

Spis treści

Streszczenie	1
1.1 Cel pracy	2
1.2 Teza pracy.....	2
1.3 Zakres pracy.....	2
Wprowadzenie	3
Kinematyka robota	7
2.1 Pojęcia podstawowe	7
2.2 Kinematyka prosta	8
2.3 Kinematyka odwrotna	10
Algorytmy kompensacji błędów ruchu robota przegubowego	13
Opis robota przegubowego	16
4.1 Konstrukcja mechaniczna	16
4.2 Napęd.....	18
4.3 Elektronika robota	19
4.3.1 Zasilanie.....	20
4.3.2 Komunikacja-	20
4.4 Oprogramowanie.....	22
4.4.1 Struktura oprogramowania i użyte biblioteki.....	22
4.4.2 Komunikacja i obsługa komend	23
4.4.3 Opracowane aplikacje sterujące	25
Uruchomienie i testy	29
5.1 Weryfikacja obliczeń kinematyki	29
5.2 Eksperymentalna weryfikacja działania algorytmu kompensacji błędów	30
5.3 Pomiary zasilania układu	31
5.4 Porównanie parametrów zaprojektowanego robota z istniejącymi rozwiązaniami	33
Podsumowanie	34
Literatura	35

Streszczenie

Celem niniejszej pracy dyplomowej jest zaprojektowanie, budowa oraz analiza kinematyki robota przegubowego o czterech stopniach swobody (ang. *Degrees of Freedom*, skrótowo 4DOF), wyposażonego w przeguby obrotowe. W pracy przedstawiono kolejne etapy realizacji projektu. W rozdziale 2 zaprezentowano teoretyczne podstawy kinematyki bezpośredniej i odwrotnej wraz z wyprowadzeniem równań pozycji efektora w przestrzeni. Rozdział 3 zawiera przegląd istniejących metod kompensacji błędów pozycji, które mają na celu zwiększenie precyzji działania manipulatora. W rozdziale 4 szczegółowo omówiono konstrukcję mechaniczną ramienia, integrację układu elektronicznego oraz konfigurację zasilania z wykorzystaniem laboratoryjnego zasilacza stabilizowanego. Rozdział 5 poświęcono testom praktycznym i eksperymentalnej weryfikacji działania systemu. W ramach projektu opracowano dwa niezależne interfejsy sterowania – aplikację komputerową oraz mobilną, wykorzystującą komunikację Bluetooth. Implementacja obejmuje również funkcjonalność aktywacji algorytmu kompensacji błędów, który koryguje odchylenia w odwzorowaniu położenia końcówki robota względem zadanych współrzędnych. Dodatkowo, przeprowadzono porównanie funkcjonalności zaprojektowanego manipulatora z wybranymi komercyjnymi rozwiązaniami o czterech stopniach swobody.

The aim of this engineering thesis is to design, build, and analyze the kinematics of an articulated robot with four degrees of freedom (4DOF), equipped with rotary joints. The thesis presents the successive stages of the project implementation. Chapter 2 introduces the theoretical foundations of forward and inverse kinematics, along with the derivation of equations describing the end-effector's position in space. Chapter 3 provides an overview of existing position error compensation methods aimed at improving the precision of the manipulator's operation. Chapter 4 discusses in detail the mechanical construction of the robotic arm, integration of the electronic system, and the power supply configuration using a stabilized laboratory power supply. Chapter 5 is dedicated to practical testing and experimental validation of the system's performance. As part of the project, two independent control interfaces were developed – a desktop application and a mobile application using Bluetooth communication. The implementation also includes functionality for activating the error compensation algorithm, which corrects deviations in the end-effector's position relative to the target coordinates. Additionally, a comparison of the designed manipulator's functionality with selected commercial 4DOF solutions was conducted.

Rozdział 1

1.1 Cel pracy

Celem pracy jest zaprojektowanie i wykonanie elektronicznego kontrolera dla manipulatora sferycznego o czterech stopniach swobody, umożliwiającego precyzyjne sterowanie ramieniem robota oraz zdalną komunikację poprzez interfejs Bluetooth. Dodatkowym celem jest opracowanie i implementacja algorytmów kompensacji błędów położenia wynikających z niedoskonałości konstrukcyjnych, kalibracji oraz działania sił zewnętrznych.

1.2 Teza pracy

Zaprojektowanie i implementacja algorytmów kinematyki oraz kompensacji błędów pozycjonowania w robocie 4DOF umożliwia zwiększenie dokładności jego działania, przy jednoczesnym zachowaniu niskich kosztów prototypowania dzięki wykorzystaniu otwartych platform sprzętowych i programistycznych.

1.3 Zakres pracy

Zakres niniejszej pracy dyplomowej obejmuje opracowanie i realizację projektu ramienia robotycznego o czterech stopniach swobody, począwszy od analizy teoretycznej, poprzez projektowanie mechaniczne i elektroniczne, aż po implementację algorytmów sterowania oraz testy funkcjonalne.

W szczególności opracowano i zrealizowano następujące elementy:

- analizę kinematyki zaprojektowanego robota sferycznego,
- dobór elementów mechanicznych i elektronicznych oraz zaprojektowanie układu sterowania,
- modyfikację zestawu ramienia robotycznego do wersji 4 DOF,
- implementację algorytmów sterujących i kompensacyjnych w środowisku Arduino,
- stworzenie aplikacji mobilnej oraz desktopowego interfejsu użytkownika do zdalnego sterowania manipulatorem,
- przeprowadzenie testów oraz eksperymentalną ocenę dokładności i skuteczności działania systemu.

Wprowadzenie

Ramię robotyczne to kluczowy element w automatyzacji, zdolny do wykonywania precyzyjnych i powtarzalnych zadań w różnych branżach, takich jak produkcja, pakowanie czy zastosowania medyczne. Zdolność ramienia do wykonywania złożonych ruchów zależy przede wszystkim od liczby stopni swobody, które określają liczbę niezależnych ruchów stawów, umożliwiających pozycjonowanie i orientację efektora końcowego. Chociaż roboty przemysłowe zazwyczaj mają sześć lub więcej stopni swobody, by naśladować zręczność ludzkiego ramienia, ramiona robotyczne o czterech stopniach swobody stanowią praktyczny kompromis między złożonością mechaniczną a funkcjonalnością.

Typowe ramię robotyczne 4 DOF zawiera stawy umożliwiające obrót podstawy, ruchy w stawie barkowym i łokciowym (w płaszczyźnie pionowej) oraz rotację nadgarstka, co pozwala na poruszanie się w przestrzeni trójwymiarowej z dodatkową rotacją efektora. Taka konfiguracja jest wystarczająca do wielu zadań, takich jak sortowanie i przenoszenie obiektów, obsługa maszyn czy układanie towarów na paletach, gdzie pełna kontrola orientacji w przestrzeni nie jest konieczna. Zredukowana złożoność manipulatorów 4 DOF upraszcza projektowanie i sterowanie oraz pozwala na ekonomiczne wdrożenie, dzięki czemu są one popularne w edukacji i małych oraz średnich przedsiębiorstwach.

Podstawą działania każdego ramienia robotycznego jest analiza kinematyczna, obejmująca rozwiązywanie problemów kinematyki prostej i odwrotnej. Kinematyka prosta pozwala obliczyć pozycję i orientację efektora na podstawie znanych kątów stawów, natomiast kinematyka odwrotna określa, jakie kąty stawów są potrzebne, aby osiągnąć zadaną pozycję efektora. Choć obliczenia kinematyki prostej nie należą do szczególnie złożonych, kinematyka odwrotna często wiąże się z koniecznością rozwiązywania nieliniowych układów równań oraz analizą wielu możliwych rozwiązań, co stanowi istotne wyzwanie w projektowaniu systemów sterowania. Dokładne modelowanie kinematyczne jest kluczowe dla planowania trajektorii ruchu, zapewniając płynność i precyzję działania manipulatora.

Najnowsze osiągnięcia w dziedzinie mikrokontrolerów, takich jak platforma Arduino, umożliwiły realizację niskokosztowych systemów sterowania dla ramion 4 DOF, integrujących serwomechanizmy i sprzężenie zwrotne z czujników, co pozwala na niezawodne sterowanie ruchem. Dodatkowo, rozwój technologii komunikacyjnych, takich jak interfejsy USB i Bluetooth, umożliwia elastyczne sterowanie urządzeniem z poziomu aplikacji komputerowych i mobilnych. Innowacje te wspierają zarówno ręczny, jak i automatyczny tryb pracy, zwiększając zakres zastosowań ramion 4 DOF w badaniach i praktyce.

Niniejsza praca dyplomowa koncentruje się na projektowaniu, modelowaniu i sterowaniu ramieniem robotycznym 4 DOF, ze szczególnym uwzględnieniem implementacji algorytmów kinematyki prostej i odwrotnej oraz metody kompensacji błędów pozycjonowania w celu zwiększenia dokładności. Projekt wykorzystuje otwarte narzędzia sprzętowe i programowe, mając na celu demonstrację skutecznego, nisko kosztowego prototypowania oraz weryfikację wpływu algorytmów kompensacji na precyzję pozycjonowania.

Przykłady popularnych robotów przegubowych 4-DOF:

1. RoArm-M2-S [\[10\]](#)

Robot czteroosiowy przeznaczony do zastosowań edukacyjnych, badawczych oraz lekkiej automatyzacji. Dzięki lekkiej konstrukcji z włókna węglowego i aluminium, zapewnia wysoką precyzję oraz łatwość integracji z różnymi systemami sterowania.



Rysunek 1.1: Robot RoArm-M2-S.

Tabela 1.1. Podstawowe parametry robota RoArm-M2-S.

Parametr	Wartość
Liczba osi	4
Maksymalny udźwig	0,5 kg
Zasięg ramienia	1 m
Powtarzalność	$\pm 0,1$ mm
Waga	0,83 kg
Kontroler	ESP32, ROS2

2. 4-DOF Multifunction Robotic Arm – IADIY [\[11\]](#)

Kompaktowe ramię robotyczne do edukacji i hobbystycznych projektów automatyki. Wyposażone w lekką konstrukcję i interfejs USB/Bluetooth, pozwala na szybkie wdrażanie projektów sterowania ruchem w środowiskach takich jak Arduino i Python. Obsługuje funkcje jak chwytak, kamera lub ssawka próżniowa.



Rysunek 1.2: Robot 4-DOF Multifunction Robotic Arm – IADIY.

Tabela 1.2. Podstawowe parametry robota 4-DOF Multifunction Robotic Arm – IADIY.

Parametr	Wartość
Liczba osi	4
Maksymalny udźwig	ok. 0,2–0,3 kg
Zasięg ramienia	ok. 35–50 cm
Powtarzalność	$\sim \pm 0,5$ mm (szac.)
Waga	ok. 1,2 kg
Kontroler	Arduino

3. Igus® robolink® DP – 4DOF [12]

Modularny przemysłowy manipulator z tworzywa sztucznego o wysokiej trwałości, zaprojektowany do lekkich zastosowań automatyzacyjnych. Umożliwia pracę z ROS i darmowym oprogramowaniem igus® Robot Control. Dostępny również w wersji IP44 (odporny na zachlapania).



Rysunek 1.3: Robot Igus® robolink® DP – 4DOF.

Tabela 1.3. Podstawowe parametry robota Igus® robolink® DP – 4DOF.

Parametr	Wartość
Liczba osi	4
Maksymalny udźwig	2–3 kg
Zasięg ramienia	790 mm
Powtarzalność	$\pm 0,5$ mm
Waga	~5–6 kg (zależnie od wersji)
Kontroler	igus® Robot Control, ROS

Przedstawione roboty charakteryzują się specyficznym układem osi przegubów, umożliwiającym realizację szerokiego zakresu pozycji i orientacji efektora w przestrzeni roboczej. Dzięki temu mogą one z powodzeniem wykonywać różnorodne zadania manipulacyjne. W dalszej części pracy przedstawiona zostanie tabela zawierająca jego parametry techniczne. Umożliwi to bezpośrednie porównanie z opisanymi wcześniej modelami i ocenę jego funkcjonalności w kontekście dostępnych na rynku rozwiązań.

Rozdział 2

Kinematyka robota

Kinematyka robota przegubowego stanowi podstawowy etap projektowania systemu sterowania, który umożliwia określenie położenia, orientacji oraz trajektorii końcówki robota względem układu bazowego [1]. Opisuje ona ruch poszczególnych członów manipulatora bez uwzględniania działających sił i momentów, koncentrując się wyłącznie na relacjach geometrycznych wynikających z zadanych kątów przegubowych.

W przypadku manipulatorów przegubowych szczególne znaczenie mają dwa główne zagadnienia: kinematyka prosta (forward kinematics) oraz kinematyka odwrotna (inverse kinematics). Ich prawidłowe opracowanie pozwala na precyzyjne wyznaczanie położenia efektora w przestrzeni roboczej oraz umożliwia sterowanie ruchem robota w sposób zgodny z wymaganiami aplikacyjnymi.

2.1 Pojęcia podstawowe

Manipulator przegubowy składa się z szeregu członów, które są ze sobą połączone za pomocą przegubów obrotowych. Każdy przegub umożliwia realizację jednego stopnia swobody, a pełna konfiguracja robota określana jest przez liczbę takich przegubów.

W przypadku zaprojektowanego robota sferycznego o czterech stopniach swobody zastosowano wyłącznie przeguby obrotowe (typu R), co umożliwia realizację ruchu efektora w przestrzeni trójwymiarowej.

Podstawowym narzędziem wykorzystywanym do analizy ruchu manipulatora jest jednorodna macierz transformacji, która umożliwia opis pozycji i orientacji danego ogniwa względem układu bazowego. W niniejszej pracy zastosowano powszechnie używaną metodę Denavita-Hartenberga (D-H), która definiuje geometrię manipulatora za pomocą czterech parametrów opisujących relacje przestrzenne między kolejnymi przegubami [2].

2.2 Kinematyka prosta

Celem analizy kinematyki prostej jest wyznaczenie położenia i orientacji efektora końcowego na podstawie znanych wartości kątów poszczególnych przegubów robota. W przypadku manipulatora przegubowego polega to na przekształceniu zestawu wartości wejściowych (kątów obrotu dla każdego przegubu) w jedną macierz transformacji, która opisuje całkowite przemieszczenie względem układu bazowego.

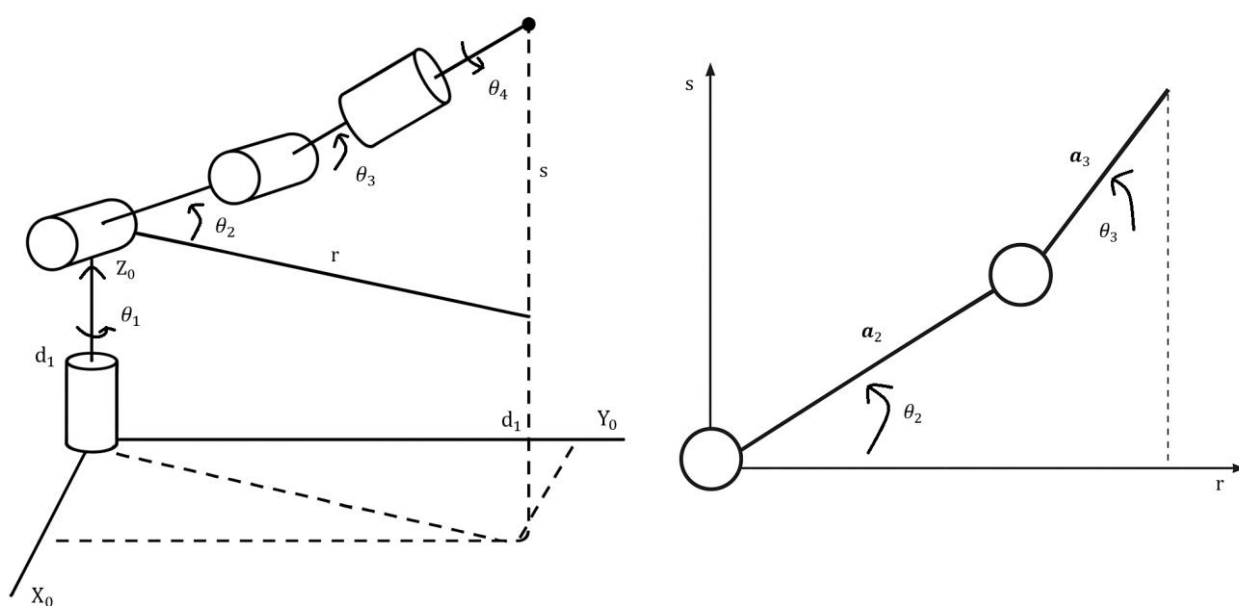
Metoda Denavita-Hartenberga polega na przypisaniu lokalnych układów współrzędnych do każdego członu manipulatora i opisaniu przekształceń między nimi za pomocą czterech parametrów:

- θ_i — kąt obrotu wokół osi z_{i-1} ,
- d_i — przesunięcie wzdłuż osi z_{i-1} ,
- a_i — długość członu wzdłuż osi x_i ,
- α_i — kąt obrotu wokół osi x_i , między osiami z_{i-1} i z_i .

Układy współrzędnych w metodzie D-H

W celu poprawnego przypisania parametrów D-H należy każdemu członowi przyporządkować lokalny układ współrzędnych zgodnie z następującymi zasadami:

- Oś z_i jest osią przegubu i wskazuje kierunek ruchu (dla przegubów obrotowych — oś obrotu),
- Oś x_i jest wspólną normalną pomiędzy osiami z_{i-1} i z_i — jest prostopadła do obu,
- Początek układu znajduje się w punkcie przecięcia osi z_i i x_i ,
- Oś y_i dopełnia układ do układu prawoskrętnego.



Rysunek 2.1: Schemat układów współrzędnych i parametrów D-H dla 4-DOF manipulatora. [3]

r - odpowiada poziomej odległości (wzdłuż osi X) pomiędzy podstawą manipulatora a końcowym efektem działania,

s - oznacza pionową odległość (wzdłuż osi Z) końcówki ramienia od podstawy.

Parametry te służą do obliczeń położenia końcówki robota oraz wyznaczania wymaganych kątów obrotu dla poszczególnych przegubów.

Macierz przekształcenia D-H

Transformację między kolejnymi układami współrzędnych opisuje się przy pomocy macierzy jednorodnej [4]:

$${}^{i-1}T_i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & \alpha_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & \alpha_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

Końcowe przekształcenie pozycji efektora względem układu bazowego robota wyraża się jako iloczyn wszystkich macierzy:

$$T_0^n = T_0^1 \times T_1^2 \times T_2^3 \times \dots \times T_{n-1}^n \quad (2.2)$$

W przypadku robota o czterech stopniach swobody powyższe równanie przyjmuje postać:

$$T_0^4 = T_0^1 \times T_1^2 \times T_2^3 \times T_3^4$$

Zaprojektowany manipulator o czterech stopniach swobody posiada wyłącznie przeguby obrotowe (typ R), co klasyfikuje go jako robota przegubowego o konfiguracji sferycznej. Konstrukcja składa się z:

- przegubu obrotowego podstawy (q_1), odpowiedzialnego za zmianę kierunku ramienia w płaszczyźnie poziomej (obrót wokół osi Z),
- dwóch przegubów zginających (q_2, q_3), realizujących ruchy w płaszczyźnie pionowej XZ,
- przegubu końcowego (q_4), który umożliwia obrót chwytaka wokół jego własnej osi.

Tabela 2.1: Parametry Denavita-Hartenberga. [3]

i	θ_i (zmienna)	d_i [mm]	a_i [mm]	α_i
1	θ_1	d_1	0	90°
2	θ_2	0	L_2	0
3	θ_3	0	L_3	0

Rzeczywiste parametry długości zaprojektowanego manipulatora są równe: $d_1 = 96$ mm, $L_2 = 66$ mm, $L_3 = 178$ mm.

2.3 Kinematyka odwrotna

Kinematyka odwrotna polega na wyznaczeniu wartości kątów przegubów robota, które pozwolą na osiągnięcie zadanego położenia końcówki manipulatora (efektora) w przestrzeni [1]. W przeciwieństwie do kinematyki prostej, w której na podstawie zadanych kątów obrotu wyznacza się pozycję końcówki, w kinematyce odwrotnej użytkownik podaje pożądane współrzędne punktu docelowego, a algorytm oblicza odpowiednie wartości θ_i dla każdego przegubu.

Rozwiązanie kinematyki odwrotnej jest znacznie bardziej złożone matematycznie i często wieloznaczne — dla jednej pozycji końcówki może istnieć więcej niż jedno możliwe ustawienie przegubów (lub brak rozwiązania, jeśli punkt jest poza przestrzenią roboczą robota).

Upraszczające założenia w modelu 4 DOF

W przypadku zaprojektowanego robota o czterech stopniach swobody przyjęto uproszczony model kinematyczny, w którym orientacja efektora nie jest uwzględniana (przegub 4 odpowiada jedynie za obrót chwytaka wokół własnej osi). Oznacza to, że kinematyka odwrotna ogranicza się do wyznaczenia kątów $\theta_1, \theta_2, \theta_3$ potrzebnych do ustawienia końcówki w przestrzeni roboczej. [4]

Procedura obliczeniowa

Zakładając, że znane są współrzędne punktu docelowego (x,y,z), można przyjąć następujące podejście:

1. Wyznaczenie kąta θ_1 – obrót podstawy:

$$\theta_1 = \arctan2(y, x) \quad (2.3)$$

2. Obliczenie rzutowanej odległości w płaszczyźnie XY oraz przesunięcia w osi Z:

$$r = \sqrt{x^2 + y^2} \quad z' = z - d_{pdstawy} \quad (2.4, 2.5)$$

3. Zastosowanie twierdzenia cosinusów do wyznaczenia kąta zgięcia „łokcia”:

$$\cos \theta_3 = \frac{r^2 + z'^2 - L_2^2 - L_3^2}{2L_2L_3} \quad (2.6)$$

4. Obliczenie kąta θ_2 z zależności geometrycznych:

$$\theta_2 = \arctan2(z', r) - \arctan2(L_3 \sin \theta_3, L_2 + L_3 \cos \theta_3) \quad (2.7)$$

5. Kąt θ_4 — dowolny lub narzucony w zależności od potrzeb.

Przykład obliczeń kinematyki odwrotnej:

Dla zadanych współrzędnych efektora w przestrzeni roboczej:

$x=28,58$ mm, $y=16,50$ mm, $z=331,16$ mm

oraz znanych długości podstawy i ramion:

$L_1=96$ mm, $L_2=66$ mm, $L_3=178$ mm

przeprowadzono obliczenia odwrotne w celu wyznaczenia wartości kątów przegubów $\theta_1, \theta_2, \theta_3$.

1. Wyznaczenie kąta obrotu podstawy:

$$\theta_1 = \arctan2(y, x) = \arctan2(16,50, 28,58) \approx 30,00^\circ$$

2. Obliczenie rzutowanej odległości w płaszczyźnie XY oraz przesunięcia w osi Z:

$$r = \sqrt{x^2 + y^2} = \sqrt{28,58^2 + 16,50^2} \approx 33,00 \text{ mm}$$

$$z' = z - d_{pdstawy} = 331,16 - 96 = 235,16 \text{ mm}$$

3. Obliczenie kąta zgięcia łokcia na podstawie twierdzenia cosinusów:

$$\cos \theta_3 = \frac{r^2 + z'^2 - L_2^2 - L_3^2}{2L_2L_3} = \cos \theta_3 = \frac{33^2 + 235,16^2 - 66^2 - 178^2}{2 \times 66 \times 178} \approx 0,867$$

$$\theta_3 = \arccos(0,867) \approx 29,99^\circ$$

4. Wyznaczenie kąta ramienia θ_2 :

$$\theta_2 = \arctan2(z', r) - \arctan2(L_3 \sin \theta_3, L_2 + L_3 \cos \theta_3)$$

$$\arctan2(235,16, 33) \approx 82,00^\circ$$

$$\arctan2(178 \times \sin(29,99^\circ), 66 + 178 \times \cos(29,99^\circ)) \approx 22,00^\circ$$

$$\theta_2 = 82,00^\circ - 22,00^\circ = 60,00^\circ$$

Wyniki końcowe: $\theta_1 = 30,00^\circ$, $\theta_2 = 60,00^\circ$, $\theta_3 = 29,99^\circ$.

Weryfikacja kinematyki odwrotnej za pomocą kinematyki prostej:

1. Konwersja kątów na radiany:

$$\theta_1 = 30,00^\circ = 0,5236 \text{ rad}, \theta_2 = 60,00^\circ = 1,0472 \text{ rad}, \theta_3 = 29,99^\circ = 0,5232 \text{ rad}.$$

2. Współrzędne x,y,z:

$$x = (L_2 \times \cos \theta_2 + L_3 \times \cos(\theta_2 + \theta_3)) \times \cos \theta_1$$

$$x = (66 \times \cos(1,0472) + 178 \times \cos(1,5704)) \times \cos(0,5236) \approx 28,58 \text{ mm}$$

$$y = (L_2 \times \cos \theta_2 + L_3 \times \cos(\theta_2 + \theta_3)) \times \sin \theta_1$$

$$y = (66 \times \cos(1,0472) + 178 \times \cos(1,5704)) \times \sin(0,5236) = 16,50 \text{ mm}$$

$$z = L_1 + L_2 \times \sin \theta_2 + L_3 \times \sin(\theta_2 + \theta_3)$$

$$z = 96 + 66 \times \sin(1,0472) + 178 \times \sin(1,5704) = 331,16 \text{ mm}$$

Wyniki końcowe (FK): $x \approx 28,58 \text{ mm}$, $y = 16,50 \text{ mm}$, $z = 331,16 \text{ mm}$

Współrzędne obliczone na podstawie wartości kątów przegubów θ_1 , θ_2 , θ_3 . całkowicie pokrywają się z zadaniem położeniem efektora. Oznacza to, że opracowany algorytm kinematyki odwrotnej działa poprawnie i jest zgodny z kinematyką prostą.

Rozdział 3

Algorytmy kompensacji błędów ruchu robota przegubowego

Roboty przegubowe, wykorzystywane powszechnie w przemyśle i zastosowaniach badawczych, charakteryzują się dużą elastycznością ruchu oraz szerokim zakresem zastosowań. Jednak ze względu na niedokładności konstrukcyjne, deformacje mechaniczne, błędy montażowe oraz ograniczenia algorytmów sterowania, rzeczywista trajektoria ruchu końcówki manipulatora często odbiega od oczekiwanej. W celu minimalizacji tych odchyleń stosuje się algorytmy kompensacji błędów ruchu [6]

Klasyfikacja metod kompensacji błędów

W literaturze i praktyce inżynierskiej wyróżnia się kilka głównych grup metod kompensacji błędów:

- **Kompensacja modelowa** – polega na korekcji parametrów modelu kinematycznego (np. długości ramion) na podstawie kalibracji.
- **Kompensacja oparta na danych** – wykorzystuje dane eksperymentalne w formie tablicy korekt (Kompensacja tablicowa ang. Look-Up Table – LUT) lub interpolacji. [5]
- **Regresja błędów** – aproksymacja błędów funkcjami matematycznymi (np. regresja liniowa).
- **Kompensacja ze sprzężeniem zwrotnym** – dynamiczne korygowanie pozycji na podstawie danych z czujników (np. kamery, enkodery, tensometry).
- **Metody adaptacyjne i uczące się** – zaawansowane podejścia oparte na sztucznej inteligencji, uczące się błędów w czasie rzeczywistym.

Przegląd metod kompensacji błędów:

1. Kompensacja modelowa (korekta parametrów kinematycznych)

Jedną z najprostszych metod jest korekta długości członów ramienia w modelu kinematycznym, wynikająca np. z niedokładnego pomiaru lub zużycia mechanicznego. Na etapie kalibracji wyznaczane są współczynniki korekcyjne a , b :

$$\begin{aligned} L1_corr &= L1 * a; \\ L2_corr &= L2 * b; \end{aligned} \tag{3.1}$$

Skorygowane wartości są następnie wykorzystywane w obliczeniach kinematyki odwrotnej, co pozwala na lepsze dopasowanie modelu matematycznego do rzeczywistego robota.

2. Kompensacja tablicowa

Metoda **Look-Up Table (LUT)** polega na utworzeniu siatki punktów kalibracyjnych w przestrzeni roboczej manipulatora, w której dla każdego punktu określana jest różnica pomiędzy jego rzeczywistą a oczekiwaną pozycją końcówki robota (efektora) [5]. Powstała w ten sposób tablica zawiera **błędy korekcyjne** $\Delta x(x,y), \Delta y(x,y)$, które zapisywane są np. w pamięci EEPROM mikrokontrolera i wykorzystywane do poprawiania wyników obliczeń kinematyki odwrotnej:

$$x_{poprawione} = x + dx(x,y), y_{poprawione} = y + dy(x,y) \quad (3.2)$$

Interpolacja pomiędzy punktami może być liniowa, dwuliniowa lub oparta na funkcjach bazowych. Dzięki temu możliwe jest częściowe wyeliminowanie systematycznych błędów pozycjonowania wynikających np. z elastyczności ramienia, luzów w przekładniach.

Przykład obliczeń korekcyjnych:

Założono, że oczekiwaną pozycją końcówki robota (efektora): $x = 180 \text{ mm}$, $y = 130 \text{ mm}$

A rzeczywista pozycją końcówki robota (efektora): $x = 182,4 \text{ mm}$, $y = 128,4 \text{ mm}$

Z tablicy LUT dla punktu $(x,y)=(180,130)$ odczytano błędy pozycjonowania:

$$dx(180,130) = -2.4 \text{ mm}, dy(180,130) = 1.6 \text{ mm}$$

Zastosowanie korekcji:

$$x_{poprawione} = 182,4 + (-2.4) = 180 \text{ mm}, y_{poprawione} = 128,4 + 1.6 = 130 \text{ mm}$$

3. Regresja błędów (korekta matematyczna)

W przypadku gdy błędy pozycjonowania zmieniają się systematycznie w przestrzeni (np. rosną wraz z odległością od środka układu), można je opisać funkcją matematyczną. Przykładowo, dla regresji liniowej:

$$dx = a_1x + a_2y + a_3, dy = b_1x + b_2y + b_3 \quad (3.3)$$

Współczynniki a_i, b_i mogą być wyznaczane na podstawie danych pomiarowych np. w programie Excel, a następnie zaimplementowane w kodzie Arduino jako korekty pozycji.

Przykład obliczeń dla korekty regresyjnej:

Przykładowe współczynniki (np. wyznaczone z pomiarów i regresji liniowej w Excelu):

$$a_1 = 0.01, a_2 = -0.005, a_3 = -1.2$$

$$b_1 = 0.002, b_2 = -0.008, b_3 = 0.6$$

Dla punktu oczekiwanego $x=150\text{ mm}$, $y=120\text{ mm}$ - obliczenie korekt:

$$dx = 0.01 \times 150 + (-0.005) \times 120 + (-1.2) = -0.3\text{ mm}$$

$$dy = 0.002 \times 150 + (-0.008) \times 120 + 0.6 = 1.86\text{ mm}$$

Zastosowanie korekcji $(x,y)=(150.3, 118,14)$:

$$x_{poprawione} = 150.3 + (-0.3) = 150\text{ mm}, y_{poprawione} = 118,14 + 1.86 = 120\text{ mm}$$

4. Kompensacja z pętlą sprzężenia zwrotnego

Zaawansowana metoda opierająca się na rzeczywistym pomiarze pozycji efektora w czasie rzeczywistym. Wymaga użycia czujników takich jak:

- kamery z analizą obrazu (np. OpenCV),
- czujniki tensometryczne wykrywające kontakt z obiektem,
- dodatkowe enkodery na efektorze.

Dane z czujników są wykorzystywane do dynamicznej korekty pozycji poprzez porównanie zadanej i aktualnej pozycji końcówki.

5. Algorytmy adaptacyjne / uczące się (np. sieci neuronowe)

Metody te wykorzystują dane z wielu cykli pracy robota, ucząc się zależności pomiędzy pozycją a występującym błędem. Dzięki temu mogą korygować również błędy nieliniowe i zmienne w czasie. Wymagają dużej mocy obliczeniowej (np. PC, GPU), dlatego stosowane są głównie w przemyśle lub badaniach naukowych.

Tabela 3.1: Porównanie metod.

Metoda	Implementacja	Wymagania sprzętowe	Typ błędów	Zastosowanie
Kompensacja modelowa	Bardzo łatwa	Brak	globalne, systemowe	Prototypy, szybkie testy
LUT (tablica korekt)	Łatwa	EEPROM / RAM	lokalne, nieliniowe	Kalibracja + Arduino
Regresja matematyczna	Średnia	Brak	systematyczne	Kalibracja w Excelu
Sprzężenie zwrotne	Trudna	Czujniki	dynamiczne	Przemysł, aplikacje z kamerami
Algorytmy uczące się	Bardzo trudna	PC / GPU	nieliniowe, zmienne	Przemysł, roboty badawcze

Rozdział 4

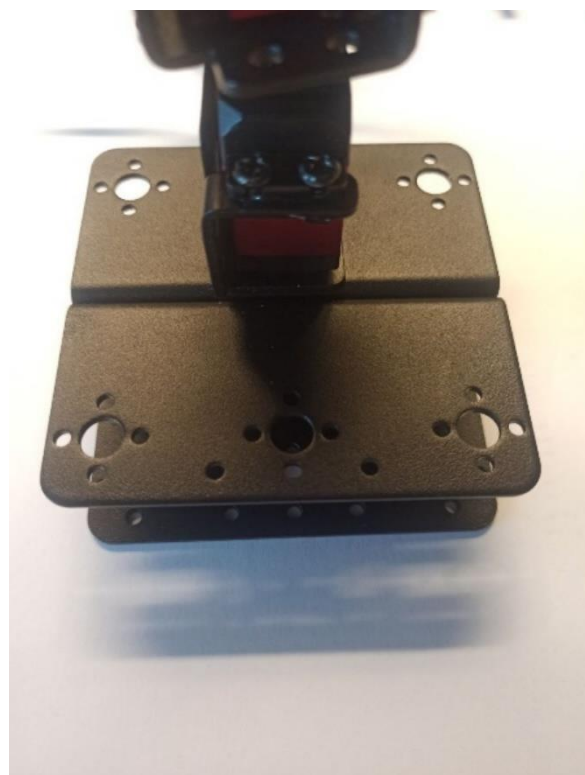
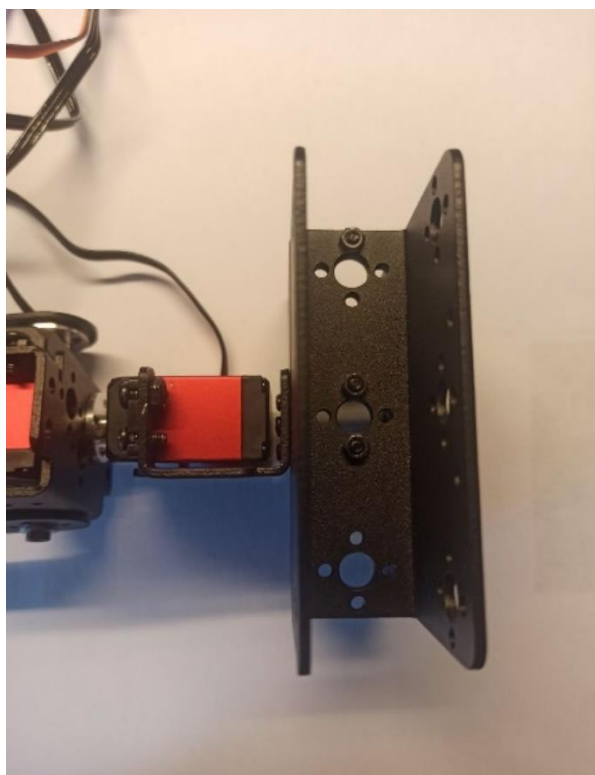
Opis robota przegubowego

4.1 Konstrukcja mechaniczna

Zaprojektowany robot przegubowy posiada cztery stopnie swobody (4DOF) i został wykonany w oparciu o zestaw metalowych elementów ramienia dostępnych komercyjnie. Manipulator składa się z podstawy, ramienia głównego, przedramienia oraz chwytaka. Ruch odbywa się wyłącznie za pomocą przegubów obrotowych (typ R), co klasyfikuje konstrukcję jako robota przegubowego o konfiguracji sferycznej.

Główne elementy robota przedstawiono na poniższych zdjęciach:

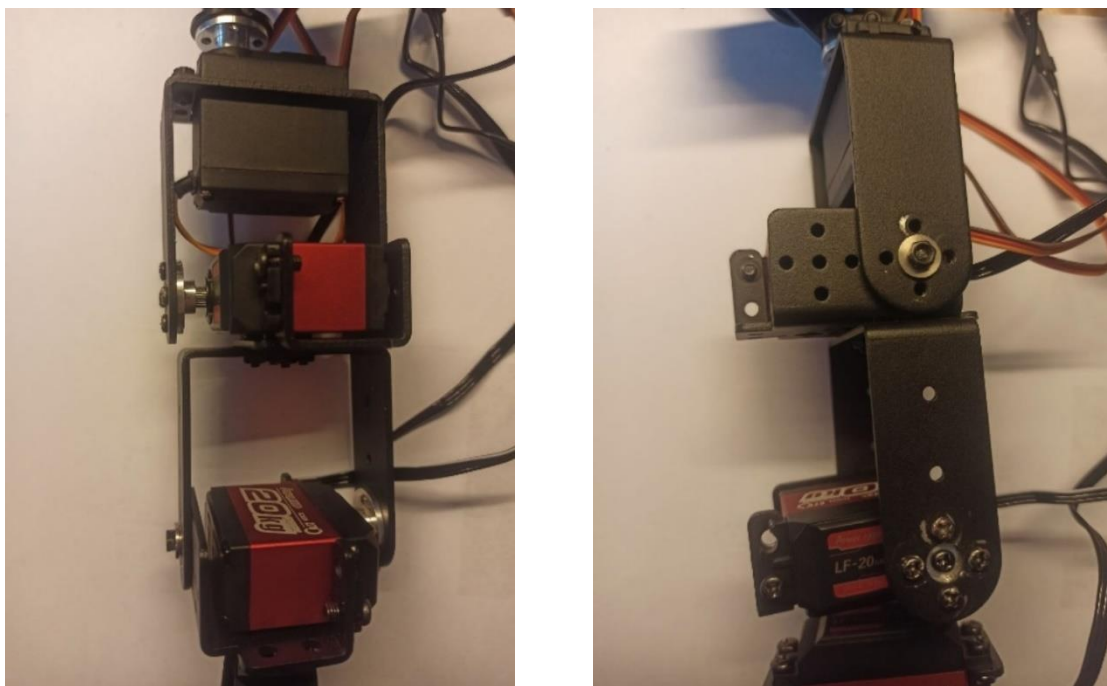
Podstawa q1 (rysunek 4.1) jest wykonana z połączonych płaskich płyt aluminiowych. Na podstawie zamontowany jest pierwszy serwomechanizm PowerHD LF-20MG, który umożliwia obrót całego ramienia wokół osi pionowej.



Rysunek 4.1: Zdjęcia podstawy robota.

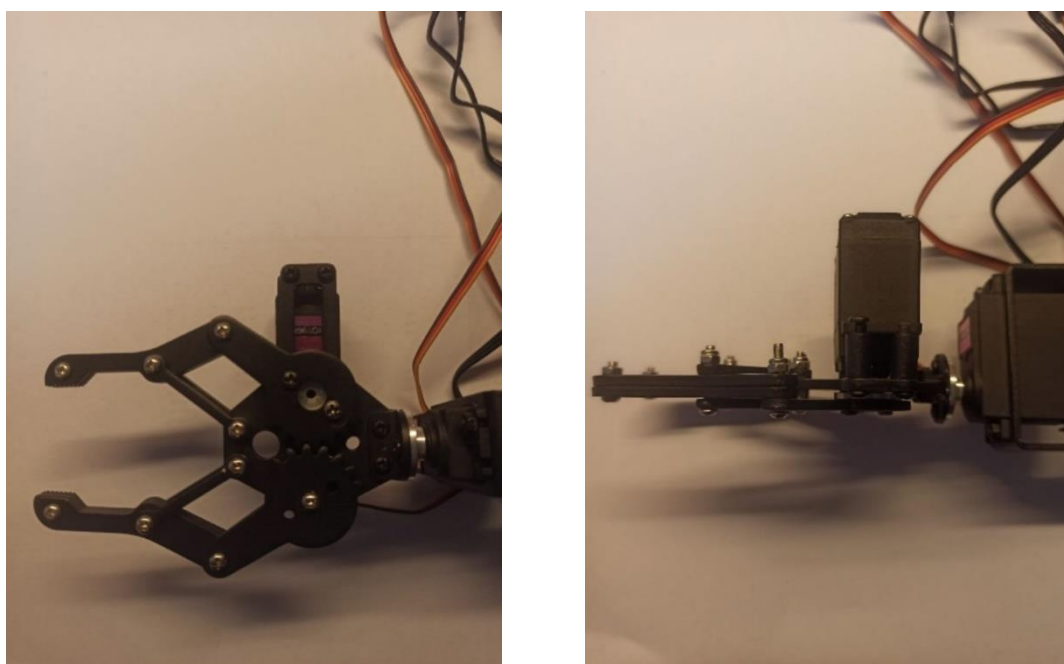
Ramię główne q_2 (rysunek 4.2) jest zamocowane do podstawy i obracane przez drugi serwomechanizm LF-20MG, ustawiony w orientacji pionowej. Konstrukcja opiera się na prostokątnych ramionach z otworami montażowymi. Na zdjęciach widoczne są także mocowania i śruby łączące elementy nośne.

Przedramię z chwytakiem łączy się z ramieniem głównym za pomocą przegubu łokciowego q_3 (rysunek 4.2), napędzanego kolejnym serwomechanizmem LF-20MG.



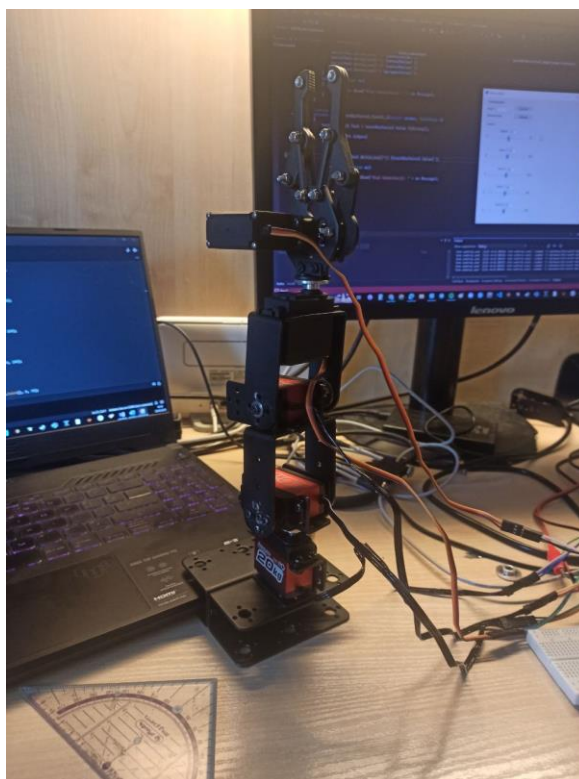
Rysunek 4.2: Widok ramienia głównego i przegubu łokciowego z zamontowanymi serwomechanizmami.

Chwytnak (rysunek 4.3) stanowi zakończenie robota i jest zamontowany na osi przegubu q_4 . Obrót realizowany jest przez serwomechanizm MG996R, natomiast sam chwyt odbywa się dzięki mechanizmowi z dwiema szczękami o ograniczonym kącie ruchu.



Rysunek 4.3: Chwytnak robota przegubowego.

Każdy przegub posiada niezależne sterowanie przy pomocy sygnałów PWM. Do napędu zastosowano trzy serwa LF-20MG oraz dwa MG996R. Dzięki metalowej konstrukcji oraz wewnętrznym przekładniom redukcyjnym serwomechanizmów, ramię jest w stanie przenosić lekkie przedmioty mimo niewielkiej mocy napędów.



Rysunek 4.4: Zdjęcie konstrukcji robota.

4.2 Napęd

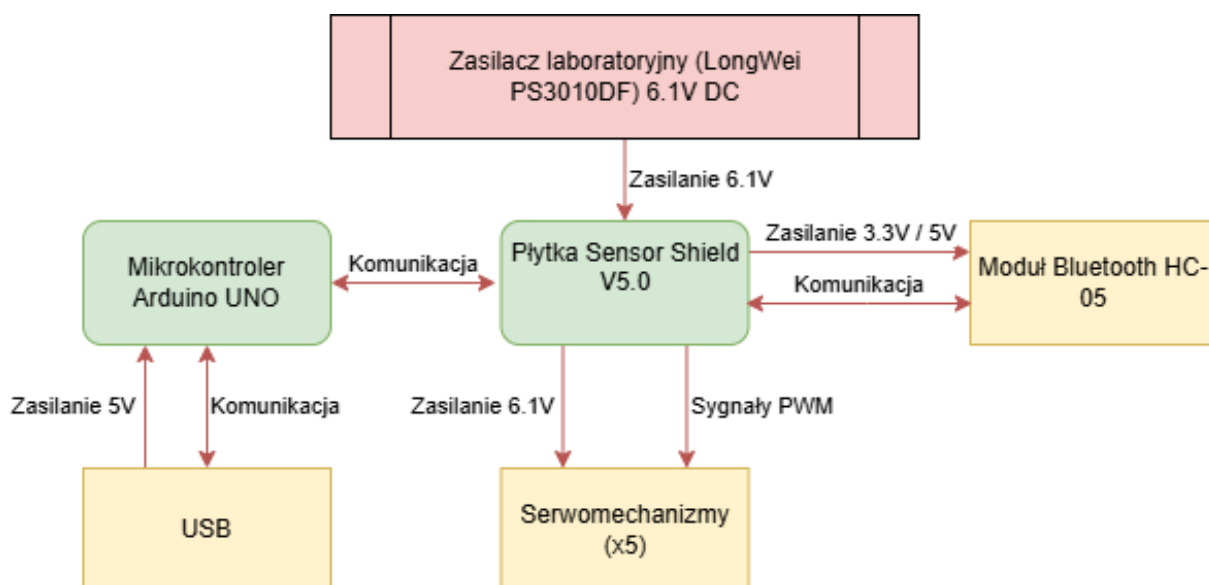
Pierwsza wersja konstrukcji robota przegubowego została wyposażona wyłącznie w serwomechanizmy MG996R, ze względu na ich szeroką dostępność i niski koszt. Jednak w trakcie testów wykazano, że serwa te nie zapewniają wystarczającego momentu obrotowego w przegubach najbardziej obciążonych – przede wszystkim w podstawie i przegubie ramienia. Podczas prób występowały spadki wydajności, zatrzymania w połowie ruchu oraz charakterystyczne „bzyczenie” wynikające z przeciążenia silnika. Z tego względu zdecydowano się na wymianę MG996R na mocniejsze serwomechanizmy PowerHD LF-20MG w trzech głównych przegubach: q_1 (obrót podstawy), q_2 (ramię) i q_3 (łokieć). Serwomechanizmy LF-20MG zapewniają znacznie wyższy moment (do 20 kg·cm), co umożliwia stabilne wykonywanie ruchów przy pełnym obciążeniu ramienia. W przegubie końcowym zastosowano dwa serwomechanizmy MG996R – jeden do obracania chwytaka, drugi do zacisku szczęk – ze względu na mniejsze wymagania momentu i ograniczony zakres ruchu.

Tabela 4.1: przedstawia zestawienie użytych serwomechanizmów wraz z ich parametrami.

Przegub	Model Serwa	Zakres ruchu	Moment obrotowy	Zasilanie
q1	LF-20MG	0–270°	20 kg·cm	6.0–7.4 V
q2	LF-20MG	0–270°	20 kg·cm	6.0–7.4 V
q3	LF-20MG	0–270°	20 kg·cm	6.0–7.4 V
q4 (obrót)	MG996R	0–180°	10 kg·cm	5.0–6.0 V
q4 (zacisk)	MG996R	0–180°	10 kg·cm	5.0–6.0 V

4.3 Elektronika robota

Sterowanie robota realizowane jest za pomocą układu mikrokontrolera Arduino Uno R3, rozszerzonego o moduł Sensor Shield V5.0, który zapewnia wygodne podłączenie serwomechanizmów oraz modułu komunikacyjnego. Zasilanie całego układu robota realizowane jest poprzez dwa niezależne źródła. Pierwszym z nich jest laboratoryjny zasilacz LongWei PS3010DF [9], ustawiony na napięcie 6,1 V DC, który dostarcza energię do Sensor Shield V5.0 – płytki rozszerzającej, rozprowadzającej napięcie do wszystkich serwomechanizmów oraz modułu komunikacyjnego Bluetooth. Przewody zasilające prowadzone są bezpośrednio do śrubowego złącza zasilania na Sensor Shieldzie. Drugim źródłem jest komputer, który dostarcza napięcie 5 V przez interfejs USB, zasilając bezpośrednio płytkę mikrokontrolera Arduino Uno R3. Taki podział źródeł pozwala na odseparowanie zasilania logicznego mikrokontrolera od zasilania części wykonawczej (serw i modułu Bluetooth), co przekłada się na stabilność całego układu.



Rysunek 4.5: Schemat układu elektronicznego robota.

W systemie zastosowano dwa interfejsy komunikacyjne:

- **USB**, który umożliwia bezpośrednie połączenie robota z komputerem do programowania i testowania,
- **Bluetooth**, zrealizowany przez moduł HC-05, który podłączony jest do linii TX/RX na Sensor Shieldzie i umożliwia bezprzewodowe sterowanie robotem z poziomu aplikacji mobilnej.

4.3.1 Zasilanie

Z Sensor Shielda napięcie 6,1 V trafia do:

- złącz PWM serwomechanizmów (które wymagają napięcia w zakresie 5–7,2 V),
- modułu Bluetooth HC-05, który może być zasilany napięciem do 6 V (VCC),
- masy wspólnej układu (GND), wspólnej z mikrokontrolerem.

Aby zapewnić poprawną komunikację pomiędzy Arduino Uno (logika 5 V) a modułem Bluetooth HC-05 (logika 3,3 V), zastosowano dzielnik napięcia zbudowany z dwóch rezystorów o wartości 1 k Ω , podłączony do linii TX z Arduino oraz do wejścia RX modułu Bluetooth. Dzielnik ten obniża poziom sygnału z 5 V do około 2,5 V, co mieści się w bezpiecznym zakresie napięć wejściowych modułu HC-05 i zapobiega jego uszkodzeniu. Choć zastosowany dzielnik napięcia powoduje niewielką stratę mocy (ok. 12,5 mW), jest to rozwiązanie wystarczające w przypadku sygnału komunikacyjnego o małym natężeniu prądu.

4.3.2 Komunikacja-

Układ sterowania manipulatorem wykorzystuje dwa podstawowe interfejsy komunikacyjne: przewodowy (USB) oraz bezprzewodowy (Bluetooth). Taka konfiguracja umożliwia elastyczne sterowanie ramieniem robota zarówno z poziomu komputera, jak i urządzenia mobilnego.

Mikrokontroler – Arduino UNO R3 [\[7\]](#)

Do realizacji zadań sterowania manipulatorem zastosowano popularną płytkę rozwojową Arduino UNO R3, opartą na mikrokontrolerze ATmega328P. Urządzenie to zostało wybrane ze względu na swoją prostotę obsługi, powszechność oraz szeroką dostępność bibliotek i dokumentacji.

Podstawowe parametry Arduino UNO R3:

- 1-rdzeniowy mikrokontroler ATmega328P (taktowanie 16 MHz),
- 32 KB pamięci Flash, 2 KB pamięci SRAM, 1 KB EEPROM,
- 14 cyfrowych wejść/wyjść (w tym 6 PWM),
- 6 wejść analogowych,
- interfejsy: UART, I2C, SPI,
- zasilanie: 5 V (z USB lub zewnętrzne przez Vin),
- wbudowany konwerter USB-UART (ATmega16U2).

Arduino UNO służy w tym projekcie jako główny kontroler systemu, odbierający dane przez UART oraz sterujący serwomechanizmami poprzez wyjścia PWM

Komunikacja przewodowa – USB

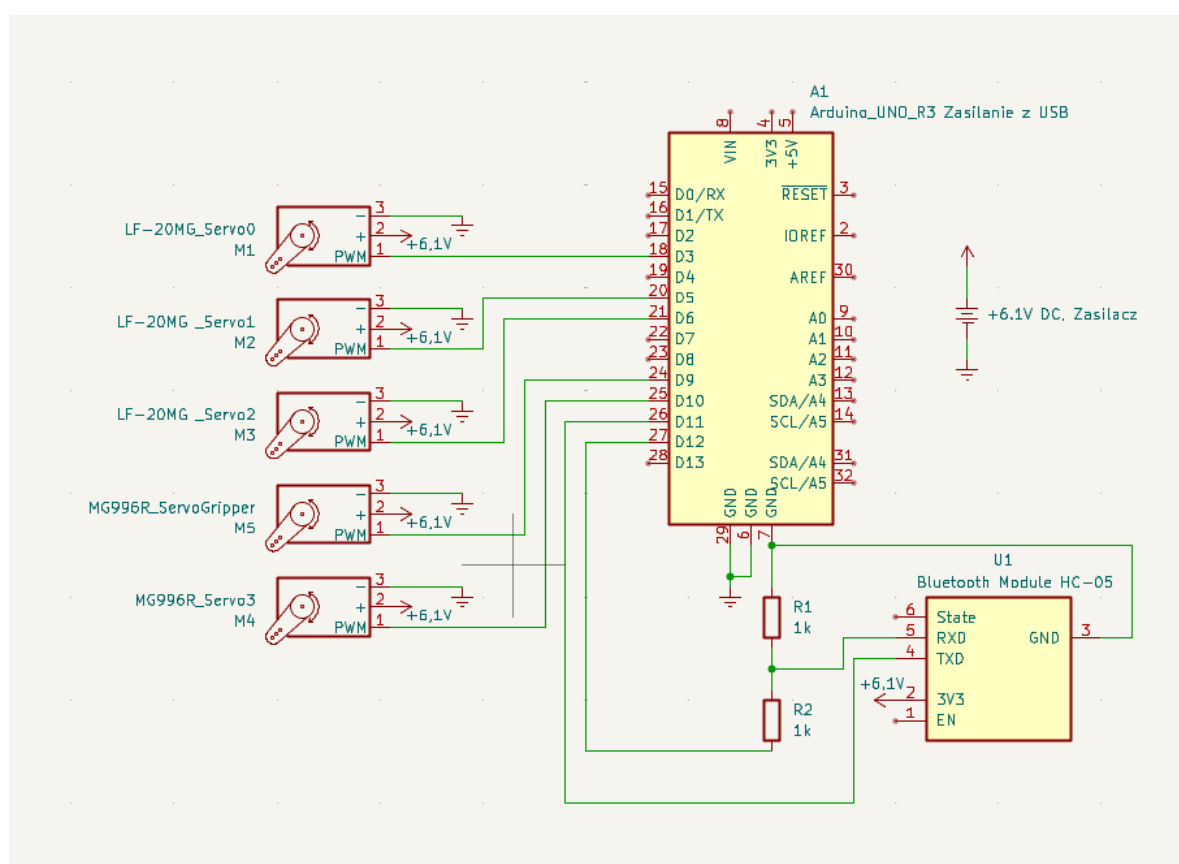
Podczas konfiguracji oraz testowania, mikrokontroler komunikuje się z komputerem PC za pomocą interfejsu USB-B, zapewniającego jednocześnie zasilanie logiczne 5 V oraz połączenie szeregowe (UART). Pozwala to na komunikację z interfejsem użytkownika stworzonym jako aplikacja komputerowa, jak również programowanie układu z poziomu środowiska Arduino IDE.

Komunikacja bezprzewodowa – Bluetooth HC-05 [8]

Do sterowania manipulatorem z poziomu aplikacji mobilnej użyto modułu Bluetooth HC-05, pracującego w trybie slave. Moduł ten komunikuje się z Arduino za pomocą interfejsu UART (RX/TX), umożliwiając bezprzewodową wymianę danych.

Podstawowe cechy modułu Bluetooth HC-05:

- interfejs komunikacyjny: **UART** (3.3 V logika),
- napięcie zasilania: **3.6–6 V** (zasilany z Sensor Shielda),
- tryb pracy: **slave** lub **master** (domyślnie slave),
- szybkość transmisji: **9600 bps** (domyślna),
- wbudowany regulator napięcia oraz dioda statusu połączenia.



Rysunek 4.6: Schemat elektroniczny układu robota.

Tabela 4.2 wyprowadzeń mikrokontrolera Arduino UNO.

Pin Arduino UNO	I/O	Funkcjonalność
D3	O	Sygnał PWM – serwomechanizm przegubu 0
D5	O	Sygnał PWM – serwomechanizm przegubu 1
D6	O	Sygnał PWM – serwomechanizm przegubu 2
D10	O	Sygnał PWM – serwomechanizm przegubu 3
D9	O	Sygnał PWM – serwomechanizm chwytaka
D11	O	RX Arduino ← TX Bluetooth (bezpośrednio)
D12	I	TX Arduino → RX Bluetooth (przez dzielnik napięcia)

4.4 Oprogramowanie

4.4.1 Struktura oprogramowania i użyte biblioteki

Oprogramowanie sterujące manipulatorem zostało zaimplementowane w środowisku Arduino IDE z wykorzystaniem języka C++ i biblioteki standardowej dla platformy Arduino. Głównym zadaniem kodu było przetwarzanie komend sterujących, wykonywanie obliczeń kinematycznych (prostych i odwrotnych), sterowanie serwomechanizmami oraz opcjonalna kompensacja błędów położenia końcówki manipulatora.

Do obsługi sprzętowej wykorzystano dwie biblioteki:

- Servo.h – biblioteka służąca do sterowania serwomechanizmami PWM,
- SoftwareSerial.h – biblioteka umożliwiająca realizację dodatkowej komunikacji szeregowej (w tym przypadku do obsługi modułu Bluetooth HC-05).

Całość oprogramowania składa się z pięciu głównych komponentów:

1. **Inicjalizacja systemu (setup())** – konfiguracja pinów serwomechanizmów, uruchomienie interfejsów komunikacyjnych oraz ustawienie pozycji początkowej ramienia.
2. **Główna pętla programu (loop())** – monitorowanie kanałów komunikacyjnych (USB oraz Bluetooth) i reagowanie na otrzymywane komendy.
3. **Funkcja handleCommand(String command)** – analizuje otrzymywaną komendę sterującą, odczytuje odpowiednie wartości zadanych kątów i przekazuje je do właściwych serw.
4. **Funkcje kinematyki:**
 - forwardKinematicsWithoutQ4() – oblicza położenie końcówki robota na podstawie zadanych kątów przegubów,

- `inverseKinematicsWithoutQ4()` – wyznacza kąty obrotu przegubów potrzebne do osiągnięcia zadanego położenia końcówki.

5. **Funkcja `applyCompensation()`** – opcjonalnie modyfikuje obliczone kąty przegubów, uwzględniając ręcznie dobrane poprawki, mające na celu kompensację rzeczywistych błędów pozycjonowania.

Wszystkie funkcje zostały ze sobą zintegrowane w sposób umożliwiający zarówno ręczne sterowanie ramieniem (za pomocą suwaków w interfejsie lub komend przez Bluetooth), jak i sterowanie automatyczne na podstawie współrzędnych przestrzennych.

4.4.2 Komunikacja i obsługa komend

Sterowanie robotem odbywa się za pomocą prostego protokołu tekstowego przesyłanego przez interfejs szeregowy. Program kontrolera został wyposażony w możliwość odbierania danych z dwóch źródeł komunikacji:

- USB (Serial) – wykorzystywane do komunikacji z komputerem oraz do testów lokalnych,
- Bluetooth (SoftwareSerial) – realizowane z użyciem modułu HC-05, umożliwiające zdalne sterowanie robotem z poziomu aplikacji mobilnej.

W kodzie zaimplementowano obsługę obu kanałów w ramach funkcji `loop()`. Odczytane linie tekstu są analizowane przez funkcję `handleCommand(String command)`, która odpowiada za rozpoznanie treści komendy i wykonanie odpowiednich akcji.

Format komend

Wysyłane komendy mają postać skrótu oznaczającego dany przegub oraz dwukropka i wartości kąta:

Tabela 4.3: Opis komend.

Komenda	Opis	Zakres wartości
S0:90	Obrót podstawy	0–180
S1:90	Kąt przegubu 1	0–180
S2:90	Kąt przegubu 2	55–150 (ograniczony zakres)
S3:90	Kąt przegubu 3	0–180
G:45	Ustawienie chwytaka (gripper)	0–70 (ograniczony zakres)

Przykład: komenda S2:120 powoduje ustawienie drugiego serwa (ramienia) na 120 stopni.

```
else if (command.startsWith("S2:")) {
    int angle = constrain(command.substring(3).toInt(), 0, 180);
    if (angle != lastServo2Angle) {
        servo2.write(angle);
        lastServo2Angle = angle;
    }
}
```

Rysunek 4.7: Fragment kodu Arduino – obsługa komendy S2 dla przegubu 2.

Obsługa i zabezpieczenia

Każda komenda jest analizowana pod kątem poprawności składni oraz wartości. Program zawiera funkcję `constrain(...)`, która ogranicza zakres możliwych kątów serwomechanizmów do bezpiecznych wartości, aby uniknąć uszkodzeń mechanicznych:

```
int angle = constrain(command.substring(3).toInt(), 0, 180);
```

Dodatkowo zastosowano zmienne `lastServoXAngle`, które zapobiegają zbędnemu wysyłaniu poleceń do serwa, jeśli wartość kąta nie uległa zmianie. Zapobiega to drganiom ramienia i przedłuża żywotność serwomechanizmów.

Komendy pozycyjne (XYZ)

Oprócz bezpośredniego ustawiania kątów, system umożliwia również przesyłanie współrzędnych przestrzennych (x, y, z), na podstawie których obliczane są odpowiednie kąty przegubów za pomocą funkcji `inverseKinematicsWithoutQ4()`. Obliczone kąty mogą być następnie automatycznie przesłane do serwomechanizmów.

```
bool inverseKinematicsWithoutQ4(float x, float y, float z,
                               int &angle1, int &angle2, int &angle3) {
    float th1 = atan2(y, x);
    float r = sqrt(x * x + y * y);

    float dz = z - L1;

    float D = (r * r + dz * dz - L2 * L2 - L3 * L3) / (2 * L2 * L3);

    if (D < -1.0 || D > 1.0) {
        return false;
    }

    float th3 = atan2(sqrt(1 - D * D), D);
    float th2 = atan2(dz, r) - atan2(L3 * sin(th3), L2 + L3 * cos(th3));

    angle1 = round(degrees(th1));
    angle2 = round(degrees(th2));
    angle3 = round(degrees(th3));

    return true;
}
```

Rysunek 4.8: Fragment kodu programu Arduino implementujący funkcję `inverseKinematicsWithoutQ4`.

W razie aktywnej kompensacji błędów, dodatkowa funkcja `applyCompensation()` wprowadza korekty do wyliczonych wartości, zapewniając wyższą precyzję pozycjonowania.

```
void applyCompensation(int &a1, int &a2, int &a3) {
    if (useCompensation) {
        a1 += round(delta1);
        a2 += round(delta2);
        a3 += round(delta3);
    }
}
```

Rysunek 4.9: Fragment kodu programu Arduino implementujący funkcję kompensacji błędów.

4.4.3 Opracowane aplikacje sterujące

Aplikacja komputerowa (.NET Framework)

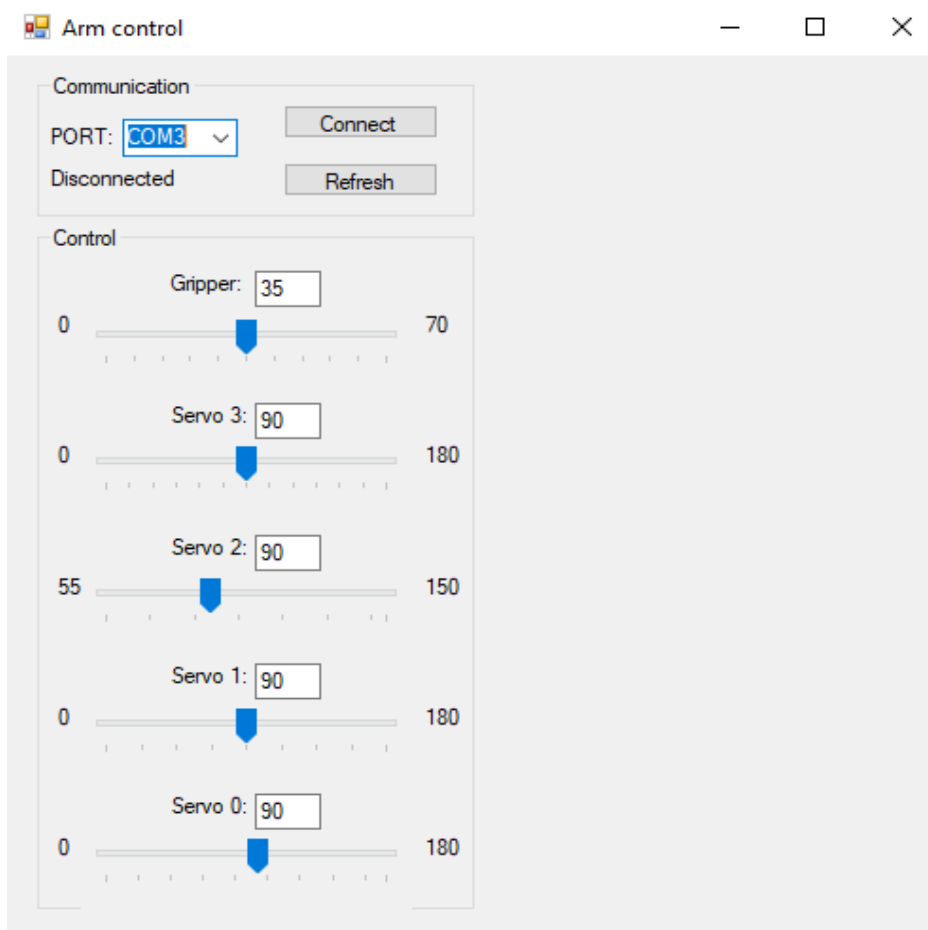
W celu umożliwienia wygodnego sterowania manipulatorem z poziomu komputera, zaprojektowano aplikację desktopową w języku C# z wykorzystaniem technologii .NET Framework. Aplikacja umożliwia użytkownikowi intuicyjne wydawanie poleceń i kontrolę pozycji serwomechanizmów za pomocą interfejsu graficznego.

Główne funkcjonalności:

- sterowanie pięcioma serwomechanizmami (S0–S3 oraz chwytakiem) za pomocą suwaków (TrackBar) i pól tekstowych (TextBox),
- dynamiczna synchronizacja wartości między suwakami a polami tekstowymi,
- wybór portu komunikacyjnego (np. COM3) z rozwijanej listy oraz inicjalizacja połączenia z mikrokontrolerem Arduino,
- przycisk Refresh do odświeżania listy dostępnych portów w czasie rzeczywistym,
- informacja o stanie połączenia (Connected/Disconnected),
- możliwość jednoczesnej zmiany wszystkich kątów przez wpisanie wartości w pola tekstowe i kliknięcie przycisku aktualizacji (Update) (po połączeniu z urządzeniem przycisk Refresh zmienia się w Update, umożliwiając jednoczesne przesłanie wszystkich wartości zadanych z pól tekstowych.).

Aplikacja została zaprojektowana z myślą o prostocie i przejrzystości, co czyni ją użyteczną zarówno podczas testów eksperymentalnych, jak i demonstracji działania systemu w warunkach laboratoryjnych.

Poniżej przedstawiono zrzut ekranu uruchomionej aplikacji komputerowej (rysunek 4.10):



Rysunek 4.10: Aplikacja komputerowa (.NET Framework).

Logika aplikacji została umieszczona w klasie Form1, dziedziczącej po Form. Komunikacja szeregową została zrealizowana przy pomocy klasy SerialPort, która umożliwia otwieranie, zamykanie oraz przesyłanie danych do portu COM.

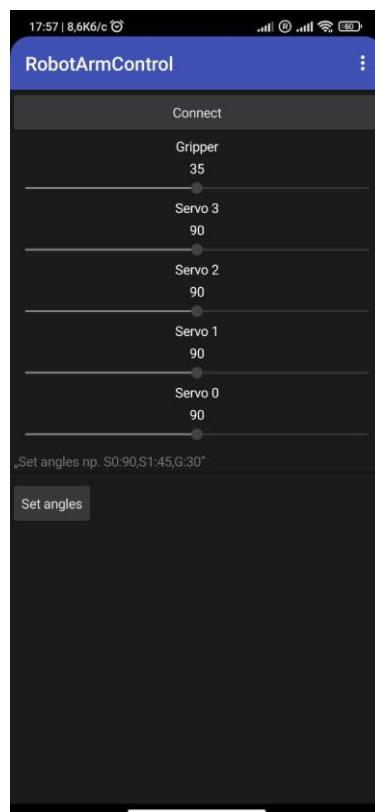
Funkcje i elementy aplikacji:

- Inicjalizacja portów COM: metoda LoadAvailablePorts() automatycznie odczytuje dostępne porty i wypełnia listę rozwijaną comboBoxPorts.
- Połączenie z Arduino: po kliknięciu przycisku Connect program otwiera wybrany port COM z prędkością transmisji 9600 baud. Status połączenia jest wyświetlany w formie kolorowego komunikatu.
- Sterowanie serwomechanizmami: każdy z pięciu przegubów robota (w tym chwytak) sterowany jest za pomocą:
 - suwaka (TrackBar) – przesuwając suwak użytkownik zmienia wartość kąta,
 - pola tekstowego (TextBox) – wpisując wartość kąta bezpośrednio z klawiatury.
 Przypisane zdarzenia (Scroll, KeyPress) synchronizują wartości obu kontrolerek i wysyłają odpowiednią komendę przez port szeregowy.

Wszystkie operacje związane z transmisją szeregową są objęte blokami try-catch, co pozwala na obsługę potencjalnych błędów transmisji, niepoprawnych danych oraz rozłączenia portu.

Aplikacja mobilna (MIT App Inventor)

W celu umożliwienia bezprzewodowego sterowania manipulatorem z poziomu urządzenia mobilnego, opracowano aplikację mobilną z wykorzystaniem środowiska **MIT App Inventor**. Interfejs umożliwia przesyłanie komend przez moduł Bluetooth HC-05 oraz pozwala użytkownikowi na sterowanie pięcioma serwomechanizmami robota (cztery osie + chwytak).



Rysunek 4.11: Aplikacja mobilna (MIT App Inventor).

Struktura aplikacji i logika działania

Aplikacja wykorzystuje dwa sposoby przesyłania danych do mikrokontrolera:

- **Bezpośrednie sterowanie suwakiem (ang. slider)** – każde przesunięcie suwaka powoduje natychmiastowe przesłanie komendy tekstowej (np. S1:90) przez `BluetoothClient.SendText`.
- **Pole tekstowe i przycisk „Set angles”** – użytkownik może ręcznie wpisać wartości w formacie protokołu (np. S0:90;S1:90;...) i wysłać całą sekwencję ustawień jednorazowo.

Kluczowe komponenty i ich funkcje:

- **ListPicker + BluetoothClient:** służy do wyszukiwania i łączenia się z dostępnymi urządzeniami Bluetooth (sparowanymi wcześniej z systemem Android).
- **Suwaki (Slider) + etykiety (Label)** – odpowiadają za przesyłanie aktualnych wartości kątów serw i ich wizualizację.

- **Pole tekstowe (TextBox)** – umożliwia ręczne wpisanie sekwencji komend w zadanym formacie.
- **Przycisk „Set angles”** – parsuje wprowadzone dane, ustawia wartości suwaków i przesyła je przez Bluetooth.
- **Bloki split, compare texts, get item i select list item** – implementują parser tekstowego polecenia w celu wyodrębnienia wartości przypisanych do danego serwomechanizmu.

Arduino odczytuje dane z portu szeregowego, parsuje je i ustawia kąty serwomechanizmów.

Rozdział 5

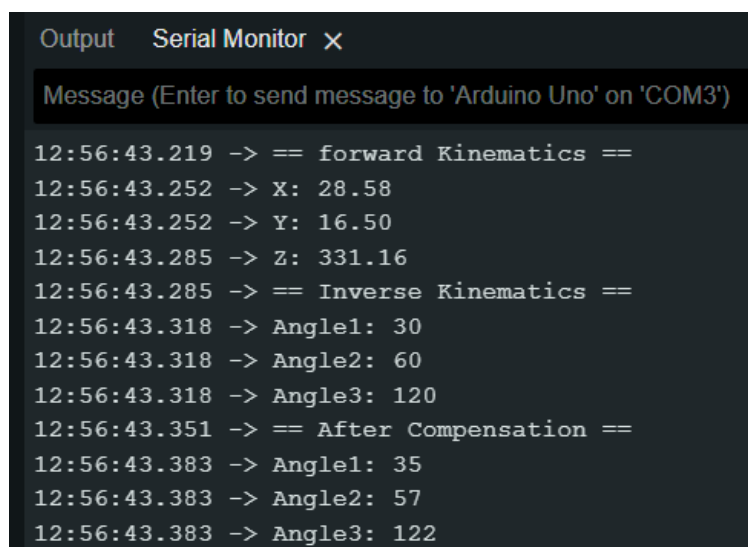
Uruchomienie i testy

5.1 Weryfikacja obliczeń kinematyki

W celu sprawdzenia poprawności działania zaimplementowanych funkcji kinematyki prostej i odwrotnej przeprowadzono testy symulacyjne oraz eksperymentalne. Poniżej przedstawiono przykładowy przypadek weryfikacyjny, w którym wykonano pełny cykl przeliczenia współrzędnych końcówki manipulatora. W odróżnieniu od obliczeń teoretycznych dla modelu kinematycznego robota, W implementacji kodu dodano 90° do trzeciego kąta (`angle3`) na końcu obliczeń, ponieważ w rzeczywistym układzie pozycja zerowa tego przegubu odpowiada wartości 90° , dodatkowo z uwzględnieniem odwróconego kierunku obrotu.

Poniższy zrzut ekranu (rysunek 5.1) przedstawia logi programu z przebiegu testu:

- Obliczenia pozycji końcówki,
- Odwrotna kinematyka,
- Przekształcone kąty po kompensacji.



```

12:56:43.219 -> == forward Kinematics ==
12:56:43.252 -> X: 28.58
12:56:43.252 -> Y: 16.50
12:56:43.285 -> Z: 331.16
12:56:43.285 -> == Inverse Kinematics ==
12:56:43.318 -> Angle1: 30
12:56:43.318 -> Angle2: 60
12:56:43.318 -> Angle3: 120
12:56:43.351 -> == After Compensation ==
12:56:43.383 -> Angle1: 35
12:56:43.383 -> Angle2: 57
12:56:43.383 -> Angle3: 122
  
```

Rysunek 5.1: Fragment logu kontrolera ilustrujący poprawność działania obliczeń kinematycznych.

Dla zadanych kątów przegubów: **Angle1 = 30°** , **Angle2 = 60°** oraz **Angle3 = 120°** , wykonano obliczenia kinematyki prostej (forward kinematics), uzyskując pozycję końcówki robota w przestrzeni: **X = 28,58 mm**, **Y = 16,50 mm**, **Z = 331,16 mm**. Następnie na podstawie tej pozycji przeprowadzono obliczenia kinematyki odwrotnej (inverse kinematics), które zwróciły dokładnie te same wartości kątów: **Angle1 = 30°** , **Angle2 = 60°** , **Angle3 = 120°** , co potwierdza poprawność zaimplementowanych funkcji `forwardKinematicsWithoutQ4()` oraz `inverseKinematicsWithoutQ4()`.

W celu zwiększenia precyzji odwzorowania ruchu, uwzględniono również działanie algorytmu kompensacji błędów. Skorygowane wartości przegubów wyniosły: **Angle1 = 35°**, **Angle2 = 57°**, **Angle3 = 122°**, co odpowiada zastosowaniu odpowiednio korekt: +5°, -3°, +2°. Dzięki temu możliwe jest ograniczenie wpływu rzeczywistych odchyłeń wynikających z luzów mechanicznych, niedokładności serwomechanizmów czy błędów przeliczeniowych. Eksperyment potwierdza, że w środowisku symulacyjnym odchylenia są marginalne, jednak w systemie rzeczywistym zastosowanie algorytmu kompensacyjnego może znacząco poprawić dokładność pozycjonowania efektora robota.

5.2 Eksperymentalna weryfikacja działania algorytmu kompensacji błędów

Celem niniejszego rozdziału jest przedstawienie procesu eksperymentalnego sprawdzenia skuteczności zaimplementowanego algorytmu kompensacji błędów położenia końcówki manipulatora. Kompensacja ta została zrealizowana poprzez dodawanie odpowiednich poprawek kątowych do wartości wyznaczonych na podstawie kinematyki odwrotnej. **Metodologia:**

Punkty testowe

Wybrano zestaw pozycji docelowych TCP w przestrzeni roboczej.

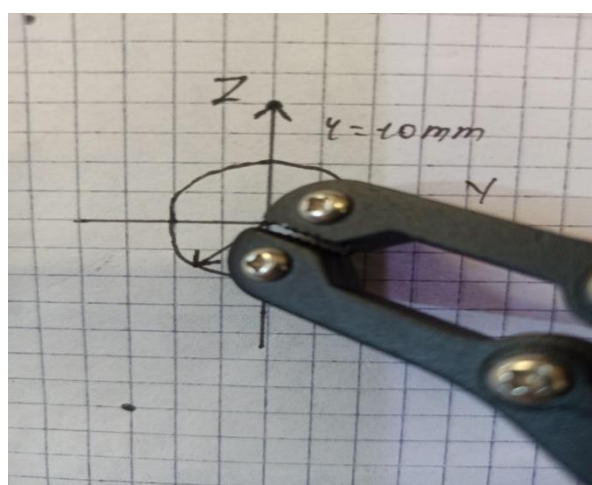
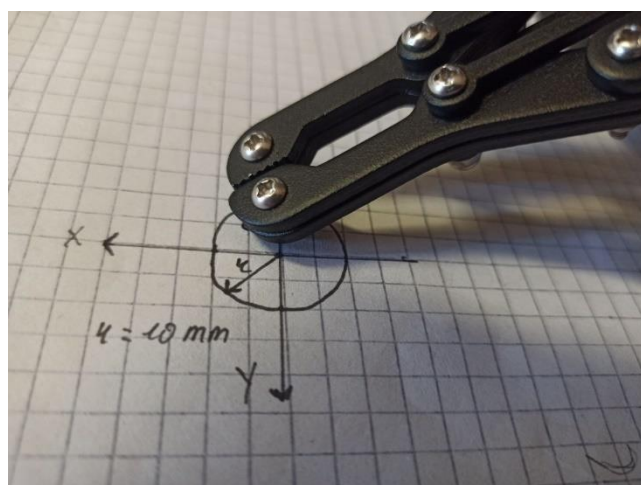
Dwa tryby pomiaru

Dla każdego punktu wykonano dwukrotne pozycjonowanie ramienia: bez kompensacji i z włączoną kompensacją.

Pomiary rzeczywistego położenia TCP

Pozycja końcówki była weryfikowana przy pomocy układu referencyjnego (rysunek 5.2)

Pomiar błędu



Dla każdej pozycji zmierzono odchylenie rzeczywistej pozycji TCP względem punktu docelowego.

Rysunek 5.2: Pomiary niedokładności zachowania pozycji.

Tabela 5.1: Wyniki pomiarów niedokładności zachowania pozycji.

Punkt	P1	P2	P3
Zadane (x, y, z) [mm]	(100, 130, 0)	(50, 210, 10)	(0, 180, 130)
Rzeczywiste bez kompensacji (x, y, z)	(104, 135, 0)	(55, 216, 7)	(4, 185, 133)
Błąd [mm]	(4, 5, 0)	(5, 6, 3)	(4, 5, 3)
Błąd względny [%]	[4.0, 3.85, 0]	[10.0, 2.86, 30.0]	[0, 2.78, 2.31]
Rzeczywiste z kompensacją (x, y, z)	(101, 133, 0)	(52, 213, 9)	(2, 183, 132)
Błąd [mm]	(1, 3, 0)	(2, 3, 1)	(2, 3, 2)
Błąd względny [%]	[1.0, 2.31, 0]	[4.0, 1.43, 10.0]	[0, 1.67, 1.54]

Na podstawie wyników zawartych w tabeli 5.1 można zauważyć, że zastosowanie algorytmu kompensacji błędów pozycyjnych skutecznie zmniejsza niedokładności w odwzorowaniu zadanych współrzędnych końcówki robota.

Dla wszystkich trzech testowanych punktów końcowych odnotowano wyraźne zmniejszenie błędów w każdym z kierunków przestrzennych (x, y, z). Na przykład, w przypadku punktu P1, błąd w osi X zmniejszył się z 4 mm do 1 mm, w osi Y z 5 mm do 3 mm, natomiast w osi Z pozycja została utrzymana bez odchyłki. Podobną poprawę zaobserwowano również w punktach P2 i P3, gdzie łączny błąd przestrzenny uległ zmniejszeniu nawet o połowę.

5.3 Pomiary zasilania układu

W celu określenia zapotrzebowania energetycznego robota oraz sprawdzenia poprawności doboru zasilania, przeprowadzono serię pomiarów napięcia i prądu pobieranego przez układ w różnych stanach pracy.

Do pomiarów wykorzystano zasilacz laboratoryjny LongWei PS3010DF o regulowanym napięciu wyjściowym w zakresie 0–30 V i maksymalnym prądzie 10 A. Pomiar został przeprowadzony przy napięciu roboczym 6,1 V, z wykorzystaniem zewnętrznego miernika oraz wskazań wbudowanego wyświetlacza.

Warunki pomiaru:

- Napięcie zasilania: 6,1 V (ustawione ręcznie),
- Obciążenie: 5 serwo mechanizmów (LF-20MG oraz MG996R),
- Pomiar prądu przy różnych stanach: bezczynność, ruch pojedynczego serwa, pełne obciążenie chwytaka.

Tabela 5.2: Pomiar poboru prądu oraz mocy przez układ robota.

Tryb pracy	Napięcie [V]	Prąd $\pm \Delta I$ [A]	Błąd Względny [%]	Moc [W]	Uwagi
Stan spoczynkowy (wszystkie serwa zatrzymane)	6.1	$0.18 \pm 0,01$	5,56	1.042	Tylko podtrzymanie pozycji
Ruch jednego serwa (q1)	6.1	$1.35 \pm 0,01$	0,74	4.578	Bez obciążenia mechanicznego
Ruch trzech serw równocześnie	6.1	$1,71 \pm 0,01$	0,58	10.255	Z obciążeniem
Ruch chwytaka (gripper) z obciążeniem	6.1	$0.36 \pm 0,01$	2,78	2.098	Chwytywanie małego przedmiotu
Zablokowany przegub + próba ruchu	6.1	$1.06 \pm 0,01$	0,94	6.025	Prąd szczytowy

Maksymalna zmierzona moc pobierana przez robota wynosiła około 10.255 W przy napięciu 6.1 V. Układ zasilania musi zapewniać stabilne zasilanie nawet przy chwilowych wzrostach poboru energii. Zasilacz PS3010DF, zdolny dostarczać do 10 A, spełnia wymagania systemu i zapewnia odpowiedni margines bezpieczeństwa.

5.4 Porównanie parametrów zaprojektowanego robota z istniejącymi rozwiązaniami

Na tle porównywanych konstrukcji zaprojektowany robot wyróżnia się najniższą masą własną (0,578 kg), kompaktowym zasięgiem ramienia (~230 mm) oraz dobrą powtarzalnością (szac. ± 3 mm). Pomimo wykorzystania prostego kontrolera Arduino UNO R3, robot umożliwia zdalne sterowanie i wykonuje operacje porównywalne z droższymi modelami edukacyjnymi. Parametry konstrukcji oraz osiągi potwierdzają, że przy odpowiednim projekcie i implementacji możliwe jest stworzenie funkcjonalnego manipulatora 4DOF do zastosowań edukacyjnych i demonstracyjnych przy niskim koszcie.

Tabela 5.3: Podstawowe parametry zaprojektowanego robota 4DOF.

Parametr	Zaprojektowany robot 4DOF	RoArm-M2-S	Multifunction Robotic Arm – IADIY	Igus® Robolink® DP – 4DOF
Liczba osi	4	4	4	4
Maksymalny udźwig	0.15 kg	0,5 kg	0,2-0,3 kg	2-3 kg
Zasięg ramienia	Około 230 mm	210 mm	350-500 mm	790 mm
Powtarzalność	$\sim \pm 3$ mm (szac.)	$\sim \pm 1$ mm (szac.)	$\sim \pm 0,5$ mm (szac.)	$\sim \pm 0,3$ mm (szac.)
Waga	0,578 kg	0,83 kg	ok. 1,2 kg	$\sim 5-6$ kg
Kontroler	Arduino UNO R3	ESP32, ROS2	Arduino	igus® Robot Control, ROS

Rozdział 6

Podsumowanie

Celem niniejszej pracy dyplomowej było zaprojektowanie, wykonanie oraz zaprogramowanie manipulatora o czterech stopniach swobody (4 DOF), umożliwiającego sterowanie efekтором w przestrzeni 3D, przy użyciu opracowanego algorytmu kinematyki odwrotnej oraz mechanizmu kompensacji błędów. Założenia pracy zostały w pełni zrealizowane, a końcowy efekt potwierdza funkcjonalność zaprojektowanego systemu.

W ramach realizacji projektu zmodyfikowano gotowe ramię robotyczne z zestawu 6DOF, ograniczając jego konstrukcję do czterech stopni swobody, co znacznie uprościło model matematyczny i umożliwiło skuteczne zastosowanie uproszczonego podejścia do kinematyki odwrotnej. Jednym z wyzwań była konieczność dobrania odpowiednich komponentów napędowych – początkowo zastosowane serwomechanizmy MG996R zostały zastąpione mocniejszymi i bardziej precyzyjnymi modelami LF-20MG. Równocześnie zrezygnowano z niestabilnych przetwornic napięcia na rzecz profesjonalnego zasilacza laboratoryjnego LongWei PS3010DF, co znacząco poprawiło niezawodność układu.

Zaprojektowany układ zrealizowano na bazie mikrokontrolera Arduino UNO, do którego podłączono wszystkie serwomechanizmy oraz moduł Bluetooth (HC-05), umożliwiający bezprzewodową komunikację z użytkownikiem. Opracowano także dwie aplikacje sterujące – mobilną oraz komputerową. Interfejs desktopowy wykonano w technologii .NET Framework, co pozwoliło na intuicyjną obsługę manipulatora oraz umożliwiło testowanie komunikacji szeregowej w czasie rzeczywistym.

W ramach pracy zaimplementowano algorytmy kinematyki prostej i odwrotnej, wykorzystujące model geometryczny manipulatora, a także zaprojektowano funkcję kompensacji błędów na podstawie wyników pomiarów. Przeprowadzone eksperymenty wykazały, że zastosowanie kompensacji znacząco poprawia dokładność pozycjonowania – w analizowanych przypadkach błąd zmniejszył się z 5–6 mm do około 2–3 mm.

Pomiar poboru prądu wykazał, że zaprojektowany układ cechuje się niskim zużyciem energii – maksymalne zużycie wynosiło około 10,26 W przy jednoczesnym ruchu trzech osi z obciążeniem w postaci niewielkiego przedmiotu o masie około 150 g, natomiast w stanie spoczynku pobór mocy ograniczał się do około 1 W. Przeprowadzone testy potwierdzają zdolność systemu do niezawodnej i stabilnej pracy.

Literatura

- [1] Kozłowski Krzysztof, Dutkiewicz Piotr, Wróblewski Waldemar; Modelowanie i sterowanie robotów. Wydawnictwo Naukowe PWN, 2016
- [2] Krzysztof Tchoń, Robert Muszyński, Robotyka, Notatki do wykładów z dziedziny automatyki i robotyki, Katedra Cybernetyki i Robotyki, Politechnika Wrocławska, Wrocław 2018
- [3] Addison, A. (2020). *How to Find Denavit-Hartenberg Parameter Tables*. Automatic Addison.
- [4] Mohammed, A. A. M., & Sunar, M. (2015). *Kinematics Modeling of a 4-DOF Robotic Arm* [Conference paper]. King Fahd University of Petroleum & Minerals.
- [5] Górski, F., Pluciennik, M., & Wichniarek, R. (2019). *Modelowanie i kompensacja błędów robota z użyciem LUT i funkcji interpolacyjnych*. *Pomiary Automatyka Robotyka*, 23(3), 123–129.
- [6] Dutka, A., & Winiarski, J. (2016). Czynniki wpływające na dokładność i powtarzalność pozycjonowania manipulatorów. *Pomiary Automatyka Robotyka*, 4, 55–61.
- [7] Dokumentacja techniczna układu mikroprocesora Arduino UNO R3
<https://docs.arduino.cc/hardware/uno-rev3>
- [8] Dokumentacja techniczna układu Bluetooth HC-05
https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf
- [9] Strona internetowa producenta zasilacza laboratoryjnego LongWei PS3010DF
<https://longweielec.com/product/ps-3010df>
- [10] Strona internetowa robota RoArm-M2-S
<https://www.waveshare.com/wiki/RoArm-M2-S>
- [11] Strona internetowa robota 4-DOF Multifunction Robotic Arm for Desktop od IADIY
<https://www.iadiy.com/4-DOF-Multifunction-Robotic-Arm-for-Desktop>
- [12] Strona internetowa robota Igus® robolink® DP – 4DOF
https://www.igus.com/product/20232?srsId=AfmBOor4RHztOVnHyrTYBwJVae3khTI9mcCTffbJ2TVxGjUfwJZMzGpU&utm_source=chatgpt.com&artNr=RL-DP-4