



WYDZIAŁ ELEKTRONIKI,  
TELEKOMUNIKACJI  
I INFORMATYKI

Imię i nazwisko studenta: Jan Górski

Nr albumu: 184294

Poziom kształcenia: Studia pierwszego stopnia

Forma studiów: stacjonarne

Kierunek studiów: Automatyka, cybernetyka i robotyka

Profil: Systemy automatyki

Imię i nazwisko studenta: Michał Blicharz

Nr albumu: 184430

Poziom kształcenia: Studia pierwszego stopnia

Forma studiów: stacjonarne

Kierunek studiów: Automatyka, cybernetyka i robotyka

Profil: Systemy automatyki

## PROJEKT DYPLOMOWY INŻYNIERSKI

Tytuł projektu w języku polskim: Budowa sześćo-osiowego robota antropomorficznego

Tytuł projektu w języku angielskim: Construction of a six-axis anthropomorphic robot

Opiekun pracy: dr inż. Piotr Fiertek

## **OŚWIADCZENIE dotyczące pracy dyplomowej zatytułowanej: Budowa sześćcio-osioowego robota antropomorficznego**

Imię i nazwisko studenta: Jan Górski  
Data i miejsce urodzenia: 26.10.2001, Gdańsk  
Nr albumu: 184294

Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki  
Kierunek: automatyka, cybernetyka i robotyka

Poziom kształcenia: pierwszy  
Forma studiów: stacjonarne

Typ pracy: projekt dyplomowy inżynierski

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2019 r. poz. 1231, z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t.j. Dz. U. z 2020 r. poz. 85, z późn. zm.),<sup>1</sup> a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

07.12.2023, Jan Górski

*Data i podpis lub uwierzytelnienie w portalu uczelnianym Moja PG*

*\*) Dokument został sporządzony w systemie teleinformatycznym, na podstawie §15 ust. 3b Rozporządzenia MNiSW z dnia 12 maja 2020 r. zmieniającego rozporządzenie w sprawie studiów (Dz.U. z 2020 r. poz. 853). Nie wymaga podpisu ani stempla.*

---

<sup>1</sup> Ustawa z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce:

Art. 312. ust. 3. W przypadku podejrzenia popełnienia przez studenta czynu, o którym mowa w art. 287 ust. 2 pkt 1–5, rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.

Art. 312. ust. 4. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 5, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o podejrzeniu popełnienia przestępstwa.

## **OŚWIADCZENIE dotyczące pracy dyplomowej zatytułowanej: Budowa sześćcio-osioowego robota antropomorficznego**

Imię i nazwisko studenta: Michał Blicharz

Data i miejsce urodzenia: 03.07.2000, Gdańsk

Nr albumu: 184430

Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki

Kierunek: automatyka, cybernetyka i robotyka

Poziom kształcenia: pierwszy

Forma studiów: stacjonarne

Typ pracy: praca dyplomowa inżynierska

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2019 r. poz. 1231, z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t.j. Dz. U. z 2020 r. poz. 85, z późn. zm.),<sup>1</sup> a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

07.12.2023, Michał Blicharz

*Data i podpis lub uwierzytelnienie w portalu uczelnianym Moja PG*

*\*) Dokument został sporządzony w systemie teleinformatycznym, na podstawie §15 ust. 3b Rozporządzenia MNiSW z dnia 12 maja 2020 r. zmieniającego rozporządzenie w sprawie studiów (Dz.U. z 2020 r. poz. 853). Nie wymaga podpisu ani stempla.*

---

<sup>1</sup> Ustawa z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce:

Art. 312. ust. 3. W przypadku podejrzenia popełnienia przez studenta czynu, o którym mowa w art. 287 ust. 2 pkt 1–5, rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.

Art. 312. ust. 4. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 5, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o podejrzeniu popełnienia przestępstwa.

## STRESZCZENIE

Poniższa praca opisuje projekt 6-osiowego robota antropomorficznego. Autorzy przedstawili proces jego projektowania i budowy. W pracy zawarto opis wykorzystanych rozwiązań mechanicznych, elektronicznych oraz programistycznych. Zaprojektowano przekładnie zapewniające odpowiedni moment obrotowy przegubów i wykonano je za pomocą drukarki 3D. Jako napędów użyto silników krokowych i serwomechanizmów. Zaproponowano algorytm rozwiązywania kinematyki odwrotnej dla robota o 6 stopniach swobody oraz rozwiązania dla ruchu z interpolacją osiową i liniową. Kalibracja robota odbywa się ręcznie. Efektem prac jest działający robot o maksymalnym zasięgu około 500 mm z możliwością rozbudowy o chwytaki i inne narzędzia.

**Słowa kluczowe:** robot antropomorficzny, kinematyka odwrotna, 6 osi swobody, interpolacja osiowa, interpolacja liniowa, ESP32

**Dziedzina nauki i techniki, zgodnie z wymogami OECD:** Nauki Inżynieryjne i Techniczne, Robotyka i Automatyka, Elektrotechnika, Elektronika i Inżynieria Informatyczna

## **ABSTRACT**

This thesis describes the design of a 6-axis anthropomorphic robot. The authors present the process of its designing and building. The paper contains a description of the mechanical, electronic and programming solutions used in the project. Appropriate gears were designed to provide torque needed by the joints and were made using a 3D printer. Stepper motors and servomechanisms were used as drives. The authors propose an algorithm for solving the inverse kinematics for a 6 degrees of freedom robot and solutions for motion with axes and linear interpolation. The robot calibration is done manually. The result of this project is a working robot with a maximum range of approximately 500 mm with a possibility of adding grippers and other tools to it.

**Keywords:** anthropomorphic robot, inverse kinematics, 6 degrees of freedom, axes interpolation, linear interpolation, ESP32

## Spis treści

<b>WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW</b>	<b>8</b>
<b>1 WSTĘP I CEL PRACY</b>	<b>10</b>
1.1 Wstęp . . . . .	10
1.2 Cel pracy . . . . .	10
1.3 Podział prac . . . . .	10
1.4 Układ pracy . . . . .	11
<b>2 PRZEGLĄD OBECNYCH ROZWIĄZAŃ (Jan Górski)</b>	<b>12</b>
2.1 Roboty przemysłowe . . . . .	12
2.1.1 FANUC M-2000iA/2300 . . . . .	12
2.1.2 ABB IRB 8700 . . . . .	13
2.2 Koboty . . . . .	14
2.2.1 Universal Robots UR5 . . . . .	14
2.2.2 KUKA LBR iisy 6 R1300 . . . . .	14
2.3 Wnioski . . . . .	15
<b>3 DOBÓR ELEMENTÓW (Michał Blicharz)</b>	<b>16</b>
3.1 Elementy mechaniczne . . . . .	16
3.1.1 Silniki . . . . .	17
3.1.2 Redukcja napędu . . . . .	17
3.2 Elementy elektroniczne . . . . .	19
3.2.1 Sterowniki silników . . . . .	20
<b>4 PROJEKT MECHANICZNY (Michał Blicharz)</b>	<b>21</b>
4.1 Nadgarstek robota (osie 4-6) . . . . .	21
4.2 Osie 1-3 . . . . .	22
4.2.1 Projekt przekładni planetarnej . . . . .	22
4.3 Podstawa robota . . . . .	25
4.4 Wykonanie . . . . .	26
4.5 Wnioski . . . . .	26
<b>5 PROJEKT ELEKTRONICZNY (Michał Blicharz)</b>	<b>28</b>

5.1 Sterowniki silników . . . . .	28
5.2 Zasilacze . . . . .	28
5.3 Schemat połączeń elektronicznych . . . . .	29
5.3.1 Komunikacja z serwomechanizmami Dynamixel AX-12A . . . . .	30
5.4 Wykonanie płytki PCB . . . . .	31
5.5 Wnioski i perspektywy dalszego rozwoju . . . . .	32
<b>6 KINEMATYKA ODWROTNA (Jan Górski)</b>	<b>34</b>
6.1 Kinematyka prosta manipulatora . . . . .	34
6.2 Charakterystyka problemu . . . . .	37
6.3 Dobór rozwiązania . . . . .	37
6.4 Algorytm rozwiązania . . . . .	38
<b>7 OPROGRAMOWANIE (Jan Górski)</b>	<b>39</b>
7.1 Komunikacja z silnikami . . . . .	39
7.1.1 Silniki krokowe . . . . .	39
7.1.2 Serwomechanizmy . . . . .	41
7.2 Interpreter rozkazów . . . . .	42
7.3 Kalibracja osi robota . . . . .	43
7.4 Kinematyka odwrotna . . . . .	45
7.5 Ruch z interpolacją osiową . . . . .	47
7.6 Ruch z interpolacją liniową . . . . .	47
7.7 Uczenie pozycji robota . . . . .	49
7.8 Pętla programowa . . . . .	49
<b>8 WYNIKI</b>	<b>51</b>
8.1 Algorytm rozwiązywania kinematyki odwrotnej (Jan Górski) . . . . .	51
8.2 Ruch z interpolacją osiową (Jan Górski) . . . . .	52
8.3 Ruch z interpolacją liniową (Jan Górski) . . . . .	55
8.4 Dokładność ruchu (Jan Górski) . . . . .	56
8.5 Maksymalny udźwig (Michał Blicharz) . . . . .	57
<b>9 PODSUMOWANIE</b>	<b>58</b>
9.1 Perspektywy rozwoju . . . . .	59
<b>WYKAZ LITERATURY</b>	<b>61</b>
<b>Załącznik nr 1: Równania kinematyki prostej</b>	<b>62</b>

## WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

- *kobot* - robot współpracujący (ang. collaborative robot)
- *PLA* - jeden z materiałów wykorzystywanych w druku 3D (polilaktyd)
- *NEMA* - standard określający wielkość silników krokowych, określa przekątną silnika w calach (np. NEMA 14 - przekątna 1,4 cala)
- *metoda D-H* - metoda przedstawienia łańcucha kinematycznego za pomocą notacji Denavita-Hartenberga
- *kinematyka odwrotna* - matematyczny proces wyznaczania wartości kątów poszczególnych przegubów robota zapewniających zadane położenie i orientację końcówki manipulatora
- *T* - macierz translacji
- *R* - macierz rotacji
- *H* - macierz transformacji
- $(x, y, z)$  - współrzędna x, y, z względem bazy robota
- $(\varphi, \psi, \theta)$  - kąt obrotu wokół osi OX, OY, OZ względem bazy robota
- $P_d$  - punkt zadany końcówki manipulatora
- $\phi^i$  - wartości kątów przegubów w  $i$ -tej iteracji algorytmu
- $E$  - błąd położenia efektora od pozycji zadanej
- $\epsilon$  - zadana dokładność rozwiązania
- $f(\phi^i)$  - funkcja kinematyki prostej (położenie efektora w  $i$ -tej iteracji algorytmu)
- $J(\phi^i)$  - jacobian funkcji kinematyki prostej
- $J^\dagger(\phi^i)$  - macierz pseudoodwrotna Moore'a-Penrose'a jacobianu funkcji kinematyki prostej
- *FreeRTOS* - system operacyjny czasu rzeczywistego na licencji MIT
- *język C++ dla Arduino* - język programowania dla platformy Arduino oparty na zmodyfikowanym języku C++
- *MicroPython* - wersja języka Python 3 zoptymalizowana do działania na mikrokontrolerach



- *UART* - uniwersalny asynchroniczny odbiornik-nadajnik, interfejs sprzętowy pozwalający na komunikację szeregową z buforem w przypadku dużych szybkości transmisji
- *I2C* - dwukierunkowa magistrala służąca do szeregowej komunikacji pomiędzy układami elektronicznymi
- *RPM* - obroty na minutę [ $\frac{obr.}{min}$ ] (ang. *revolutions per minute*)

# **1 WSTĘP I CEL PRACY**

## **1.1 Wstęp**

Robotem przemysłowym nazywamy automatycznie sterowalną, programowalną, wielozadaniową, stacjonarną lub mobilną maszynę o co najmniej trzech stopniach swobody, posiadającą właściwości manipulacyjne bądź lokomocyjne dla zastosowań przemysłowych. Ich funkcje i ruchy można dynamicznie zmieniać, nie wpływając na ich konstrukcję mechaniczną. Główną zaletą robotów przemysłowych jest ich bardzo szerokie zastosowanie, zaczynając od przemysłu ciężkiego na medycynie kończąc. Odwołując się do tematu pracy, robot antropomorficzny to robot o szeregowym łańcuchu kinematycznym posiadający wszystkie osie obrotowe. Minimalna liczba osi (przegubów) wynosi trzy. Dodatkowe przeguby pozwalają na większą elastyczność ruchu manipulatora w ograniczonej przestrzeni, co w przypadku 6 stopni swobody, umożliwia uzyskanie zadanego punktu w dowolnej orientacji narzędzia.

## **1.2 Cel pracy**

Celem projektu inżynierskiego było zaprojektowanie i budowa ramienia robotycznego o sześciu stopniach swobody. W konstrukcji robota wykorzystano silniki krokowe i serwomechanizmy wraz z przekładniami zapewniającymi uzyskanie odpowiedniego momentu obrotowego na osiach robota. Należało opracować i zaimplementować prosty zestaw komend sterujących robotem (prosty język programowania robota). Kontroler powinien umożliwiać przemieszczanie narzędzia robota z interpolacją osiową i liniową. Jako główną jednostkę sterującą wykorzystano mikrokontroler ESP32. Do sterowania silnikami krokowymi wykorzystano sterowniki posiadające enkodery magnetyczne, umożliwiające odczyt aktualnej pozycji wału silnika i pracę w zamkniętej pętli sterowania. Zastosowane sterowniki umożliwiają komunikację z wykorzystaniem interfejsu UART. Do napędu ostatnich trzech osi robota wykorzystano serwomechanizmy Dynamixel AX-12A.

## **1.3 Podział prac**

Poszczególne zadania zostały rozdzielone pomiędzy autorów projektu w następujący sposób. Michał Blicharz dobrał elementy konstrukcyjne robota (silniki, sterowniki silników, materiały konstrukcyjne, zasilanie, mikrokontroler). Zajmował się projektem części elektronicznej, w tym

wykonaniem płytki PCB układu sterującego. Jego głównym zadaniem było zaprojektowanie i wykonanie konstrukcji mechanicznej robota i przekładni planetarnych. Przeprowadził również testy silników i przekładni.

Jan Górski odpowiedzialny był za oprogramowanie i część matematyczną projektu. Napisał główny program sterujący robotem oraz biblioteki w języku C do mikrokontrolera ESP32, realizujące odpowiednie sterowanie silnikami. Jego zadaniem było wyznaczenie rozwiązania problemu kinematyki odwrotnej oraz jego implementacja programowa. Oprogramował również prosty interfejs użytkownika oraz realizację ruchu narzędzia z interpolacją osiową i liniową. Zaprojektował też trzy ostatnie stopnie swobody robota i przeprowadził przegląd rozwiązań dostępnych na rynku.

#### **1.4 Układ pracy**

W rozdziale 1 przedstawiono wprowadzenie do omawianego tematu, wyjaśniono, czym jest robot 6-osiowy oraz przedstawiono cele opisywanego projektu. Rozdział 2 zawiera przegląd aktualnych rozwiązań, które były źródłem inspiracji podczas projektowania powstałej w ramach pracy konstrukcji robota. W rozdziale 3 przedstawiono dobór elementów (zarówno elektronicznych, jak i mechanicznych) wykorzystanych w projekcie. Rozdział 4 skupia się na projekcie i konstrukcji mechanicznej robota. W rozdziale 5 opisano rozwiązania elektroniczne robota. Kinematykę odwrotną oraz dobór i algorytm jej rozwiązywania przedstawiono w rozdziale 6. W rozdziale 7 opisano oprogramowanie robota: sterowanie silnikami, komunikację z użytkownikiem, ruch z interpolacją osiową i liniową. W rozdziale 8 autorzy przedstawili osiągnięte rezultaty przeprowadzonych testów. W rozdziale 9 zawarto podsumowanie pracy wraz z perspektywami dalszego rozwoju.

## 2 PRZEGLĄD OBECNYCH ROZWIĄZAŃ (Jan Górski)

Obecnie na rynku znaleźć można wiele robotów antropomorficznych. Powszechnie używa się ich przy automatyzacji procesów produkcyjnych. Liderami w tej dziedzinie są takie firmy jak: ABB, FANUC, KUKA, Universal Robots, Mitsubishi, Kawasaki, Yaskawa. Dobór konstrukcji mechanicznej robota zależy od wymagań użytkownika. Przykładowo w fabrykach samochodów, gdzie przenoszone elementy posiadają duży ciężar, głównym kryterium jest m.in. udźwig. Z kolei w branży elektronicznej, gdzie komponenty są lekkie, ważniejsza jest precyzja i szybkość. Analiza obecnych rozwiązań pozwala na wybór konstrukcji mechanicznej, spełniającej opisane w rozdziale 3 wymagania projektu inżynierskiego.

### 2.1 Roboty przemysłowe

Dominującym polem zastosowań ramion robotycznych jest przemysł. Głównymi wymaganiami w projektowaniu rozwiązań dla tej branży są precyzja i niezawodność. Dodatkowo często wymagany jest również duży udźwig, gdy robot ma pracować z ciężkimi elementami. Poniżej przedstawiono przykładowe modele robotów przemysłowych tych firm, które jako podstawowe kryterium przyjmują duży udźwig. Pozwala to na wytypowanie rozwiązań mechanicznych, które mogą być przydatne w projekcie. Przeskalowanie spotykanych rozwiązań może pomóc w maksymalizacji osiągnięć robota przy ograniczonym budżecie na zakup komponentów.

#### 2.1.1 FANUC M-2000iA/2300

Robot FANUC M-2000iA/2300 posiada udźwig do 2300 kg i zasięg 3,734 m [1], co według producenta czyni go najmocniejszym robotem dostępnym na rynku. Znaczący wpływ na tę cechę ma umieszczenie silników pierwszych trzech osi łańcucha kinematycznego w bazie robota. Napęd przekazywany jest z podstawy do przegubu robota za pomocą cięgien. Minimalizuje to masę ruchomą manipulatora, dzięki czemu równocześnie zostaje zmniejszona inercja robota. Równocześnie zwiększony zostaje moment obrotowy przegubów, gdyż robot nie musi dźwigać kolejnych napędów. Również zastosowanie nadgarstka sferycznego wpływa pozytywnie na zwiększenie udźwigu. Obliczenia kinematyki odwrotnej stają się prostsze, gdyż możliwe jest podzielenie łańcucha kinematycznego robota na dwie części odpowiedzialne kolejno za położenie i orientację efektora.



Rysunek 2.1: Robot FANUC M-2000iA/2300 [2]

#### 2.1.2 ABB IRB 8700

Maksymalny udźwig i zasięg robota ABB IRB 8700 wynosi odpowiednio 800 kg i 4,2 m [3]. W tym przypadku producent również zastosował przeniesienie napędu z użyciem cięgien. Robot jest jednak 25% szybszy od konkurencyjnych konstrukcji dzięki użyciu pojedynczej przekładni na każdą oś napędu. Roboty tej klasy zazwyczaj posiadają podwójne napędy lub przekładnie. Sprawdziło to, że cena i awaryjność zostały zmniejszone. Dodatkowo model ten posiada amortyzatory sprężynowe i możliwość balansowania obciążenia w czasie ruchu. Podobnie jak w konstrukcjach innych firm, robot wyposażony jest w nadgarstek sferyczny.



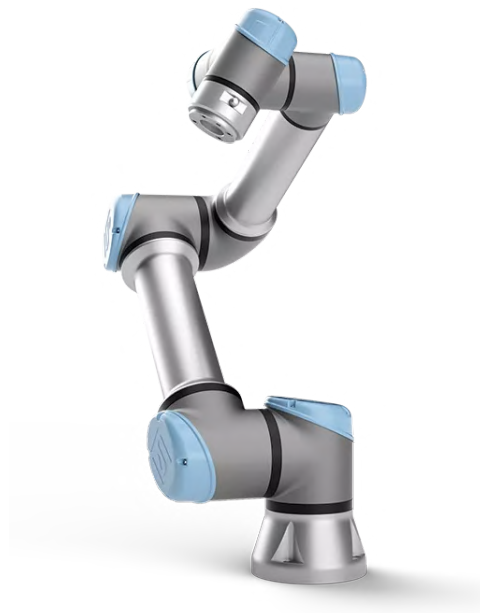
Rysunek 2.2: Robot ABB IRB 8700 [4]

## 2.2 Koboty

W ciągu ostatnich lat mocno rozwinęła się branża kobotów. Charakteryzują się one mniejszym udźwigiem. W zamian koboty umożliwiają współpracę z ludźmi dzięki zastosowaniu czujników obciążenia poszczególnych osi robota oraz zmniejszeniu ich masy i wymiarów. W ich konstrukcji stosuje się lekkie materiały o zaokrąglonych krawędziach. Szybkie zahamowanie ruchu kobotów następuje już przy wykryciu niewielkich różnic w obciążeniach osi robota względem przewidywanej wielkości uwzględniające konfigurację i dynamikę robota. Tego typu roboty nie muszą być oddzielane od operatorów na stanowisku pracy.

### 2.2.1 Universal Robots UR5

Jednym z najbardziej rozpoznawalnych kobotów z pewnością jest UR5. Jego udźwig wynosi 5 kg, natomiast zasięg 0,85 m. Dodatkowym atutem jest niska waga robota, wynosząca 18,4 kg [5]. Jednolita konstrukcja pozwala na wielokrotne użycie już raz zaprojektowanych elementów. Łączenie poszczególnych stopni swobody za pomocą rur umożliwia uporządkowanie okablowania. Zmniejsza to ryzyko wypadku wynikającego z możliwości zahaczenia o wystające elementy robota przez operatora, np. ubraniem.



Rysunek 2.3: Robot UR5 [6]

### 2.2.2 KUKA LBR iiwa 6 R1300

Podobną konstrukcję posiada robot KUKA LBR iiwa 6 R1300 o udźwigu wynoszącym 6 kg i zasięgu 1,3 m [7]. Charakteryzuje się on jednak innym łańcuchem kinematycznym niż UR5.

Wartą odnotowania funkcją tego robota, występującą również w UR5, jest możliwość programowania ścieżki robota poprzez ręczną zmianę położenia członów robota. Pozwala to na szybkie i intuicyjne programowanie ruchu robota. Podobnie jak robot UR5, Charakteryzuje się on obłymi kształtami i niską wagą.



Rysunek 2.4: Robot KUKA LBR iisy 6 R1300 [8]

### 2.3 Wnioski

Na podstawie przeprowadzonej analizy obecnych rozwiązań wysunięto wnioski o doborze rozwiązań kinematycznych. Głównymi kryteriami doboru łańcucha kinematycznego, a co za tym idzie konstrukcji robota, były cena podzespołów, prostota budowy poszczególnych osi manipulatora oraz dostępność elementów. Podczas projektowania konstrukcji mechanicznej robota postanowiono wzorować się na rozwiązaniach zastosowanych w robocie UR5. Wybór ten pozwala na zastosowanie najprostszego rozwiązania konstrukcji mechanicznej (powtarzalność elementów w kolejnych stopniach swobody). Jako że udźwig nie był krytycznym parametrem robota, postanowiono zrezygnować z dodatkowej siły oferowanej przy potencjalnym wykorzystaniu cięgien.

Nadgarstek sferyczny umożliwiłby znaczące uproszczenie równań kinematyki odwrotnej, dzięki możliwości zastosowania rozprężenia kinematycznego. Stwarzał on jednak ryzyko potencjalnych problemów, jeśli chodzi o złożoność mechaniczną i realizację przeniesienia napędów. Z tego powodu zdecydowano się go wykluczyć. Większa prostota konstrukcji składającej się z powtarzalnych elementów mechanicznych, mimo bardziej skomplikowanych równań kinematyki odwrotnej, została uznana za korzystniejszą.

### 3 DOBÓR ELEMENTÓW (Michał Blicharz)

Podstawowymi założeniami przy projektowaniu konstrukcji robota był jego zasięg, maksymalny udźwig oraz dokładność pozycjonowania narzędzia. Poniżej przedstawiono zakładane wartości głównych parametrów robota:

- zasięg: co najmniej 500 mm,
- maksymalny udźwig: nie mniej niż 250 g,
- powtarzalność: nie gorsza niż 15 mm.

#### 3.1 Elementy mechaniczne

Dobór elementów mechanicznych był kluczowy dla uzyskania zakładanych parametrów. Zdecydowano się na wykorzystanie silników krokowych z przekładnią. Główną zaletą silników krokowych jest prostota uzyskania stałego kąta obrotu (obrót o stały krok). Kolejną ich zaletą jest stosunkowo niewielka cena oraz dostępność. Z kolei wadami silników krokowych są: niewielki moment obrotowy, możliwość zgubienia kroku oraz największa z nich, praca krokowa, która skutkuje brakiem płynnego ruchu. Ostatni z problemów silników krokowych w tym wypadku jest pomijalny, ponieważ prędkość obrotowa silników jest na tyle duża, że problem braku płynności ruchu jest trudny do zauważenia gołym okiem i nie wpływa istotnie na pracę robota. Pierwszy z tych problemów można rozwiązać, stosując odpowiednią przekładnię. Problem gubienia kroków udało się wyeliminować, stosując kontroler silników zintegrowany z enkoderem umożliwiającym pracę w zamkniętej pętli sprzężenia zwrotnego.

Kolejnym ważnym elementem konstrukcji mechanicznej, który należy wybrać, jest materiał, z którego wykonana zostanie główna konstrukcja ramienia łącząca kolejne stopnie swobody. W tym wypadku rozważano dwa rozwiązania: druk 3D (PLA) oraz zastosowanie rury PVC. Wybór rozwiązania oparto o kryteria punktowane w zależności od ich istotności (tabela 3.1). Dla obu rozwiązań przyporządkowano punkty w każdej kategorii. O wyborze decydowała sumaryczna ilość punktów. Finalnie zdecydowano się na wykorzystanie rury PVC.



		Możliwe rozwiązania	
Kryterium	Maksymalna liczba pkt	Rura PVC	Druk 3D
Łatwość obróbki	10	8	10
Wytrzymałość	30	30	26
Waga	20	15	13
Cena	10	9	10
Dostępność	10	10	10
Suma punktów		72	69

Tabela 3.1: Tabela pokazująca proces wyboru materiału na łączniki stopni swobody

### 3.1.1 Silniki

W ostatnich trzech stopniach swobody zdecydowano się na użycie serwomechanizmów Dynamixel AX-12A [9]. Głównymi ich zaletami jest niska waga (54.5 g) oraz niewielkie wymiary. Dodatkową zaletą była dostępność (zostały pozyskane bezpłatnie z zasobów koła naukowego SKAR). Serwomechanizmów Dynamixel AX-12A nie można było zastosować w pierwszych trzech osiach robota, ze względu na niewystarczający moment obrotowy. Zwiększenie momentu za pomocą przekładni nie było możliwe z uwagi na ograniczenie zakresu ruchu serwomechanizmów. Z tego powodu w pierwszych stopniach zdecydowano się użyć silniki NEMA. W przypadku trzeciej osi wykorzystano silnik NEMA 17. Jest on kompromisem pomiędzy wagą a momentem obrotowym. Oszacowano, że w połączeniu z przekładnią 40:1, jego moment obrotowy będzie wystarczający. Silnik NEMA 17 jest ponad dwukrotnie lżejszy od NEMA 23. W drugiej osi zastosowano wspomniany silnik NEMA 23, ponieważ jego waga w tym miejscu nie jest tak krytyczna, a zapewnia ponad dwukrotnie większy moment obrotowy. Przy obrocie pierwszej osi nie jest konieczne dźwiganie masy całego robota, dzięki czemu można zastosować mniejszy i tańszy silnik NEMA 17.

### 3.1.2 Redukcja napędu

Przy wyborze elementów mechanicznych robota należy również uwzględnić dobór typu redukcji napędu. Redukcja napędu jest konieczna, ponieważ silniki o wymiarach i wadze odpowiedniej do rozmiaru robota, nie posiadają momentu obrotowego pozwalającego na uzyskanie zakładanych parametrów pracy robota. Wybór redukcji napędu był kluczowy dla udźwigu i dokładności robota. Znaczącą rolę przy doborze przekładni stanowiła możliwość wykonania jej w technologii druku 3D. Rozważano takie rozwiązania jak: przekładnia pasowa, przekładnia harmoniczna, przekładnia cykloidalna oraz przekładnia planetarna. Poniżej, pokrótce przedstawiono te rozwiązania.

## Przekładnia pasowa

Największymi zaletami przekładni pasowej są: prostota, niewielka waga oraz stosunkowo niewielka cena. Najprostsza przekładnia pasowa opiera się na wykorzystaniu dwóch kół pasowych połączonych pasem. Jedno koło jest kołem napędzającym, drugie napędzanym. Do koła napędzającego dołączany jest silnik (wejście przekładni). Koło napędzane jest wyjściem przekładni. Przełożenie takiej przekładni opisuje się wzorem  $i = \frac{d_1}{d_2}$ , gdzie:  $i$  jest teoretycznym przełożeniem przekładni,  $d_1$  i  $d_2$  średnicami skutecznymi koła odpowiednio napędzającego i napędzanego. Głównymi problemami z wykorzystaniem przekładni pasowej w tym rozwiązaniu są duży rozmiar oraz niewielkie przełożenie przekładni.

## Przekładnia falowa (harmoniczna)

Kolejnym typem przekładni jest przekładnia falowa. Głównymi zaletami tego rozwiązania są: niska waga, niewielki rozmiar, duże przełożenie oraz bardzo małe luzy. Przekładnie harmoniczne składają się z trzech głównych elementów. Są nimi: koło zębate sztywne, generator fali, koło zębate podatne. Ruch obrotowy generatora fali powoduje odkształcanie się elementu podatnego. Skutkiem zniekształcenia jest przesunięcie się koła podatnego względem sztywnego. Pozwala to na uzyskanie dużego przełożenia. Przekładnia ta jest bardzo wytrzymała i pozwala przenosić duży moment obrotowy, ponieważ jednocześnie zazębione jest około 30% zębów. Nie zdecydowano się na wykorzystanie tej przekładni z powodu konieczności spełnienia bardzo rygorystycznych reguł jej projektowania. Dodatkowo wykonanie odpowiednio wytrzymałego, a jednocześnie elastycznego koła podatnego, byłoby bardzo trudne w technice druku 3D. Znalaziono jedną możliwą do kupienia przekładnię falową pasującą do silnika NEMA17 o przełożeniu zbliżonym do założeń. Koszt takiego rozwiązania jest jednak bardzo wysoki (na dzień 30.11.2023 \$1051.00) [10].

## Przekładnia cykloidalna

Jest to najbardziej skomplikowana z opisywanych przekładni. Jej praca opiera się na wprawieniu w ruch mimośrodowy koła cykloidalnego. Przekładnia ta jest stosunkowo prosta do zaprojektowania i wykonania w technologii druku 3D. Główną wadą jest duża liczba elementów, a co za tym idzie wysoka cena. Kolejną niedogodnością tej przekładni jest możliwość pojawienia się wibracji przy wysokiej prędkości obrotowej. Wibracje można wyeliminować, dodając drugie koło cykloidalne obrócone o 180°, jednak powoduje to zwiększenie stopnia złożoności całej konstrukcji. Zaletami takiej przekładni są: nieduży rozmiar, niewielki luz międzyzębny oraz bardzo duże przełożenie.

## Przekładnia planetarna

Przekładnie planetarne są jednymi z najczęściej wykorzystywanych współpracujących kół zębatach. Charakteryzują się stosunkowo prostą konstrukcją oraz są niewiele większe od przekładni falowych i cykloidalnych. Największą wadą tego rozwiązania jest stosunkowo duży luz, wynikający z zastosowania przekładni zębatach (luz międzyzębne). Ich zaletami są: niska cena, duża dostępność, prostota wykonania w technologii druku 3D oraz możliwość zawarcia kilku stopni przekładni w jednej obudowie. Pozwala to na znaczne zwiększenie przełożenia takiej przekładni, przy nieznacznym zwiększeniu jej rozmiaru.

## Wnioski

Ze względu na możliwość połączenia kilku stopni w jednej obudowie ostatecznie zdecydowano się na zastosowanie przekładni planetarnej. Umożliwia to zwiększenie jej przełożenia. Kolejną zaletą wykorzystania przekładni planetarnej jest prostota wykonania przekładni w technologii druku 3D.

### 3.2 Elementy elektroniczne

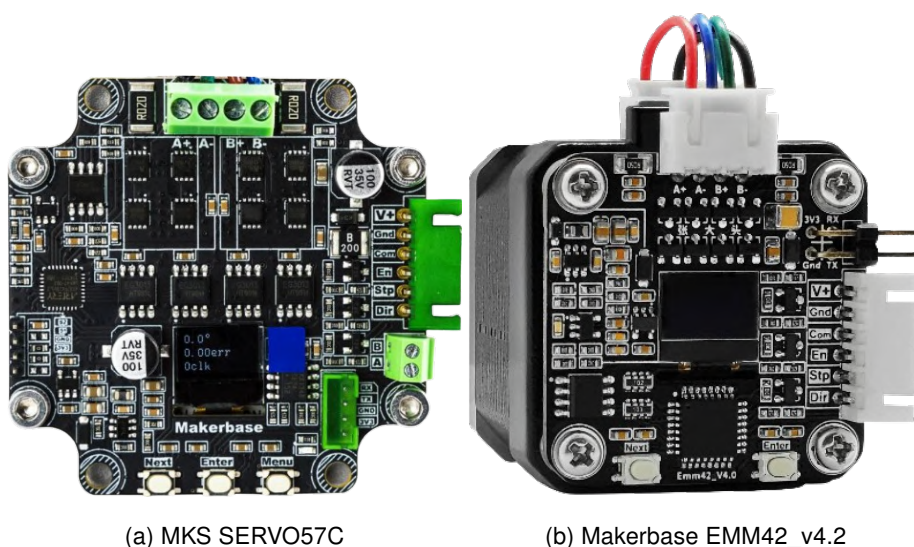
Najważniejszym elementem elektronicznym, który należy dobrać, jest mikrokontroler zarządzający komunikacją pomiędzy silnikami a użytkownikiem. Rozważano trzy rozwiązania: ESP32, Arduino i Raspberry Pi. Kryteria wyboru mikrokontrolera przedstawiono w tabeli 3.2 (podobnie jak w przypadku wyboru materiału konstrukcyjnego, decyduje suma punktów). Zgodnie z wynikami umieszczonymi w tabeli zdecydowano się na wykorzystanie mikrokontrolera ESP32. Następnie należało do niego dobrać komputer odpowiedzialny za liczenie kinematyki odwrotnej. Po analizie dostępnych rozwiązań początkowo zdecydowano się na użycie Raspberry Pi 3B+, jego główną zaletą jest dobry stosunek wydajności do ceny. Niestety w trakcie testów okazało się, że jego wydajność obliczeniowa jest niewystarczająca. Zdecydowano się na jego zastąpienie komputerem PC pracującym pod systemem operacyjnym Ubuntu.

		Możliwe rozwiązania		
Kryterium	Maksymalna liczba pkt	ESP32	Arduino	Raspberry PI
Cena	20	19	15	6
Dostępność	10	5	8	6
Łatwość użycia	20	15	17	18
Ilość wbudowanych standardów komunikacji	20	20	15	10
Suma punktów		59	55	40

Tabela 3.2: Tabela pokazująca proces wyboru mikrokontrolera sterującego silnikami

### 3.2.1 Sterowniki silników

W projekcie zdecydowano się na wykorzystanie sterowników silników krokowych z zamkniętą pętlą sterowania. Wybrano odpowiednio Makerbase EMM42\_v4.2 (rysunek 3.1b) dla silników NEMA 17 oraz MKS SERVO57C (rysunek 3.1a) dla silników NEMA 23. Ich głównymi zaletami są: kompensacja zgubionych kroków oraz możliwość komunikacji po interfejsie UART. Komunikacja po interfejsie UART daje możliwość łatwej zmiany parametrów pracy silników z poziomu programu sterującego. Sterownik MKS SERVO57C umożliwia pracę silnika z prądem do 5,2 A oraz napięciem od 12 V do 24 V. Producent zastosował w tym sterowniku 14-bitowy absolutny enkoder magnetyczny, umożliwiający odczyt aktualnej pozycji wału silnika z dokładnością do  $0,022^\circ$ . Natomiast w przypadku sterownika Makerbase EMM42\_v4.2 możliwa jest praca z prądem do 3 A oraz przy napięciu od 7 V do 27 V. Producent tego sterownika nie podaje dokładności zastosowanego enkodera. Podaje on natomiast dokładność ustalania pozycji, która jest większa od  $0,08^\circ$ . Oba sterowniki umożliwiają sterowanie silnikami za pomocą komend wysyłanych przez interfejs UART lub z użyciem złączy STEP i DIR.



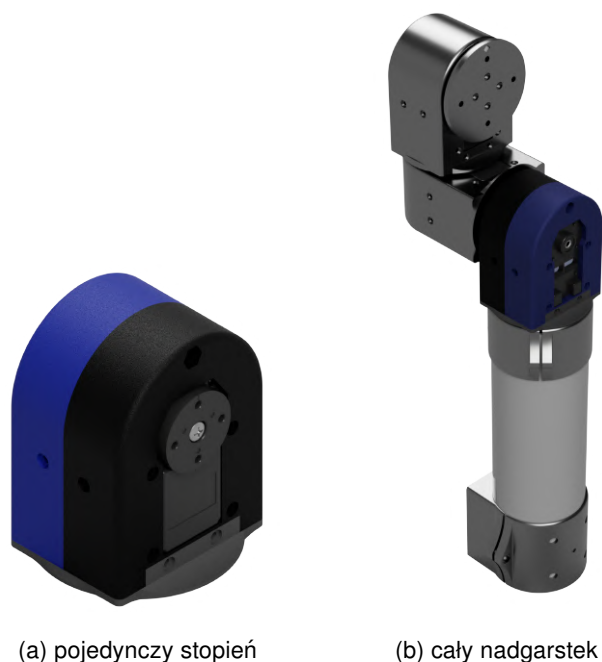
Rysunek 3.1: Zastosowane sterowniki silników

## **4 PROJEKT MECHANICZNY (Michał Blicharz)**

Zgodnie z rozdziałem 2.3 zdecydowano się na zaprojektowanie konstrukcji wzorowanej na kobocie UR5 [5]. Przy projektowaniu dużą wagę przykładano do tego, żeby projekt jednego elementu mógł być wykorzystany w wielu miejscach przy jego niewielkiej modyfikacji. W kolejnych podrozdziałach opisano projekt poszczególnych podzespołów robota. Kolejność podrozdziałów pokazuje kolejne etapy projektu. Projekt konstrukcji mechanicznej został wykonany w programie Autodesk Fusion 360.

### **4.1 Nadgarstek robota (osie 4-6)**

Przy konstrukcji nadgarstka zdecydowano się na wykorzystanie trzech identycznych segmentów, składających się z silnika wraz z obudową służącą do jego montażu. Przy ostatnim stopniu swobody zastosowano dodatkową flanszę umożliwiającą montaż chwytaka. Konstrukcję pojedynczego segmentu przedstawiono na rysunku montażowym 4.1a (dla łatwiejszego odróżnienia elementów, każdy został zaznaczony osobnym kolorem). Na rysunku montażowym 4.1b widać złożenie osi 4-6 wraz ze sposobem montażu do łącznika oraz do wcześniejszych osi.



Rysunek 4.1: Rysunek złożeniowy

## 4.2 Oś 1-3

W osiach 1-3 konieczne jest wykorzystanie redukcji napędu w celu uzyskania odpowiedniego momentu obrotowego. Zdecydowano się na wykorzystanie jednego projektu przekładni planetarnej dla wszystkich trzech osi, w celu uproszczenia projektu i ułatwienia przyszłego serwisowania robota (poszczególne przekładnie różnią się tylko jednym elementem). Zdecydowano się na wykorzystanie przełożenia 38,4:1. Stanowi ono kompromis pomiędzy prędkością obrotową a momentem obrotowym osi robota. Takie przełożenie pozwala uzyskać zakładany udźwig dla wszystkich osi, utrzymując jednocześnie zadowalającą prędkość obrotu osi.

### 4.2.1 Projekt przekładni planetarnej

Przy projektowaniu przekładni zdecydowano się na wykorzystanie zębów pochylonych. Zaletami tego rozwiązania względem zębów prostych są: cichsza praca, większa wytrzymałość na obciążenia wynikająca z jednoczesnej pracy większej liczby zębów (względem zębów prostych), co powoduje mniejsze zużycie pojedynczych zębów oraz możliwość przenoszenia większych momentów, dzięki większej powierzchni styku przylegających zębów. Głównymi wadami tego rozwiązania są: większa złożoność procesu produkcji oraz wyższe tarcie pomiędzy zębami. Pierwszy z tych problemów jest pomijalny w przypadku wykonywania przekładni zębatej w technologii druku 3D. Dla drukarki 3D wykonanie pochylonych zębów nie jest problemem. Zdecydowano się na wykorzystanie przekładni o dwóch stopniach. Sposób połączenia stopni przekładni pokazano na rysunku 4.2. Rozwiązanie to pozwala na zmniejszenie średnicy przekładni przy nieznacznym

zwiększeniu jej długości. Przy projektowaniu przekładni planetarnych musi zostać spełniony wzór:

$$Z_r = Z_s + 2 \cdot Z_p, \quad (4.1)$$

gdzie

- $Z_r$  - liczba zębów pierścienia,
- $Z_s$  - liczba zębów słońca (centralne koło zębate),
- $Z_p$  - liczba zębów planet (koła zębate obracające się wokół słońca).

Przełożenie takiej przekładni opisuje wzór:

$$\omega_{r2} = \frac{\omega_{s1} - \frac{\omega_{s1} \cdot Z_{r1} \cdot Z_{p2}}{Z_{r2} \cdot Z_{p1}}}{1 + \frac{Z_{r1}}{Z_{s1}}}, \quad (4.2)$$

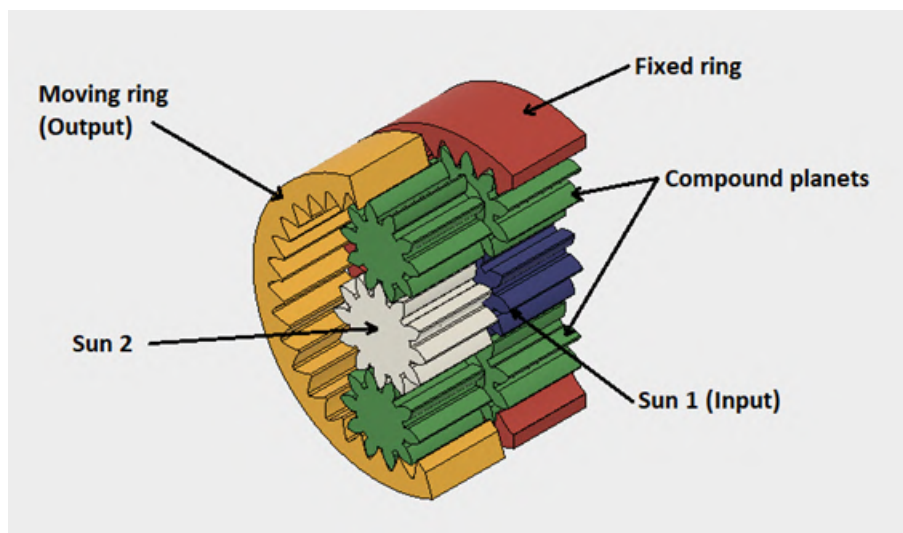
gdzie

- $\omega_{r2}$  - prędkość obrotowa drugiego pierścienia w  $\frac{\text{obr.}}{\text{min}}$  (wyjście przekładni),
- $\omega_{s1}$  - prędkość obrotowa pierwszego słońca w  $\frac{\text{obr.}}{\text{min}}$  (wejście przekładni),
- $Z_{r1}$  - liczba zębów pierwszego pierścienia,
- $Z_{r2}$  - liczba zębów drugiego pierścienia,
- $Z_{p1}$  - liczba zębów planet pierwszego stopnia przekładni,
- $Z_{p2}$  - liczba zębów planet drugiego stopnia przekładni,
- $Z_{s1}$  - liczba zębów słońca pierwszego stopnia przekładni.

Do obliczenia liczby zębów wykorzystano program ze strony JuanGg projects [11]. Otrzymano następujące wyniki:

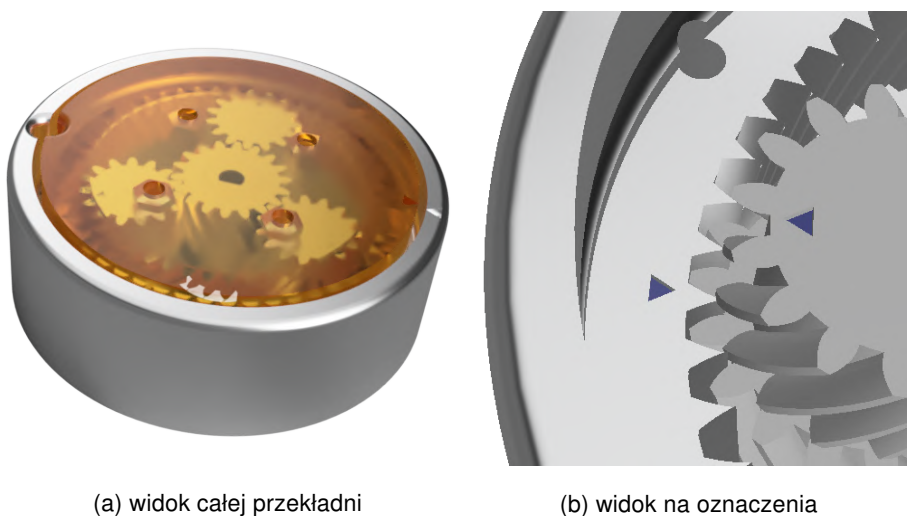
- $Z_{r1} = 51,$
- $Z_{r2} = 48,$
- $Z_{p1} = 18,$
- $Z_{p2} = 15,$
- $Z_{s1} = 15,$
- $Z_{s2} = 18.$

Następnie sprawdzono poprawność otrzymanych wyników ze wzorami 4.1 oraz 4.2. Dla ułatwienia obliczeń za  $\omega_{s1}$  przyjęto 1. Pierwsze równanie zostało spełnione dla obu stopni, co oznacza, że spełniają one podstawowe założenie (wzór 4.1) umożliwiające prawidłowe zaprojektowanie przekładni planetarnej. Z równania 4.2 otrzymano wynik  $\omega_{r2} = 0,026 \frac{\text{obr.}}{\text{min}}$ . Oznacza to, że dobrane liczby zębów spełniają założenie oraz że przekładnia ta jest możliwa do wykonania.



Rysunek 4.2: Schematyczne przedstawienie przekładni [12]

Jako łożyska pomiędzy obudową przekładni a jej wyjściem zdecydowano się użyć plastikowych kulek o średnicy 6,2 mm. Na zdjęciu 4.3a pokazano finalny projekt przekładni wykorzystanej w projekcie. W celu łatwiejszego montażu przekładni zastosowano strzałki pokazujące poprawne położenie planet względem pierścienia (zdjęcie 4.3b).



(a) widok całej przekładni

(b) widok na oznaczenia

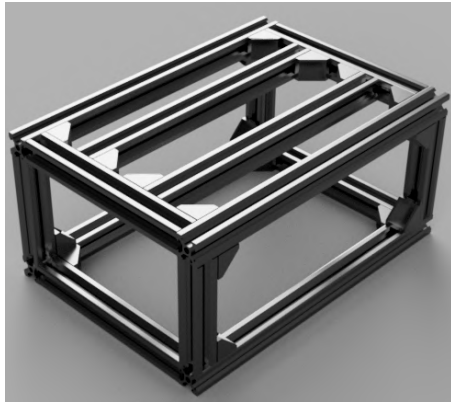
Rysunek 4.3: Projekt przekładni wykorzystanej w projekcie

Na rysunku 4.2 kolorem niebieskim zaznaczono słońce pierwszego stopnia przekładni, które jest nałożone na wał silnika i jest wejściem wykorzystanej przekładni. Kolorem czerwonym zaznaczono nieruchomy pierścień pierwszego stopnia. Na zielono zaznaczono planety przekładni, stanowią one połączenie pomiędzy pierwszym a drugim jej stopniem. Wyjście przekładni, które jest pierścieniem drugiego stopnia, zaznaczono kolorem pomarańczowym. Słońce drugiego stopnia przekładni zaznaczono kolorem białym. Obraca się ono swobodnie i jego zadaniem jest utrzymanie odpowiedniego położenia planet.

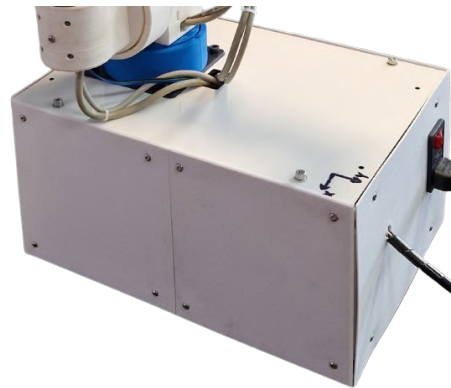


### 4.3 Podstawa robota

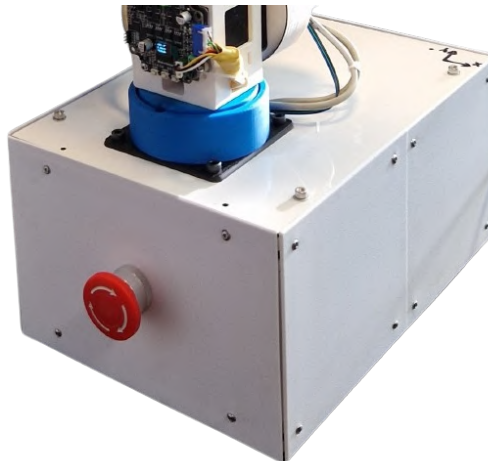
Szkielet podstawy robota postanowiono wykonać z profili aluminiowych 20 mm na 20 mm typu V-slot. Główną zaletą takich profili jest ich stosunkowo niewielka cena, modułowość oraz duża wytrzymałość. Konstrukcję stelaża przedstawiono na rysunku 4.4a. W górnej części podstawy zastosowano dwa dodatkowe profile umożliwiające pewny montaż pierwszej osi robota. Cała podstawa została ze wszystkich stron zamknięta panelami w celu uniemożliwienia dostępu do wnętrza konstrukcji. Na panelach tych umieszczono: gniazdo C14 z bezpiecznikiem i włącznikiem, wyłącznik bezpieczeństwa oraz wyprowadzono przewody umożliwiające sterowanie robotem za pomocą komputera. Od wewnętrznej strony do dolnego panelu został przymocowany zasilacz, podstawa pod elektronikę sterującą oraz puszka z przetwornicami. Na rysunkach 4.4c oraz 4.4b pokazano zdjęcia zbudowanej podstawy robota.



(a) projekt 3D



(b) fizyczna realizacja (widok od tyłu)

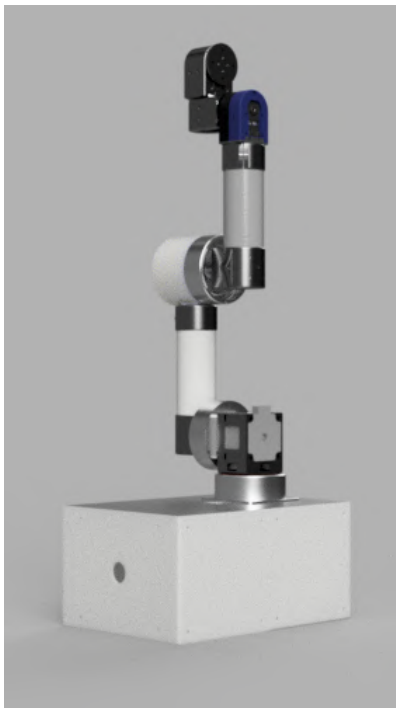


(c) fizyczna realizacja (widok od przodu)

Rysunek 4.4: Podstawa robota

#### 4.4 Wykonanie

Większą część elementów robota wykonano w technologii druku 3D z materiału PLA na drukarkach Bambu Lab X1C oraz Bambu Lab P1S [13]. Niektóre z elementów wykonane zostały w kilku wstępnych wersjach przed ustaleniem ostatecznego kształtu i wymiarów. Wynikało to m.in. z niedoskonałości innych użytych materiałów. Na zdjęciu 4.5 pokazano finalny projekt robota.



Rysunek 4.5: Finalny projekt robota

#### 4.5 Wnioski

W tabeli 4.1 przedstawiono parametry efektorów poszczególnych osi robota. Masa finalnego robota wynosi około 5 kg. Podstawa wykonana z profili aluminiowych jest wystarczająco ciężka, żeby umożliwić robotowi pracę bez stałego zamocowania do podłoża. W celu poprawy funkcjonalności i estetyki robota planowane jest w przyszłości zastąpienie przewodów wychodzących z robota portami.

Numer osi	Wybrany silnik	Moment obrotowy silnika [Nm]	Masa silnika [g]	Redukcja napędu	Moment obrotowy po redukcji [Nm]	Masa całości [g]
1	Nema 17	0,44	269	38,4	16,9	381
2	Nema 23	1,06	680	38,4	40,7	792
3	Nema 17	0,44	269	38,4	16,9	381
4	Dynamixel AX-12A	1,5	54,6	1	1,5	54,6
5	Dynamixel AX-12A	1,5	54,6	1	1,5	54,6
6	Dynamixel AX-12A	1,5	54,6	1	1,5	54,6

Tabela 4.1: Parametry efektorów kolejnych osi robota (wyniki zaokrąglono do dwóch miejsc po przecinku)

## 5 PROJEKT ELEKTRONICZNY (Michał Blicharz)

W tym rozdziale omówiono projekt elektroniczny robota, wykorzystane rozwiązania i sposób ich działania. Dodatkowo zaprezentowano problemy, które pojawiły się w trakcie testów oraz możliwe ich rozwiązania.

### 5.1 Sterowniki silników

Początkowo do komunikacji z silnikami planowano wykorzystać wyłącznie interfejs UART. Jednak podczas testów i prób implementacji ruchu z interpolacją osiową okazało się, że ten sposób komunikacji jest niewystarczający. Z tego powodu konieczna okazała się dodatkowa implementacja sterowania z użyciem złączy STEP/DIR. Więcej szczegółów związanych z problemami z komunikacją ze sterownikami zawarto w podrozdziale 7.1.1. Głównym problemem był brak kompletnej i szczegółowej dokumentacji technicznej.

### 5.2 Zasilacze

W projekcie należało zastosować trzy różne napięcia zasilające: 24 V dla silników NEMA, 11,1 V dla serwomechanizmów Dynamixel AX-12A oraz 5 V dla mikrokontrolera i układów na płytce PCB. Zdecydowano się na zastosowanie jednego zasilacza 24 V oraz dwóch przetwornic typu bulk. W tabeli 5.1 przedstawiono parametry elektryczne silników użytych w projekcie.

numer osi robota	1	2	3	4	5	6
maksymalny pobór prądu [A]	1,75	2	1,75	1,62	1,62	1,62
napięcie zasilania [V]	24	24	24	11,1	11,1	11,1
maksymalna moc [W]	42	48	42	18	18	18

Tabela 5.1: Parametry zasilania poszczególnych osi robota (zaokrąglone do 2 miejsc po przecinku)

Moc pobierana przez mikrokontroler oraz układy na płytce PCB jest pomijalna przy doborze zasilacza. Przyjęto sprawność przetwornicy z 24 V na 11,1 V, na poziomie 90%. Po uwzględnieniu tego założenia wyliczono, że minimalna moc zasilacza powinna wynosić 186 W. Zdecydowano się na wykorzystanie zasilacza HRP-200-24 firmy MEAN WELL [14] o nominalnej mocy wyjściowej

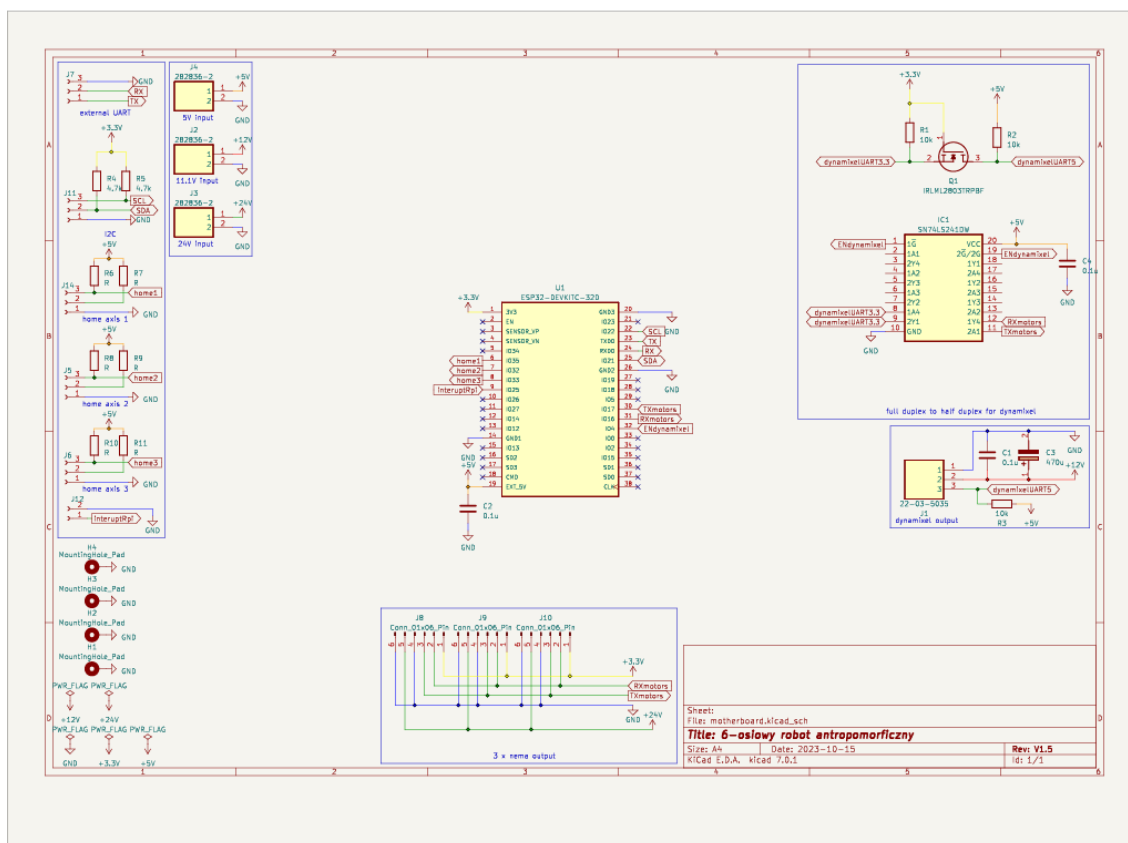
200 W. Jako przetwornice bulk wykorzystano układy Solectro XL4016 [15] dla 11,1 V oraz Solectro LM2596 [16] dla 5 V. Początkowo planowano wykorzystać dwa moduły Solectro LM2596, oparte na układzie LM2596S-ADJ, jednak maksymalny prąd oferowany przez ten układ wynosi 3 A. Nie jest to wystarczające, gdyż maksymalny prąd, pobierany na linii 11,1 V, wynosi 4,86 A. Z tego powodu zdecydowano się na wykorzystanie większego modułu Solectro XL4016 opartego na układzie XL4016 [17], którego maksymalny prąd obciążenia wynosi 8 A. Jest to wartość w pełni wystarczająca do pracy w projektowanym układzie. Na rysunku 5.1 pokazano zdjęcia wykorzystanego zasilacza oraz przetwornic. W celu zapewnienia łatwiejszego transportu zastosowano zamontowane na obudowie złącze C14 zintegrowane z bezpiecznikiem i wyłącznikiem zasilania.



Rysunek 5.1: Wykorzystany zasilacz i przetwornice

### 5.3 Schemat połączeń elektronicznych

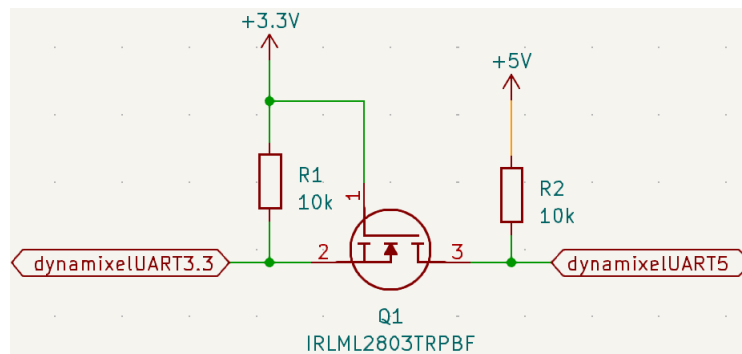
Do zaprojektowania schematu połączeń elektronicznych wykorzystano program KiCad [19]. Na schemacie 5.2 pokazano projekt elektroniczny. Na środku schematu widać wykorzystany mikrokontroler, dookoła niego pokazano wszystkie wykorzystane elementy ułożone w blokach funkcyjnych. Najbardziej rozbudowanym i skomplikowanym blokiem jest blok odpowiedzialny za komunikację z serwomechanizmami Dynamixel AX-12A.



Rysunek 5.2: Schemat obwodu elektrycznego

### 5.3.1 Komunikacja z serwomechanizmami Dynamixel AX-12A

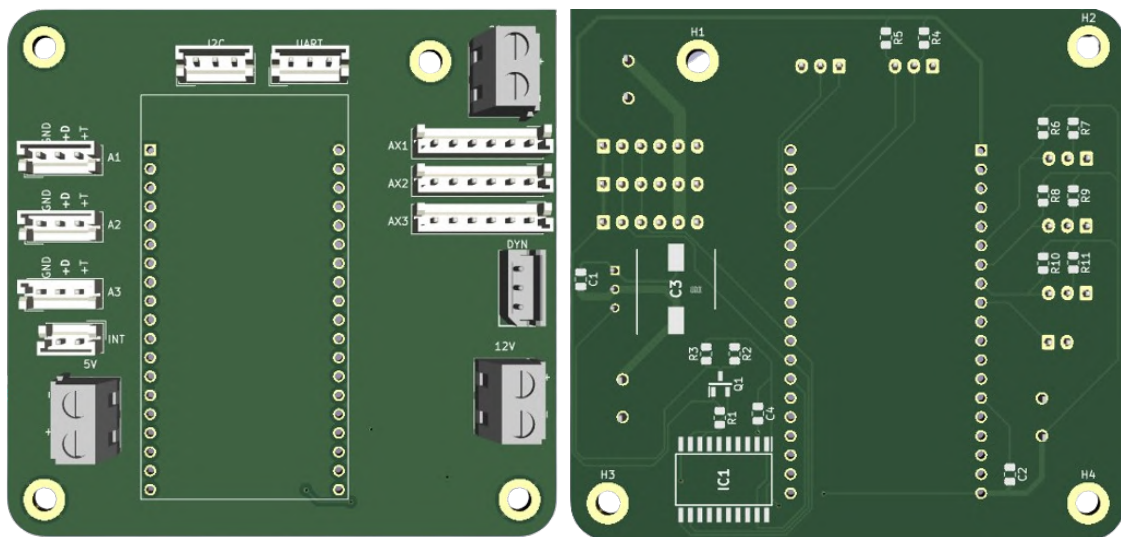
Do komunikacji z serwomechanizmami konieczne było wykonanie układu umożliwiającego zamianę interfejsu UART w trybie full-duplex, używanego do komunikacji pomiędzy mikrokontrolerem a sterownikami silników NEMA, na interfejs UART w trybie half-duplex, który jest obsługiwany przez serwomechanizmy. W tym celu wykorzystano bufor trójstanowy SN74LS241DW [20], do którego, oprócz standardowych linii RX i TX wykorzystywanych przez interfejs UART, doprowadzono sygnał *enable*, określający aktualny kierunek komunikacji. Konieczne było zastosowanie konwertera stanów logicznych. Konwerter postanowiono zrealizować z wykorzystaniem tranzystora MOSFET typu N (IRML2803TRPBF [21]) w układzie wspólnej bramki. Głównymi kryteriami doboru tranzystora użytego w tym układzie były czas przełączania oraz napięcie pracy. Układ do konwersji stanów logicznych przedstawiono na schemacie 5.3. Pozwala on na zamianę poziomu napięć w obu kierunkach ( $3,3\text{ V} \rightarrow 5\text{ V}$  oraz  $5\text{ V} \rightarrow 3,3\text{ V}$ ). W celu minimalizacji rozmiarów układu zamiana poziomu napięć następuje bezpośrednio na linii wychodzącej do serwomechanizmów.



Rysunek 5.3: Układ wykorzystywany do konwersji stanów logicznych

#### 5.4 Wykonanie płytki PCB

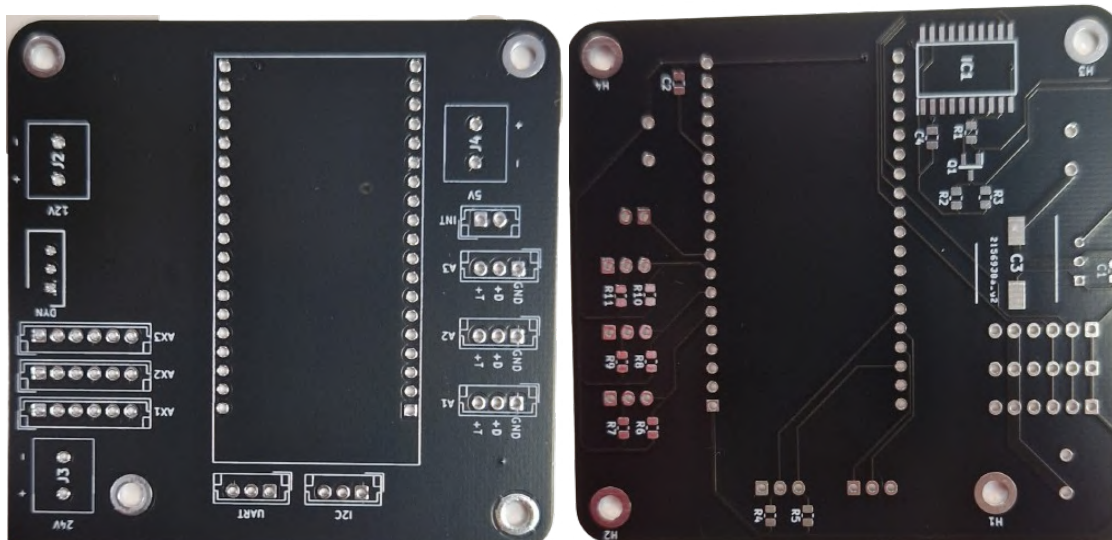
Po wykonaniu schematu połączeń elektronicznych należało zaprojektować oraz wykonać fizyczną realizację połączeń. Przy projektowaniu płytki bardzo dużą uwagę przykładano do poprawnego umiejscowienia kondensatorów przeciwzakłóceńowych. Na rysunku 5.4 pokazano projekt płytki PCB, natomiast na rysunku 5.5 widoczna jest jej fizyczna realizacja.



(a) przednia strona zaprojektowanej płytki

(b) tylna strona zaprojektowanej płytki

Rysunek 5.4: Rendery 3D przedstawiające zaprojektowaną płytkę



(a) przednia strona zaprojektowanej płytki

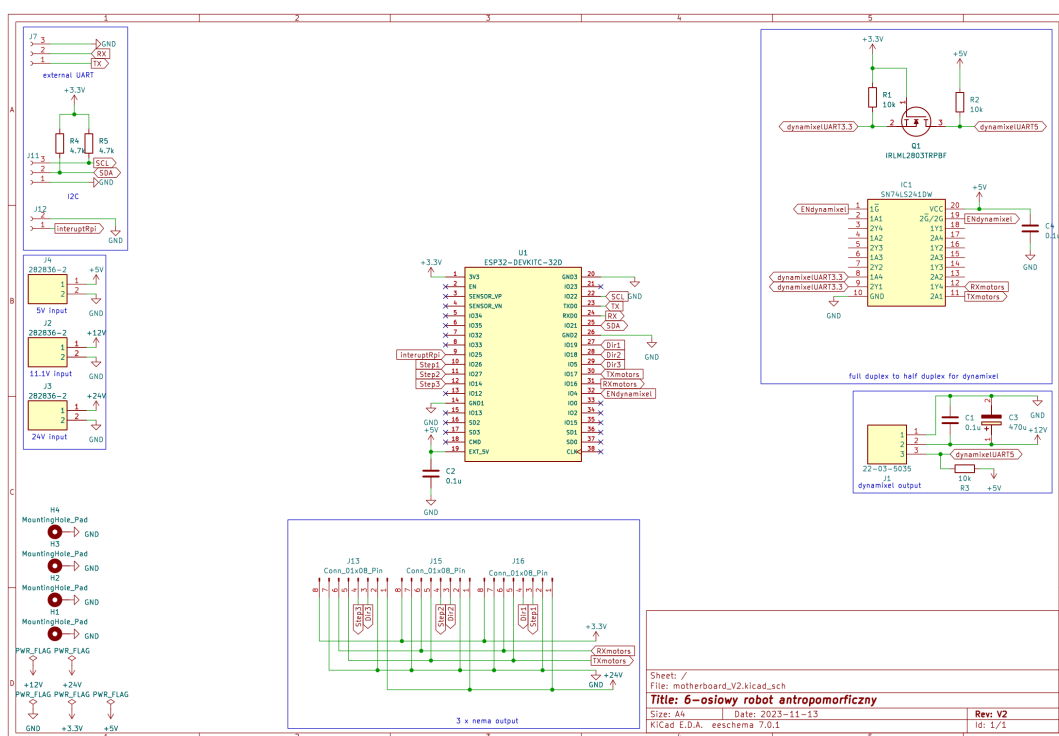
(b) tylnia strona zaprojektowanej płytki

Rysunek 5.5: Zdjęcia przedstawiające zaprojektowaną płytkę

## 5.5 Wnioski i perspektywy dalszego rozwoju

W trakcie testów okazało się, że konieczne jest dodanie obsługi interfejsu STEP/DIR dla pierwszych trzech osi. Nastąpiło to jednak już po wysłaniu projektu do produkcji, zbyt późno, aby wykonać nową płytkę PCB. Z tego powodu zdecydowano się na wykorzystanie płytki przedstawionej na rysunku 5.5 z dodanymi połączeniami do interfejsu STEP/DIR. Na rysunku 5.6 przedstawiono poprawiony schemat połączeń. W dalszej perspektywie możliwe jest zamówienie nowej wersji płytki. Co więcej, w robocie uwzględnione jest wewnętrzne zasilanie 5 V, które aktualnie nie jest używane, ponieważ obecnie robot może pracować jedynie podłączony do komputera PC, który zapewnia zasilanie układów sterujących. Jeżeli zdecydowano by się na dodanie do robota wewnętrznego mikrokomputera, zastępującego aktualnie używany komputer PC, jest to możliwe bez wprowadzania dużych modyfikacji w układzie zasilania.





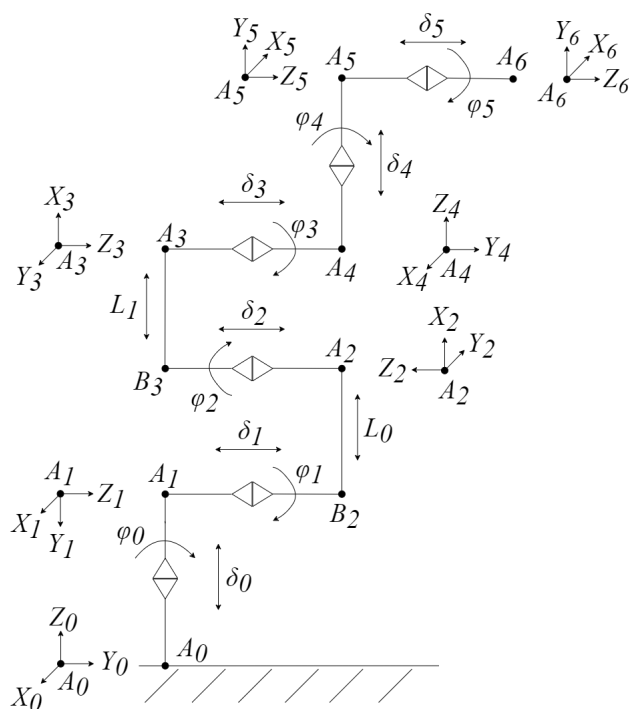
Rysunek 5.6: Zaktualizowana wersja schematu elektronicznego

## 6 KINEMATYKA ODWROTNA (Jan Górski)

W celu efektywnego sterowania położeniem końcówki robota niezbędne jest wyznaczenie jego kinematyki odwrotnej. Kinematyka odwrotna to matematyczny proces wyznaczania wartości kątów poszczególnych przegubów manipulatora tak, aby uzyskać zadane położenie i orientację jego końcówki (narzędzia).

### 6.1 Kinematyka prosta manipulatora

Pierwszym etapem rozwiązania problemu kinematyki odwrotnej jest wyznaczenie łańcucha kinematycznego konstrukcji oraz jego równań kinematyki prostej. Do przedstawienia zależności łańcucha kinematycznego zdecydowano się użyć notacji Denavita-Hartenberga (D-H). Pozwala ona w łatwy sposób uzyskać równania kinematyki prostej. Rysunek 6.1 przedstawia schemat łańcucha kinematycznego projektowanego robota.



Rysunek 6.1: Łańcuch kinematyczny manipulatora

Następnie należy wyznaczyć parametry D-H opisujące łańcuch kinematyczny robota. Za ich pomocą można skonstruować macierz transformacji reprezentującą przejścia geometryczne między kolejnymi przegubami. Parametry zostały przedstawione w tabeli 6.1.

$i \rightarrow i + 1$	$\varphi'_i$	$\delta'_i$	$\lambda'_i$	$\alpha'_i$
$0 \rightarrow 1$	$\varphi_0$	$\delta_0$	0	$-\frac{\pi}{2}$
$1 \rightarrow 2$	$\varphi_1 - \frac{\pi}{2}$	$\delta_1$	$L_0$	$\pi$
$2 \rightarrow 3$	$\varphi_2$	$\delta_2$	$L_1$	$\pi$
$3 \rightarrow 4$	$\varphi_3 + \frac{\pi}{2}$	$\delta_3$	0	$\frac{\pi}{2}$
$4 \rightarrow 5$	$\varphi_4 + \pi$	$\delta_4$	0	$\frac{\pi}{2}$
$5 \rightarrow 6$	$\varphi_5$	$\delta_5$	0	0

Tabela 6.1: Tabela parametrów D-H manipulatora

Oznaczenia kolumn tabeli:

- $i \rightarrow i + 1$  - numeracja transformacji geometrycznych wzdłuż kolejnych części łańcucha kinematycznego,
- $\varphi'_i$  - obrót wokół osi  $OZ_i$  lokalnego układu współrzędnych,
- $\delta'_i$  - przesunięcie wzdłuż osi  $OZ_i$  lokalnego układu współrzędnych,
- $\lambda'_i$  - przesunięcie wzdłuż osi  $OX_{i+1}$  lokalnego układu współrzędnych,
- $\alpha'_i$  - obrót wokół osi  $OX_{i+1}$  lokalnego układu współrzędnych.

Wymiary manipulatora:

- $\delta_0 = |A_0 A_1| = 77,5 \text{ mm}$ ,
- $\delta_1 = |A_1 B_2| = 96,4 \text{ mm}$ ,
- $L_0 = |B_2 A_2| = 196,2 \text{ mm}$ ,
- $\delta_2 = |A_2 B_3| = 83,6 \text{ mm}$ ,
- $L_1 = |B_3 A_3| = 192,5 \text{ mm}$ ,
- $\delta_3 = |A_3 A_4| = 62,4 \text{ mm}$ ,
- $\delta_4 = |A_4 A_5| = 62,4 \text{ mm}$ ,
- $\delta_5 = |A_5 A_6| = 25,9 \text{ mm}$ .

Macierz transformacji między bazą a końcówką robota można wyznaczyć za pomocą następujących równań.

$$R_{Z_i} = \begin{bmatrix} \cos \varphi'_i & -\sin \varphi'_i & 0 & 0 \\ \sin \varphi'_i & \cos \varphi'_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.1)$$

$$T_{Z_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta'_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.2)$$

$$T_{X_i} = \begin{bmatrix} 1 & 0 & 0 & \lambda'_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.3)$$

$$R_{X_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha'_i & -\sin \alpha'_i & 0 \\ 0 & \sin \alpha'_i & \cos \alpha'_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.4)$$

$$H_i^{i+1} = R_{Z_i} \cdot T_{Z_i} \cdot T_{X_i} \cdot R_{X_i} \quad (6.5)$$

$$H_0^6 = H_0^1 \cdot H_1^2 \cdot H_2^3 \cdot H_3^4 \cdot H_4^5 \cdot H_5^6 \quad (6.6)$$

$$H_0^6 = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & T_x \\ R_{yx} & R_{yy} & R_{yz} & T_y \\ R_{zx} & R_{zy} & R_{zz} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.7)$$

$$T_0^6 = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (6.8)$$

$$\begin{aligned} R_0^6 &= \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{bmatrix} = R_Z \cdot R_Y \cdot R_X = \\ &= \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \psi & 0 & \sin \psi \\ 0 & 1 & 0 \\ -\sin \psi & 0 & \cos \psi \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{bmatrix} \end{aligned} \quad (6.9)$$

Z równań 6.8 i 6.9 można wyznaczyć równania kinematyki prostej. Przedstawiono je w sekcji: *Załącznik nr 1: Równania kinematyki prostej*.

## 6.2 Charakterystyka problemu

Otrzymane równania kinematyki prostej są bardzo złożone i trudne do analitycznego przekształcenia w równania kinematyki odwrotnej. Z tego powodu zdecydowano się na metodę rozwiązywania iteracyjnego, która poprzez minimalizację błędu położenia efektora pozwoli na znalezienie wymaganych wartości kątów dla poszczególnych przegubów manipulatora. Zazwyczaj jednak algorytmy iteracyjne mają problem ze znalezieniem rozwiązania w punktach osobliwych kinematyki manipulatora. Nie są też one w stanie wykryć, czy zadany punkt jest osiągalny. Po zadaniu punktu poza zasięgiem manipulatora algorytm nigdy nie będzie w stanie zminimalizować funkcji błędu do żądanej wielkości. Dodatkowo metody iteracyjne są wyraźnie bardziej kosztowne obliczeniowo w porównaniu do rozwiązań analitycznych. Każdorazowe żądanie rozwiązania kinematyki odwrotnej oznacza wykonanie od kilku do kilkudziesięciu iteracji pętli algorytmu. Rozwiązania iteracyjne, zależnie od warunków początkowych, pozwalają na osiągnięcie jednego punktu w przestrzeni na kilka sposobów. Źle dobrane warunki początkowe mogą uniemożliwić prawidłowe działanie algorytmu.

## 6.3 Dobór rozwiązania

Istnieje wiele możliwości rozwiązania tego problemu metodami iteracyjnymi [22] [23]. Zdecydowano się na użycie metody Newtona-Raphsona do minimalizacji błędu położenia efektora. Co więcej, w celu uniknięcia problemów związanych z odwracalnością jakobianu, zastosowano w zamian macierz pseudoodwrotną Moore'a-Penrose'a jakobianu. Jej użycie gwarantuje odwracalność. Wadą tego rozwiązania jest konieczność zapewnienia dobrych warunków początkowych. Zbyt oddalone od optimum uniemożliwią jego prawidłowe znalezienie. Inną wadą tego podejścia jest niestabilna praca w punktach osobliwych kinematyki. Jako że są one rzadko osiągalne w praktyce, a algorytm ten zapewnia znaczną prostotę i szybkość implementacji, zdecydowano się na jego użycie.

## 6.4 Algorytm rozwiązania

Algorytm metody numerycznej minimalizacji Newtona-Raphsona [24]:

- punkt zadany:

$$P_d = \begin{bmatrix} x_d & y_d & z_d & \varphi_d & \psi_d & \theta_d \end{bmatrix}^T \quad (6.10)$$

- początkowe wartości kątów przegubów:

$$\phi^i = \begin{bmatrix} \varphi_0^i & \varphi_1^i & \varphi_2^i & \varphi_3^i & \varphi_4^i & \varphi_5^i \end{bmatrix}^T \quad (6.11)$$

- dopóki  $|E| > \epsilon$ :

- oblicz błąd położenia:

$$E = P_d - f(\phi^i) \quad (6.12)$$

- wyznacz macierz pseudoodwrotną Moore'a-Penrose'a jacobianu funkcji kinematyki prostej:

$$J^\dagger(\phi^i) = J(\phi^i)^T \cdot (J(\phi^i) \cdot J(\phi^i)^T)^{-1} \quad (6.13)$$

- oblicz nowe wartości kątów przegubów:

$$\phi^{i+1} = \theta^i + J^\dagger(\phi^i) \cdot E \quad (6.14)$$

- inkrementuj licznik iteracji:

$$i++ \quad (6.15)$$

## 7 OPROGRAMOWANIE (Jan Górski)

Ze względu na użycie mikrokontrolera ESP32, zdecydowano się napisać oprogramowanie robota w języku programowania C z wykorzystaniem FreeRTOS oraz platformy programistycznej ESP IDF. Wybór ten pozwolił na pełne wykorzystanie zasobów mikrokontrolera oraz na pewność wykonywania krytycznych czasowo instrukcji. Innymi rozważanymi językami programowania były: C++ dla platformy Arduino i MicroPython. Pierwszy pozwala na szybkie i proste pisanie programów, ale nie daje pełnej kontroli nad działaniem poszczególnych bibliotek napisanych przez innych autorów. Drugi, jako język interpretowany, jest wyraźnie wolniejszy i dodatkowo nie wspiera wszystkich funkcji wielowątkowości ESP32. Jest za to łatwiejszy w użyciu niż język C, dzięki prostszej składni i dynamicznym typach zmiennych.

### 7.1 Komunikacja z silnikami

#### 7.1.1 Silniki krokowe

Silniki krokowe NEMA 17 i NEMA 23 sterowane są kolejno poprzez sterowniki EMM42\_v4.2 i MKS SERVO57C. Oba sterowniki działają na tej samej zasadzie. Jedyna różnica w sterowaniu to składnia poszczególnych komend. Sterowniki działają w trybie *vFOC* [25] [26], umożliwiającym sterowanie za pomocą sygnałów STEP/DIR z równoczesną kontrolą parametrów silnika poprzez interfejs UART. Dzięki pętli sprzężenia zwrotnego z enkodera magnetycznego umożliwiają kontrolę i korekcję zgubionych kroków oraz automatyczne dopasowanie prądu zasilającego silniki.

Początkowo zamierzano sterować silnikami tylko i wyłącznie w trybie UART. Niestety w trakcie prac nad projektem okazało się, że tryb ten nie umożliwia pełnej kontroli nad prędkościami i przyspieszeniami silników. Producent założył, że prędkość silników będzie regulowana za pomocą liczby określającej bieg silnika. Nie sprecyzował jednak dokładnej prędkości przypisanej do danego biegu. Zauważono, że prędkość silnika zależy od obciążenia na wale silnika oraz zastosowanego przyspieszenia. Użytkownik nie ma możliwości dokładnego sterowania parametrami. Z tego powodu zdecydowano się na użycie trybu *vFOC*.

Sterowanie STEP/DIR pozwala na uzyskanie pełnej kontroli nad ruchem silników. Sygnał STEP jest sygnałem prostokątnym taktującym wykonywanie kroków. Sygnał DIR determinuje kierunek obrotu wału silnika. Stan niski oznacza kierunek zgodny z kierunkiem obrotu wskazówek

zegara, a wysoki — przeciwny. Zmiany szybkości obrotu silnika uzyskano poprzez zmianę okresu (czasu trwania jednego kroku) sygnału STEP. W bibliotece obsługującej sterowniki realizacja tego trybu pracy wygląda następująco. Każdy z silników ma przypisany do siebie zegar mikrokontrolera. Każdy z tych zegarów generuje przerwanie z okresem równym okresowi sygnału STEP. W funkcji obsługi przerwania następuje generacja nowego impulsu oraz dekrementacja licznika kroków pozostałych do końca ruchu. Umożliwia to równoległą pracę silników i innych funkcji, np. odczytu położenia z enkodera.

W celu zapewnienia płynnego ruchu silników niezbędne było zaimplementowanie kontroli przyspieszeń. Interpolacja osiowa wymaga pełnej kontroli nad całkowitym czasem ruchu poszczególnych silników z uwzględnieniem przyspieszeń i hamowań. Każdy z silników ma do pokonania pewną odległość kątową wyrażoną daną liczbą kroków  $n_c$  (inną dla każdej osi robota). Założono, że czas trwania fazy przyspieszania i hamowania  $t_0$  ma trwać po  $a\% = 10\%$  całkowitego czasu trwania ruchu  $t_c$  (ruchu z zadaną prędkością  $\omega_{goal}$  [RPM], liczonego bez fazy przyspieszania i hamowania). Przyspieszenie  $a$  jest równe zmianie prędkości od zera do prędkości zadanej  $\omega_{goal}$  w czasie  $t_0$ . Wyliczona zostaje liczba kroków  $n_0$ , które zostaną wykonane z wyznaczonym przyspieszeniem  $a$  w zadanym czasie  $t_0$ . Następnie, w celu uproszczenia algorytmu sterowania, obliczona zostaje zmiana prędkości  $\Delta\omega$  tak, aby była ona stała w każdym wykonanym kroku. W fazie przyspieszania, z każdym wykonanym krokiem prędkość jest zwiększana o wyliczoną wartość  $\Delta\omega$ . Analogiczne obliczenia są przeprowadzane dla fazy hamowania. Dzięki temu uzyskano eksponentialną krzywą przyspieszenia, a całkowity czas ruchu stanowi 120% czasu ruchu  $t_c$  (z zadaną prędkością bez uwzględnienia przyspieszania i hamowania).

Poniżej przedstawiono równania używane do obliczeń przyspieszeń ( $t_{goal}$  - czas trwania jednego kroku wykonywanego z prędkością zadaną  $\omega_{goal}$ ;  $\Delta t$  - zmiana okresu wykonywania kroków w fazie przyspieszania i hamowania).

$$t_{goal} = \frac{1}{\omega_{goal}} \left[ \frac{min}{obr.} \right] \quad (7.1)$$

$$t_c = t_{goal} \cdot n_c \left[ \frac{min}{obr.} \right] \quad (7.2)$$

$$t_0 = a\% \cdot t_c \left[ \frac{min}{obr.} \right] \quad (7.3)$$

$$a = \frac{\omega_{goal}}{t_0} \left[ \frac{obr.^2}{min^2} \right] \quad (7.4)$$

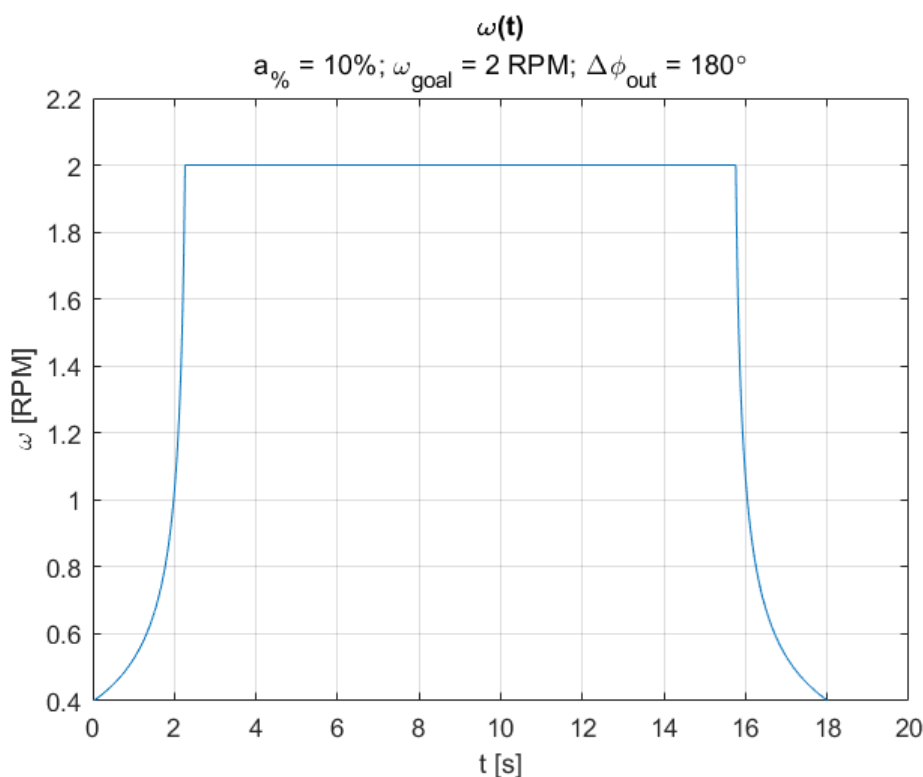
$$n_0 = \frac{a \cdot t_0^2}{2} [-] \quad (7.5)$$

$$\Delta t = \frac{t_0}{n_0} \cdot 2 \left[ \frac{min}{obr.} \right] \quad (7.6)$$

$$\Delta\omega = \frac{1}{\Delta t} \left[ \frac{obr.}{min} \right] \quad (7.7)$$



Na rysunku 7.1 przedstawiono wykres prędkości silnika za przekładnią  $\omega$  w funkcji czasu  $t$  dla fazy przyspieszenia  $a_{\%} = 10\%$ , prędkości zadanej  $\omega_{goal} = 2 \text{ RPM}$  i obrotu wyjścia przekładni o  $\Delta\phi_{out} = 180^\circ$ .



Rysunek 7.1: Wykres prędkości silników krokowych

Powyższy sposób realizacji przyspieszeń pozwala na uzyskanie prostoty implementacji programowej. W fazie przyspieszenia, z każdym wykonanym krokiem, zmieniany jest okres wywoływania przerwania o tę samą wartość. Zmniejsza to liczbę obliczeń wykonywanych w funkcji obsługi przerwania w stosunku do podejścia z przyspieszeniem liniowym. Wadą zaimplementowanego rozwiązania jest występowanie dużych zmian prędkości w końcowej fazie przyspieszania i w początkowej fazie hamowania. Dla dłuższych odcinków drogi przyspieszenie będzie stosunkowo małe, podczas gdy dla krótkich odcinków przyspieszenie będzie bardziej gwałtowne. W celu uzyskania wspomnianego przyspieszenia liniowego należałoby zmieniać wartość okresu przerwania o wartość zależną od pozostałej do wykonania liczby kroków w fazie przyspieszenia. Zwiększałoby to poziom złożoności algorytmu.

### 7.1.2 Serwomechanizmy

Serwomechanizmy Dynamixel AX-12A sterowane są w pełni poprzez interfejs UART. Bibliotekę dla mikrokontrolera ESP32 napisano w języku C. Poszczególne instrukcje sformułowane są w protokole Dynamixel Protocol 1.0 [9]. Za ich pomocą można zadawać każdemu serwome-

chanizmowi pozycję docelową oraz prędkość, z jaką ma się obracać. Istnieje również możliwość odczytu aktualnej pozycji wału wraz z jego obciążeniem. Serwomechanizm zwraca również kody ewentualnych błędów, umożliwiając szybką diagnostykę problemów. Prędkość silnika zadaje się jako wielokrotność prędkości minimalnej równej 0,111 RPM, przy czym maksymalna prędkość wynosi około 114 RPM. Z tego względu, w tych silnikach niemożliwa jest realizacja przyspieszenia działającego w reżimie oczekiwanego przez użytkownika czasu trwania ruchu.

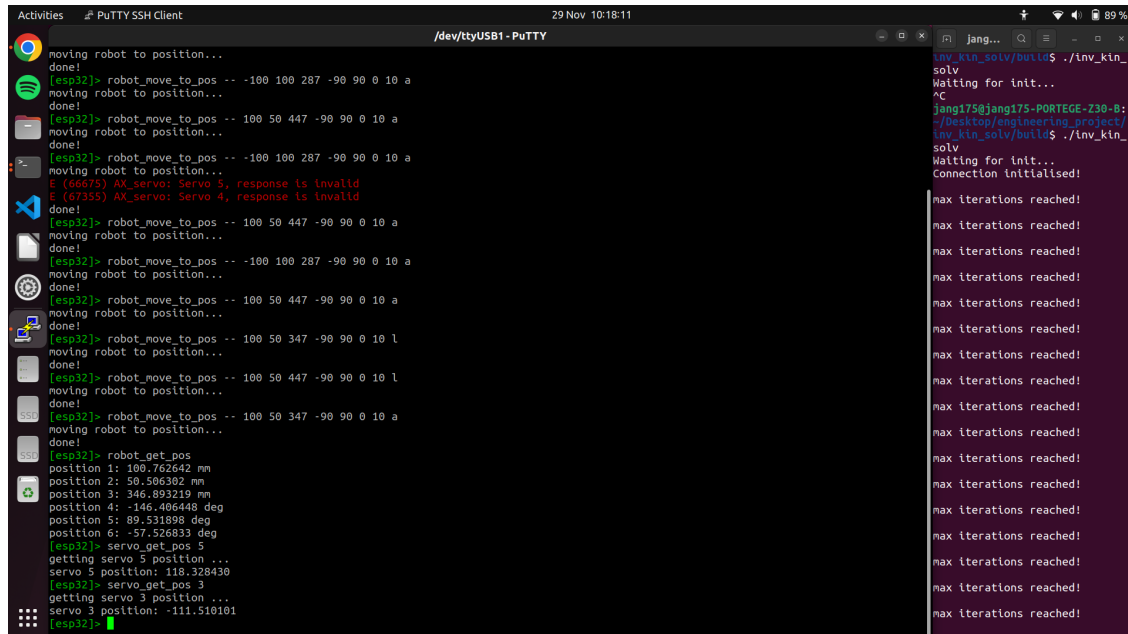
## 7.2 Interpreter rozkazów

Interakcja użytkownika z robotem odbywa się poprzez terminal komputera. Mikrokontroler ESP32, za pomocą biblioteki Console [27], obsługuje terminalowe komendy zdefiniowane wcześniej w programie. Wyświetlane są przy nich również odpowiedzi dla użytkownika ułatwiające ich obsługę. Ze względu na rozpoznawanie tzw. *escape sequences* preferowanym terminalem do sterowania robotem jest Putty [28].

Lista komend:

- **help** - wyświetla listę dostępnych komend i ich opisy,
- **servo\_move <DOF> <pos> <rpm>** - przesunąć oś robota o numerze <DOF> do pozycji kątownej (w stopniach) <pos> z prędkością <rpm> (w *RPM*),
- **servo\_get\_pos <DOF>** - wyświetli aktualną pozycję osi robota o numerze <DOF>,
- **robot\_move\_to\_pos <x> <y> <z> <phi> <psi> <theta> <speed> <inter>** - przesunąć efektor manipulatora do pozycji <x>, <y>, <z> i orientacji <phi>, <psi>, <theta> z prędkością <speed> i z interpolacją <inter> (n — wszystkie osie poruszają się z prędkością maksymalną [*RPM*], a — z interpolacją osiową [*RPM*], l — z interpolacją liniową [ $\frac{mm}{s}$ ]),
- **servo\_set\_zero\_pos <DOF>** - ustaw aktualną pozycję osi robota o numerze <DOF> jako pozycję zerową,
- **robot\_go\_to\_zero\_pos <rpm>** - przesunąć efektor manipulatora do pozycji zerowej z prędkością dla wszystkich osi robota <rpm> (w *RPM*),
- **robot\_get\_pos** - wyświetli aktualną pozycję efektora manipulatora,
- **robot\_learn\_pos <max\_speed> <delay\_ms> <inter>** - naucz robota nowej pozycji, do której ma dojechać z prędkością maksymalną <max\_speed>, z wybraną interpolacją <inter> (n — wszystkie osie poruszają się z prędkością maksymalną [*RPM*], a — z interpolacją osiową [*RPM*], l — z interpolacją liniową [ $\frac{mm}{s}$ ]) i poczekaj <delay\_ms> (w *ms*),
- **robot\_reset\_learned\_pos** - zresetuj nauczone pozycje robota,
- **robot\_move\_to\_learned\_pos** - przesunąć efektor manipulatora przez wcześniej nauczone pozycje z wybraną interpolacją.

Naciśnięcie klawiszy *CTRL + C* powoduje wyjście z terminalu i zapisanie aktualnych pozycji robota do pamięci nieulotnej mikrokontrolera ESP32. Zasilanie manipulatora będzie mogło być wtedy bezpiecznie wyłączone bez ryzyka dekalibracji pozycji zerowej (rozdz. 7.3). Zrzut ekranu terminalu wraz z pokazaniem działania przykładowych komend zamieszczono poniżej.



```
moving robot to position...
done!
[esp32]> robot_move_to_pos -- -100 100 287 -90 90 0 10 a
moving robot to position...
done!
[esp32]> robot_move_to_pos -- 100 50 447 -90 90 0 10 a
moving robot to position...
done!
[esp32]> robot_move_to_pos -- -100 100 287 -90 90 0 10 a
moving robot to position...
done!
E (66675) AX_servo: Servo 5, response is invalid
E (67355) AX_servo: Servo 4, response is invalid
[esp32]> robot_move_to_pos -- 100 50 447 -90 90 0 10 a
moving robot to position...
done!
[esp32]> robot_move_to_pos -- -100 100 287 -90 90 0 10 a
moving robot to position...
done!
[esp32]> robot_move_to_pos -- 100 50 447 -90 90 0 10 a
moving robot to position...
done!
[esp32]> robot_move_to_pos -- 100 50 347 -90 90 0 10 l
moving robot to position...
done!
[esp32]> robot_move_to_pos -- 100 50 447 -90 90 0 10 l
moving robot to position...
done!
[esp32]> robot_move_to_pos -- 100 50 347 -90 90 0 10 a
moving robot to position...
done!
[esp32]> robot_get_pos
position 1: 100.762642 mm
position 2: 50.506302 mm
position 3: 346.893219 mm
position 4: -146.406448 deg
position 5: 89.531898 deg
position 6: -57.526833 deg
[esp32]> servo_get_pos 5
getting servo 5 position ...
servo 5 position: 118.328430
[esp32]> servo_get_pos 3
getting servo 3 position ...
servo 3 position: -111.510101
[esp32]>
```

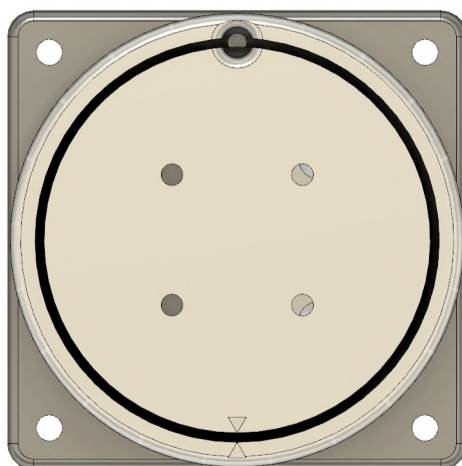
```
inv_kin_solv/build$ ./inv_kin_
solv
Waiting for init...
^C
jang175@jang175-ORTEGE-Z30-B:
~/Desktop/engineering_project/
inv_kin_solv/build$ ./inv_kin_
solv
Waiting for init...
Connection initialised!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
max iterations reached!
```

Rysunek 7.2: Przykładowa obsługa robota za pomocą terminalu

### 7.3 Kalibracja osi robota

Robot powinien znać swoją pozycję zerową, przy której ustalone są zerowe pozycje poszczególnych osi. Przy pierwszym uruchomieniu manipulatora należy przeprowadzić procedurę kalibracji robota. Klasyczną metodą znajdowania pozycji zerowej jest zastosowanie czujników krańcowych. Osiągnięcie przez robota pozycji powodującej zwarcie czujnika krańcowego oznacza, że dana pozycja jest pozycją zerową. Rozwiązanie to wymaga zastosowania dodatkowych przewodów oraz zwiększa poziom złożoności konstrukcji mechanicznej przegubów, czego starano się w tym projekcie uniknąć.

Problem kalibracji osi robota postanowiono rozwiązać metodą programową z ręcznym ustalaniem pozycji zerowych poszczególnych osi robota. Kontroler zapisuje raz ustaloną przez użytkownika pozycję zerową (wskazania enkoderów) w pamięci nieulotnej mikrokontrolera. Użytkownik powinien ustawić zerowe ułożenie osi manipulatora poprzez ich ułożenie w sposób powodujący zrównanie położenia strzałek znajdujących się na przekładniach (rysunek 7.3). Następnie, po wpisaniu odpowiedniej komendy (rozdz. 7.2), wskazania enkoderów zostają zapisane w pamięci nieulotnej mikrokontrolera jako nowa pozycja zerowa.



Rysunek 7.3: Przekładnia w położeniu zerowym

Sterowniki EMM42\_v4.2 po zaniku zasilania resetują stan rejestrów enkoderów do  $0^\circ$ . Z kolei, w tej samej sytuacji, sterowniki MKS SERVO57C przeskalowują odczytywane wartości enkoderów do zakresu od  $-180^\circ$  do  $180^\circ$ . W związku z tym otrzymanie poprawnej wartości pozycji zerowej enkoderów wymaga zapisania w pamięci nieulotnej mikrokontrolera aktualnych wskazań enkoderów przed wyłączeniem zasilania. Operacja ta ustawia w pamięci nieulotnej flagę oznaczającą, że robot został wyłączony poprawnie bez dekalibracji. Następnie, po ponownym uruchomieniu robota, należy odczytać wartości zapisane w pamięci i wyliczyć odpowiednie przesunięcia wskazań enkoderów względem punktu zerowego. Odczytywana jest również flaga poprawnego wyłączenia robota. Jeżeli flaga nie została ustawiona, w terminalu pojawi się informacja, że robot został wyłączony niepoprawnie i należy ponownie przeprowadzić kalibrację. Flaga ta jest kasowana po każdej inicjalizacji robota. Wadą tego rozwiązania jest to, że w przypadku przesunięcia pierwszych trzech osi robota, podczas gdy będzie on wyłączony, nastąpi dekalibracja pozycji zerowej. Problem ten nie występuje w ostatnich trzech osiach robota, ponieważ serwomechanizmy AX-12A odczytują swoją aktualną pozycję za pomocą potencjometrów.



Rysunek 7.4: Robot w pozycji zerowej

Zastosowane rozwiązanie kalibracji robota jest wystarczające do prawidłowej jego pracy. Opór występujący na przekładniach planetarnych jest na tyle duży, że nie ma możliwości przypadkowej zmiany położenia pierwszych trzech osi robota podczas braku zasilania. Przed wyłączeniem zasilania robota wymagane jest jednak bezpieczne zakończenie pracy programu poprzez naciśnięcie klawiszy *CTRL + C* w terminalu użytkownika. Dalej występuje problem dekalibracji robota w przypadku nagłego zaniku zasilania. W celu jego rozwiązania należałoby zastosować system przełączania zasilania mikrokontrolera na zasilanie awaryjne (np. zasilanie akumulatorowe), gdy zostanie wykryty zanik napięcia. Mikrokontroler powinien wtedy obsłużyć funkcję przerwania zapisującą obecne położenia osi robota do pamięci nieulotnej. Następnie zasilanie awaryjne powinno zostać wyłączone, aby nie powodować jego szybkiego rozładowania. Nie udało się zrealizować tej funkcjonalności ze względu na ograniczenia czasowe.

#### 7.4 Kinematyka odwrotna

W celu zapewnienia jak najszybszego wykonywania obliczeń algorytm do wyznaczania kinematyki odwrotnej zdecydowano się napisać w języku C. Na etapie testów okazało się jednak, że mikrokontroler ESP32 nie jest w stanie wykonywać obliczeń numerycznych z akceptowalną szyb-

kością. W związku z tym zdecydowano się na użycie komputera jednopłytkowego Raspberry Pi 3B+ również oprogramowanego w języku C. Jego wydajność pozwala na wykonywanie obliczeń o wiele szybciej. Raspberry Pi komunikuje się z mikrokontrolerem ESP32 za pomocą protokołu I2C. Mikrokontroler wysyła do układu Raspberry Pi wymagane położenie efektora oraz aktualne wartości kątów poszczególnych przegubów. Aktualne wartości kątów są używane jako wartości początkowe dla algorytmu rozwiązywania zagadnienia kinematyki odwrotnej. Raspberry Pi odbiera je, przeprowadza obliczenia kinematyki odwrotnej i zwraca obliczone wartości kątów z powrotem do mikrokontrolera.

Na późniejszym etapie prac stwierdzono, że komputer jednopłytkowy Raspberry Pi także wykonuje obliczenia zbyt wolno. W związku z tym postanowiono przenieść program do rozwiązywania kinematyki odwrotnej na komputer PC z systemem Linux Ubuntu. Interfejs I2C zastąpiono interfejsem UART.

Algorytm wyznaczania wartości kątów manipulatora przedstawiono w rozdziale 6.4. W celu realizacji obliczeń macierzowych napisano autorską bibliotekę w języku C. Macierze reprezentowane są przez tablice dwuwymiarowe liczb typu *double* (typ zmiennych zmiennoprzecinkowych o podwójnej precyzji) dla zachowania większej precyzji obliczeń. Jakobian funkcji kinematyki prostej jest obliczany poprzez różniczkowanie cząstkowe numeryczne w przód według wzoru 7.8.

$$\frac{df(a_0, \dots, a_k, \dots, a_n)}{da_k} = \frac{f(a_0, \dots, a_k + h, \dots, a_n) - f(a_0, \dots, a_k, \dots, a_n)}{h} \quad (7.8)$$

Maksymalna liczba iteracji algorytmu została ustalona na 100 poprzez testowanie jego działania dla różnych wartości danych wejściowych. Wartość kroku różniczkowania  $h$ , wynoszącą 0,0001 rad, dobrano tak, aby uzyskać zadowalającą dokładność obliczeń. Pozwala to na wykonywanie ruchów pomiędzy znacznie oddalonymi od siebie punktami. Akceptowalne wartości błędów (zawarte w macierzy  $\epsilon$ ) wynoszą 0,5 mm błędu położenia i 1° błędu orientacji. W algorytmie zaimplementowano również zabezpieczenie przed ewentualnym wystąpieniem wartości *NaN* (ang. *Not a Number*) oraz przed przekroczeniem maksymalnej liczby iteracji. W przypadku wystąpienia któregośkolwiek z tych zdarzeń algorytm zwraca początkowe wartości kątów. To rozwiązanie sprawia, że robot nie będzie próbował osiągnąć pozycji poza jego fizycznym zasięgiem.

Jeśli nie uda się znaleźć rozwiązania w zakresie zadanego błędu, algorytm ponawia poszukiwanie rozwiązania z innymi warunkami początkowymi kątów manipulatora. Nowe warunki początkowe utworzone są poprzez dodanie do oryginalnych wartości początkowych losowych wartości kątów z przedziału od -90° do 90°. Żeby zbędnie nie wydłużać obliczeń dla pozycji nieosiągalnych, maksymalna liczba restartów została ustalona na 10.

Zaimplementowano zabezpieczenie przed przekroczeniem zakresu ruchu osi robota. Pozwala ono na znalezienie rozwiązania biorącego pod uwagę ograniczenia ruchu serwosilników Dynamixel AX-12A wynoszące od -150° do 150°. W przypadku, gdy w czasie iteracji algorytmu wyznaczone wartości kątów trzech ostatnich osi robota znajdują się poza tym zakresem, zostają one przesunięte do najbliższego punktu granicznego. Brak takiego zabezpieczenia powoduje, że nie-

które pozycje narzędzia są nieosiągalne, nawet jeśli znajdują się w fizycznym zasięgu robota. Przyczyną jest przekraczanie dozwolonego zakresu ruchu serwosilników.

## 7.5 Ruch z interpolacją osiową

Ruch z interpolacją osiową oznacza, że wszystkie przeguby powinny rozpocząć i zakończyć swój ruch w tym samym czasie. Po wyznaczeniu kątów docelowych osi manipulatora na podstawie aktualnych  $\varphi_{i-1}$  i zadanych pozycji kątowych  $\varphi_i$  przegubów program sterujący sprawdza, który przegub ma najdłuższą drogę kątową  $\phi_{\max}$  do pokonania. Dla wyznaczonego przegubu zostaje przypisana prędkość maksymalna  $\omega_{\max}$ , którą definiuje użytkownik. Wyliczony zostaje czas ruchu tego stopnia swobody  $t_c$  przy prędkości maksymalnej. Ten czas jest całkowitym czasem trwania zadanego ruchu dla wszystkich osi robota. Dla pozostałych przegubów wyliczane są prędkości  $\omega_i$ , pozwalające na pokonanie ich dróg w tym samym czasie całkowitym. Ponieważ czas ruchu z przyspieszeniami w silnikach krokowych trwa dokładnie 120% czasu trwania ruchu z zadaną prędkością bez przyspieszeń (podrozdz.: 7.1.1), zachowana jest pełna kontrola nad czasem trwania ruchu wszystkich silników. Prędkość zadaną dla serwomechanizmów AX-12A, które nie realizują przyspieszeń, obniża się 1,2 raza w celu skompensowania dłuższego czasu trwania ruchu silników krokowych realizujących planowane przyspieszanie i hamowanie. W ten sposób wszystkie osie osiągają zadane pozycje w tym samym czasie. Ze względu na dolne ograniczenia prędkości obrotowych silników AX-12A, ruch z interpolacją osiową nie będzie zawsze działać poprawnie dla bardzo małych prędkości zadanych przez użytkownika.

Wzory opisujące proces wyznaczania prędkości poszczególnych osi manipulatora przedstawiono poniżej.

$$\phi_i = |\varphi_i - \varphi_{i-1}| \text{ [}^\circ\text{]} \quad (7.9)$$

$$\phi_{\max} = \max(\phi_i) \text{ [}^\circ\text{]} \quad (7.10)$$

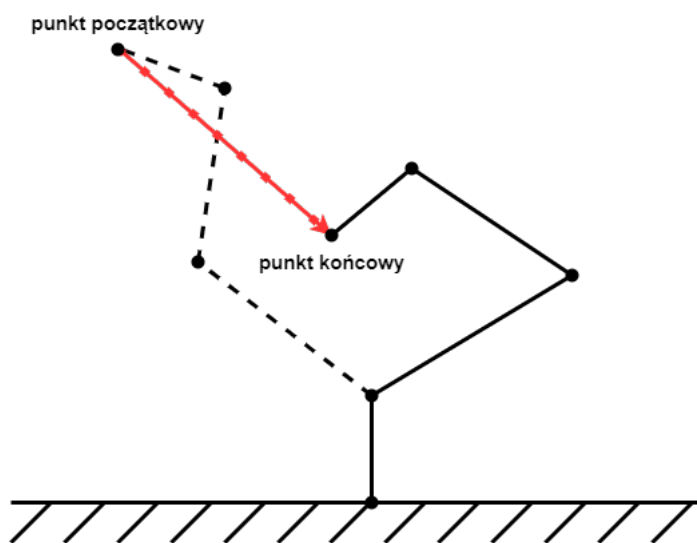
$$t_c = \frac{\phi_{\max}}{\omega_{\max}} \left[ \frac{^\circ}{\frac{\text{obr.}}{\text{min}}} \right] \quad (7.11)$$

$$\omega_i = \frac{\phi_i}{t_c} \left[ \frac{^\circ}{\frac{\text{obr.}}{\text{min}}} = \frac{\text{obr.}}{\text{min}} \right] \quad (7.12)$$

## 7.6 Ruch z interpolacją liniową

Ruch z interpolacją liniową polega na ruchu efektora do zadanego położenia po linii prostej z zadaną prędkością liniową. W celu wyznaczenia prostoliniowej trajektorii ruchu podzielono odcinek łączący punkt początkowy i końcowy, na równe segmenty (rysunek 7.5). Długość segmentów  $s_c$  została wyznaczona w programie w ten sposób, że najdłuższa składowa X, Y lub Z trajektorii ma około 3 mm (tak, aby liczba wszystkich segmentów była liczbą całkowitą). Do położenia początkowego dodaje się wyliczony segment i ustawia tę pozycję jako docelową. Na podstawie długości segmentu oblicza się czas  $t_c$ , w którym ten zostanie pokonany z zadaną prędkością

liniową  $v \left[ \frac{mm}{s} \right]$ . Na podstawie aktualnych  $\varphi_{i-1}$  i zadanych pozycji kątowych  $\varphi_i$  przegubów wyznacza się, który z przegubów ma najdłuższą drogę kątową  $\phi_{\max}$  do pokonania. Czas  $t_c$  pozwala określić prędkość kątową  $\omega_{\max}$  osi robota z najdłuższą do pokonania drogą kątową. Prędkości kątowe pozostałych osi robota  $\omega_i$  są wyliczane tak jak w przypadku ruchu z interpolacją osiową (rozdz. 7.5). W ten sposób wszystkie osie zakończą ruch w czasie  $t_c$ . Dzięki temu rozwiązaniu efektor przesuwany się do nowej pozycji docelowej z zadaną prędkością liniową. Po osiągnięciu nowej pozycji docelowej proces powtarza się aż do osiągnięcia pozycji końcowej. W tym trybie należy wyłączyć realizację przyspieszeń, gdyż powodowałyby one nierówny ruch efektora.



Rysunek 7.5: Przedstawienie podziału ścieżki na krótkie odcinki w ruchu z interpolacją liniową

Wzory wykorzystywane do wyznaczania prędkości poszczególnych osi manipulatora przedstawiono poniżej.

$$s_c = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2} \text{ [mm]} \quad (7.13)$$

$$t_c = \frac{s_c}{v} \text{ [s]} \quad (7.14)$$

$$\phi_i = |\varphi_i - \varphi_{i-1}| \text{ [}^\circ\text{]} \quad (7.15)$$

$$\phi_{\max} = \max(\phi_i) \text{ [}^\circ\text{]} \quad (7.16)$$

$$\omega_{\max} = \frac{\phi_{\max}}{t_c} \left[ \frac{^\circ}{s} \right] = \frac{\frac{\phi_{\max}}{360^\circ}}{\frac{t_c}{60 \frac{s}{min}}} \left[ \frac{obr.}{min} \right] \quad (7.17)$$

W celu przyspieszenia wykonywania licznych obliczeń kinematyki odwrotnej zdecydowano się na wykorzystanie wielowątkowości oferowanej przez FreeRTOS. Wątek główny programu odpowiada za ruch silników. Wysyła on odpowiednie sygnały sterujące i sprawdza, czy wszystkie silniki osiągnęły zadane pozycje. Równolegle wykonywane jest zadanie wyznaczania kinematyki



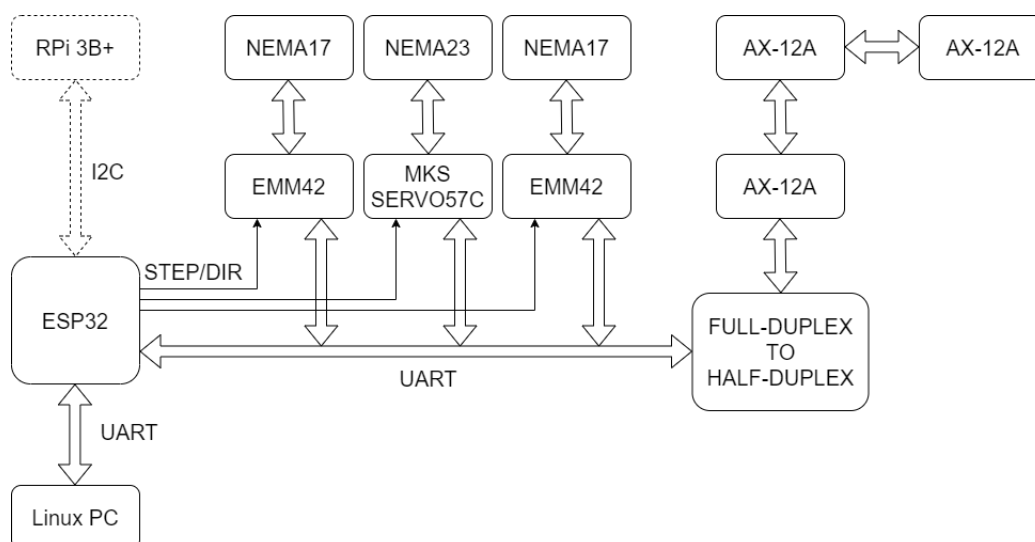
odwrotnej. Wątek ten wysyła aktualne pozycje osi manipulatora oraz zadane pozycje docelowe efektora do komputera PC. Następnie oczekuje na odpowiedź z obliczonymi wartościami wymaganych kątów przegubów robota. Wartości te wysyłane są do pętli głównej programu za pomocą kolejki FIFO. Rozwiązanie to pozwala na znaczną poprawę wydajności programu, gdyż wątek główny nie musi czekać na wykonanie czasochłonnych obliczeń kinematyki odwrotnej. W tym samym czasie może monitorować stan silników. Wartości kątów osi robota, odpowiadające kolejnym punktom zadanej trajektorii, mogą być obliczane w czasie, gdy silniki nie osiągnęły jeszcze poprzednich zadanych położeń.

### **7.7 Uczenie pozycji robota**

Robot posiada opcję uczenia i odtwarzania zdefiniowanych przez użytkownika pozycji. Maksymalna liczba możliwych do zapisania pozycji wynosi 20. W celu nauczania robota nowej pozycji należy najpierw dojechać do niej końcówką manipulatora. Następnie za pomocą terminalu użytkownika powinno się wywołać komendę zapamiętania danej pozycji. Dodatkowo należy zdefiniować prędkość maksymalną osiągnięcia tej pozycji, czas pozostania w danej pozycji oraz rodzaj interpolacji. Kroki te powtarza się dla wszystkich pozycji, które robot ma po kolei osiągnąć. Po zakończeniu nauki za pomocą odpowiedniej komendy można wywołać realizację zapamiętanej sekwencji ruchów. Nauczone pozycje można w każdej chwili skasować. Są one również traczone po wyłączeniu robota, gdyż nie są zapisywane w pamięci nieulotnej.

### **7.8 Pętla programowa**

Na rysunku 7.6 przedstawiono schemat blokowy komunikacji robota. Rolę jednostki centralnej pełni mikrokontroler ESP32. Po włączeniu zasilania manipulatora następuje inicjalizacja wejść i wyjść mikrokontrolera oraz protokołów komunikacyjnych. Następnie mikrokontroler oczekuje na sygnał kontrolny od komputera PC (wcześniej Raspberry Pi 3B+), sygnalizujący jego gotowość do pracy. Po jego otrzymaniu robot odczytuje aktualne położenia przegubów i ustawia się w pozycji zerowej. W razie potrzeby użytkownik powinien wykonać jego ponowną kalibrację. Robot jest gotowy do pracy.



Rysunek 7.6: Schemat komunikacji

Mikrokontroler otrzymuje komendy sterujące od użytkownika z komputera PC poprzez interfejs UART. Użytkownik może zadać ruch z interpolacją osiową, liniową lub tak, żeby wszystkie osie robota poruszały się z tą samą zadaną prędkością. Następnie ESP32 za pomocą UART wysyła żądane koordynaty do komputera PC (wcześniej przez I2C do Raspberry Pi 3B+). Komputer przeprowadza obliczenia kinematyki odwrotnej i zwraca obliczone wartości kątów do mikrokontrolera ESP32. Kolejnym etapem jest wysłanie odpowiednich sygnałów do sterowników silników. Silniki krokowe, za pośrednictwem sterowników EMM42\_v4.2 i MKS SERVO57C, sterowane są za pomocą sygnałów STEP/DIR. Sterowniki zwracają informacje o stanie silników (np. położenie wału, błąd sterowania itd.) poprzez UART. Serwomechanizmy AX-12A komunikują się w pełni za pomocą interfejsu UART half-duplex. Z tego powodu wymagany jest konwerter UART full-duplex do half-duplex. Mikrokontroler odczytuje położenie wałów silników i po osiągnięciu zadanych wartości przesyła informację o powodzeniu operacji do terminalu użytkownika.

## 8 WYNIKI

Poniżej przedstawiono uzyskane rezultaty testów wraz z ich analizą i komentarzem.

### 8.1 Algorytm rozwiązywania kinematyki odwrotnej (Jan Górski)

Algorytm wyznaczania kinematyki odwrotnej testowano za pomocą osobnego programu testowego dla różnych wartości początkowych konfiguracji osi robota oraz różnych zadanych położeń i orientacji docelowych. Wyniki kilku przykładowych testów zamieszczono w tabeli 8.1 (dane zostały zaokrąglone do 4 miejsc po przecinku).

pozycja początkowa	pozycja docelowa	pozycja obliczona	błąd położenia	liczba iteracji
$x \approx -352,0628 \text{ mm}$ $y \approx -86,5196 \text{ mm}$ $z \approx 370,3132 \text{ mm}$ $\varphi \approx 87,2228^\circ$ $\psi \approx 50,9523^\circ$ $\theta \approx -153,1275^\circ$	$x = -200 \text{ mm}$ $y = -150 \text{ mm}$ $z = 300 \text{ mm}$ $\varphi = -90^\circ$ $\psi = 0^\circ$ $\theta = 90^\circ$	$x \approx -200,0149 \text{ mm}$ $y \approx -149,9818 \text{ mm}$ $z \approx 299,9202 \text{ mm}$ $\varphi \approx -89,9988^\circ$ $\psi \approx -0,0007^\circ$ $\theta \approx 89,9994^\circ$	$\Delta x \approx 0,0149 \text{ mm}$ $\Delta y \approx -0,0182 \text{ mm}$ $\Delta z \approx 0,0799 \text{ mm}$ $\Delta \varphi \approx -0,0012^\circ$ $\Delta \psi \approx 0,0007^\circ$ $\Delta \theta \approx 0,0006^\circ$	41
$x \approx -228,6903 \text{ mm}$ $y \approx 199,5199 \text{ mm}$ $z \approx -131,3227 \text{ mm}$ $\varphi \approx 104,1509^\circ$ $\psi \approx 69,247558^\circ$ $\theta \approx -20,1611^\circ$	$x = 134 \text{ mm}$ $y = 123 \text{ mm}$ $z = 462 \text{ mm}$ $\varphi = 64^\circ$ $\psi = -24^\circ$ $\theta = 64^\circ$	$x \approx 133,6871 \text{ mm}$ $y \approx 123,0284 \text{ mm}$ $z \approx 461,5081 \text{ mm}$ $\varphi \approx 64,0386^\circ$ $\psi \approx -24,012^\circ$ $\theta \approx 63,9965^\circ$	$\Delta x \approx 0,3129 \text{ mm}$ $\Delta y \approx -0,0284 \text{ mm}$ $\Delta z \approx 0,4919 \text{ mm}$ $\Delta \varphi \approx -0,0386^\circ$ $\Delta \psi \approx 0,012^\circ$ $\Delta \theta \approx 0,0035^\circ$	16
$x \approx 316,2664 \text{ mm}$ $y \approx -6,2321 \text{ mm}$ $z \approx 55,9835 \text{ mm}$ $\varphi \approx 162,4285^\circ$ $\psi \approx 34,4102^\circ$ $\theta \approx 65,2447^\circ$	$x = 310,6 \text{ mm}$ $y = -10,1 \text{ mm}$ $z = 127,4 \text{ mm}$ $\varphi = 164,7^\circ$ $\psi = 44,7^\circ$ $\theta = 64,2^\circ$	$x \approx 310,5375 \text{ mm}$ $y \approx -10,1215 \text{ mm}$ $z \approx 127,4232 \text{ mm}$ $\varphi \approx 164,7016^\circ$ $\psi \approx 44,7022^\circ$ $\theta \approx 64,2004^\circ$	$\Delta x \approx 0,0625 \text{ mm}$ $\Delta y \approx 0,0215 \text{ mm}$ $\Delta z \approx -0,0232 \text{ mm}$ $\Delta \varphi \approx -0,0016^\circ$ $\Delta \psi \approx -0,0022^\circ$ $\Delta \theta \approx -0,0004^\circ$	35

Tabela 8.1: Testy algorytmu rozwiązującego zagadnienia kinematyki odwrotnej

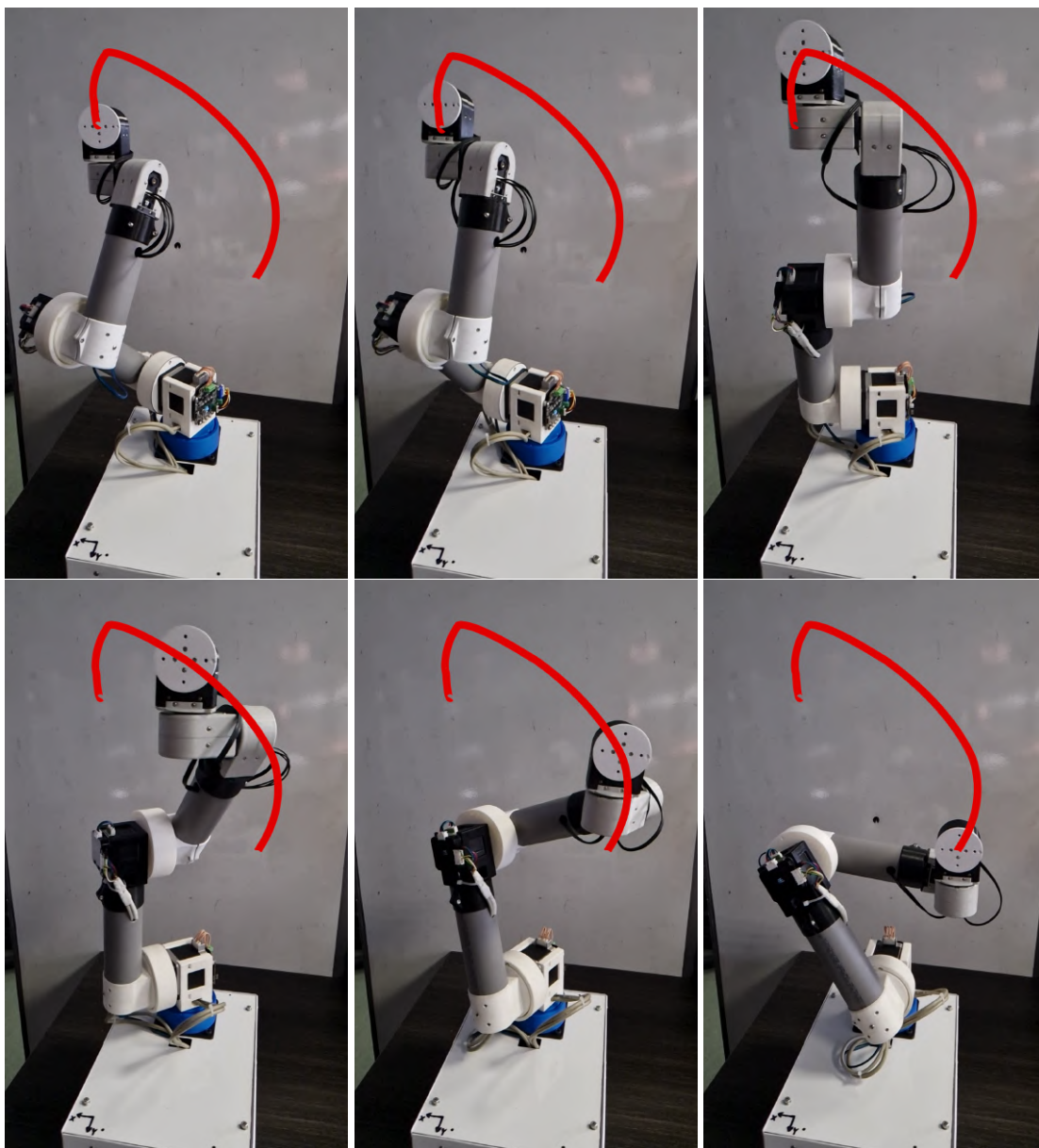
Algorytm działa poprawnie dla większości przypadków. Dobrana maksymalna liczba iteracji jest wystarczająca do znalezienia rozwiązania dla przypadku, gdy pozycja i orientacja docelowa znacząco różnią się od pozycji i orientacji startowych poszukiwań. W przypadku, gdy zadana pozycja znajduje się poza fizycznym zasięgiem robota, zwracana jest pozycja początkowa — robot nie wykonuje ruchu. Na etapie testów zauważono jednak, że w dalszym ciągu nie wszystkie pozycje są osiągalne przy danych warunkach początkowych. Dodanie restartów algorytmu z innymi, losowymi pozycjami początkowymi, pozwoliło na zmniejszenie częstotliwości występowania tego problemu, aczkolwiek nie został on w pełni rozwiązany.

## **8.2 Ruch z interpolacją osiową (Jan Górski)**

Ruch z interpolacją osiową działa poprawnie w większości przypadków. Występują jednak sytuacje, w których osie robota nie osiągają zadanych pozycji w tym samym czasie. W przypadku, gdy zadana maksymalna prędkość obrotowa przegubów jest zbyt niska, część osi osiągnie pozycję docelową w innym czasie niż pozostałe. Spowodowane jest to dolnym ograniczeniem prędkości serwośników AX-12A, wynoszącym 0,111 RPM. Desynchronizacja ruchów osi następuje, gdy wyliczona prędkość trzech ostatnich przegubów jest niższa od tej wartości.

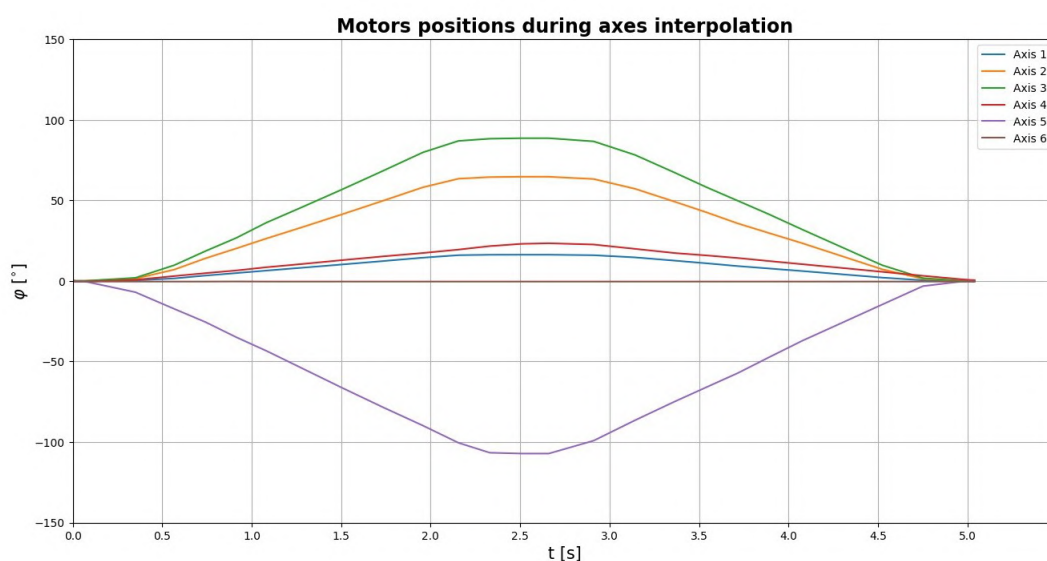
Zauważono również nieprawidłowe działanie interpolacji osiowej w przypadku wykonywania nagłych, szybkich ruchów przy dużym wyciągnięciu ramienia. W takiej sytuacji występują duże przeciążenia wynikające ze znacznych momentów bezwładności poruszających się przegubów. Sterowniki silników próbują kompensować te siły, jednak nie zawsze daje to zadowalający efekt. W razie wystąpienia tego typu problemów należy zmniejszyć prędkość maksymalną osi robota.

Na rysunku 8.1 przedstawiono przykład realizacji ruchu robota z interpolacją osiową. Trajektorię ruchu wyznaczono analizując zdjęcia poklatkowe ruchu robota. Kolorem czerwonym zaznaczono położenia końcówki manipulatora na poszczególnych klatkach. Ponieważ wszystkie osie robota zaczynają i kończą swój ruch w tym samym momencie, kształt trajektorii może znacząco odbiegać od prostej. Droga pokonywana przez końcówkę manipulatora jest dłuższa niż w przypadku ruchu z interpolacją liniową.

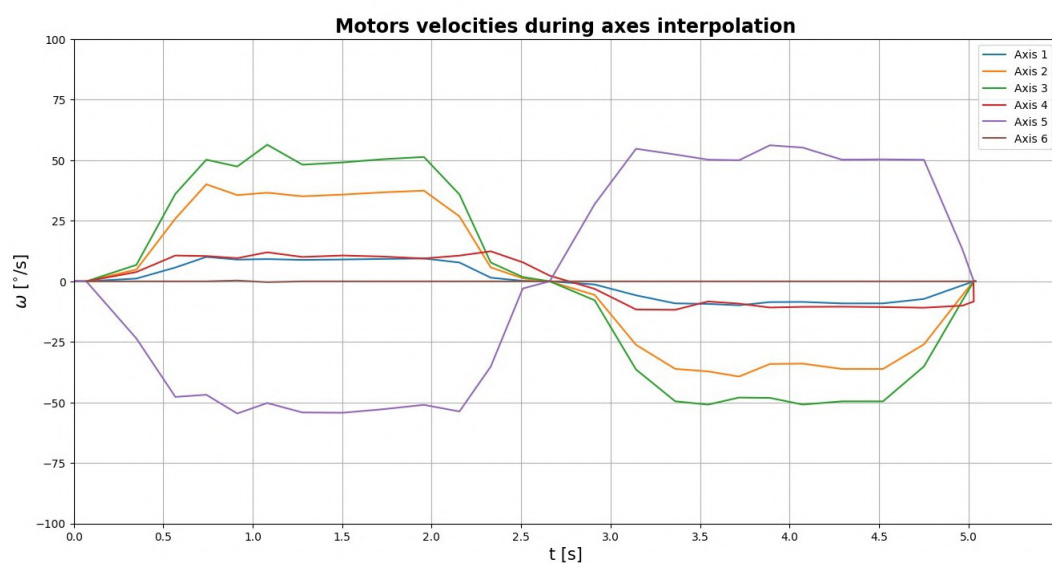


Rysunek 8.1: Przykład realizacji ruchu z interpolacją osiową

Na rysunku 8.2 przedstawiono wykres pomierzonego położenia kąowego przegubów robota w czasie ruchu z interpolacją osiową z jednej pozycji do drugiej i z powrotem. Z kolei na rysunku 8.3 przedstawiono wykres prędkości poszczególnych osi robota (uzyskanych numerycznie metodą różnic skończonych, stosując różnice w tył) w czasie wykonywania tych ruchów. Widoczne jest, że wszystkie osie rzeczywiście pokonują zadane im drogi kątowe w przybliżeniu w tym samym czasie, niezależnie od ich długości. Dla osi 1-3 widoczne są zadane fazy przyspieszania i hamowania, które trwają po około 10% początkowej i końcowej części ruchu.



Rysunek 8.2: Wykres położenia osi robota w funkcji czasu dla ruchu z interpolacją osiową

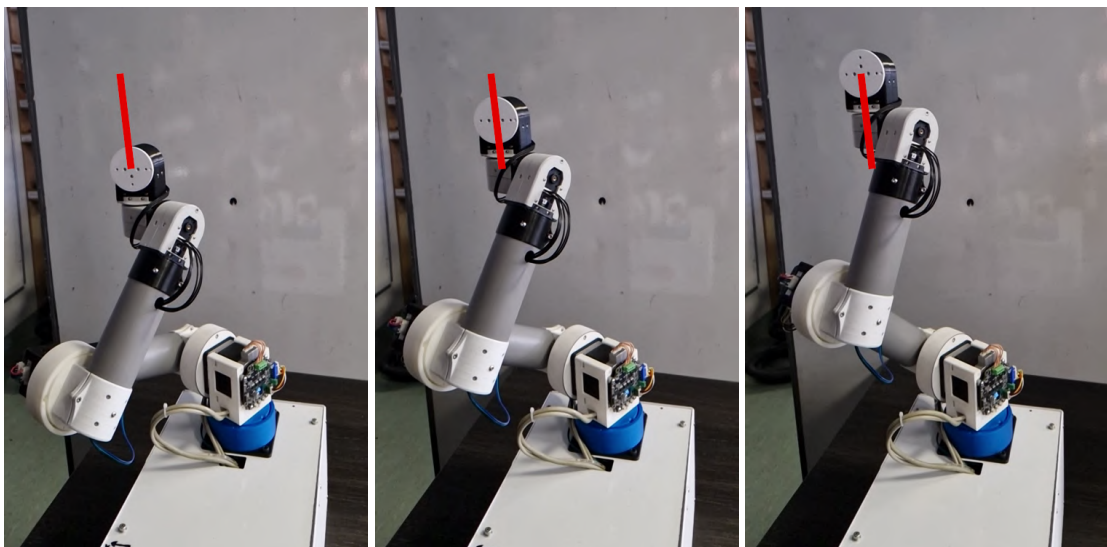


Rysunek 8.3: Wykres prędkości osi robota w funkcji czasu dla ruchu z interpolacją osiową

### 8.3 Ruch z interpolacją liniową (Jan Górski)

Robot wykonuje ruch z interpolacją liniową poprawnie po wyznaczonej trajektorii. Jednak w przypadku, gdy zadana prędkość liniowa jest zbyt duża, występuje skokowy ruch manipulatora. Spowodowane jest to występowaniem opóźnienia związanego z rozwiązywaniem zagadnienia kinematyki odwrotnej dla kolejnych segmentów trajektorii ruchu. Problemu nie udało się całkowicie rozwiązać za pomocą operacji wielowątkowych na mikrokontrolerze ESP32. Czas wykonywania obliczeń pomiędzy końcem poprzedniego kroku a początkiem następnego jest zbyt długi. W celu wyeliminowania tego problemu należałoby zastosować inne podejście do realizacji ruchu z interpolacją liniową lub zoptymalizować działanie obecnego programu. Innym problemem występującym w stosowanym podejściu jest brak realizacji przyspieszenia na początku i hamowania na końcu ruchu efektora. Z tego względu dynamiczne ruchy mogą powodować przeciążenia silników. Występują również problemy z trajektorią ruchu przechodzącą przez punkty osobliwe manipulatora. W tych miejscach przeguby robota pokonują duże drogi kątowe przy równoczesnym niewielkim przesunięciu efektora. W takich sytuacjach prędkość zadana dla tych przegubów może być większa niż ich maksymalna możliwa prędkość obrotowa. Generuje to opóźnienia w realizacji ruchu wzdłuż zadanej trajektorii. Z drugiej strony, jeśli zadane prędkości są niższe niż minimalne prędkości osiągalne przez serwomechanizmy AX-12A, robot nie jest w stanie wykonać ruchu dokładnie zadaną prędkością liniową. W takiej sytuacji należy zwiększyć prędkość liniową.

Na rysunkach 8.4 przedstawiono przykład realizacji ruchu z interpolacją liniową wzdłuż osi OZ bazowego układu współrzędnych robota z efekтором zwróconym zgodnie z kierunkiem osi OY układu bazowego. Kolorem czerwonym zaznaczono, analogicznie jak na rysunku 8.1, trajektorię ruchu końcówki robota. Widać, że jest ona linią prostą. Efektor porusza się po niej z zadaną prędkością liniową.



Rysunek 8.4: Przykład realizacji ruchu z interpolacją liniową

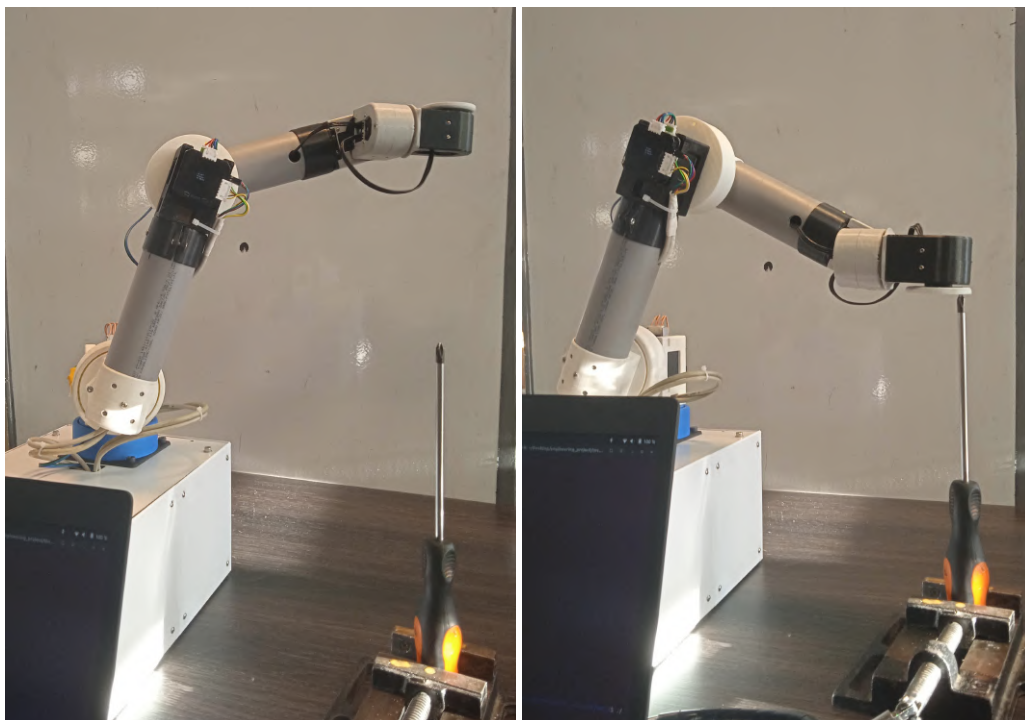


#### 8.4 Dokładność ruchu (Jan Górski)

W celu sprawdzenia dokładności ruchu robota wykonano serię 6 ruchów z interpolacją osiową. Każdy z nich polegał na ruchu do pozycji referencyjnej (wyznaczonej przez końcówkę wkrętaka krzyżowego na rysunku 8.5). Pozycję referencyjną zdefiniowano, dojeżdżając do niej za pierwszym razem końcówką manipulatora. Następnie przesuwno manipulator do losowej pozycji w przestrzeni i wracano do pozycji referencyjnej, za każdym razem z inną orientacją końcówki robota. Za pomocą suwmiarki mierzono odległość od pozycji referencyjnej. Wyniki pomiarów zamieszczono w tabeli 8.2.

numer pomiaru	przesunięcie od pozycji docelowej [mm]
1	7,8
2	5,4
3	5,6
4	7,5
5	8,1
6	6,7
średnia	6,85

Tabela 8.2: Testy dokładności ruchu robota



Rysunek 8.5: Stanowisko do pomiaru dokładności ruchu robota



Uzyskane wartości, zamieszczone w tabeli 8.2, wynikają z luzów na poszczególnych osiach robota oraz z niedokładności wyznaczania kinematyki odwrotnej. Luzy wszystkich przegubów kumulują się i powodują niedokładność pozycjonowania końcówki efektora wynoszącą średnio około 6,85 mm.

### 8.5 Maksymalny udźwig (Michał Blicharz)

W celu sprawdzenia maksymalnego udźwigu poszczególnych osi robota wykonywano nimi ruch od położeń pokazanych na rysunku 8.6 do pionu. Ruchy powtarzano, każdorazowo zwiększając obciążenie przyłożone na końcu ramienia.

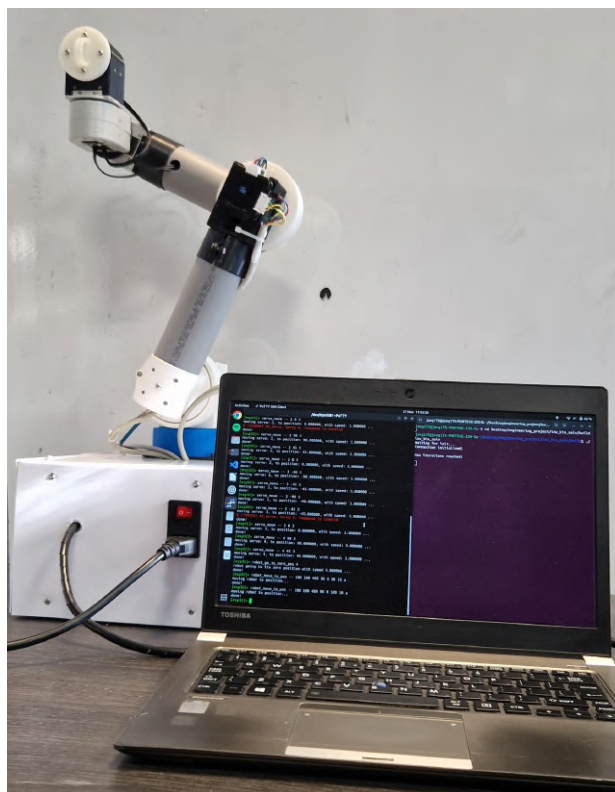


Rysunek 8.6: Pozycje początkowe do pomiaru udźwigu poszczególnych osi robota

Maksymalny udźwig, jaki udało się uzyskać dla czwartej osi, wyniósł 400 g i wynikał z maksymalnego momentu serwomechanizmów Dynamixel AX-12A. Na drugiej i trzeciej osi maksymalny udźwig wyniósł 500 g. Wynikało to również z ograniczenia serwomechanizmów Dynamixel. Pomimo że nie wykonywały one żadnego ruchu, uginały się pod obciążeniem i nie mogły utrzymać zadanej pozycji.

## 9 PODSUMOWANIE

Celem projektu było zbudowanie 6-osiowego ramienia robotycznego. Zadanie zostało w pełni zrealizowane. Efektem prac jest robot bazujący na konstrukcji koboty UR5 obsługiwany za pomocą komend wysyłanych z terminala uruchomionego na komputerze PC (rysunek 9.1). Jego zasięg wynosi 528,6 mm, a maksymalny udźwig to około 400-500 g. Zaimplementowano szereg instrukcji sterujących, umożliwiających ruch robota, a także uczenie i odczytywanie jego pozycji. Zadany ruch może odbywać się z interpolacją osiową, liniową lub z taką samą prędkością zadaną dla wszystkich osi. Kalibracja robota wykonywana jest poprzez ręczne ustawienie manipulatora w pozycji zerowej. Kinematyka odwrotna manipulatora obliczana jest na komputerze PC z systemem Linux. Program realizujący obliczenia został napisany w języku C. Realizacja projektu wymagała zaprojektowania dedykowanej płytki PCB. Na potrzeby projektu zostały również zaprojektowane przekładnie planetarne zapewniające odpowiedni moment obrotowy osi robota.



Rysunek 9.1: Widok zbudowanego robota

## 9.1 Perspektywy rozwoju

W czasie realizacji projektu zauważono kilka problemów, które mogłyby zostać rozwiązane w przyszłości. Jednym z nich jest brak możliwości oprogramowania przyspieszeń w serwosilnikach AX-12A. Rozwiązanie polegałoby na zastosowaniu innych serwomechanizmów, dających taką możliwość. Poprawić można również sposób realizacji przyspieszeń silników krokowych. Obecny posiada wiele wad w stosunku do przyspieszenia liniowego. Innym problemem jest skokowy ruch z interpolacją liniową manipulatora, który można wyeliminować poprzez zastosowanie innego podejścia do realizacji ruchu lub zoptymalizowanie działania obecnego programu. Warto również zwrócić uwagę na fakt, że robot nie posiada automatycznej procedury kalibracji. Można ją zrealizować, stosując czujniki krańcowe, pozwalające na automatyczny dojazd robota do pozycji zerowej. Warty rozważenia jest również system detekcji zaniku zasilania, umożliwiający zapisanie aktualnych pozycji przegubów robota, w celu uniknięcia dekalibracji układu. Celowym wydaje się także poprawienie tolerancji dopasowania elementów przekładni planetarnych w celu zmniejszenia luzów na osiach. Robota można również dostosować do konkretnych zadań, stosując odpowiednie chwytaki, czy to do przenoszenia obiektów, czy do trzymania narzędzi. Kolejnym możliwym usprawnieniem jest optymalizacja kodu w celu zastąpienia komputera PC innym mikrokomputerem umieszczonym w bazie robota i wysyłanie do niego komend poprzez specjalnie zaprojektowany interfejs.

## WYKAZ LITERATURY

- [1] „FANUC M-2000iA/2300.” (2023), adr.: <https://www.fanuc.eu/pl/en/robots/robot-filter-page/m-2000-series/m-2000ia-2300> (term. wiz. 15. 10. 2023).
- [2] „Zdjęcie FANUC M-2000iA/2300.” (2023), adr.: <https://www.fanuc.eu/~media/corporate/products/robots/m2000/m2000ia1700/fea-ro-pr-m20001700-l-1.jpg> (term. wiz. 24. 10. 2023).
- [3] „ABB IRB 8700.” (2023), adr.: <https://new.abb.com/products/robotics/robots/articulated-robots/irb-8700> (term. wiz. 15. 10. 2023).
- [4] „Zdjęcie ABB IRB 8700.” (2023), adr.: [https://www07.abb.com/images/librariesprovider89/default-album/irb-8700.jpg?sfvrsn=e3613a09\\_1](https://www07.abb.com/images/librariesprovider89/default-album/irb-8700.jpg?sfvrsn=e3613a09_1) (term. wiz. 24. 10. 2023).
- [5] „Universal Robots UR5.” (2023), adr.: <https://www.universal-robots.com/products/ur5-robot/> (term. wiz. 15. 10. 2023).
- [6] „Zdjęcie UR5.” (2023), adr.: <https://a.storyblok.com/f/169662/1000x1000/36a6c788d2/ur5e.png/m/fit-in/1266x1776> (term. wiz. 24. 10. 2023).
- [7] „KUKA LBR iisy 6 R1300.” (2023), adr.: <https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iisy-cobot> (term. wiz. 15. 10. 2023).
- [8] „Zdjęcie KUKA LBR iisy 6 R1300.” (2023), adr.: <https://www.kuka.com/-/media/kuka-corporate/images/products/robots/cta-images/lbr-iisy.png?rev=-1&w=767&hash=63C954054475822C43723300F85B15C0> (term. wiz. 24. 10. 2023).
- [9] „Dokumentacja serwosilnika Dynamixel AX-12A.” (2023), adr.: <https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/> (term. wiz. 10. 11. 2023).
- [10] „Strona sklepu oferującego przekładnie falową do silnika NEMA17.” (2023), adr.: [https://www.automationdirect.com/adc/shopping/catalog/motion\\_control/high-precision-inline\\_strain\\_wave\\_gearboxes/stepper\\_motor\\_gearboxes/hpgcn17-505m](https://www.automationdirect.com/adc/shopping/catalog/motion_control/high-precision-inline_strain_wave_gearboxes/stepper_motor_gearboxes/hpgcn17-505m) (term. wiz. 30. 11. 2023).
- [11] „Kalkulator przekładni planetarnej.” (2023), adr.: <https://juangg-projects.blogspot.com/2018/02/split-ring-compound-epicyclicplanetary.html> (term. wiz. 23. 11. 2023).
- [12] „Schematyczny rysunek wykorzystanej przekładni.” (2023), adr.: <https://juangg-projects.blogspot.com/2018/02/split-ring-compound-epicyclicplanetary.html> (term. wiz. 23. 11. 2023).
- [13] „Oficjalna strona producenta drukarek bambu lab.” (2023), adr.: <https://bambulab.com/en-eu> (term. wiz. 23. 11. 2023).
- [14] „Dokumentacja techniczna zasilacza.” (2023), adr.: <https://www.meanwell.com/productPdf.aspx?i=438> (term. wiz. 23. 11. 2023).

- [15] „Przetwornica bulk oparta o moduł XL4016.” (2023), adr.: <https://solectroshop.com/pl/przetwornice-step-down/34642-przetwornica-step-down-xl4016-dcdc-200w-4v-38v-do-125-35v-5905323230231.html> (term. wiz. 23. 11. 2023).
- [16] „Wykorzystana przetwornica bulk oparta o układ LM2596S-ADJ.” (2023), adr.: <https://solectroshop.com/pl/przetwornice-step-down/1082-przetwornica-step-down-lm2596-dc-32-35v-do-125-32v-3a-5905323231313.html> (term. wiz. 23. 11. 2023).
- [17] „Dokumentacja techniczna XL4016.” (2023), adr.: <https://www.xlsemi.com/datasheet/XL4016-EN.pdf> (term. wiz. 23. 11. 2023).
- [18] „Zasilacz.” (2023), adr.: <https://www.digikey.pl/en/products/detail/mean-well-usa-inc/HRP-200-24/7704297> (term. wiz. 23. 11. 2023).
- [19] „Strona główna oprogramowania KiCad.” (2023), adr.: <https://www.kicad.org> (term. wiz. 24. 11. 2023).
- [20] „Dokumentacja techniczna bufora sn74ls241.” (2023), adr.: [https://www.ti.com/lit/ds/symlink/sn74ls241.pdf?ts=1701347315121&ref\\_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FSN74LS241](https://www.ti.com/lit/ds/symlink/sn74ls241.pdf?ts=1701347315121&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FSN74LS241) (term. wiz. 30. 11. 2023).
- [21] „Dokumentacja techniczna tranzystora IRLML2803.” (2023), adr.: [https://www.mouser.pl/datasheet/2/196/Infineon\\_IRLML2803\\_DataSheet\\_v01\\_01\\_EN-3363559.pdf](https://www.mouser.pl/datasheet/2/196/Infineon_IRLML2803_DataSheet_v01_01_EN-3363559.pdf) (term. wiz. 30. 11. 2023).
- [22] S. R. Buss. „Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods.” CA 92093-0112, Department of Mathematics University of California, San Diego La Jolla. (paź. 2009).
- [23] R. Nilsson. „Inverse Kinematics.” 2009:142 CIV - ISSN: 1402-1617 - ISRN: LTU-EX-09/142-SE, Luleå University of Technology. (2009).
- [24] K. Lynch i F. Park. „Numerical Inverse Kinematics,” Northwestern University. (2023), adr.: <https://modernrobotics.northwestern.edu/nu-gm-book-resource/6-2-numerical-inverse-kinematics-part-1-of-2/> (term. wiz. 27. 11. 2023).
- [25] „Dokumentacja sterownika EMM42\_v4.2.” (2023), adr.: [https://github.com/Macrobase-tech/Emm42/blob/main/Datasheet/Emm\\_V4.x%E6%AD%A5%E8%BF%9B%E9%97%AD%E7%8E%AF%E9%A9%B1%E5%8A%A8%E5%99%A8%E8%AF%B4%E6%98%8E%E4%B9%A6Rev1.0.pdf](https://github.com/Macrobase-tech/Emm42/blob/main/Datasheet/Emm_V4.x%E6%AD%A5%E8%BF%9B%E9%97%AD%E7%8E%AF%E9%A9%B1%E5%8A%A8%E5%99%A8%E8%AF%B4%E6%98%8E%E4%B9%A6Rev1.0.pdf) (term. wiz. 10. 11. 2023).
- [26] „Dokumentacja sterownika MKS SERVO57C.” (2022), adr.: <https://github.com/makerbase-mks/MKS-SERVO57C/blob/main/User%20Manual/MKS%20SERVO57C%20User%20Manual%20V1.0.2.pdf> (term. wiz. 10. 11. 2023).
- [27] „Dokumentacja biblioteki ESP Console.” (2023), adr.: <https://docs.espressif.com/projects/esp-idf/en/v5.1.1/esp32/api-reference/system/console.html> (term. wiz. 12. 11. 2023).
- [28] „Strona główna projektu Putty.” (2023), adr.: <https://www.putty.org/> (term. wiz. 12. 11. 2023).

## Załącznik nr 1: Równania kinematyki prostej

Poniżej przedstawiono analityczne równania kinematyki prostej budowanego manipulatora.

$$\begin{aligned} x = & \delta_2 \cdot \sin(\varphi_0) - \delta_1 \cdot \sin(\varphi_0) - \delta_3 \cdot \sin(\varphi_0) - \delta_5 \cdot \cos(\varphi_4) \cdot \sin(\varphi_0) + L0 \cdot \cos(\varphi_0) \cdot \sin(\varphi_1) - L1 \cdot \cos(\varphi_0) \cdot \\ & \cos(\varphi_1) \cdot \sin(\varphi_2) + L1 \cdot \cos(\varphi_0) \cdot \cos(\varphi_2) \cdot \sin(\varphi_1) + \delta_4 \cdot \cos(\varphi_0) \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin(\varphi_3) - \delta_4 \cdot \cos(\varphi_0) \cdot \\ & \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \sin(\varphi_2) + \delta_4 \cdot \cos(\varphi_0) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_1) + \delta_4 \cdot \cos(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \\ & \sin(\varphi_3) - \delta_5 \cdot \cos(\varphi_0) \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_4) - \delta_5 \cdot \cos(\varphi_0) \cdot \cos(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \\ & \sin(\varphi_4) + \delta_5 \cdot \cos(\varphi_0) \cdot \cos(\varphi_2) \cdot \sin(\varphi_1) \cdot \sin(\varphi_3) \cdot \sin(\varphi_4) - \delta_5 \cdot \cos(\varphi_0) \cdot \cos(\varphi_3) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_4) \end{aligned}$$

$$\begin{aligned} y = & \delta_1 \cdot \cos(\varphi_0) - \delta_2 \cdot \cos(\varphi_0) + \delta_3 \cdot \cos(\varphi_0) + \delta_5 \cdot \cos(\varphi_0) \cdot \cos(\varphi_4) + L0 \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) - \\ & L1 \cdot \cos(\varphi_1) \cdot \sin(\varphi_0) \cdot \sin(\varphi_2) + L1 \cdot \cos(\varphi_2) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) + \delta_4 \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin(\varphi_0) \cdot \sin(\varphi_3) - \\ & \delta_4 \cdot \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_2) + \delta_4 \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) + \delta_4 \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \\ & \sin(\varphi_2) \cdot \sin(\varphi_3) - \delta_5 \cdot \cos(\varphi_1) \cdot \sin(\varphi_0) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_4) + \delta_5 \cdot \cos(\varphi_2) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_3) \cdot \\ & \sin(\varphi_4) - \delta_5 \cdot \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_4) - \delta_5 \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_4) \end{aligned}$$

$$\begin{aligned} z = & \delta_0 + L0 \cdot \cos(\varphi_1) + L1 \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) + L1 \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) + \delta_4 \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \\ & \cos(\varphi_3) + \delta_4 \cdot \cos(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) - \delta_4 \cdot \cos(\varphi_2) \cdot \sin(\varphi_1) \cdot \sin(\varphi_3) + \delta_4 \cdot \cos(\varphi_3) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) + \\ & \delta_5 \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_4) - \delta_5 \cdot \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \sin(\varphi_2) \cdot \sin(\varphi_4) + \delta_5 \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \\ & \sin(\varphi_1) \cdot \sin(\varphi_4) + \delta_5 \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_4) \end{aligned}$$

$$\begin{aligned} \varphi = & atan2(\cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \cos(\varphi_5) + \cos(\varphi_1) \cdot \cos(\varphi_5) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) - \cos(\varphi_2) \cdot \cos(\varphi_5) \cdot \\ & \sin(\varphi_1) \cdot \sin(\varphi_3) + \cos(\varphi_3) \cdot \cos(\varphi_5) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) - \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_4) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5) + \\ & \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \sin(\varphi_2) \cdot \sin(\varphi_5) - \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \sin(\varphi_1) \cdot \sin(\varphi_5) - \cos(\varphi_4) \cdot \\ & \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5), \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_4) - \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \sin(\varphi_2) \cdot \\ & \sin(\varphi_4) + \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_1) \cdot \sin(\varphi_4) + \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_4)) \end{aligned}$$

$$\begin{aligned} \psi = & atan2(\cos(\varphi_2) \cdot \sin(\varphi_1) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5) - \cos(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5) - \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \\ & \cos(\varphi_3) \cdot \sin(\varphi_5) - \cos(\varphi_3) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_5) - \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_3) + \\ & \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_2) - \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_1) - \cos(\varphi_4) \cdot \\ & \cos(\varphi_5) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3), sqrt((\cos(\varphi_0) \cdot \cos(\varphi_5) \cdot \sin(\varphi_4) - \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin(\varphi_0) \cdot \sin(\varphi_3) \cdot \\ & \sin(\varphi_5) + \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_2) \cdot \sin(\varphi_5) - \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_5) - \\ & \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_0) + \\ & \cos(\varphi_1) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_0) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) - \cos(\varphi_2) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_3) + \end{aligned}$$

$$\begin{aligned} \theta = & \operatorname{atan2}(\cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin(\varphi_0) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5) - \cos(\varphi_0) \cdot \cos(\varphi_5) \cdot \sin(\varphi_4) - \cos(\varphi_1) \cdot \\ & \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_2) \cdot \sin(\varphi_5) + \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_5) + \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \\ & \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5) - \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_0) - \cos(\varphi_1) \cdot \cos(\varphi_4) \cdot \\ & \cos(\varphi_5) \cdot \sin(\varphi_0) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) + \cos(\varphi_2) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_3) - \cos(\varphi_3) \cdot \\ & \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2), \cos(\varphi_5) \cdot \sin(\varphi_0) \cdot \sin(\varphi_4) + \cos(\varphi_0) \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \\ & \sin(\varphi_3) \cdot \sin(\varphi_5) - \cos(\varphi_0) \cdot \cos(\varphi_1) \cdot \cos(\varphi_3) \cdot \sin(\varphi_2) \cdot \sin(\varphi_5) + \cos(\varphi_0) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \sin(\varphi_1) \cdot \\ & \sin(\varphi_5) + \cos(\varphi_0) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) \cdot \sin(\varphi_5) - \cos(\varphi_0) \cdot \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \\ & \cos(\varphi_5) - \cos(\varphi_0) \cdot \cos(\varphi_1) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_2) \cdot \sin(\varphi_3) + \cos(\varphi_0) \cdot \cos(\varphi_2) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \\ & \sin(\varphi_1) \cdot \sin(\varphi_3) - \cos(\varphi_0) \cdot \cos(\varphi_3) \cdot \cos(\varphi_4) \cdot \cos(\varphi_5) \cdot \sin(\varphi_1) \cdot \sin(\varphi_2)) \end{aligned}$$