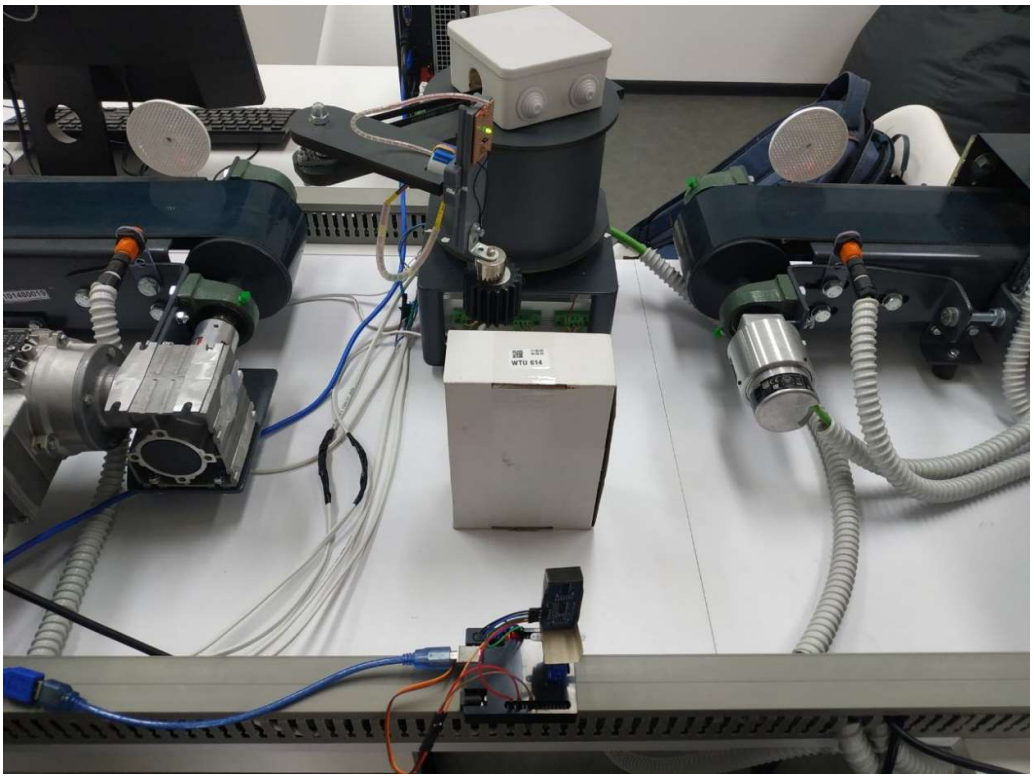

Рисунок 3.30 – Переміщення стріли маніпулятору для уникнення перешкоди

Висновки за розділом

За результатами розділу було розроблено алгоритм керування системою керування маніпулятором, де описано принцип роботи системи та взаємодію її компонентів. Також було наведено блок-схему, що містить у собі алгоритм роботи системи.

Було наведено макетну схему системи пошуку фізичних перешкод у зоні дії маніпулятору, де наведено всі підключення відповідних елементів.

Наведено структурну схему на якій зображено загальний вигляд всієї системи. Зображено різні частини та компоненти системи та їх підключення до інших частин системи для їх коректної роботи.

Було розроблено програму у Arduino IDE, що виявляє перешкоди у робочій зоні маніпулятору за допомогою ультразвукового датчику, що закріплений на серводвигуні. Дані, отримані з ультразвукового датчика, передаються на мікроконтролер Arduino Uno, який аналізує їх і визначає наявність та координати перешкод.

Наведено розроблену програму у Processing, що використовується для візуалізації роботи системи пошуку фізичних перешкод у зоні дії маніпулятору.

Також було наведено програму у Node Red для обробки отриманих з радару даних та подальшої їх відправки до UaExpert і навпаки.

Розроблено програму у Tia Portal, що на базі отриманих даних коригує переміщення ланок маніпулятору за наявності перешкод у зоні його дії.

ВИСНОВКИ

Результатом кваліфікаційної роботи є розроблена система автоматичного керування рухом маніпулятора SCARA-робота з можливістю автоматичного пошуку траси при наявності фізичних перешкод.

Розв'язання задачі автоматизації пошуку траси дозволило забезпечити виявлення перешкод у робочому просторі маніпулятора, розробити алгоритм адаптивного керування рухом маніпулятора, що дозволяє уникати перешкод та інтегрувати апаратне та програмне забезпечення для взаємодії між датчиками, контролерами та приводами маніпулятора.

Для розробки системи керування рухом маніпулятора SCARA-робота з можливістю автоматичного пошуку траси при наявності фізичних перешкод на стенді Siemens КНУ було прийнято рішення про модернізацію стенду, шляхом додання додаткових елементів обладнання.

Для вирішення поставленої задачі було використано ультразвуковий датчик HC-SR04 для виявлення перешкод, серводвигун SG90 для сканування робочого простору, мікроконтролер Arduino Uno для попередньої обробки даних.

За допомогою встановлених датчиків наявна можливість сканування робочої зони маніпулятора та на основі отриманих даних коригувати рух ланок маніпулятора для подальшого його руху з уникненням перешкод на його шляху.

Програма для вирішення поставленої задачі у створенні автоматичної системи керування процесом руху ланок маніпулятора була розроблена для контролеру SIMATIC S7-1200 у робочому середовищі TIA Portal. Програма для сканування робочої зони керованого маніпулятору була розроблена у середовищі Arduino IDE. Програма для візуалізації процесу сканування робочої зони керованого маніпулятору була розроблена у середовищі Processing. У Node Red була розроблена програма для оброки отриманих даних та зв’язку датчиків системи з ПЛК лабораторного стенду.

У результаті була створена система, що забезпечує точне та гнучке керування маніпулятором SCARA з можливістю врахування динамічних змін у робочому просторі. Система показала високу ефективність в автоматизації процесу руху маніпулятора та мінімізації ризиків зіткнення з перешкодами. Розроблена система може бути використана у навчальних цілях, у дослідницьких проектах або для автоматизації виробничих процесів із маніпуляторами SCARA.

Подальше вдосконалення розробленої системи можливе у таких напрямах:

- використання більш чутливих датчиків для покращення точності виявлення перешкод;
- інтеграція додаткових модулів керування, які враховують складніші алгоритми планування траєкторій;
- розширення функціональності програмного забезпечення для покращення інтерфейсу оператора та збільшення адаптивності системи;
- оптимізація часу роботи таймерів розробленої програми для зменшення часу простою під час виконання програми.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Лисенко С. М. Напрямки досліджень та розвитку комп'ютерної інженерії. К. : ХНУ, 2015. 142 с.

2. Репніков Н. Б. Теорія автоматичного керування: класика і сучасність; підручник. К. : НТУУ «КПІ», 2011. 328 с.

3. Дмитрів В. Т. Динаміка і точність роботів: навч. посібник. Львів: Видавництво Львівської політехніки, 2021. 200с.

4. Попович М. Г. Електромеханічні системи автоматичного керування та електроприводи. К.: Либідь, 2005. 680 с.

5. SIMATIC S7-1200, CPU 1215C - Industry Mall. URL: https://mall.industry.siemens.com/mall/en/en/Catalog/Product/6ES7215-1AG40-0XB0 (Дата звернення 22.08.24)

6. Казачковський М. М. Комплектні електроприводи: навч. посібник. Дніпропетровськ : Національний гірничий університет, 2003. 226 с.

7. 28BYJ-48 – 5V Stepper Motor Mouser Electronics. URL: https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf (Дата звернення 29.08.24)

8. Module: TB6560 bipolar stepper motor driver. URL: https://www.mantech.co.za/Datasheets/Products/TB6560_1688_SGT.pdf (Дата звернення 13.09.24)

9. Попович М. Г., Ковальчук О. В. Теорія автоматичного керування: Підручник. Київ: «Либідь», 2007. 656 с.

10. KY-040 Arduino Rotary Encoder User Manual: URL: https://www.epitran.it/ebayDrive/datasheet/25.pdf (Дата звернення 16.09.24)

11. Система медичного маніпулятора URL: https://patents.google.com/patent/US20180036090A1/en?oq=20180036090 (дата звернення 28.09.2024)

12. Спосіб і пристрій планування беззіткненого руху маніпулятора URL: https://patents.google.com/patent/EP3623116A1/en (дата звернення 29.09.2024)

13. Спосіб та система предиктивного уникнення зіткнення маніпулятора з людиною URL: https://patents.google.com/patent/EP2685996C1/ (дата звернення 10.10.2024)

14. Маніпулятор і спосіб керування ним URL: https://patents.google.com/patent/US8606402 (дата звернення 14.10.2024)

15. Синхронізація процесу фарбування багаторукового робота URL: https://patents.google.com/patent/US9227322B2/en?oq=US9227322B2 (дата звернення 15.10.2024)

16. How HC-SR04 Ultrasonic Sensor Works & Interface It With Arduino URL: https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/ (дата звернення 15.10.2024)

17. SG90 9 g Micro Servo Datasheet URL: https://www.friendlywire.com/projects/ne555-servo-safe/SG90-datasheet.pdf (дата звернення 22.10.2024)

18. UNO R3 URL: https://docs.arduino.cc/hardware/uno-rev3/ (дата звернення 22.10.2024)

19. Цвіркун Л. І., Грулер Г. О. Робототехніка та мехатроніка: навчальний посібник. Д. : Національний гірничий університет, 2007. 216 с

20. Derive and Apply Inverse Kinematics to Two-Link Robot Arm. URL: https://www.mathworks.com/help/symbolic/derive-and-apply-inverse-kinematics-to-robot-arm.html (Дата звернення 26.10.24)

21. Modeling and Control of 2-DOF Robot Arm. 2018. Volume 6, Issue 11. URL: https://www.ijeert.org/papers/v6-i11/3.pdf (Дата звернення 26.10.24)

22. Simple math behind calculating distance using Ultrasonic sensor HC-SR04. URL: https://medium.com/@adityavijaynarkar/simple-math-behind-calculating-distance-using-ultrasonic-sensor-hc-sr04-66ed5a6aa214 (Дата звернення 26.10.24)

23. Mathematical Modeling and Analytic Characterization of an Ultrasound Proximity URL:Sensor https://www.redalyc.org/journal/932/93265444001/html/ (Дата звернення 05.11.24)

24. HC-SR04 Ultrasonic Sensor Module User Guide URL: https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf (Дата звернення 06.11.24)

25. Visualization with Arduino and Processing URL: https://www.arduino.cc/education/visualization-with-arduino-and-processing/ (Дата звернення 08.11.24)

26. Building Secure OPC-UA Server in Node-RED. URL: https://flowfuse.com/node-red/protocol/opa-ua/ (Дата звернення 08.11.24)

27. Маринич І. А., Тронь В. В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 151 “Автоматизація та комп’ютерно-інтегровані технології”. Кривий Ріг : Видавничий центр КНУ, 2022. 50с.

28. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

29. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

30. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові.
Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)

31. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).

ДОДАТОК А

Код програми TIA Portal

```
IF "Start_ObstMode" AND NOT "Conveyor".StatusBits.Error AND NOT "Axis_1".StatusBits.Error
   AND NOT "Axis_2".StatusBits.Error AND NOT "Axis_3".StatusBits.Error
THEN
   "AUTO_mode_work" := 1;
ELSE
   "AUTO_mode_work" := 0;
   "Start_AUT" := 0;
END_IF;
IF NOT "AUTO_mode_work" AND "Switch_AUT/MAN" THEN
   "Data_block_1"."stop D1" := 1;
   "Data_block_1"."stop D2" := 1;
   "Data_block_1"."stop D3" := 1;
   "Data_block_1"."conv stop" := 1;
END_IF;

IF "AUTO_mode_work" THEN

   IF NOT "STEP_0_END" THEN
      "STEP_0" := 1;
   ELSE
      "STEP_0" := 0;
   END_IF;



   //Шаг 0:
   IF "STEP_0" THEN
```

```
    "Data_block_1"."On/off D1" := 1;
    "Data_block_1"."On/off D2" := 1;
    "Data_block_1"."On/off D3" := 1;
    "Data_block_1"."stop D1" := 0;
    "Data_block_1"."stop D2" := 0;
    "Data_block_1"."stop D3" := 0;
    "Data_block_1"."home D1" := 1
    "Data_block_1"."home D2" := 1;
    "Data_block_1"."home D3" := 1;
    #Start_Timer_0 := 1;
    "SecondCycle" := 0;


    "IEC_Tim_OFF_0_DB".TON(IN := #Start_Timer_0,
                           PT := T#1s,
                           ET => #ET_TIME_ON_0,
                           Q => #Timer_0_END);

    IF #Timer_0_END THEN
        "Data_block_1"."home D1" := 0
        "Data_block_1"."home D2" := 0;
        "Data_block_1"."home D3" := 0;
        "STEP_0_END" := 1;
        "STEP_1" := 1;
    END_IF;
END_IF;


//Шаг 1:
IF "STEP_1" THEN
```

```
"Data_block_1"."angle D3" := -43;
"Data_block_1"."move D3" := 1;
#Start_Timer_1 := 1;

"IEC_Tim_OFF_1_DB".TON(IN := #Start_Timer_1,
                       PT := T#12s,
                       ET => #ET_TIME_ON_1,
                       Q => #Timer_1_END);

IF #Timer_1_END THEN
   "Data_block_1"."move D3" := 0;
   "magnit" := 1;
   #Start_Timer_2 := 1;
END_IF;

"IEC_Tim_OFF_2_DB".TON(IN := #Start_Timer_2,
                       PT := T#1s,
                       ET => #ET_TIME_ON_2,
                       Q => #Timer_2_END);



IF #Timer_2_END THEN
   "Data_block_1"."angle D3" := 40;
   "Data_block_1"."move D3" := 1;
   #Start_Timer_3 := 1;
END_IF;

"IEC_Tim_OFF_3_DB".TON(IN := #Start_Timer_3,
                       PT := T#20s,
```

```
                    ET => #ET_TIME_ON_3,
                    Q => #Timer_3_END);

   IF #Timer_3_END THEN
      "Data_block_1"."move D3" := 0;
      "STEP_1" := 0;
      "STEP_2" := 1;
   END_IF;
END_IF;



//Шаг 2:
IF "STEP_2" THEN
   IF #obstacleStartAngle > 0 AND #obstacleStartDistance > 0 THEN
      "Data_block_1"."angle D1" := #obstacleStartAngle - 15;
      "Data_block_1"."move D1" := 1;
      #Start_Timer_4 := 1;

      "IEC_Tim_OFF_4_DB".TON(IN := #Start_Timer_4,
                    PT := T#10s,
                    ET => #ET_TIME_ON_4,
                    Q => #Timer_4_END);

      IF #Timer_4_END THEN
         "Data_block_1"."move D1" := 0;
         "STEP_2" := 0;
         "STEP_3.2" := 1;
      END_IF;
   ELSE
      "STEP_2" := 0;
```

```
      "STEP_3" := 1;
   END_IF;
END_IF;

//Шаг 3.2:
IF "STEP_3.2" THEN
   "Data_block_1"."angle D2" := -70;
   "Data_block_1"."move D2" := 1;
   #Start_Timer_15 := 1;

   "IEC_Timer_15_DB".TON(IN:=#Start_Timer_15,
                         PT:=T#10s,
                         ET=>#ET_TIME_ON_15,
                         Q=>#Timer_15_END);
   IF #Timer_15_END THEN
      "Data_block_1"."move D2" := 0;
      "STEP_3.2" := 0;
      "STEP_4.2" := 1;
   END_IF;
END_IF;

//Шаг 4.2:
IF "STEP_4.2" THEN
   "Data_block_1"."angle D1" := 180;
   "Data_block_1"."move D1" := 1;
   #Start_Timer_16 := 1;

   "IEC_Timer_16_DB".TON(IN:=#Start_Timer_16,
                         PT:=T#10s,
                         Q=>#Timer_16_END,
```

```
                ET=>#ET_TIME_ON_16);

   IF #Timer_16_END THEN
      "Data_block_1"."move D1" := 0;
      "STEP_4.2" := 0;
      "STEP_5.2" := 1;
   END_IF;
END_IF;

//Шаг 5.2:
IF "STEP_5.2" THEN
   "Data_block_1"."angle D2" := 0;
   "Data_block_1"."move D2" := 1;
   #Start_Timer_17 := 1;

   "IEC_Timer_17_DB".TON(IN:=#Start_Timer_17,
                PT:=T#10s,
                Q=>#Timer_17_END,
                ET=>#ET_TIME_ON_17);

   IF #Timer_17_END THEN
      "Data_block_1"."move D2" := 0;
      "STEP_5.2" := 0;
      "STEP_5" := 1;
   END_IF;
END_IF;

//Шаг 3:
IF "STEP_3" THEN
   "Data_block_1"."angle D1" := 90;
```

```
   "Data_block_1"."move D1" := 1;
   #Start_Timer_5 := 1;

   "IEC_Tim_OFF_5_DB".TON(IN := #Start_Timer_5,
                          PT := T#10s,
                          ET => #ET_TIME_ON_5,
                          Q => #Timer_5_END);

   IF #Timer_5_END THEN
      "Data_block_1"."move D1" := 0;
      "STEP_3" := 0;
      "STEP_4" := 1;
   END_IF;
END_IF;


//Шаг 4:
IF "STEP_4" THEN
   "Data_block_1"."angle D2" := 0;
   "Data_block_1"."move D2" := 1;
   #Start_Timer_6 := 1;

   "IEC_Tim_OFF_6_DB".TON(IN := #Start_Timer_6,
                          PT := T#10s,
                          ET => #ET_TIME_ON_6,
                          Q => #Timer_6_END);

   IF #Timer_6_END THEN
      "Data_block_1"."move D2" := 0;
      "STEP_4" := 0;
```

```
      "STEP_5" := 1;
   END_IF;
END_IF;



//Шаг 5:
IF "STEP_5" THEN
   "Data_block_1"."angle D3" := -30;
   "Data_block_1"."move D3" := 1;
   #Start_Timer_7 := 1;

   "IEC_Tim_OFF_7_DB".TON(IN := #Start_Timer_7,
                          PT := T#20s,
                          ET => #ET_TIME_ON_7,
                          Q => #Timer_7_END);

   IF #Timer_7_END THEN
      "Data_block_1"."move D3" := 0;
      "magnit" := 0;
      #Start_Timer_8 := 1;
   END_IF;

   "IEC_Tim_OFF_8_DB".TON(IN := #Start_Timer_8,
                          PT := T#1s,
                          ET => #ET_TIME_ON_8,
                          Q => #Timer_8_END);

   IF #Timer_8_END THEN
      "Data_block_1"."angle D3" := 40;
      "Data_block_1"."move D3" := 1;
```

```
      "Data_block_1"."conv ON/OFF" := 1;
      "Data_block_1"."conv stop" := 0;
      #Start_Timer_9 := 1;
   END_IF;

   "IEC_Tim_OFF_9_DB".TON(IN := #Start_Timer_9,
                          PT := T#15s,
                          ET => #ET_TIME_ON_9,
                          Q => #Timer_9_END);

   IF #Timer_9_END THEN
      "Data_block_1"."move D3" := 0;
      "STEP_5" := 0;
      "STEP_6" := 1;
   END_IF;
END_IF;


//Шаг 6:
IF "STEP_6" THEN
   "Data_block_1"."conv home" := 1;

   IF "RightSensor" THEN
      "Data_block_1"."conv position" := 820;
      "Data_block_1"."conv speed" := 50;
      "Data_block_1"."conv move" := 1;
      "Data_block_1"."angle D2" := 90;
      "Data_block_1"."move D2" := 1;
      "Start_Timer_10" := 1;
   END_IF;
```

```
   "IEC_Tim_OFF_10_DB".TON(IN := "Start_Timer_10",
                           PT := T#20s,
                           ET => #ET_TIME_ON_10,
                           Q => #Timer_10_END);

   IF #Timer_10_END AND "LeftSensor" THEN
      "Data_block_1"."move D2" := 0;
      "Data_block_1"."conv move" := 0;
      "Data_block_1"."conv ON/OFF" := 0;
      "Data_block_1"."conv home" := 0;
      "Data_block_1"."conv speed" := 0;
      "Start_Timer_10" := 0;
      "Data_block_1"."conv position" := 0;
      "STEP_6" := 0;
      "STEP_7" := 1;
   END_IF;
END_IF;


//Шаг 7:
IF "STEP_7" THEN
   "Data_block_1"."angle D1" := 0;
   "Data_block_1"."move D1" := 1;
   #Start_Timer_11 := 1;

   "IEC_Tim_OFF_11_DB".TON(IN := #Start_Timer_11,
                           PT := T#10s,
                           ET => #ET_TIME_ON_11,
                           Q => #Timer_11_END);
```

```
    IF #Timer_11_END THEN
        "Data_block_1"."move D1" := 0;
        "Data_block_1"."conv home" := 1;
        "STEP_7" := 0;
        "STEP_8" := 1;
    END_IF;
END_IF;

//Шаг 8:
IF "STEP_8" THEN
    "Data_block_1"."angle D2" := 0;
    "Data_block_1"."move D2" := 1;
    #Start_Timer_12 := 1;

    "IEC_Tim_OFF_12_DB".TON(IN := #Start_Timer_12,
                            PT := T#10s,
                            ET => #ET_TIME_ON_12,
                            Q => #Timer_12_END);

    IF #Timer_12_END THEN
        "Data_block_1"."move D2" := 0;
        "Data_block_1"."angle D3" := 0;
        "Data_block_1"."move D3" := 1;
        #Start_Timer_13 := 1;
    END_IF;

    "IEC_Tim_OFF_13_DB".TON(IN := #Start_Timer_13,
                            PT := T#15s,
                            ET => #ET_TIME_ON_13,
```

```
                Q => #Timer_13_END);

IF #Timer_13_END THEN
    "Data_block_1"."move D3" := 0;
    "Data_block_1"."angle D1" := 0;
    "STEP_8" := 0;
    "Data_block_1"."On/off D1" := 0;
    "Data_block_1"."On/off D2" := 0;
    "Data_block_1"."On/off D3" := 0;
    "Data_block_1"."conv home" := 0;
    "Switch_AUT/MAN" := 0;
    "Start_ObstMode" := 0;
    #obstacleStartDistance := 0;
    #obstacleStartAngle := 0;
    #obstacleEndDistance := 0;
    #obstacleEndAngle := 0;
    #RadarStart := 0;
    RESET_TIMER("IEC_Tim_OFF_0_DB");
    RESET_TIMER("IEC_Tim_OFF_1_DB");
    RESET_TIMER("IEC_Tim_OFF_2_DB");
    RESET_TIMER("IEC_Tim_OFF_3_DB");
    RESET_TIMER("IEC_Tim_OFF_4_DB");
    RESET_TIMER("IEC_Tim_OFF_5_DB");
    RESET_TIMER("IEC_Tim_OFF_6_DB");
    RESET_TIMER("IEC_Tim_OFF_7_DB");
    RESET_TIMER("IEC_Tim_OFF_8_DB");
    RESET_TIMER("IEC_Tim_OFF_9_DB");
    RESET_TIMER("IEC_Tim_OFF_10_DB");
    RESET_TIMER("IEC_Tim_OFF_11_DB");
    RESET_TIMER("IEC_Tim_OFF_12_DB");
```

```
        RESET_TIMER("IEC_Tim_OFF_13_DB");
        RESET_TIMER("IEC_Tim_OFF_14_DB");
        RESET_TIMER("IEC_Timer_15_DB");
        RESET_TIMER("IEC_Timer_16_DB");
        RESET_TIMER("IEC_Timer_17_DB");
      END_IF;
   END_IF;
END_IF;
```

Код програми Node-Red

```
// Отримуємо дані з msg.payload
let input = msg.payload.trim(); // Видаляємо зайві пробіли, якщо є
// Розділяємо дані на масив пар "кут.дистанція"
let dataPairs = input.split('.');
// Ініціалізуємо змінні для результату
let ObstacleStartDistance = null;
let ObstacleStartAngle = null;
let ObstacleEndDistance = null;
let ObstacleEndAngle = null;
let currentObject = null;
let skipCount = 0; // Лічильник пропущених точок
const MAX_SKIP_COUNT = 3; // Максимальна кількість пропусків, яку можна ігнорувати
const MIN_POINTS_FOR_OBJECT = 5; // Мінімальна кількість точок для визнання об'єкта
// Проходимо по всіх парах даних
dataPairs.forEach(pair => {
   let [angle, distance] = pair.split(',').map(Number); // Розбиваємо на кут та дистанцію
```

```
// Перевіряємо, чи дистанція менше 20 см
if (distance < 20) {
  if (!currentObject) {
    // Починається новий об'єкт
    currentObject = {
      startAngle: angle,
      startDistance: distance,
      endAngle: angle,
      endDistance: distance,
      pointsCount: 1 // Лічильник точок в об'єкті
    };
    skipCount = 0; // Скидаємо лічильник пропусків
  } else {
    // Продовження поточного об'єкта
    currentObject.endAngle = angle;
    currentObject.endDistance = distance;
    currentObject.pointsCount += 1;
    skipCount = 0; // Скидаємо лічильник пропусків
  }
} else {
  if (currentObject) {
    if (skipCount < MAX_SKIP_COUNT) {
      // Ігноруємо пропуски, але не завершуємо об'єкт
      skipCount++;
    } else {
      // Об'єкт завершено
      if (currentObject.pointsCount >= MIN_POINTS_FOR_OBJECT) {
        ObstacleStartAngle = currentObject.startAngle;
        ObstacleStartDistance = currentObject.startDistance;
```

```
                ObstacleEndAngle = currentObject.endAngle;
                ObstacleEndDistance = currentObject.endDistance;
            }
            currentObject = null; // Скидаємо поточний об'єкт
            skipCount = 0; // Скидаємо лічильник пропусків
        }
    }
  }
});
// Якщо останній об'єкт не завершений, але існує, фіксуємо його
if (currentObject && currentObject.pointsCount >= MIN_POINTS_FOR_OBJECT) {
  ObstacleStartAngle = currentObject.startAngle;
  ObstacleStartDistance = currentObject.startDistance;
  ObstacleEndAngle = currentObject.endAngle;
  ObstacleEndDistance = currentObject.endDistance;
}
// Формуємо повідомлення для кожного виходу
let msg1 = { payload: ObstacleStartDistance }; // Для першого виходу
let msg2 = { payload: ObstacleStartAngle };    // Для другого виходу
let msg3 = { payload: ObstacleEndDistance };   // Для третього виходу
let msg4 = { payload: ObstacleEndAngle };      // Для четвертого виходу
// Повертаємо масив повідомлень
return [msg1, msg2, msg3, msg4];
```

Код програми Processing

```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from the serial port
import java.io.IOException;
```

```
Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
//PFont orcFont;
void setup() {
size (1280, 720); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
 smooth();
 myPort = new Serial(this,"COM3", 9600); // starts the serial communication
 myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So actually it reads this: angle,distance.
 //orcFont = loadFont("OCRAExtended-30.vlw");
}
void draw() {
  fill(98,245,31);
  //textFont(orcFont);
  // simulating motion blur and slow fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);
  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
```

```
  drawLine();
  drawObject();
  drawText();
}
void serialEvent (Serial myPort) { // starts reading data from the Serial Port
  // reads the data from the Serial Port up to the character '.' and puts it into the String variable "data".
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);
  index1 = data.indexOf(","); // find the character ',' and puts it into the variable "index1"
  angle= data.substring(0, index1); // read the data from position "0" to position of the variable index1 or thats the value of the angle the Arduino Board sent into the Serial Port distance= data.substring(index1+1, data.length()); // read the data from position "index1" to the end of the data pr thats the value of the distance
  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}
void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
```

```
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}
void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
```

```
 translate(width/2,height-height*0.074); // moves the starting coordinats to new location
 line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle))); // draws the line according to the angle
 popMatrix();
}
void drawText() { // draws the texts on the screen
 pushMatrix();
 if(iDistance>40) {
 noObject = "Out of Range";
 }
 else {
 noObject = "In Range";
 }
 fill(0,0,0);
 noStroke();
 rect(0, height-height*0.0648, width, height);
 fill(98,245,31);
 textSize(25);
 text("10cm",width-width*0.3854,height-height*0.0833);
 text("20cm",width-width*0.281,height-height*0.0833);
 text("30cm",width-width*0.177,height-height*0.0833);
 text("40cm",width-width*0.0729,height-height*0.0833);
 textSize(40);
 text("Object: " + noObject, width-width*0.875, height-height*0.0277);
 text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
 text("Distance: ", width-width*0.26, height-height*0.0277);
 if(iDistance<40) {
 text("        " + iDistance +" cm", width-width*0.225, height-height*0.0277);
 }
```

```
  textSize(25);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-
width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-
width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();
  translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-
width/2*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();
  translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-
width/2*sin(radians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();
  translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-
width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
```

Код програми Arduino EDI

```
// Includes the Servo library
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;
Servo myServo; // Creates a servo object for controlling the servo motor
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor attached
}
void loop() {
  if (Serial.available() > 0) {
    // Read the incoming data as a string
    String TiaPortalStart = Serial.readStringUntil('\n'); // Reads until newline character
    if(TiaPortalStart = "StartProgram"){
      // rotates the servo motor from 15 to 165 degrees
      for(int i=15;i<=165;i++){
        myServo.write(i);
        delay(30);
        distance = calculateDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree
        Serial.print(i);
        Serial.print(",");
        Serial.print(distance);
        Serial.print(".");
```

```
      }
      myServo.write(90);
    }
  }
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds
  distance= duration*0.034/2;
  return distance;
}
```

ДОДАТОК Б

Проект методичних вказівок

Лабораторна робота №1

Тема роботи: Вивчення промислового обладнання встановленого на стенді №3 з системою пошуку траси маніпулятора при наявності фізичних перешкод.

Мета роботи: На основі стенду №3 з системою пошуку траси маніпулятора при наявності фізичних перешкод навести технічні характеристики використовуваних у системі датчиків, дати розгорнутий опис контролеру та елементів системи керування маніпулятора, що використовується на стенді.

Теоретичні відомості

Система керування роботом-маніпулятором використовує ультразвуковий датчик HC-SR04, серводвигун SG90 та плату Arduino UNO. Ультразвуковий датчик використовується для знаходження перешкоди на шляху руху ланок маніпулятора. Серводвигун використовується для повертання ультразвукового датчику для подальшого зчитування робочого простору маніпулятора у конусі 150 градусів перед собою. Всі датчики та інші елементи системи сканування робочого простору маніпулятора підключені до плати Arduino UNO, що містить у собі програму для роботи системи сканування та надає живлення(5V) усім підключеним елементам.

Для реалізації системи необхідно зчитувати робочій простір ланок маніпулятора для подальшого коригування руху маніпулятора. Для цього використовується ультразвуковий датчик HC-SR04. (рис. 1).

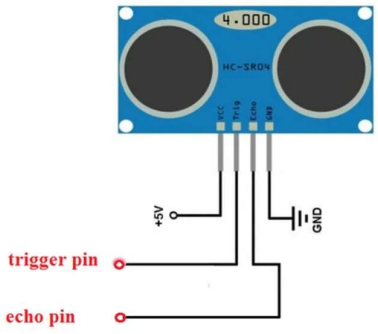

Рисунок 1 – Загальний вигляд енкодера hw-040

Датчик працює наступним чином. Для сенсорних елементів використовуються п'єзоелектричні кристали. П'єзоелектричні кристали коливаються на високих частотах коли до них подається електричний сигнал. П'єзоелектричні кристали генеруватимуть електричний сигнал, коли ультразвукова хвиля потрапила на поверхню датчика в зворотному напрямку (рис. 2).

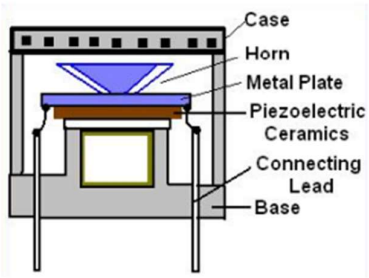

Рисунок 2– Зображення конструкції ультразвукового датчику

SG90 являє собою якісний серводвигун з пусковим моментом 2 кг/див. Усередині корпусу знаходиться невеликий модуль керування, який під дією вхідного сигналу подає живлення електродвигун. Вхідний сигнал керування має дані про необхідне положення валу. Для визначення положення валу на даний момент часу редуктор з'єднаний із двигуном змінного резистора. Електроніка Tower Pro SG90 визначає різницю поточного положення редуктора та необхідного положення. Модуль управління використовуючи опір змінного резистора надає живлення відповідної полярності на двигун для повороту редуктора, що приводить у відповідність положення передане сигналом управління.

Інформація про необхідне положення валу міститься у шпаруватості імпульсів керованого сигналу. Частота керованого сигналу повинна бути постійна і складати 50 Гц.(рис. 3).

Рисунок 3 – Зображення модуля аналогового вводу SM 1231

Arduino Uno – це плата для створення прототипів у сфері електроніки та програмування. Вона базується на мікроконтролері ATmega328P, що забезпечує гарну обчислювальну потужність для багатьох проектів. Arduino Uno має 14 цифрових виводів (6 з яких можуть працювати в режимі PWM) і 6 аналогових входів, що дозволяють зчитувати значення з різноманітних сенсорів або взаємодіяти з іншими пристроями.

Плата працює на частоті 16 МГц і оснащена 32 КБ флеш-пам’яті для зберігання програми, з яких 0,5 КБ зайняті завантажувачем. Для тимчасового зберігання даних доступно 2 КБ оперативної пам’яті SRAM і 1 КБ EEPROM для збереження даних, які залишаються навіть після вимкнення живлення (рис. 4).

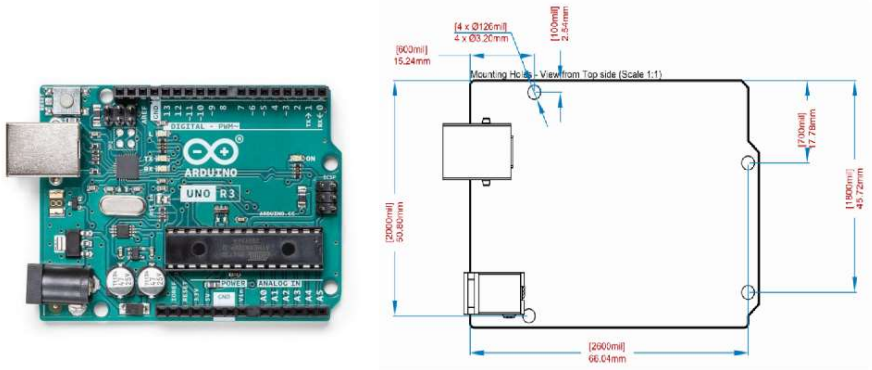

Рисунок 4 – Зображення Arduino Uno

Arduino Uno підтримує живлення через USB-порт або зовнішній адаптер постійного струму з напругою 7–12 В. Вбудований стабілізатор напруги забезпечує оптимальні умови для роботи плати та підключених пристроїв. Для підключення до комп’ютера використовується USB-інтерфейс на базі мікросхеми ATmega16U2, що дозволяє програмувати плату без додаткових програматорів.

На платі також є кнопка для перезавантаження, а також кілька світлодіодів: живлення (PWR) і індикатори активності цифрових пінів (TX і RX). Інтуїтивно зрозуміле програмне середовище Arduino IDE дозволяє писати, компілювати та завантажувати код на плату за допомогою мови програмування, що базується на C/C++.

Завдання:

1. Навести розгорнутий опис та характеристики ультразвукового датчику HC-SR04.
2. Навести розгорнутий опис та характеристики серводвигуна SG90.
3. Навести розгорнутий опис та характеристики Arduino UNO.

Запитання для самоконтролю

1. Що таке ультразвуковий датчик?
2. Що таке сервопривід?
3. Поясніть принцип роботи ультразвукового датчику HC-SR04.
4. Для чого використовується пін Trig у ультразвуковому датчику HC-SR04?
5. Що таке пллата Arduino Uno? Яке її призначення у системі?
6. Пояснить принцип роботи та схему підключення плати Arduino.
7. Для чого використовується пін PWM у сервоприводі SG90?
8. Опишіть які групи пінів наявні на платі Arduino Uno та їх використання.

Лабораторна робота №2

Тема роботи: Ознайомлення з принципом роботи системи пошуку траси маніпулятора при наявності фізичних перешкод на основі стенду №3.

Мета роботи: Ознайомитись з принципом роботи системи пошуку траси маніпулятора при наявності фізичних перешкод, навчитись працювати з екраном оператора системи.

Теоретичні відомості

Метою системи є виявлення підходу до керування рухом маніпулятора SCARA-робота з можливістю автоматичного пошуку траси при наявності фізичних перешкод. У результаті розробки системи було забезпечено виявлення перешкод у робочому просторі маніпулятора, розробити алгоритм адаптивного керування рухом маніпулятора, що дозволяє уникати перешкод та інтегрувати апаратне та програмне забезпечення для взаємодії між датчиками, контролерами та приводами маніпулятора.

Для взаємодії оператора з системою використовується HMI-панель.

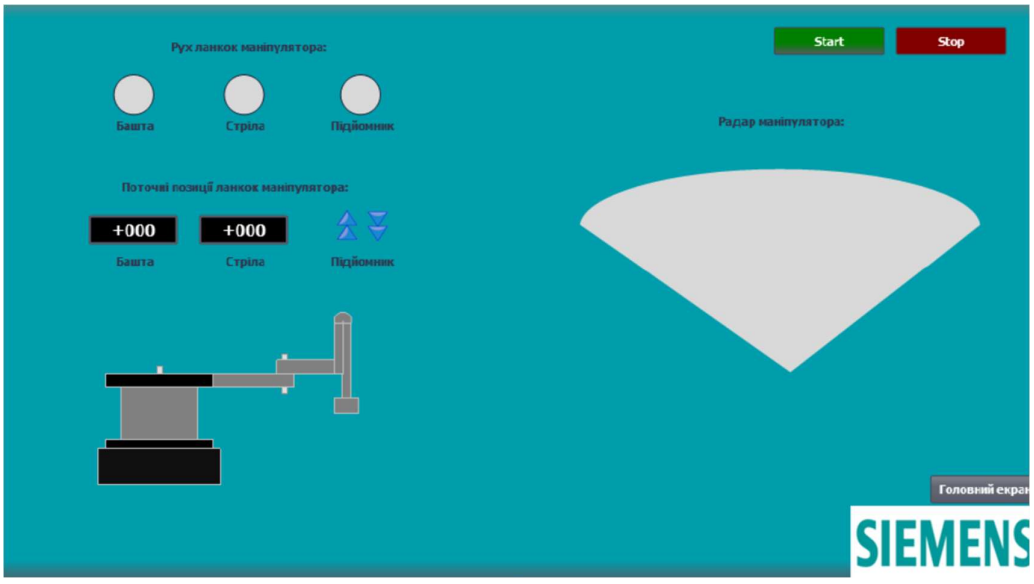

Рисунок 1 – Вигляд HMI-панелі на момент початку роботи

При початку роботи програми програма у Tia Portal передає повідомлення у Node-Red стосовно початку роботи програми системи, яке в свою чергу передається до плати Arduiono. При надходженні цього сигналу починається робота програма плати.

Система управління маніпулятором побудована на основі взаємодії кількох ключових компонентів. Виявлення перешкод у робочому просторі здійснюється за допомогою ультразвукового датчика, встановленого на серводвигуні. Серводвигун забезпечує обертання датчика для сканування простору у 150 градусів. Отримані дані з ультразвукового датчика передаються на мікроконтролер Arduino Uno, який обробляє ці дані та визначає наявність і координати перешкод.

Далі Arduino передає інформацію на персональний комп'ютер у Node-Red, який виконує додаткову обробку отриманих даних та координує обмін інформацією між Arduino та програмованим логічним контролером (ПЛК) Siemens S7-1200. При надходженні даних з плати Arduino, Node-Red виконує подальшу обробку цих даних в результаті чого система отримує позицію, що відповідає початку перешкоди та позицію, що являє собою кінець перешкоди. Далі отримані дані передаються до ПЛК, де програма контролера за необхідності змінює траєкторію ланок маніпулятора для перенесення вантажу на інший конвеєр.

ПЛК виступає основним контролером, який приймає оброблені дані про перешкоди та формує відповідні команди для керування маніпулятором.

Для реалізації руху маніпулятора ПЛК надсилає сигнали на драйвери крокових двигунів, які, у свою чергу, керують переміщенням ланок маніпулятора. Це дозволяє виконувати необхідні маневри, враховуючи розташування перешкод у робочому просторі. Усі компоненти системи з'єднані в єдину інтегровану структуру, де кожен елемент виконує свою роль, забезпечуючи ефективне функціонування маніпулятора.

Така система дозволяє сканувати робочий простір, виявляти перешкоди, та точно керувати положенням ланок маніпулятора в режимі реального часу.

Для перевірки роботи системи необхідно виставити перешкоду на шляху маніпулятору. Відкривши необхідні проекти у Node-Red, UaExpert та TiaPortal, можна натиснути кнопку «Start» на HMI панелі для початку роботи системи.

Завдання

Використовуючи проект системи, ознайомитись з принципом роботи системи та навчитися працювати з нею через HMI-панель. Перевірити роботу системи при різних позиціях перешкоди на шляху маніпулятора. Описати різницю у виконанні програми системи та пояснити чому було отримано саме такі результати.

Запитання для самоконтролю

1. Що таке HMI-панель? Для чого вона використовується?
2. Які системи керування маніпулятором ви знаєте?
3. Наведіть алгоритм дій для запису циклу руху маніпулятора.
4. Поясніть для чого використовується код програми Arduino IDE.
5. Поясніть для чого необхідний блок Fucntion node у проекті Node-Red.
6. Наведіть принцип отримання даних з OPC серверу у Tia Portal.
7. Які елементи на HMI-панелі використовуються для візуалізації процесу роботи з системою.
8. Для чого необхідний OPC interface у проекті Tia Portal.