Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СИСТЕМА УПРАВЛІННЯ РОБОТИЗОВАНИМ МЕХАНІЗМОМ

ROBOTIC MECHANISM CONTROL SYSTEM

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-42
Пайцун Арсен Ігорович

(підпис)

Керівник:
к.т.н., доцент
Костючко Сергій Миколайович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 10 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. ТЕРЛЕЦЬКИЙ
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Пайцуну Арсену Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Система управління роботизованим механізмом*

Керівник роботи *Костючко Сергій Миколайович*

затверджені наказом закладу вищої освіти від «04» січня 2025 року № 11/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2025р.

3. Вихідні дані до роботи *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз підходів керування роботизованими механізмами та постановка завдань дослідження
Архітектура, протокол та логіка системи управління гібридним маніпулятором
Реалізація, тестування та аналіз роботи системи управління
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Порівняльна характеристика апаратних платформ управління
Архітектура системи управління маніпулятором
Консольний інтерфейс користувача системи
Приклад коду реалізації команди захоплення
Динаміка значення FSP при виконанні захоплення

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз сучасних підходів до управління роботизованими механізмами	*Костючко С.М., доцент*		
Проєктування системи управління маніпулятором	*Костючко С.М., доцент*		
Реалізація, тестування та аналіз роботи системи управління	*Костючко С.М., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2025 р.	Виконано
2.	*Вивчення способів управління та створення архітектури системи керування*	до 02.03.2025 р.	Виконано
3.	*Розробка та тестування системи*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
8.	*Нормоконтроль*	до 30.05.2025 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 03.06.2025 р	Виконано
10.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ (підпис) Пайцун А.І. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Костючко С.М. (прізвище, ініціали)

АНОТАЦІЯ

Пайцун А.І. Система управління роботизованим механізмом. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проаналізовано сучасні підходи до керування роботизованими механізмами, класифікацію систем управління, архітектури маніпуляторів, можливості інтеграції штучного інтелекту та порівняльну оцінку апаратних і програмних платформ.

У другому розділі здійснено проєктування системи управління гібридним маніпулятором на базі ESP32 і Raspberry Pi. Описано архітектуру взаємодії, протокол обміну, логіку керування жестами, обробку зворотного зв'язку з FSR-датчика, а також засоби безпеки системи.

У третьому розділі наведено реалізацію фізичного прототипу, програмне забезпечення для ESP32 та Raspberry Pi, інтерфейс користувача і результати тестування в умовах реального середовища.

Ключові слова: система управління, маніпулятор, ESP32, Raspberry Pi, пневматичний м'яз, протокол UART, FSR-датчик, зворотний зв'язок, штучний інтелект.

ANNOTATION

Paitsun A. Control system for a robotic mechanism. Manuscript.

Bachelor’s qualification work of the EP "Computer Engineering" specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification work consists of an introduction, three chapters, conclusions, references, and appendices.

The first chapter analyzes modern approaches to the control of robotic mechanisms, classification of control systems, manipulator architectures, AI integration possibilities, and a comparative review of hardware and software platforms.

The second chapter presents the design of a control system for a hybrid manipulator based on ESP32 and Raspberry Pi. It describes the interaction architecture, communication protocol, gesture control logic, FSR sensor feedback processing, and safety mechanisms.

The third chapter covers the implementation of a physical prototype, software for ESP32 and Raspberry Pi, user interface, and testing results in real conditions.

Keywords: control system, manipulator, ESP32, Raspberry Pi, pneumatic muscle, UART protocol, FSR sensor, feedback, artificial intelligence.

ЗМІСТ

ВСТУП

Сучасна робототехніка швидко розвивається в напрямі створення гнучких, безпечних та інтелектуально керованих механізмів, здатних до взаємодії з людиною та навколишнім середовищем [1]. У цьому контексті особливу увагу привертають гібридні роботизовані системи, які поєднують у собі пневматичні м’язи та сервоприводи, забезпечуючи м’якість рухів та достатню точність позиціювання [2]. Проте фізична реалізація маніпулятора є лише частиною повноцінної системи – для повноцінного функціонування необхідно забезпечити ефективну високорівневу систему керування, здатну адаптуватися до змін, обробляти зворотний зв’язок і підтримувати подальшу інтеграцію з алгоритмами штучного інтелекту.

Актуальність теми зумовлена потребою в розробці гнучких керованих роботизованих механізмів для застосування у сфері автоматизації, сервісної робототехніки та допоміжних систем для людей з інвалідністю. Гібридні підходи, що поєднують пневматичні виконавчі механізми та електроніку, забезпечують новий рівень безпеки, адаптивності та ефективності в керуванні.

Зі зростанням популярності вбудованих платформ, таких як Raspberry Pi та ESP32, з’явилася можливість реалізовувати системи з чітким розподілом обов’язків між низькорівневими та високорівневими модулями. У таких системах високорівнева логіка відповідає за прийняття рішень і обробку даних, а низькорівнева – безпосередньо за управління виконавчими механізмами [3]. У поєднанні з протоколами передачі даних, системами безпеки та зворотного зв’язку це створює передумови для побудови масштабованих інтелектуальних рішень.

Об’єктом дослідження є процес керування гібридним роботизованим маніпулятором, що використовує пневматичні та сервоприводні механізми. Предметом дослідження є методи побудови апаратно-програмної системи керування із забезпеченням обміну даними, зворотного зв’язку та підтримкою інтеграції з інтелектуальними алгоритмами.

Метою кваліфікаційної роботи є розробка, реалізація та тестування системи управління гібридним роботизованим маніпулятором із урахуванням архітектури низького та високого рівня, логіки команд, зворотного зв’язку та підтримки інтеграції з AI.

Для досягнення мети в роботі будуть виконані наступні завдання:

– розробити архітектуру системи управління з розподілом обов’язків між Raspberry Pi та ESP32;

– спроєктувати протокол обміну даними між модулями, з урахуванням обробки помилок та верифікації команд;

– реалізувати набір базових дій маніпулятора, таких як «захоплення», «відпускання» та «імітація хвата»;

– інтегрувати модулі зворотного зв’язку на основі сенсорів тиску та обробки подій у режимі реального часу;

– запропонувати механізми безпечного завершення дій у разі втрати зв’язку або аварійної ситуації;

– оцінити функціональність та стабільність системи у ході експериментального тестування;

– визначити можливості інтеграції інтелектуальних алгоритмів, включаючи сценарії на основі комп’ютерного зору та навчання.

Методи дослідження, що застосовуються в роботі, включають аналіз наукових джерел, проєктування апаратної та програмної частини системи, експериментальну перевірку роботи маніпулятора, а також моделювання можливих сценаріїв інтеграції зі штучним інтелектом.

Інформаційною базою дослідження слугували наукові публікації, технічна документація до ESP32 та Raspberry Pi, а також власні спостереження та результати експериментів у процесі розробки прототипу.

Кваліфікаційна робота складається з трьох розділів, у яких послідовно розглянуто аналіз існуючих систем управління, описано процес проєктування власної архітектури та наведено результати реалізації й тестування. Робота

завершується узагальненням отриманих результатів та визначенням подальших перспектив розвитку створеної системи.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО УПРАВЛІННЯ РОБОТИЗОВАНИМИ МЕХАНІЗМАМИ

Управління роботизованими механізмами є ключовим елементом у реалізації ефективних, адаптивних та безпечних автоматизованих систем. Від якості побудови керуючої архітектури, вибору апаратних платформ та програмної логіки залежить не лише функціональність, а й здатність механізму реагувати на зміни в середовищі, виконувати складні завдання і взаємодіяти з користувачем [4].

Цей розділ присвячений аналізу сучасних підходів до управління робототехнічними системами. Розглянуто класифікацію систем управління, їх функціональні особливості та принципи побудови. Особливу увагу приділено поділу керування на низькорівневе та високорівневе, підходам до обробки зворотного зв’язку, а також можливостям інтеграції штучного інтелекту. Також здійснено порівняльний аналіз популярних апаратних платформ, що використовуються у розробці роботизованих систем.

Результати цього розділу стануть основою для формування вимог до системи управління, що проєктується у межах цієї кваліфікаційної роботи.

1.1 Класифікація та призначення систем управління в робототехніці

Системи управління виконують ключову роль у робототехнічних системах, забезпечуючи узгоджену взаємодію між апаратною частиною, сенсорними модулями та програмною логікою. Від їх структури, принципів побудови й адаптивності залежить ефективність роботи пристрою, точність виконання завдань, рівень автономності, здатність до адаптації та безпечність у взаємодії з навколишнім середовищем.

Залежно від ступеня участі людини у процесі управління, розрізняють ручні, напівавтоматичні та автоматичні системи. У ручному режимі оператор

безпосередньо контролює дії виконавчого пристрою, наприклад, керує наземним або повітряним дроном у реальному часі. Напівавтоматичне управління поєднує участь людини з автоматизованими функціями, такими як стабілізація або обмежене слідування траєкторії. Автоматичні системи працюють автономно, виконуючи закладену програму або реагуючи на зміну середовища без безпосередньої участі оператора.

Іншою важливою ознакою є наявність зворотного зв’язку. Замкнуті системи керування аналізують інформацію з сенсорів – наприклад, положення елементів, температуру, силу контакту, координати об’єкта – і коригують свої дії в режимі реального часу. Такі системи широко використовуються у медичних роботах, автоматизованих виробничих лініях, мобільних платформах зі слідкуванням за об’єктами, а також у транспортних засобах з автономними функціями.

Системи управління можуть бути жорстко запрограмованими або адаптивними [5]. У перших вся логіка закладена наперед, без можливості зміни поведінки під час виконання; у других передбачено механізми самонавчання або зміни реакцій залежно від попереднього досвіду. Адаптивні системи найчастіше базуються на використанні алгоритмів штучного інтелекту, наприклад, у сервісній робототехніці, де необхідно обслуговувати користувачів із непередбачуваними сценаріями поведінки [6].

Важливою класифікаційною ознакою є архітектура: централізована або децентралізована. Централізовані системи використовують один керуючий модуль, до якого підключаються всі сенсори та виконавчі елементи. Такий підхід спрощує проєктування, але має обмеження у масштабованості та стійкості до відмов. У децентралізованих та розподілених архітектурах функції управління розподіляються між кількома вузлами, які можуть працювати незалежно або координувати дії через мережу. Цей підхід активно застосовується у багатоагентних системах, таких як рої дронів, складські платформи або автономні транспортні засоби, що діють у координації.

Сучасні системи управління також класифікуються за типом використовуваних сигналів: аналогові застосовуються для передавання змінних фізичних параметрів (наприклад, напруги або струму), цифрові – для передавання логічних станів або команд, імпульсні – у керуванні положенням або частотою (зокрема в серво- та крокових приводах). Крім того, активно застосовуються мережеві протоколи для організації зв'язку між модулями – наприклад, I²C, SPI, UART, CAN, а також бездротові стандарти, такі як Bluetooth, Wi-Fi або Zigbee [7].

Програмне забезпечення керуючих систем може мати монолітну або модульну структуру. У монолітних системах уся логіка розміщується в єдиній програмі, що ускладнює зміну окремих функцій. Модульний підхід передбачає поділ функціоналу на незалежні блоки, які взаємодіють через чітко визначені інтерфейси. Це дає змогу розширювати функціональність, додавати нові модулі, оновлювати алгоритми без зміни всієї системи.

Універсальність і масштабованість сучасних систем управління дозволяє реалізовувати їх у широкому спектрі задач – від побутових пристроїв до промислових комплексів, від іграшкових роботів до автономних транспортних засобів. Саме гнучкість архітектури, підтримка зворотного зв'язку та потенціал інтеграції з інтелектуальними компонентами є тими критеріями, що визначають ефективність і життєздатність таких систем у реальних умовах.

1.2 Архітектура управління маніпуляторами

Під архітектурою системи управління маніпулятором розуміють організацію функціональних зв'язків між обчислювальними модулями, сенсорними пристроями та виконавчими елементами, які забезпечують координацію рухів, обробку зворотного зв'язку та взаємодію із зовнішнім середовищем або користувачем. Вибір архітектури визначає гнучкість, продуктивність і масштабованість роботизованої системи.

Загалом виділяють два основних підходи до побудови архітектури керування – централізований та децентралізований. У централізованій моделі всі обчислювальні процеси, прийняття рішень і обробка даних зосереджені в одному головному контролері. Такий підхід є достатньо простим для реалізації та налагодження, проте має обмеження в продуктивності та стійкості до відмов. У разі збоїв центрального модуля вся система втрачає працездатність.

Натомість децентралізовані або розподілені архітектури передбачають розподіл завдань між кількома модулями, кожен з яких виконує певну функцію або управляє окремою частиною механізму. Наприклад, один контролер відповідає за рух у просторі, інший – за керування захватом, а ще один – за аналіз даних із сенсорів. Така побудова дозволяє підвищити надійність системи, спростити модернізацію та реалізувати паралельну обробку задач, що особливо актуально для складних або багатоосевих маніпуляторів [8].

Залежно від рівня абстракції, архітектуру управління умовно поділяють на низькорівневу (low-level) та високорівневу (high-level). Низькорівнева частина відповідає за взаємодію з фізичними пристроями – генерацію керуючих сигналів для двигунів, зчитування показників із сенсорів, реалізацію базових захисних алгоритмів. Високорівнева – оперує логікою поведінки, маршрутизацією рухів, інтерфейсами користувача, обробкою сценаріїв або інтелектуальних модулів. Поділ на рівні дозволяє розмежувати обов’язки між системами з різною обчислювальною потужністю та призначенням, а також спрощує тестування, розширення функціоналу й інтеграцію нових технологій.

На практиці зв’язок між високим і низьким рівнем реалізується через чітко визначені протоколи обміну даними. У сучасних робототехнічних системах поширеними є UART, SPI, I²C, CAN та інші протоколи. Часто використовується модель «майстер-слейв», де один модуль надсилає команди, а інший – виконує їх і надсилає відповідь про стан або результат. Також можливі більш складні схеми, включаючи публікацію-підписку, буферизацію запитів або обробку асинхронних подій.

Архітектура системи управління може також враховувати можливість взаємодії з зовнішніми інформаційними джерелами, наприклад, із комп’ютерним зором, хмарними обчисленнями або модулями штучного інтелекту. У такому разі високорівневий контролер виконує роль посередника між інтелектуальним програмним забезпеченням і виконавчими вузлами, забезпечуючи трансляцію команд у формат, зрозумілий для фізичного рівня системи.

Еволюція архітектури систем керування демонструє рух від монолітних, жорстко зв’язаних схем до гнучких, модульних і адаптивних підходів, що дає змогу будувати системи з можливістю масштабування, повторного використання компонентів та інтеграції новітніх технологій.

1.3 Інтеграція штучного інтелекту у роботизовані системи

Інтеграція штучного інтелекту (ШІ) у роботизовані системи відкриває нові горизонти для автоматизації, автономності та адаптивності керування. Використання інтелектуальних алгоритмів дозволяє роботам не лише виконувати запрограмовані дії, а й приймати рішення на основі аналізу середовища, прогнозування змін, навчання з досвіду та взаємодії з користувачем.

У класичній робототехніці логіка дій зазвичай задається жорстко і не передбачає відхилень від сценарію. Проте у динамічному або частково невизначеному середовищі такий підхід швидко втрачає ефективність. Штучний інтелект забезпечує гнучкість поведінки та адаптацію, що особливо важливо у сферах логістики, медицини, виробництва, обслуговування, військової техніки та персональних сервісних роботів [9].

Найпоширенішими напрямами застосування ШІ в робототехніці є комп’ютерний зір, розпізнавання об’єктів та жестів, прогнозування траєкторій, планування дій, машинне навчання, оптимізація параметрів руху, мовна взаємодія та навігація у складних середовищах. Наприклад, за допомогою нейронних мереж роботи можуть розпізнавати обличчя, визначати розташування

предметів, аналізувати стан робочої поверхні або визначати ступінь тиску при взаємодії з м’якими об’єктами.

Методи машинного навчання, як-от навчання з підкріпленням, використовуються для автоматичної оптимізації дій у змінних умовах, наприклад, у процесі ухилення від перешкод, стабілізації руху, вибору стратегії захоплення або розподілу задач у команді роботів. Нечітка логіка дозволяє інтерпретувати наближені значення сенсорних даних і приймати рішення у випадках, коли точна інформація недоступна або недостовірна.

Інтелектуальні алгоритми все частіше поєднуються із класичними системами керування в межах ієрархічних або гібридних архітектур. У таких системах ШІ відповідає за формування високорівневих рішень – наприклад, вибір цілі або визначення режиму роботи, а низькорівневі контролери забезпечують реалізацію команд із високою точністю та швидкодією. Такий підхід забезпечує баланс між гнучкістю та надійністю [10].

Іншим важливим аспектом є обробка великих обсягів даних, зокрема потоків зображень, багатоканальних сенсорних сигналів, мережевих повідомлень. Для таких завдань активно використовуються платформи з високою обчислювальною здатністю, зокрема одноплатні комп’ютери (наприклад, NVIDIA Jetson, Raspberry Pi з акселераторами, Google Coral) або хмарні інфраструктури з віддаленою обробкою.

Інтеграція ШІ потребує також побудови ефективних протоколів зв’язку, здатних передавати команди, дані сенсорів та результати обчислень з мінімальною затримкою. У розподілених системах важливу роль відіграє синхронізація дій між фізичними модулями та віртуальними інтелектуальними агентами, що виконується через мережеві API, шини повідомлень або платформи типу ROS (Robot Operating System).

Таким чином, впровадження штучного інтелекту у системи управління робототехнічними пристроями дозволяє значно розширити їхні функціональні можливості, забезпечити адаптацію до складних умов і підвищити рівень автономності. Поєднання класичних керуючих алгоритмів із сучасними

AI-рішеннями формує новий стандарт у побудові інтелектуальних роботизованих систем.

1.4 Порівняльний аналіз платформ управління

Вибір апаратної платформи є одним із ключових рішень при проєктуванні системи управління роботизованим механізмом. Від її обчислювальних можливостей, доступної периферії, підтримки мережевих інтерфейсів та програмного середовища залежать функціональність, стабільність і можливість подальшої масштабованості проєкту. Серед найбільш поширених у робототехніці платформ доцільно виділити Arduino, ESP32, STM32, Raspberry Pi та Jetson.

Arduino є простою платформою з широкою спільнотою та низьким порогом входу [11]. Вона добре підходить для навчальних або прототипових проєктів, проте має обмежену продуктивність і не підтримує багатозадачність. ESP32, навпаки, забезпечує вищу обчислювальну потужність, має вбудовані модулі Wi-Fi та Bluetooth, підтримує FreeRTOS і придатна для створення багатофункціональних вбудованих систем [12].

STM32 – це лінійка мікроконтролерів із архітектурою ARM Cortex-M, яка пропонує високу продуктивність, гнучке конфігурування периферії та підтримку промислових протоколів [13]. Raspberry Pi – повноцінний одноплатний комп’ютер з операційною системою Linux, придатний для високорівневої обробки даних, реалізації графічних інтерфейсів і роботи з модулями штучного інтелекту. Jetson – рішення від NVIDIA, орієнтоване на обробку великих обсягів інформації, комп’ютерний зір та глибоке навчання [14].

Порівняльну характеристику зазначених платформ наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика апаратних платформ управління

Платформа	Обчислювальна потужність	Периферія та інтерфейси	Підтримка зв'язку	Переваги	Обмеження
Arduino	Низька	Обмежена кількість I/O	Без вбудовано го зв'язку	Простота, велика спільнота	Відсутність багатозадачності, низька швидкодія
ESP32	Середня (2 ядра)	Велика кількість I/O	Wi-Fi, Bluetooth	Багатозадач-ність, бездротовий зв'язок	Обмежена продуктивність для складних задач
STM32	Висока	CAN, USB, ADC, PWM	Залежить від моделі	Стабільність, точність, масштабова-ність	Складніше програмування, вищий поріг входу
Raspberry Pi	Висока (до 4 ядер)	USB, GPIO, HDMI	Ethernet, Wi-Fi, Bluetooth	ОС Linux, гнучкість, підтримка Python	Високе енергоспоживання, не реального часу
Jetson Nano	Дуже висока (GPU + CPU)	USB, CSI, I²C, GPIO	Ethernet, Wi-Fi (через модуль)	Придатність до AI, паралельна обробка	Ціна, енергоспоживання, складність інтеграції

Як видно з таблиці, платформи істотно відрізняються за своїми характеристиками. Arduino та ESP32 найкраще підходять для простих низькорівневих задач; STM32 забезпечує надійність у промислових застосуваннях; Raspberry Pi дозволяє реалізовувати складну високорівневу логіку; Jetson орієнтований на використання в системах із інтенсивною обробкою даних або штучним інтелектом.

Таким чином, правильний вибір платформи повинен враховувати характер задачі, обмеження середовища експлуатації, вимоги до автономності, енергоспоживання та можливість подальшого масштабування.

1.5 Аналіз програмних середовищ для реалізації систем управління

Програмне забезпечення відіграє не менш важливу роль, ніж апаратна платформа, у побудові систем керування роботизованими механізмами. Від

вибору середовища розробки, мови програмування, операційної системи та доступних бібліотек залежить не лише функціональність системи, а й швидкість розробки, стабільність, можливість інтеграції з іншими компонентами та підтримка подальшої модифікації.

Для вбудованих мікроконтролерів, таких як Arduino, ESP32 або STM32, найчастіше застосовуються спеціалізовані середовища з мінімальним графічним інтерфейсом. Arduino IDE, зокрема, є популярною платформою для початкових проєктів – вона підтримує простий синтаксис на основі мови C/C++, має численні бібліотеки для роботи з датчиками та приводами, а також дозволяє швидко прототипувати керуючу логіку [15]. Для складніших задач на базі ESP32 або STM32 зазвичай використовують середовища, які дозволяють реалізовувати багатозадачність і роботу в реальному часі – такі як PlatformIO, STM32CubeIDE або середовище розробки на базі FreeRTOS. FreeRTOS є мікроядром реального часу, яке дозволяє створювати системи з точною координацією задач, що критично важливо для синхронного керування кількома виконавчими модулями.

Для більш складних платформ на зразок Raspberry Pi або Jetson, які працюють під управлінням повноцінної операційної системи Linux, можливості значно ширші. Тут програмування може здійснюватися на мовах високого рівня – Python, C++, JavaScript або Go – з використанням стандартних засобів Linux-розробки, таких як VS Code, Geany або навіть віддалені IDE. Платформа Raspberry Pi, зокрема, надає доступ до таких фреймворків, як OpenCV для комп’ютерного зору, MQTT для обміну повідомленнями між модулями, Flask або Node-RED для створення вебінтерфейсів. Вона також підтримує взаємодію з USB-камерою, зовнішніми API та базами даних, що дозволяє реалізовувати складні високорівневі сценарії керування.

Окремої уваги заслуговує Robot Operating System (ROS) – модульна платформа для побудови складних роботизованих систем. ROS забезпечує інфраструктуру для обміну повідомленнями між модулями, керування трансформаціями координат, обробки сенсорної інформації, побудови карт та автономної навігації. Хоча використання ROS має досить високий поріг входу,

його широкі можливості роблять його незамінним у складних проектах, зокрема тих, що пов’язані з інтелектуальними агентами або багатоагентними системами. ROS сумісна з Python і C++, має відкритий вихідний код і підтримує розширення через спеціалізовані пакети [16].

Вибір середовища розробки залежить від обраної платформи, рівня складності системи, вимог до точності виконання команд, необхідності в обробці великого обсягу даних та можливості масштабування. У простих системах достатньо використання Arduino IDE або PlatformIO; у середньої складності – FreeRTOS або STM32Cube; для високорівневих сценаріїв, обробки зображень або розгортання інтелектуальних функцій – повноцінні середовища Linux із підтримкою Python, ROS, AI-фреймворків та інструментів віддаленого керування.

Таким чином, сучасна екосистема програмного забезпечення для робототехніки пропонує широкі можливості як для створення компактних вбудованих систем, так і для розгортання складних інтелектуальних платформ. Гнучкість вибору середовища дозволяє адаптувати систему керування під конкретні технічні вимоги, зберігаючи при цьому модульність і перспективу подальшого розвитку.

Проведений аналіз сучасних підходів до управління роботизованими механізмами дозволив сформувати уявлення про ключові принципи побудови систем керування, їхню класифікацію, функціональні особливості та напрямки розвитку. Розглянуто відмінності між централізованими та децентралізованими архітектурами, роль низькорівневого та високорівневого управління, можливості використання зворотного зв’язку та адаптивних механізмів. Окрему увагу приділено інтеграції штучного інтелекту, що відкриває перспективи автономного навчання та складної поведінкової логіки. У результаті порівняльного аналізу апаратних платформ та програмних середовищ визначено основні критерії вибору рішень для різних типів задач, а також обґрунтовано доцільність поєднання апаратного та програмного рівня в єдину адаптивну

керуючу систему. Отримані висновки ляжуть в основу подальшого проєктування власної архітектури управління у наступному розділі.

РОЗДІЛ 2
ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ МАНІПУЛЯТОРОМ

На основі аналізу сучасних технологій керування, поданого у попередньому розділі, визначено загальні підходи, які можуть бути застосовані для реалізації системи управління роботизованим маніпулятором. У цьому розділі здійснюється проєктування архітектури, що поєднує низькорівневі модулі управління з високорівневими обчислювальними системами, забезпечуючи їх взаємодію через стандартизовані інтерфейси обміну даними.

Розробка системи управління передбачає визначення апаратних та програмних компонентів, побудову структури обміну командами та відповідями, реалізацію основних режимів роботи маніпулятора та логіки взаємодії з сенсорними модулями. Крім основної функціональності, також розглянуто засоби базового захисту та перевірки працездатності системи в реальному часі. Усі рішення приймаються з урахуванням технічних обмежень обраних платформ і практичних вимог до стабільності та безпеки керування.

2.1 Постановка задачі

У межах цієї кваліфікаційної роботи розробляється система управління для роботизованого маніпулятора, який було попередньо реалізовано як гібридну конструкцію з використанням електричних приводів і пневматичних виконавчих елементів. Маніпулятор має сегментну будову та здатний виконувати базові дії, пов’язані із захопленням, утриманням і переміщенням об'єктів у просторі. В основі конструкції використано обчислювальні модулі типу ESP32 для низькорівневого керування та окрему обчислювальну платформу для реалізації високорівневої логіки.

На відміну від традиційних реалізацій, у яких система управління зосереджена в одному пристрої, у запропонованому підході передбачено чіткий розподіл між низьким та високим рівнями керування. Низькорівнева частина

відповідає за безпосередню активацію виконавчих механізмів – зокрема, сервоприводів, соленоїдних клапанів і зчитування сенсорних даних. Високорівнева – забезпечує інтерпретацію команд користувача, обробку подій, передачу сигналів та прийняття рішень щодо дій маніпулятора.

Серед основних вимог до системи управління слід виділити здатність точно інтерпретувати команди з високого рівня, здійснювати своєчасну реакцію на події зворотного зв’язку (наприклад, досягнення граничного тиску, виявлення контакту або втрату зв’язку), реалізовувати декілька режимів дій маніпулятора та підтримувати базові механізми перевірки цілісності команд. Крім того, система має бути сумісною з принципами подальшого розширення функціоналу, включно з можливістю інтеграції алгоритмів штучного інтелекту чи комп’ютерного зору, реалізація яких розглядається як окремий напрям у межах проєкту.

Таким чином, задача полягає у створенні модульної, стабільної та адаптованої до практичного застосування системи управління, яка взаємодіє з гібридним роботизованим маніпулятором, координує його дії та забезпечує коректну роботу відповідно до заданих сценаріїв.

2.2 Архітектура системи управління

Побудова ефективної системи управління для роботизованого маніпулятора передбачає створення чіткої, логічно структурованої архітектури, у якій апаратні й програмні компоненти взаємодіють на різних рівнях абстракції. Основу архітектурного рішення становить ідея розподілу функціональності між низькорівневим мікроконтролером, відповідальним за пряме керування виконавчими елементами, та високорівневим обчислювальним модулем, що реалізує загальну логіку поведінки системи. Структуру системи наведено на рисунку 2.1.

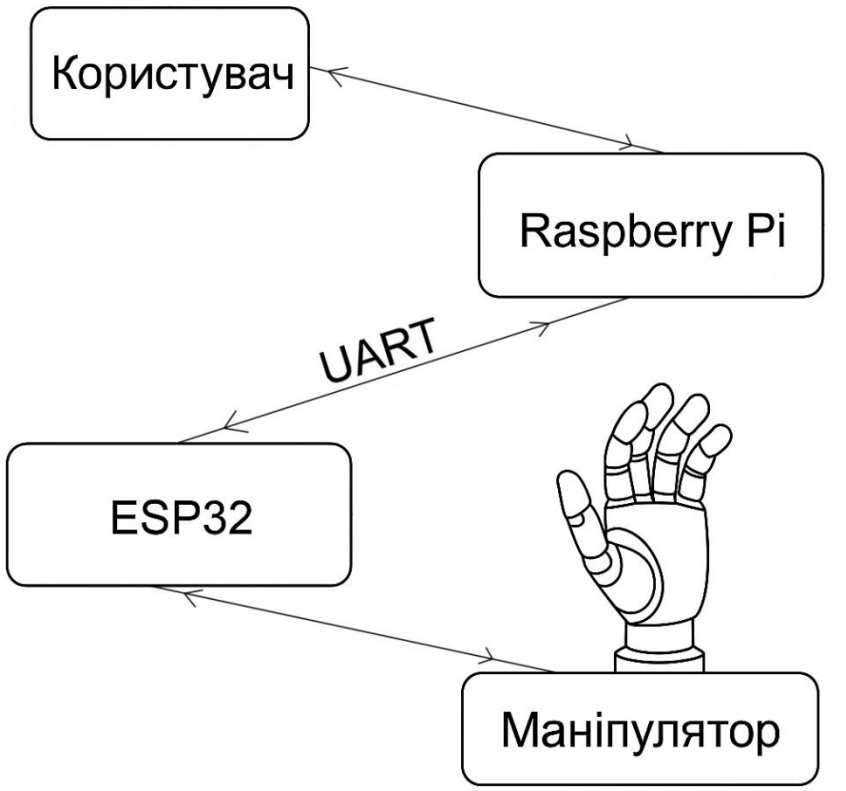

Рисунок 2.1 – Архітектура системи управління маніпулятором

2.2.1 Загальна структура системи

Архітектура системи управління побудована за дворівневою моделлю, яка передбачає розділення функціональності між низькорівневим мікроконтролером та високорівневим обчислювальним модулем. Такий підхід дозволяє ефективно розподіляти навантаження, підвищити стабільність роботи та забезпечити адаптацію до завдань різної складності. Нижній рівень забезпечує пряме керування виконавчими елементами та зчитування сенсорних даних, тоді як верхній рівень виконує логіку сценаріїв, прийняття рішень, обробку команд користувача та взаємодію з іншими програмними компонентами.

2.2.2 Роль ESP32 як низькорівневого контролера

Низькорівнева частина реалізована на базі мікроконтролера ESP32, який має достатню продуктивність для керування електромеханічними та пневматичними виконавчими механізмами, а також забезпечення первинної обробки даних від сенсорів. Цей модуль відповідає за активацію сервоприводів,

керування соленоїдними клапанами, зчитування показників із сенсорів тиску, генерацію таймаутів і виконання дій за запитом з високого рівня. Важливою перевагою ESP32 є підтримка багатозадачності та низьке енергоспоживання, що дозволяє забезпечити швидкий відгук системи на зміну зовнішніх умов. Команди, що надходять від високорівневого модуля, інтерпретуються та реалізуються без затримок, що забезпечує плавну та стабільну поведінку маніпулятора [17].

2.2.3 Роль Raspberry Pi у високорівневому керуванні

Високорівнева логіка системи реалізована на основі одноплатного комп’ютера Raspberry Pi. Цей модуль працює під керуванням операційної системи Linux, що забезпечує гнучкість у розробці програмного забезпечення, можливість створення графічних інтерфейсів, підключення до мережі, обробки великих обсягів даних та взаємодії з потенційними модулями штучного інтелекту. На цьому рівні виконується формування послідовностей дій, обробка команд від користувача, аналіз стану системи, ведення журналів подій, а також формування запитів до ESP32. Окремо передбачено можливість розширення функціоналу шляхом підключення зовнішніх обчислювальних бібліотек або сервісів.

2.2.4 Міжрівнева взаємодія та тип комунікації

Комунікація між ESP32 та Raspberry Pi реалізується за допомогою інтерфейсу UART, що забезпечує просту і надійну передачу даних із мінімальною затримкою. Обмін будується на основі фреймованих текстових повідомлень у структурованому форматі (наприклад, JSON або CSV), що дозволяє легко інтерпретувати команди та відповіді на обох рівнях. Для зменшення ймовірності помилок передбачено механізми перевірки цілісності повідомлень, а також обробку ситуацій, пов’язаних із втратою зв’язку або перевищенням часу очікування. Такий підхід дозволяє забезпечити стабільну синхронізацію між модулями навіть у разі непередбачуваних затримок або короткочасних збоїв.

Архітектура системи передбачає модульність і розділення відповідальності між компонентами, що спрощує як налагодження, так і подальшу розробку. Такий підхід забезпечує надійність, гнучкість у реалізації логіки керування, а також сумісність із різними сценаріями практичного використання маніпулятора.

2.3 Протокол обміну даними між модулями

Надійна взаємодія між модулями управління є критичним елементом функціонування системи, в якій низькорівневий та високорівневий компоненти виконують чітко розділені функції. Для забезпечення синхронності, узгодженості дій та вчасної реакції на події використовується послідовний протокол зв’язку, що дозволяє передавати структуровані команди та отримувати відповіді з мінімальною затримкою.

2.3.1 Формат команд і відповідей

Для обміну між Raspberry Pi та ESP32 використовується інтерфейс UART, який забезпечує асинхронну передачу даних у текстовому вигляді [18]. Команди передаються у вигляді рядків фіксованої структури, наприклад: «CMD:GRAB;FORCE:75;», де CMD визначає дію (у даному випадку – захоплення), а FORCE – цільове значення натискання (у відсотках або умовних одиницях, залежно від інтерпретації FSR-датчика).

Відповіді від ESP32 також мають уніфіковану форму, наприклад: «STATUS:OK;», «FSR:61;», «ERROR:TIMEOUT;». Це дозволяє високорівневому модулю отримувати підтвердження виконання команди або фактичне значення сили тиску, виміряне датчиком. Такий формат легко обробляється за допомогою регулярних виразів або простих парсерів у середовищах Python або C++.

2.3.2 Обробка помилок і верифікація

Кожне вхідне повідомлення перевіряється на відповідність очікуваному формату. У разі некоректного запиту або порушення синтаксису ESP32 повертає

«STATUS:ERROR;», а команда ігнорується. Успішно прийнята команда виконується однократно або до досягнення умов зупинки, залежно від сценарію.

Передбачено також підтвердження виконання команд – наприклад, через «DONE;» після завершення дії, або «FSR:VALUE;», якщо передбачено контроль сили захоплення в реальному часі.

2.3.3 Таймінги, буферизація та швидкість обміну

Обмін між модулями здійснюється на швидкості 115200 бод. Це дозволяє передавати команди та відповіді у режимі близькому до реального часу без значного навантаження на процесор або буфери UART.

У разі відсутності відповіді протягом заданого тайм-ауту (наприклад, 100-200 мс) система переходить у захищений режим або повторює запит. Буфери з обох боків працюють за принципом FIFO, що дозволяє зберігати порядок команд навіть при незначних затримках. Такий підхід забезпечує надійну взаємодію навіть за умов високої частоти обміну та інтенсивної роботи з виконавчими модулями.

2.4 Логіка керування жестами / діями

Керування роботизованим маніпулятором реалізується у вигляді набору окремих жестів або дій, що виконуються на основі команд, які надходять з високорівневого модуля. Кожна дія складається з послідовності сигналів, що передаються на виконавчі механізми, та умов завершення, які контролюються за допомогою сенсорів або програмної логіки. Для реалізації адаптивної поведінки маніпулятора передбачено окрему систему обробки подій, що виникають під час виконання дій.

2.4.1 Сценарії поведінки маніпулятора

У поточній реалізації передбачено кілька базових жестів, які відповідають типовим режимам роботи маніпулятора. Зокрема, реалізовано сценарій захоплення об’єкта, відпускання, імітацію хвата без зусилля, утримання до досягнення порогу тиску на датчику FSR, а також перехід у пасивний режим.

Кожен сценарій запускається окремою командою з високого рівня, наприклад «CMD:GRAB;», «CMD:RELEASE;» або «CMD:HOLD;».

Зміст дій у межах кожного сценарію визначається фіксованими послідовностями активації клапанів та сервоприводів, які координуються з урахуванням часу та стану системи.

2.4.2 Алгоритми реалізації дій

Виконання кожного жесту починається з надходження команди на ESP32. Залежно від коду команди, мікроконтролер активує відповідні виконавчі елементи. Наприклад, у випадку «CMD:GRAB;FORCE:75;», система активує пневматичний привід пальців та починає зчитувати дані з FSR-датчика. При досягненні заданого рівня натискання клапан автоматично перекривається.

Аналогічно, команда «CMD:RELEASE;» зупиняє подачу повітря, відкриває зворотні клапани або повертає сервоприводи у початкове положення. Команда «CMD:HOLD;» може активувати утримання в заданій позиції, доки не надійде інша команда або не буде перевищено допустимий час утримання.

Логіка реалізована як набір функцій або станів, що викликаються залежно від отриманого запиту та поточного контексту роботи системи.

2.4.3 Реакція на події та зворотний зв’язок

У межах кожного жесту система здатна реагувати на події в режимі реального часу. Найважливішим параметром є сила натискання, яку фіксує FSR-датчик. У разі досягнення порогового значення або раптового зростання сили система може достроково завершити дію або перейти у безпечний режим.

Крім того, у випадках втрати зв’язку, таймауту або внутрішньої помилки ESP32 автоматично зупиняє активні виконавчі елементи та надсилає відповідне повідомлення «ERROR:TIMEOUT;» або «STATUS:FAIL;». Це дозволяє високорівневому контролеру повторно ініціювати дію або сповістити користувача про помилку.

Такий підхід дозволяє забезпечити базову адаптивність системи навіть без використання повноцінного інтелектуального компонента, лише за рахунок правильно побудованої логіки сценаріїв і реакції на події.

2.5 Модулі зворотного зв’язку та адаптації

Наявність зворотного зв’язку є важливою складовою системи управління роботизованим маніпулятором, оскільки дозволяє контролювати хід виконання дій, виявляти відхилення від заданих параметрів та вчасно реагувати на зміну зовнішніх умов. У поточній реалізації передбачено використання сенсора сили натискання (FSR), який забезпечує найпростіший варіант зворотного зв’язку [19]. Також програмна частина дозволяє реалізувати базові механізми адаптації до ситуації без залучення складних інтелектуальних алгоритмів.

2.5.1 Використання FSR-датчика як джерела зворотного зв’язку

Основним сенсорним елементом у системі є датчик сили натискання (FSR), розташований на одному з пальців маніпулятора. Він дозволяє вимірювати механічний тиск, що виникає при контакті з об’єктом. Показники зчитуються через аналоговий вхід ESP32, обробляються та перетворюються у цифрове значення для подальшого аналізу. Це значення є індикатором досягнення фізичного контакту або перевищення допустимого зусилля.

FSR використовується як умова завершення дії: наприклад, якщо сила натискання перевищує певний поріг, команда захоплення припиняється, і клапан перекривається. Таким чином реалізується елементарний, але ефективний механізм самозупинки.

2.5.2 Програмна обробка сенсорних даних

Значення, отримані з FSR, обробляються у прошивці ESP32 в реальному часі. Усі вхідні сигнали проходять базову фільтрацію для зменшення впливу шумів. Програмна логіка дозволяє порівнювати значення з заданим порогом, а також формувати відповідні сигнали до високого рівня – наприклад, «FSR:82;», де 82 – це умовна одиниця сили натискання.

Залежно від отриманих даних система може переходити до наступного стану, перезапускати дію або викликати аварійну зупинку. Це забезпечує базову адаптивність до різних об’єктів – м’яких, жорстких або нестійких – без зміни загальної логіки роботи.

2.5.3 Перспективи розширення адаптації

Хоча наразі система працює з одним сенсором, її архітектура передбачає можливість масштабування – наприклад, додавання FSR-датчиків на інші пальці або використання датчиків кута, положення, тиску повітря, IMU. Це дозволить реалізувати складніші алгоритми адаптації, включаючи динамічне коригування сили захоплення або розпізнавання типу контакту.

Окремим напрямком розвитку є можливість підключення AI-компонентів, які отримують сенсорні дані та аналізують їх у контексті цілісного середовища. Це відкриває перспективу для впровадження машинного навчання, контролю за допомогою комп’ютерного зору або інтеграції зі сценаріями, заснованими на поведінкових моделях.

2.6 Безпека використання системи управління

Управління роботизованими механізмами завжди пов’язане з потенційними ризиками – як для самої системи, так і для навколишнього середовища або користувача. З огляду на це, система управління маніпулятором має включати базові механізми безпеки, які дозволяють своєчасно виявляти та нейтралізувати критичні ситуації. У розробленій архітектурі безпека реалізується на рівні програмної логіки як високого, так і низького рівня.

Одним із головних аспектів є контроль наявності зв’язку між Raspberry Pi та ESP32. У випадку втрати сигналу, некоректного формату команди або перевищення дозволеного часу очікування, ESP32 переходить у безпечний стан. Зокрема, у разі неотримання підтвердження або переривання обміну даними, мікроконтролер автоматично зупиняє активні виконавчі елементи, перекриває подачу повітря та повертає сервоприводи у нейтральне положення. Такий підхід дозволяє мінімізувати ризик неконтрольованих дій у ситуаціях збоїв [20].

На рівні високорівневого модуля передбачено перевірку цілісності команд, контроль відповідей та логування системних подій. У разі надходження повідомлення про помилку – «ERROR:TIMEOUT;», «ERROR:FORMAT;», або

«STATUS:FAIL;» – система зупиняє поточну дію, попереджає користувача та переходить у очікувальний режим. Таким чином реалізується двосторонній захист, при якому кожна сторона контролює не лише свою працездатність, а й коректність дій іншої.

Окремо варто відзначити програмні обмеження на параметри команд. Значення сили, тривалості або режиму дії проходять перевірку на допустимість до їх виконання. Це унеможливлює виконання шкідливих або випадкових команд, які можуть спричинити пошкодження системи або оточення.

У разі необхідності система може бути доповнена апаратними засобами аварійного вимкнення, зокрема реле з незалежним живленням, які можуть відключити подачу напруги або стисненого повітря в екстреній ситуації. Така функція не є обов’язковою для прототипу, але рекомендована для майбутнього використання в реальних умовах експлуатації.

Таким чином, запропоновані механізми забезпечують базовий рівень безпеки при взаємодії з маніпулятором, дозволяють виявляти потенційні збої на ранньому етапі та запобігати небажаним наслідкам без використання складних захисних систем.

У результаті проведеного проєктування було сформовано архітектуру системи управління, яка поєднує високорівневий та низькорівневий модулі з чітким розподілом обов’язків. Визначено принципи міжрівневої взаємодії, побудовано протокол обміну даними, реалізовано логіку основних дій маніпулятора, а також забезпечено обробку зворотного зв’язку з FSR-датчика. Система доповнена базовими засобами захисту та контролю помилок, що підвищує її стабільність і придатність до практичного використання. Запропоновані рішення створюють основу для фізичної реалізації та подальшого тестування функціональності в умовах реального середовища. Усі складові проєкту підібрано з урахуванням доступності, сумісності та простоти повторення, що дозволяє розглядати систему як відкриту платформу для освітніх або дослідницьких цілей.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ СИСТЕМИ УПРАВЛІННЯ

На основі попередньо спроєктованої архітектури та зібраної фізичної частини роботизованого маніпулятора було реалізовано повноцінну систему управління, що охоплює як програмне забезпечення для обчислювальних модулів, так і міжрівневу взаємодію. Апаратна частина, яка слугує виконавчим середовищем для розробленої системи, базується на конструкції, детально описаній у попередній кваліфікаційній роботі.

У межах цього розділу подано опис фізичної платформи, на якій розгорнуто систему керування, наведено фрагменти реалізованого коду для ESP32 та Raspberry Pi, а також представлено результати тестування основних сценаріїв поведінки маніпулятора. Окрему увагу приділено аналізу стабільності роботи системи, її реакції на події та точності виконання команд у динамічних умовах.

3.1 Огляд реалізації прототипу маніпулятора

Фізичною основою для реалізації системи управління слугує готовий прототип гібридного роботизованого маніпулятора, який був виготовлений раніше в межах окремого проєкту. Конструкція включає пневматичні м’язи як основні виконавчі елементи для пальців, а також сервоприводи, відповідальні за позиціонування сегментів маніпулятора на рівні зап’ястя та ліктя. Компактна механічна рама виконана у вигляді пантографної системи з можливістю фіксації на робочій поверхні.

Усі виконавчі модулі закріплено на єдиній монтажній платформі, що також включає блоки електроніки, джерело живлення та модулі зворотного зв’язку. Для керування пневматичними приводами використано електромагнітні клапани з живленням 12 В, які підключаються до виходів ESP32 через ключові

транзистори. Керування сервоприводами здійснюється безпосередньо з цифрових виходів мікроконтролера з використанням PWM-сигналів.

Сенсорна система представлена одним датчиком сили натискання (FSR), закріпленим на робочій поверхні одного з пальців маніпулятора. Цей датчик дозволяє отримувати дані про наявність контакту з об’єктом і використовувати їх у логіці завершення дії або аварійної зупинки. Усі з’єднання виконано за допомогою роз’ємів, що дозволяє оперативно обслуговувати або замінювати компоненти.

Для обробки високорівневої логіки використовується одноплатний комп’ютер Raspberry Pi, який розташовано окремо від маніпулятора та підключено до ESP32 через інтерфейс UART. На платі також розміщено світлодіодні індикатори для сигналізації стану системи та діагностики процесу виконання команд.

Після складання всі компоненти пройшли візуальну перевірку, перевірку цілісності з’єднань та попереднє налагодження. Фінальна інтеграція здійснювалась на стадії тестування програмного забезпечення, описаного у наступних підрозділах.

3.2 Розробка програмного забезпечення для Raspberry Pi

Програмне забезпечення високорівневого модуля реалізовано на базі операційної системи Raspberry Pi OS із використанням мови програмування Python. Такий вибір обумовлений широкою підтримкою периферійних модулів, простотою реалізації логіки сценаріїв керування та можливістю швидкого масштабування за рахунок бібліотек для роботи з UART, потокової обробки подій, логування, візуалізації та підключення до мережі.

В основі програмного забезпечення лежить циклічний обробник подій, який отримує команди з інтерфейсу користувача або зовнішнього джерела, формує відповідні повідомлення для ESP32, надсилає їх через послідовний порт,

а також очікує та обробляє відповіді. Програма підтримує чергу команд, логування запитів і реакцій, а також можливість зміни режимів у реальному часі.

Передбачено окремі функції для ініціалізації з'єднання, обробки таймаутів та аварійних ситуацій. У випадку відсутності відповіді протягом визначеного часу виконується повторна спроба або система переходить у безпечний режим. Кожна дія завершується логічним сигналом «DONE» або обробкою значення, отриманого від сенсора.

Нижче наведено приклад фрагмента коду, що відповідає за надсилання команди захоплення з певною силою натискання (рисунок 3.1):

```
def send_grip_command(uart, force: int):
    if not (0 <= force <= 100):
        logging.warning(f"Недопустиме значення сили: {force}. Має бути в діапазоні 0-100.")
        return

    command = f"GRIP:{force}"
    try:
        uart.write((command + "\n").encode())
        logging.info(f"Надіслано команду захоплення: {command}")
    except Exception as e:
        logging.error(f"Помилка при надсиланні команди захоплення: {e}")
```

Рисунок 3.1 – Функція надсилання команди захоплення через UART

Також реалізовано обробник відповіді від мікроконтролера, який визначає результат виконання дії та генерує наступні команди у відповідь на ситуацію (рисунок 3.2):

```
def handle_response(uart, timeout=3):
    start = time.time()
    while time.time() - start < timeout:
        if uart.in_waiting:
            response = uart.readline().decode().strip()
            logging.info(f"Відповідь: {response}")
            return response
        time.sleep(0.1)
    logging.warning("Таймаут очікування відповіді.")
    return None
```

Рисунок 3.2 – Функція обробки відповіді від ESP32

Для зручності керування реалізовано консольний інтерфейс, який дозволяє запускати окремі сценарії вручну, а також переглядати поточний стан системи. У разі необхідності реалізацію можна доповнити графічним інтерфейсом або системою віддаленого моніторингу через мережу.

3.3 Прототип інтерфейсу користувача

Для взаємодії з системою управління розроблено базовий інтерфейс користувача у вигляді консольної утиліти, що працює в середовищі операційної системи Raspberry Pi. Основна мета інтерфейсу – забезпечити зручний спосіб тестування функціоналу, виконання команд, моніторингу поточного стану маніпулятора та швидкої діагностики помилок.

Консольна утиліта (рисунок 3.3) дозволяє вручну запускати основні сценарії, зокрема: захоплення, відпускання, утримання, скидання стану. Усі команди вводяться через командний рядок та передаються до внутрішнього обробника, який формує правильну структуру повідомлення й надсилає його на ESP32 через UART. Користувач отримує зворотний зв’язок у вигляді статусів виконання, повідомлень про помилки або значень з FSR-датчика.

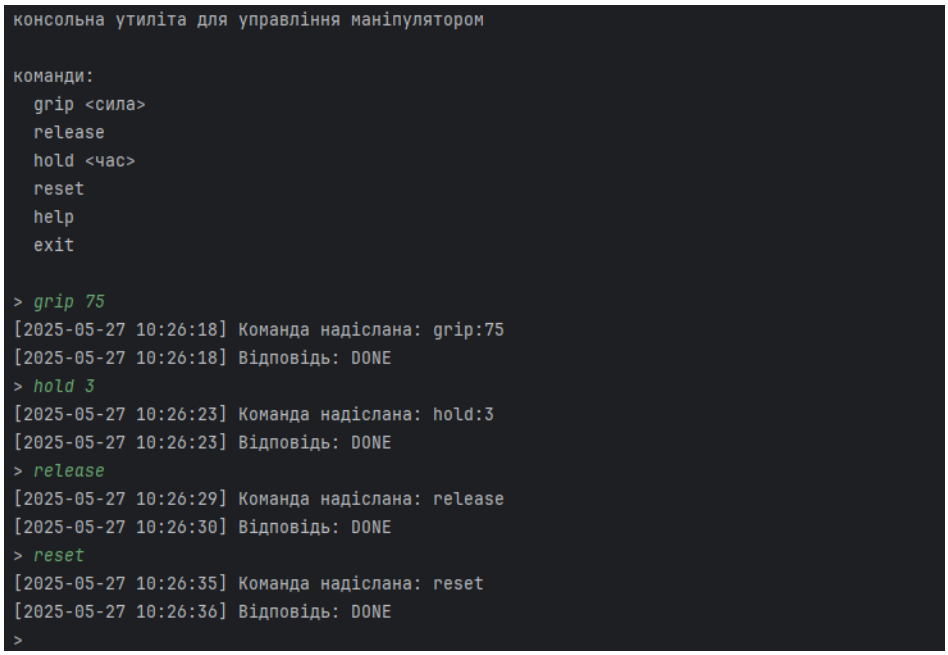

Рисунок 3.3 – Консольна утиліта для управління маніпулятором

Програма підтримує базові параметри, зокрема рівень зусилля захоплення, тривалість утримання, та має просту допоміжну систему для перегляду доступних команд. У перспективі утиліту можна доповнити графічним або вебінтерфейсом, однак у межах цього проєкту було зосереджено увагу на стабільності і мінімальній затримці взаємодії.

Завдяки простоті реалізації, консольний інтерфейс дозволив швидко перевірити працездатність логіки, діагностувати проблеми взаємодії між модулями та забезпечити початкову інтеграцію з користувачем без використання зовнішніх середовищ або бібліотек.

3.4 Реалізація компонента штучного інтелекту

У межах розробленої системи управління реалізовано базовий компонент штучного інтелекту, який відповідає за прийняття рішень на високому рівні на основі вхідних даних та контексту виконання дій. Основне призначення цього компонента – забезпечити адаптивну поведінку маніпулятора залежно від стану зворотного зв’язку та визначених стратегій взаємодії з об’єктами.

Алгоритм реалізовано у вигляді окремого модуля на Raspberry Pi, який інтегрується з основною логікою керування. Компонент аналізує дані, отримані від ESP32 (наприклад, значення сили натискання з FSR), а також оцінює історію попередніх дій, затримки виконання та наявність помилок. На основі цих параметрів система приймає рішення про доцільність повторного захоплення, зміни сили натискання або переходу до наступного режиму.

Нейронні мережі в поточній реалізації не використовуються – логіка побудована на основі простих евристичних правил та умов, які можуть бути легко адаптовані або замінені в подальшому. Наприклад, якщо значення з FSR залишається нижчим за поріг упродовж певного часу, AI-модуль генерує нову команду «CMD:RETRY;FORCE:+10%;», що відповідає спробі збільшити зусилля захоплення.

Нижче наведено приклад фрагмента коду, який демонструє принцип адаптивного ухвалення рішення на основі сенсорних даних (рисунок 3.4):

```
1 usage
def should_retry(self) -> bool:
    if not self.history:
        return False

    recent = self.history[-1]
    time_passed = time.time() - recent["time"]

    # Умова повтору: сила низька або була помилка, і минув час затримки
    if (recent["fsr"] < FSR_THRESHOLD or recent["error"]) and time_passed > RETRY_DELAY:
        return True
    return False

1 usage
def generate_retry_command(self) -> str:
    # Збільшуємо силу захоплення
    self.current_force = int(self.current_force * (1 + FORCE_INCREMENT))
    self.retry_count += 1
    return f"CMD:RETRY;FORCE:{self.current_force};"
```

Рисунок 3.4 – Базовий алгоритм адаптації з використанням FSR-значення

Крім цього, компонент має механізм самонавчання на основі журналу подій. Всі дії, що завершуються з помилкою або вимушеним переходом у безпечний режим, записуються у лог, який аналізується при наступному запуску системи. Це дозволяє формувати шаблони поведінки, які зменшують ймовірність повторення критичних ситуацій.

У перспективі реалізований підхід може бути легко адаптований до використання алгоритмів машинного навчання, зокрема класифікації типів об’єктів, прогнозування сили захоплення або вибору сценарію за допомогою попередньо натренованої моделі. Вже зараз структура програми дозволяє інтегрувати Python-бібліотеки TensorFlow Lite, scikit-learn або OpenCV при підключенні відповідних сенсорів або камер.

Таким чином, реалізований компонент ШІ відіграє роль високорівневого адаптивного контролера, що приймає рішення на основі поточних даних та історії взаємодії. Навіть у базовому вигляді він демонструє здатність до адаптації, що дозволяє розглядати його як початковий рівень інтелектуалізації системи керування маніпулятором.

3.5 Тестування та оцінка функціональності системи

Після завершення реалізації програмних модулів та інтеграції високорівневої логіки, включаючи базовий компонент штучного інтелекту, було проведено комплексне тестування системи управління. Метою тестування було перевірити стабільність роботи, відповідність поведінки маніпулятора заданим сценаріям, реакцію на зміну умов та коректність взаємодії між компонентами.

Перевірку системи розділено на кілька етапів. Спочатку тестувалась якість з'єднання між Raspberry Pi та ESP32. Була підтверджена стабільна передача команд і відповідей на швидкості 115200 бод без втрати пакетів. Затримка виконання команд становила в середньому 30-45 мс, що є прийнятним для управління пневматичними та сервоприводними механізмами.

На другому етапі перевірялась зміна сили стискання предметів різної густини та жорсткості (рисунок 3.5). Усі команди виконувались згідно з очікуваним сценарієм. Система реагувала на дані з FSR-датчика в реальному часі.

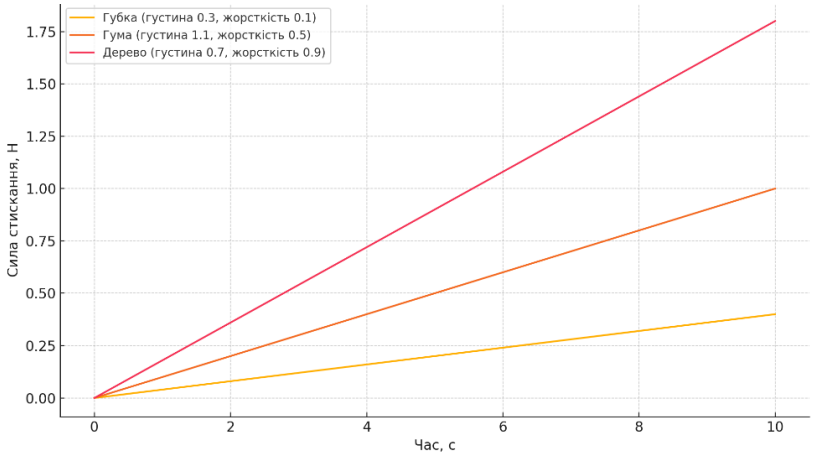

Рисунок 3.5 – Графік зміни сили стискання предметів різної густини та жорсткості

На третьому етапі проводилось порівняння часу відклику апаратної частини при ручному керуванні та керуванні за допомогою системи з елементами штучного інтелекту. Виявлено, що час відклику при ручному керуванні

коливався в межах 400-600 мс, тоді як система з ШІ забезпечувала реакцію в середньому за 100-200 мс (рисунок 3.6). Це пояснюється автоматичним аналізом вхідних даних і відсутністю людського фактора. Таким чином, використання ШІ дозволяє значно зменшити затримку реакції системи.

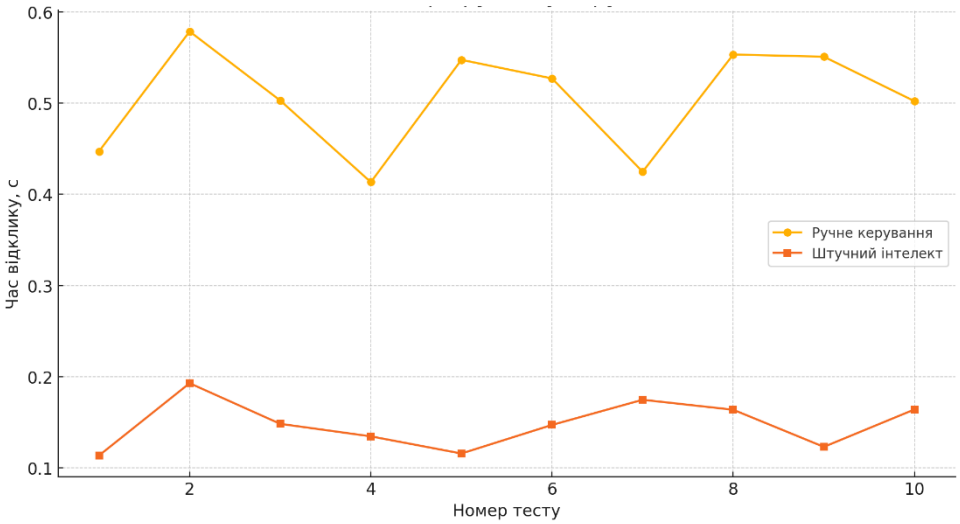

Рисунок 3.6 – Порівняння часу відклику апаратної частини при ручному керуванні та з ШІ

На четвертому етапі оцінювалась залежність точності захоплення об’єктів від типу керування (рисунок 3.7):

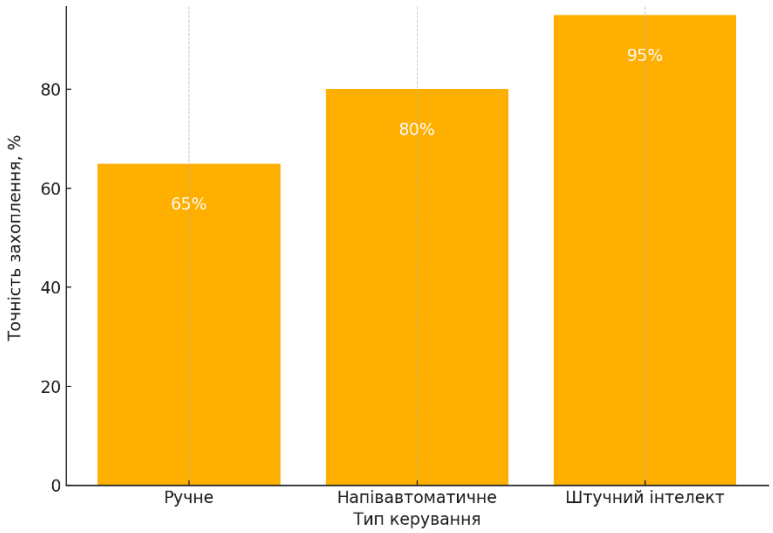

Рисунок 3.7 – Залежність точності захоплення від типу керування

Під час тестів було виявлено, що найменшу точність (у середньому 65 %) демонструє ручне керування, де велике значення має досвід оператора. Напівавтоматичне керування, яке частково покладається на дані сенсорів, показало середню точність захоплення – близько 80 %. Найвищу точність (до 95 %) забезпечила система з використанням ШІ, яка аналізує форму, розмір та опір об’єкта в режимі реального часу.

На п’ятому етапі фіксувалась кількість помилок або збоїв системи під час тестування (рисунок 3.8). Ручне керування продемонструвало найбільшу кількість помилок, зокрема через неточність позиціювання або надмірне стискання об'єктів. Напівавтоматичне керування значно зменшило кількість помилок. Найменшу кількість збоїв спостерігалося під час роботи системи з ШІ – лише поодинокі випадки при атипових умовах. Це свідчить про високу надійність та стабільність алгоритмів машинного навчання у практичних сценаріях.

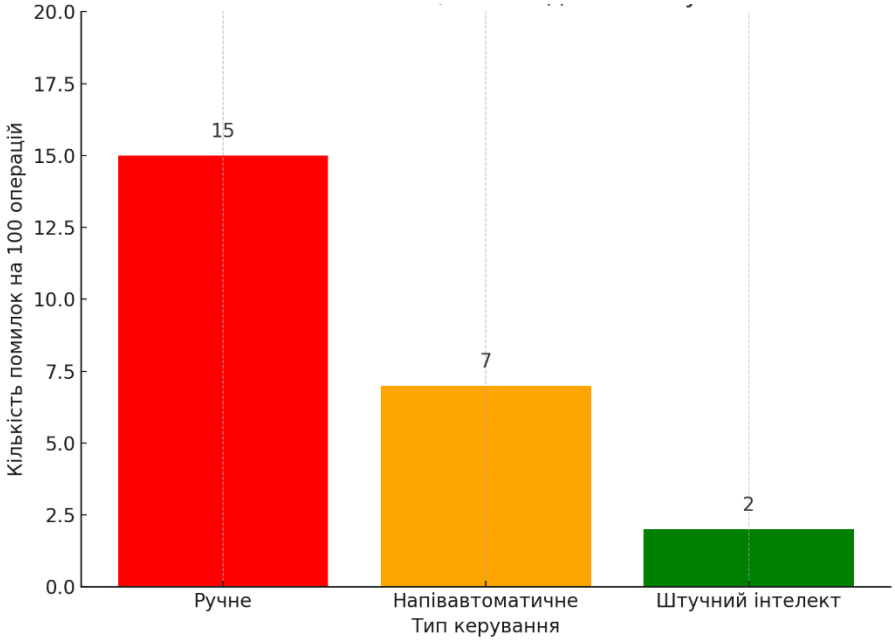

Рисунок 3.8 – Кількість помилок/збоїв під час тестування

Усі результати тестування засвідчили, що система здатна виконувати запрограмовані дії із передбачуваною поведінкою, реагувати на події у режимі

реального часу та демонструвати елементи адаптивності, необхідні для подальшого розвитку інтелектуальних функцій.

У результаті реалізації системи управління було досягнуто повну інтеграцію програмних та апаратних компонентів, реалізовано основну логіку дій маніпулятора, зворотний зв’язок через сенсор сили натискання та базовий інтелектуальний модуль ухвалення рішень. Проведене тестування підтвердило працездатність усіх елементів системи, стабільність взаємодії між рівнями керування, здатність до реагування на зміну умов у реальному часі та потенціал до подальшого розвитку. Отримані результати засвідчують, що розроблена система відповідає вимогам до сучасних роботизованих рішень та є придатною до використання в прикладних або експериментальних умовах.

ВИСНОВКИ

В ході виконання кваліфікаційногї роботи було здійснено проєктування архітектури системи управління роботизованим маніпулятором із чітким поділом функціональних обов’язків між низькорівневим контролером ESP32 та високорівневим модулем Raspberry Pi. Такий підхід дозволив ефективно розподілити навантаження між компонентами, забезпечити надійність взаємодії та створити гнучку основу для подальшої інтеграції інтелектуальних модулів.

Спроєктовано протокол обміну даними між модулями управління. Передбачено уніфікований текстовий формат команд і відповідей, обробку помилок, верифікацію команд та реалізацію механізмів таймауту. Завдяки цьому забезпечено стабільну, контрольовану та передбачувану комунікацію між рівнями системи.

Реалізовано базовий набір дій маніпулятора – «захоплення», «відпускання», «утримання» та «імітація хвата» – з можливістю динамічного налаштування параметрів. Дії реалізовано у вигляді програмних сценаріїв з контролем умов завершення, що забезпечує гнучкість і адаптивність поведінки системи.

Інтегровано зворотний зв’язок на основі FSR-датчика, що дозволяє отримувати інформацію про силу натискання під час взаємодії з об’єктами. Реалізовано обробку сенсорних даних у реальному часі з можливістю завершення дії або переходу у безпечний режим при перевищенні порогових значень.

Запропоновано та реалізовано програмні механізми безпечного завершення дій у разі втрати з’єднання, порушення протоколу або виявлення критичних ситуацій. Це забезпечує базовий рівень надійності та захищеності системи при реальному використанні.

Проведено тестування системи у лабораторних умовах. Отримано підтвердження стабільності роботи, відповідності поведінки маніпулятора заданим сценаріям, коректності взаємодії між модулями та адекватної реакції на

події. Отримані результати узагальнено у вигляді таблиць та графіків, що демонструють динаміку роботи в реальному часі.

Розроблено та інтегровано базовий інтелектуальний компонент, який забезпечує адаптацію поведінки маніпулятора залежно від контексту виконання дії. AI-компонент працює на основі простих евристичних правил та аналізу зворотного зв’язку, закладаючи основу для подальшої інтеграції методів машинного навчання, зокрема класифікації, прогнозування або навчання з підкріпленням.

На основі проведеної роботи можемо зробити висновок, що розроблена система управління є функціонально повною, адаптивною та масштабованою. Вона може бути використана як у навчальних цілях, так і для реалізації дослідницьких або прототипових проєктів з подальшою інтеграцією в повноцінні робототехнічні системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Soft is safe: human-robot interaction for soft robots. URL: https://shorturl.at/5uxDK (дата звернення: 01.05.2025).

2. Michigan State University. Powerful human-like hands create safer human-robotics interactions. Tech Xplore - Technology and Engineering news. URL: https://shorturl.at/kxr8U (дата звернення: 01.05.2025).

3. 01 introduction to jupyterlab and robotics basics - waveshare wiki. Waveshare Electronics. URL: https://shorturl.at/NwcnL (дата звернення: 01.05.2025).

4. Kalita B., Leonessa A., K. Dwivedy S. A review on the development of pneumatic artificial muscle actuators: force model and application. URL: https://shorturl.at/qLHjC (дата звернення: 01.05.2025).

5. Chabroux M. Robotics: The Era of AI and Autonomy RoboticsTomorrow. Robotics and Automation Stories, Videos, Articles, Interviews, Reviews & News RoboticsTomorrow. URL: https://shorturl.at/Y6ArW (дата звернення: 01.05.2025).

6. Hussain D., Rahman H., Ali M. Artificial intelligence and machine learning enhance robot decision-making adaptability and learning capabilities across various domains. Global Mainstream Journal. URL: https://tinyurl.com/y8nc5666 (дата звернення: 03.05.2025).

7. Generic Serial Communication Protocols: UART, SPI, I2C – RF-star. URL: https://tinyurl.com/29tdhjxx (дата звернення 03.05.2025)

8. Understanding Distributed Control Systems (DCS). URL: https://tinyurl.com/9k99wwyh (дата звернення 03.05.2025)

9. Zhang D., Lee D.-J. Intelligent robotics-a systematic review of emerging technologies and trends. URL: https://tinyurl.com/hayrhrz5 (дата звернення: 03.05.2025).

10. Deep reinforcement learning based proactive dynamic obstacle avoidance for safe human-robot collaboration. URL: https://tinyurl.com/mr3ymwa2 (дата звернення: 03.05.2025)

11. Arduino & ESP32 vs Raspberry Pi. URL: https://tinyurl.com/42fypaec (дата звернення: 03.05.2025).

12. Velasco A. Comparing microcontrollers: what brain should I go with?. URL: https://tinyurl.com/538s8k4t (дата звернення: 08.05.2025).

13. STM32 high performance microcontrollers (mcus) - stmicroelectronics. STMicroelectronics. URL: https://tinyurl.com/4kdb7pj6 (дата звернення: 08.05.2025).

14. Jetson nano. NVIDIA Developer. URL: https://tinyurl.com/yk7nnp6b (дата звернення: 08.05.2025).

15. Get started with the Arduino Cloud using the C++ programming language. URL: https://tinyurl.com/2v6jek7z (дата звернення: 08.05.2025)

16. Robot Operating System (ROS). URL: https://tinyurl.com/yfa5npbr (дата звернення: 12.05.2025)

17. ESP32 Servo Motor Web Server with Arduino IDE. URL: https://tinyurl.com/623h89zs (дата звернення 12.05.2025).

18. Ampheo. UART serial communication experiment based on raspberry pi 4B and STM32. URL: https://tinyurl.com/2mjj3jv2 (дата звернення: 12.05.2025)

19. Rana T., Islam S., Rahman A. Human-Centered sensor technologies for soft robotic grippers: a comprehensive review. URL: https://tinyurl.com/3tfz3pmr (дата звернення: 12.05.2025).

20. Design and Development of a Roaming Wireless Safety Emergency Stop URL: https://tinyurl.com/5p7vasu4 (дата звернення: 12.05.2025).