

**A G H**

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**  
**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Projekt dyplomowy

*System sterowania robotem kroczącym na czterech nogach*  
*Control system for a four-legged walking robot*

Autor: *Dawid Antosz*  
Kierunek studiów: *Automatyka i Robotyka*  
Opiekun pracy: *dr inż. Maciej Rosół*

Kraków, 2024



## **Spis treści**

<b>1. Wstęp.....</b>	<b>7</b>
1.1. Cel pracy.....	8
1.2. Zakres pracy .....	8
<b>2. Przegląd istniejących konstrukcji .....</b>	<b>9</b>
2.1. „Mini cheetah”, Instytut Technologii w Massachusetts .....	9
2.2. „Morti”, Instytut Maxa Plancka w Stuttgarcie .....	11
<b>3. Projekt robota kroczącego na czterech nogach.....</b>	<b>13</b>
3.1. Założenia projektowe .....	13
3.2. Dobór elementów .....	14
3.3. Konstrukcja robota .....	21
3.4. Kinematyka prosta.....	25
3.5. Kinematyka odwrotna .....	31
3.6. Implementacja systemu sterowania.....	34
<b>4. Prezentacja działania programu.....</b>	<b>43</b>
<b>5. Podsumowanie oraz wnioski .....</b>	<b>47</b>



## **1. Wstęp**

Rozważając ogólne tło dotychczasowych osiągnięć techniki w zakresie robotyki, zauważalne jest, że odgrywa ona coraz istotniejszą rolę, nie tylko w przemyśle, ale również na rynku konsumenckim. Jednym z licznych przykładów, chociażby coraz to powszechniejszych robotów sprzątających, jest dowodem tego, jak technologia staje się ogólnie powszechna oraz w coraz to znaczącym stopniu wpływa na nasze życie.

Roboty kroczące są klasyfikowane jako roboty, których sposób lokomocji opiera się na mechanizmie wzorującym się na odnóżach żywych organizmów. Jest to ich największą przewagą w porównaniu do konwencjonalnych robotów kołowych, ze względu na większą elastyczność w przemieszczaniu się po różnego rodzaju nawierzchniach. Mimo iż wymaga to większej złożoności oraz zużycia energii, przewaga ta może jednak wykazywać potencjał wykorzystania w zadaniach eksploracji trudno dostępnych terenów, gdzie pojazd kołowy miałby znaczne trudności w osiągnięciu zamierzonego celu. [1]

Wspomniana wcześniej kwestia dotycząca dużego zapotrzebowania na energię, zwłaszcza w robotyce amatorskiej, gdzie powszechnie stosuje się tradycyjne pakiety akumulatorów, głównie Li-Po, niewątpliwie stawia wyzwanie przed praktycznością tych konstrukcji. Problem krótkiego czasu pracy na jednym ładowaniu stanowi istotną barierę, zwłaszcza w przypadku, gdy konieczne jest pokonywanie dłuższych dystansów. Dlatego istnieje potrzeba dalszego rozwoju technologii, takich jak opracowanie wydajniejszych źródeł zasilania, aby zwiększyć ogólny zasięg działania rozważanych konstrukcji.

Jednocześnie kluczowym elementem postępu jest rozwijanie skutecznych systemów sterowania, umożliwiających tym robotom pełne wykorzystanie ich potencjału. W temacie powstałych do tej pory tego typu konstrukcji warto wspomnieć o organizacji takich jak DARPA i Boston Dynamics które opracowały znane roboty kroczące dedykowane zastosowaniom wojskowym, jak chociażby projekt „BigDog” czy „LS3” [2]. Opracowane urządzenia miały pełnić funkcje towarzyszącego żołnierzom w terenie niedostępnym dla pojazdów kołowych, przenoszącego ładunek pojazdu w charakterze „konia” z zamiarem zmniejszenia obciążenia żołnierza. Mimo iż projekt „BigDog” ostatecznie uznano za zbyt głośny dla tego konkretnego zastosowania, uzyskane w jego ramach wyniki stały się ogromnym źródłem inspiracji dla środowiska badaczy tej kategorii robotyki.

Finalizując wprowadzenie do omawianego tematu, warto nakreślić motywację podjęcia się tego zagadnienia, która wynika z głębokiego zainteresowania robotami tego typu, ciekawości oraz chęci poszerzenia wiedzy w tym obszarze robotyki mobilnej. Dalsza część niniejszej pracy będzie koncentrować się na konkretnych aspektach projektowania i implementacji własnej konstrukcji omawianego robota.

## 1.1. Cel pracy

Główym celem niniejszej pracy jest zaprojektowanie i zaimplementowanie systemu sterowania dla robota kroczącego na czterech nogach. W ramach projektu zostanie zrealizowana zarówno konstrukcja mechaniczna robota, jak i oprogramowanie umożliwiające pełną kontrolę i sterowanie jego ruchami. Przedstawiony zostanie cały procesu projektowania oraz realizacji systemu sterowania dla urządzeń tego typu, mając na celu poszerzenie wiedzy w tym obszarze robotyki mobilnej.

## 1.2. Zakres pracy

Zakres pracy obejmuje przeprowadzenie badań dotyczących robotów kroczących, realizację konstrukcji mechanicznej, opracowanie układu elektrycznego odpowiedzialnego za zasilanie robota, implementację oprogramowania sterującego, wyniki testów działania robota, weryfikacja uzyskanych wyników pracy oraz szczegółową dokumentację techniczną projektu. Faza projektowa uwzględnia wcześniejszą analizę komponentów dostępnych na rynku konsumenckim w aktualnym czasie, pod kątem spełnienia wymagań technicznych oraz optymalizacji kosztów.

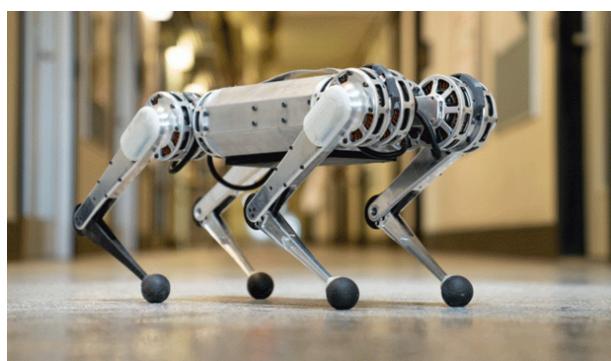
Praca została podzielona na kilka rozdziałów. Po wprowadzeniu przedstawionym we wstępie, w rozdziale drugim dokonano prezentacji kilku przykładowych konstrukcji robotów omawianego typu. Rozdział trzeci, stanowiący główną część pracy, w kolejnych punktach analizuje etapy budowy. Począwszy od przedstawienia założeń dotyczących projektowanego urządzenia, omawiane są konkretne etapy budowy, obejmujące dobór elementów, analizę konstrukcji oraz szczegółową implementację oprogramowania. W czwartym rozdziale przedstawione zostały rezultaty pracy, natomiast w ostatnim, piątym rozdziale, zawarte zostały wnioski wraz z końcowym podsumowaniem.

## **2. Przegląd istniejących konstrukcji**

W niniejszym rozdziale dokonano analizy kilku istniejących konstrukcji robotów kroczących rozważanego typu. Przeprowadzony zostanie staranny przegląd dostępnych źródeł dotyczących tych robotów, a następnie na ich podstawie przedstawione zostaną ich najciekawsze aspekty.

### **2.1. „Mini cheetah”, Instytut Technologii w Massachusetts**

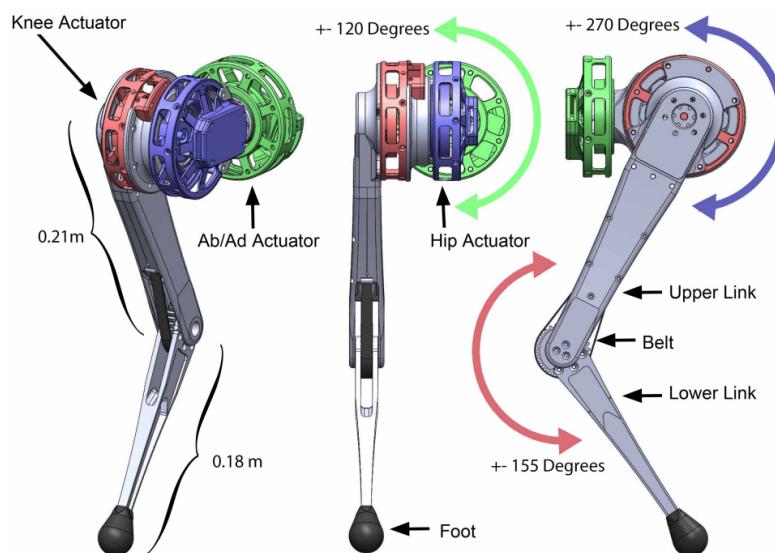
Robotem o wyjątkowej dynamice ruchu, zdolnym do wykonywania salt oraz podskakiwania jest „Mini Cheetah” – projekt opracowany przez badaczy z Instytutu Technologii w Massachusetts. Konstrukcja robota została zaprojektowana z myślą o modułowości, umożliwiając szybką wymianę uszkodzonych części, takich jak kończyny czy silniki. Korpus robota to lekka blacha aluminiowa, w którym mieści się akumulator, zasilacz logiczny, układ pomiarowy wychylenia kątowego (VN-100), odbiornik bezprzewodowy oraz komputer sterujący. Ogólny wygląd robota przedstawiono na rysunku 2.1. Program robota umożliwia automatyczne przywracanie równowagi po jej utracie z powodu działania siłowego.[3][4] Każda noga napędzana jest trzema silnikami, co zapewnia jej trzy stopnie swobody oraz rozległy zakres ruchu. Lekka konstrukcja, charakteryzująca się wysokim momentem obrotowym i niską bezwładnością, umożliwia robotowi wykonywanie szybkich, dynamicznych manewrów oraz uderzanie w ziemię z dużą siłą bez ryzyka uszkodzenia przekładni lub kończyn. Ten aspekt sprawia że pod względem mechanicznym robot jest szczególnie efektywny.[3][4]



**Rys. 2.1.** Robot "Mini cheetah", MIT. [3]

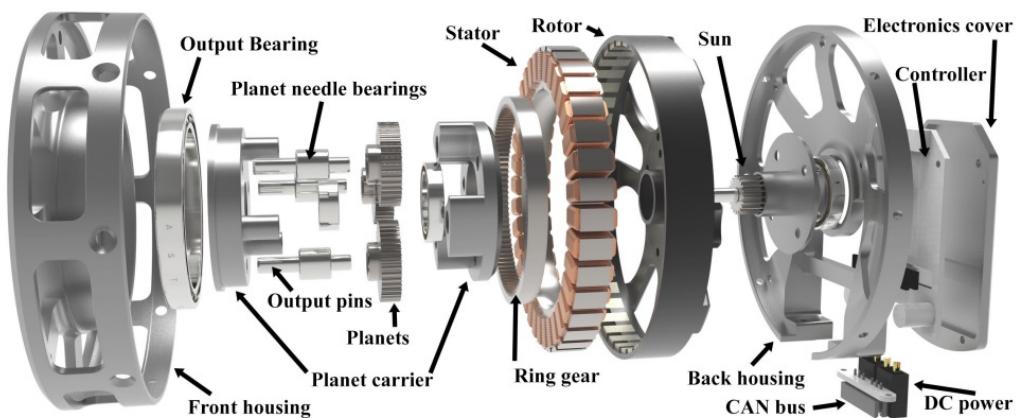
Pojedyncza noga robota została zaprojektowana tak, aby zmaksymalizować zakres ruchu przy jednocześnie minimalnej masie konstrukcji. Jej wygląd przedstawiono na rysunku 2.2. Zakres ruchu

umożliwia obrócenie robota do góry nogami oraz jego pracę zarówno w konfiguracji z kolanem do przodu, jak i kolanem do tyłu. Silniki biodra i kolan są umieszczone wspólnie w biodrzu, aby zminimalizować moment bezwładności. Wykorzystano przekładnie pasowe, umożliwiające przeniesienie momentu obrotowego na staw kolanowy zapewniająca dodatkowe przełożenie 1,55:1. Każda nogi jest w stanie wytworzyć siłę 150 N w najbardziej wymagającej konfiguracji. Ponadto nogi uzyskują stosunkowo duże przyspieszenia kątowe.



**Rys. 2.2.** Schemat przedstawiający nogę "Mini Cheetah", zielony - silownik ruchu poprzecznego, fioletowy - silownik biodrowy, czerwony - silownik kolanowy. [4]

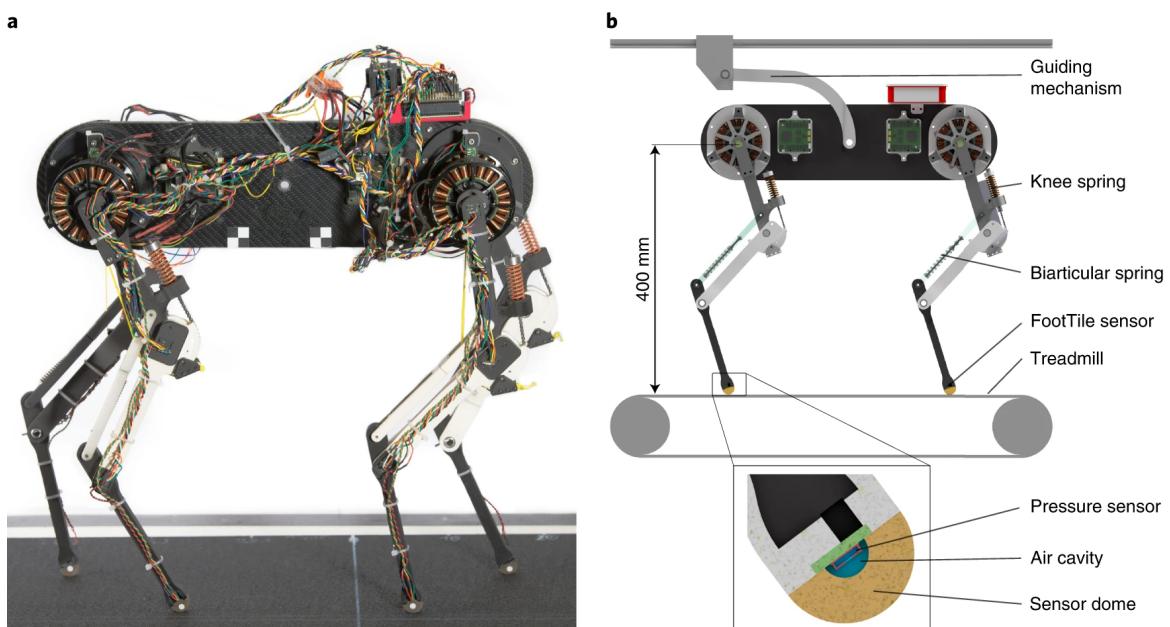
Rysunek 2.3 ilustruje pojedynczy układ napędowy robota który został zaprojektowany z myślą o uzyskaniu dużej gęstości momentu obrotowego. W każdym z tych układów znajduje się silnik elektryczny, jednostopniowa przekładnia planetarna o przełożeniu 6:1, oraz sterownik silnika ze zintegrowanym czujnikiem położenia. [4]



**Rys. 2.3.** Szczegółowy schemat budowy elementu wykonawczego. [4]

## 2.2. „*Morti*”, Instytut Maxa Plancka w Stuttgarcie

Naukowcy z Instytutu Maxa Plancka (MPI-IS) w Stuttgarcie przeprowadzili badanie, w którym skupili się na zrozumieniu procesu nauki chodzenia u zwierząt. W ramach tego badania powstał czworonożny robot o nazwie „*Morti*” którego wygląd został przedstawiony na rysunku 2.4. Autorzy wykorzystali w swojej pracy algorytm uczenia, który dostosowuje centralny generator wzorców za pomocą zamkniętej pętli sprzężenia zwrotnego. Robot „*Morti*” uczy się dopasowywać kontroler do swojej mechaniki i uczy się chodzić w ciągu jednej godziny, a także w krótkim czasie potrafi optymalizować swój ruch. W badaniu wykorzystano algorytm optymalizacji Bayesa, który kieruje procesem uczenia, dopasowując zmierzone informacje z czujnika stopy do danych docelowych. W trakcie nauki, robot stale porównuje wysłane i oczekiwane informacje z czujnika dostosowując wzorce kontroli motorycznej. [5]



**Rys. 2.4.** a - Robot "Morti", b – mocowanie robota do bieżni oraz przekrój czujnika wykrywającego kontakt z podłożem. [5]

Dane z czujników umieszczonych na stopach robota są ciągle porównywane z oczekiwany kontaktem z ziemią przewidywanym przez układ CPG (Centralny Generator Wzorców) robota. Jeżeli robot potknie się, algorytm uczący się modyfikuje odległość, jaką nogę przesuwa się w przód i w tył, szybkość ruchu nóg oraz czas, przez jaki nogi pozostaje na ziemi. Robot „*Morti*” jest wyjątkowo ciekawym przykładem ze względu na zastosowania technik uczenia maszynowego w sterowaniu robotem. [5]



### **3. Projekt robota kroczącego na czterech nogach**

W tym rozdziale przedstawiono etapy projektowania i realizacji robota kroczącego na czterech nogach. Opisano przyjęte założenia projektowe dotyczące konstrukcji mechanicznej oraz elementów elektronicznych robota.

#### **3.1. Założenia projektowe**

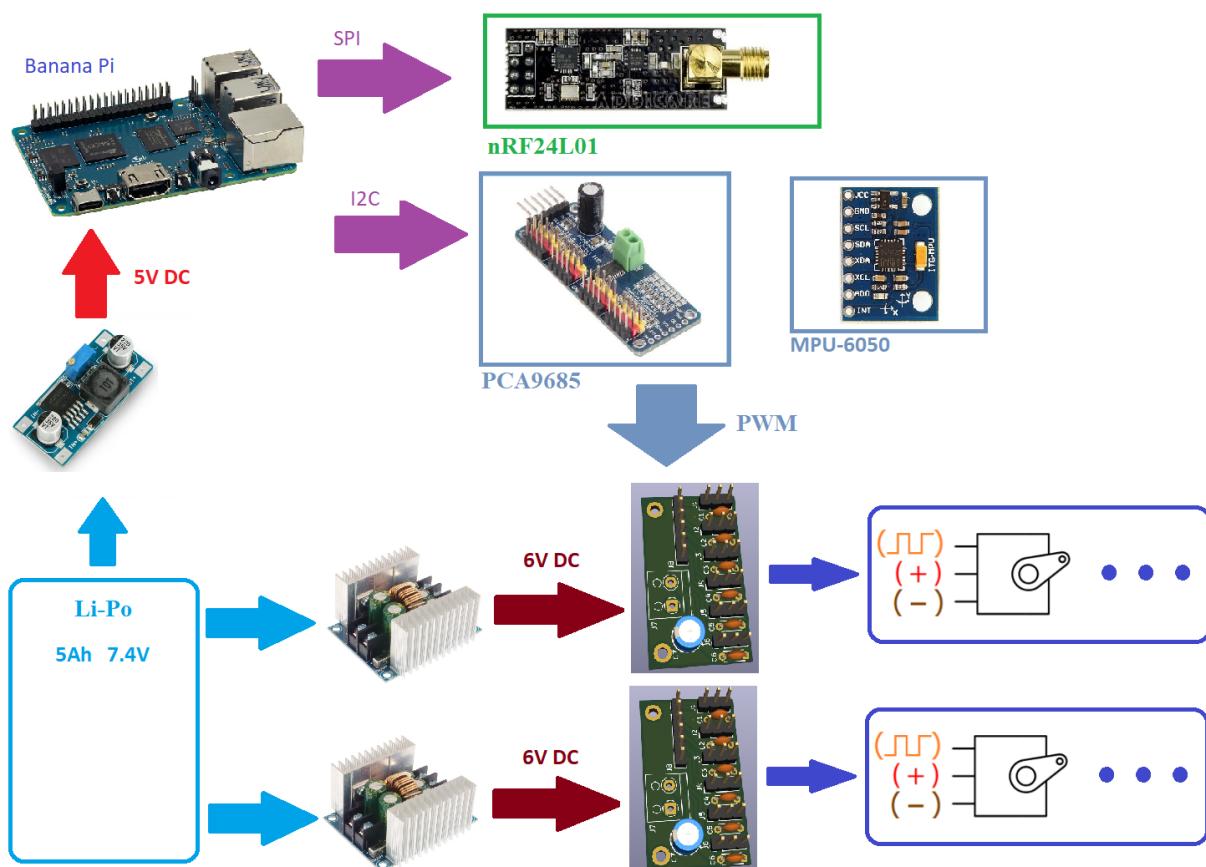
W niniejszym podrozdziale zostaną przedstawione główne założenia, które stanowią podstawę realizowanej pracy:

- Konstrukcja robota wyposażona w dwie pary nóg umożliwiające poruszanie się po różnego rodzaju nawierzchniach.
- System sterowania będzie odpowiedzialny za regulację i koordynację ruchów nóg, mając na celu osiągnięcie zamierzonej pozycji robota na podstawie wcześniej przetworzonych danych sterowania zadanych przez operatora.
- Komunikacja oraz sterowanie zdalne pozwolą na pełną mobilność robota oraz umożliwią jego kontrolę z pewnej odległości.
- Zasilanie baterijne pozwali na niezależne działanie urządzenia, eliminując konieczności jego stałego połączenia do zasilania sieciowego.
- Robot będzie zdolny do wykonywania podstawowych zadań, takich jak przemieszczanie się w wyznaczonym kierunku oraz osiąganie określonej pozycji.

## 3.2. Dobór elementów

W ramach tego etapu projektowania, po dokładnym określaniu wymagań, przeprowadzone zostaną rozważania dotyczące wyboru konkretnych elementów elektronicznych sterowania. Uwzględniając kompatybilność z resztą modułów, analiza wyboru elementów opierać się będzie na dostępnych komponentach na rynku, jednocześnie uwzględniając przewidziany budżet projektu.

Na rysunku 3.1 zaprezentowano schemat blokowy ilustrujący układ połączeń między poszczególnymi elementami. W trakcie procesu projektowania staranne rozważanie funkcjonowania całego układu ma na celu minimalizację ryzyka wystąpienia potencjalnych błędów w dalszych etapach realizacji projektu.



Rys. 3.1. Ilustracja koncepcyjnego schematu połączeń elektrycznych robota.

Począwszy od zasilania głównej jednostki sterującej, zasilanej napięciem 5V, doprowadzonym przez złącze USB typu C, przewiduje się podłączenie do portów posiadających funkcjonalność interfejsu SPI moduł nRF24L01. Z dostępnych peryferii wykorzystana będzie również magistrala I2C, podłączone zostaną przez nią moduł sterownika PCA9685, generującego sygnał PWM dla poszczególnych serwomechanizmów, a także moduł akcelerometru MPU-6050. Kolejnym etapem jest zasilanie części

wykonawczej za pomocą dwóch przetwornic ustawionych na wartość 6V. Zarówno zasilanie, jaki i sygnały poszczególnych kanałów, zostaną doprowadzone do płytka widocznych na rysunku 3.8, umożliwiających podłączenie kolejnych serwomechanizmów.

- **Część mechaniczno-elektryczna:**

Jako element wykonawczy odpowiedzialny za sterowanie ruchem nóg robota, zostaną wykorzystane serwomechanizmy modelarskie. Spośród dostępnych na rynku serwomechanizmów spełniających wymagane parametry, wybrano model MG-996R firmy Tower Pro przedstawiony na rysunku 3.2.



Rys. 3.2. Serwo TowerPro MG-996R (12sztuk).

Zakres kątowy pracy serwomechanizmu obejmuje wartości od 0 do 180 stopni, co jest zupełnie wystarczające dla analizowanej konstrukcji. Napięcie zasilania dla elementu roboczego zawiera się w przedziale od 4.8V do 7.2V.[6] W przypadku bezpośredniego podłączenia do w pełni naładowanego pakietu litowo-polimerowego, osiągającego w górnym zakresie 8.4V, przekraczamy ustalony zakres napięcia. W związku z powyższym konieczne staje się zastosowanie przetwornicy w celu dostosowania napięcia do określonego zakresu.

W warunkach normalnej pracy, pobór prądu mieści się w przedziale od 500 do 900 mA przy napięciu 6V. Natomiast w przypadku pracy pod obciążeniem, pobór prądu jest na poziomie 2.5 A przy tym samym napięciu. Układ sterowany jest za pomocą standardowego sygnału PWM dla serwomechanizmów modelarskich, charakteryzującego się częstotliwością 50 Hz oraz zakresem wypełnienia od 0 do 2 ms. Zgodnie z dostarczoną przez producenta dokumentacją techniczną, przy napięciu zasilającym wynoszącym 6V, serwomechanizm osiąga moment obrotowy o wartości 11kg·cm oraz prędkość 0.15s/60°. Parametry te stanowiły najbardziej istotne kryteria decyzyjne. [6]

- **Zasilanie:**

W celu zapewnienia stabilnego źródła zasilania, zdecydowano się na wykorzystanie pakietu litowo-polimerowego o pojemności 5 Ah, współczynnika rozładowania 50C oraz napięciu 7.4V. To rozwiązanie gwarantuje dostateczną wydajność prądową, jednocześnie stanowiąc pewne i wygodne źródło energii.

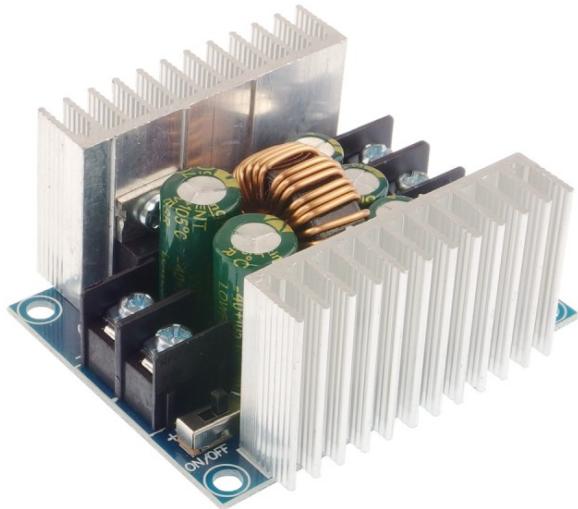
Zalecane przez producenta napięcie pracy dla serwomechanizmów oraz napięcie zasilania części logicznej jest niższe niż otrzymane napięcie z pakietu litowo-polimerowego które przy pełnym naładowaniu wynosi 8.4V. W związku z tym konieczne staje się użycie przetwornic typu „step down”, umożliwiających regulację napięcia do pożąданej wartości.

Do regulacji napięcia części logicznej zastosowana zostanie przetwornica typu step-down LM2596, przy założeniu, że pobierany prąd na wyjściu nie przekroczy 3 A. Przetwornica posiada zdolność do pracy w zakresie od 3,3 V do 35 V napięcia wejściowego oraz umożliwia uzyskanie maksymalnego ciągłego prądu wyjściowego wynoszącego 3 A przy zastosowaniu radiatorka.



Rys. 3.3. Przetwornica step-down LM2596S.[7]

Regulacja napięcia części wykonawczej, przy uwzględnieniu poboru prądu wynikającego z jej obciążenia o wartości 30 A, napotyka wyzwanie związane z kosztami przetwornic o odpowiedniej wydajności prądowej i wymaga odpowiedniego dostosowania tej części zasilania. W celu umożliwienia poboru tak dużej mocy, zdecydowano się na rozdzielenie zasilania dla każdej pary nóg. Taka konfiguracja znaczco redukuje zapotrzebowanie na moc dla pojedynczej przetwornicy do 15 A. Pozwoli to również na bardziej wygodne podłączenie oraz mniejsze wykorzystanie przestrzeni. W tym celu zostaną użyte dwie przetwornice step-down, których wygląd przedstawiono na rysunku 3.4. Wybrany model przetwornicy jest zdolny do pracy z maksymalnym ciągłym prądem wyjściowym wynoszącym 20A, co zapewni dostateczny zapas mocy. Przetwornica cechuje się napięciem wejściowym od 6 do 40 V, oraz regulacją napięcia wyjściowego w zakresie od 1.2 do 35 V, co jest dostateczne dla potrzeb projektu. W ramach niniejszego projektu wartość napięcia wyjściowego zostanie ustalona na 6V.

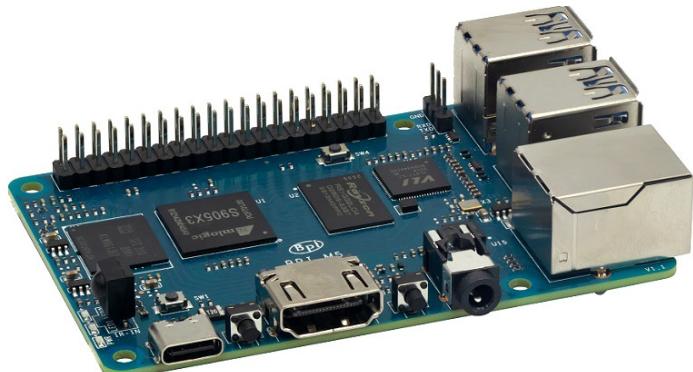


Rys. 3.4. Przetwornica Step-Down 20A.[8]

- **Część logiczna:**

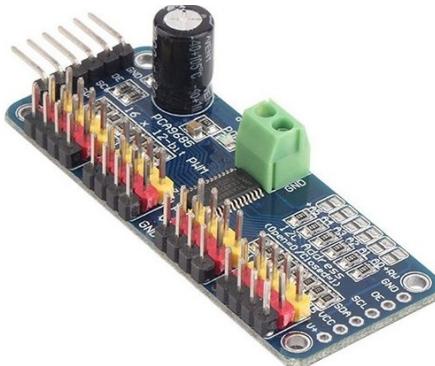
Jako centralną jednostkę obliczeniową wybrano jednopłytkowy komputer Banana PI M5, stanowiący aktualnie atrakcyjną alternatywę wśród komputerów jednopłytkowych, zwłaszcza w porównaniu do powszechnie wykorzystywanych układów Raspberry Pi. Wybór ten podyktowany jest zainteresowaniem różnorodnych rozwiązań w tej kategorii oraz został uzasadniony oferowanymi możliwościami tej jednostki. Minikomputer posiada czterordzeniowy procesor o częstotliwości taktowania 2GHz, 4Gb pamięci RAM, wbudowaną pamięć 16GB oraz slot karty SD obsługujący karty o pojemności do 2TB wraz z szeregiem dostępnych portów, co czyni go w pełni wystarczającym dla potrzeb niniejszej pracy. Dodatkowo, w przypadku dalszego rozwoju projektu, komputer Banana PI M5 gwarantuje dostateczny zapas mocy obliczeniowej.

Rysunek 3.5 przedstawia wygląd wybranego minikomputera. Banana PI M5 charakteryzuje się umiarkowanym poborem prądu wynoszącym 2A [5]. Nawet po uwzględnieniu dodatkowych modułów, których pobór energii jest stosunkowo niewielki, utrzymano się w akceptowalnym zakresie poboru mocy. Wyjścia ogólnego przeznaczenia (GPIO) są kompatybilne z popularnym układem Raspberry Pi, a producent gwarantuje również pełne wsparcie dla bibliotek dostępnych dla platform Raspberry Pi. Jedną z obsługiwanych dystrybucji systemu operacyjnego Linux na Banana Pi jest Ubuntu. Obraz systemu został pobrany ze strony producenta i zainstalowany na karcie SD o pojemności 64GB. Wybór Ubuntu umożliwia korzystanie z bogatej bazy bibliotek, co jest istotne w kontekście obsługi modułów niezbędnych do realizacji projektu.



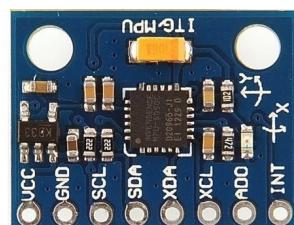
Rys. 3.5. Komputer jednopłytkowy Banana Pi BPI-M5.[9]

Do sterowania kolejnymi serwomechanizmami niezbędne jest wykorzystanie dodatkowego modułu sterownika PWM. Po przeprowadzeniu analizy dostępnych na rynku rozwiązań, z uwagi na niski koszt, podjęta została decyzja o wyborze modułu 16-kanałowego sterownika PWM o rozdzielczości 12 bitowej z interfejsem I<sub>2</sub>C, wykorzystującego układ PCA9685 [10]. Omawiany moduł został przedstawiony na rysunku 3.6.



Rys. 3.6. Sterownik serwomechanizmów PCA9685.[11]

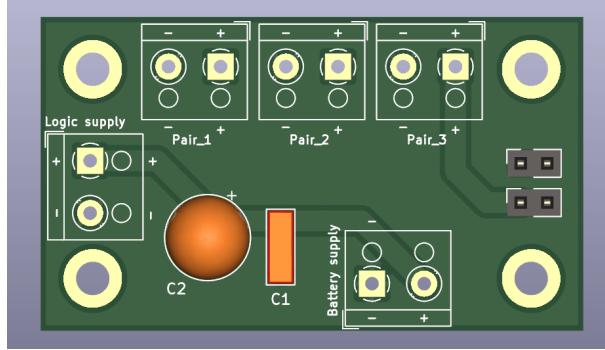
Dodatkowo, w celu przyszłego poszerzenia funkcjonalności projektu, zamontowany został moduł trzyosiowego akcelerometru MPU6050, którego ilustracja znajduje się na rysunku 3.7. Jego potencjalne wykorzystanie pozwoli na pozyskiwanie informacji zwrotnej dotyczącej płynności ruchu robota oraz umożliwi monitorowanie jego pochylenia oraz orientacji.



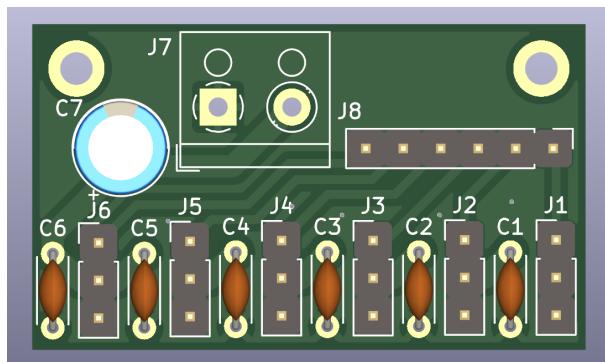
Rys. 3.7. Moduł MPU-6050.[12]

W celu usystematyzowania układu połączeń, przewiduje się wykonanie dwóch płyt drukowanych umożliwiających uporządkowanie przewodów zasilających oraz sygnałowych. Pierwsza z nich widoczna na rysunku 3.7, przeznaczona jest do rozdzielenia zasilania baterii poprzez przyłutowane terminale śrubowe dla poszczególnych par nóg, jak i również osobnego podłączenia wejścia przetwornicy zasilającej logikę. Druga płytka, przedstawiona na rysunku 3.8, została zaprojektowana z myślą o separacji doprowadzonego zasilania od linii sygnałowych PWM dla serwomechanizmów. Takie rozwiązanie pozwoli uniknąć konieczności stosowania złącz przedłużających kable od serwomechanizmów oraz bezpośredniego ich podłączenia do sterownika PWM, co mogłoby być kłopotliwe ze względu na wymagany duży pobór prądu. W ramach rozważanego rozwiązania do sterownika przyłutowane zostaną jedynie dwie listwy po 6 kabli sygnałowych, przeznaczonych dla poszczególnych par nóg.

W celu eliminacji potencjalnych fluktuacji zasilania wynikających z pracy silników, na obu płytach przewidziano przyłutowanie kondensatorów filtrujących. Ich obecność ma na celu zapewnienie stabilnego zasilania bez potencjalnych wahań.



Rys. 3.8. Płytki obwodu drukowanego służące do rozdzielenia zasilania logiki układu od części wykonawczej.



Rys. 3.9. Płytki obwodu drukowanego służące do rozdzielenia zasilania od linii sygnałowych serwomechanizmów.

- **Komunikacja:**

Komputer Banana PI M5 umożliwia komunikację przez sieć Wi-Fi przy użyciu protokołu SSH. Będzie to stanowić domyślny sposób uruchamiania skryptów języka Python, które będą obsługiwać wspomniane moduły oraz realizować komendy sterowania robotem. Ze względu na znaczne ograniczenia zasięgu sieci, zaimplementowano dodatkowy moduł radiowy, aby umożliwić sterowanie robotem z większej odległości. Sterowanie urządzeniem będzie odbywać się drogą radiową przy użyciu modułu nRF24L01, którego ilustracja została przedstawiona na rysunku 3.10. Moduł ten umożliwia dwustronną komunikację w paśmie 2.4 GHz, posiada wbudowany wzmacniacz oraz antenę, zapewniające moc nadajnika wynoszącą 22 dBm. Zasięg transmisji deklarowany przez producenta wynosi 1000 m. Układ komunikuje się z mikrokontrolerami poprzez interfejs SPI.[13]



Rys. 3.10. Moduł bezprzewodowy nRF24L01.[14]

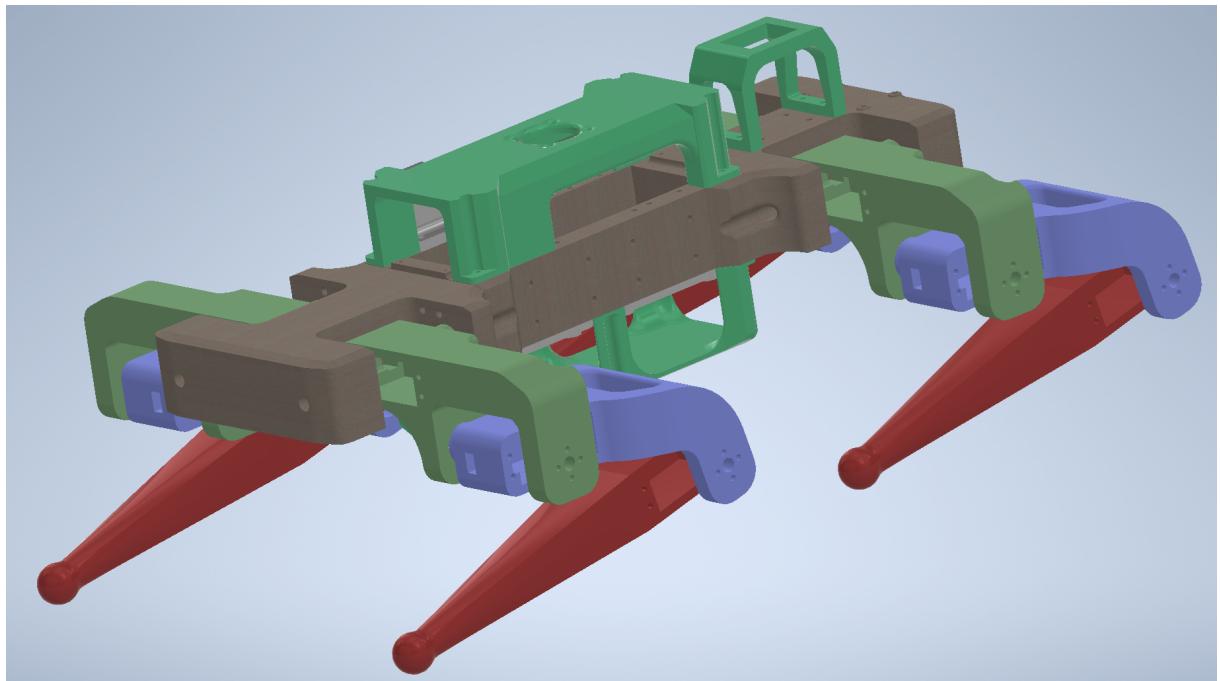
Od strony operatora konieczne jest również umożliwienie komunikacji poprzez podany moduł. W tym celu zostanie zbudowany pilot dla operatora, który będzie składał się z następujących elementów:

- Głównej jednostki sterującej, płytka rozwojowa „BlackPill” z mikrokontrolerem STM32F11CEU6,
- Dwóch modułów joysticków,
- Modułu komunikacji radiowej nRF24L01 pracujący w trybie nadajnika,
- Dwóch ogniw Li-ion Samsung INR18650 połączonych równolegle,
- Przetwornicy obniżającej napięcie do poziomu 3.3V.

W celu zaprogramowania układu zostanie wykorzystane środowisko programistyczne STM32CubeIDE.

### 3.3. Konstrukcja robota

Zaprojektowana konstrukcja robota została stworzona z priorytetem umieszczenia najcięższych elementów w samym jej środku, w celu zrównoważenia środka masy, co umożliwia równomierne obciążenie każdej z kończyn robota. Rysunek 3.11 prezentuje złożony model konstrukcji robota, wykonany w programie Autodesk Inventor.



Rys. 3.11. Złożony model konstrukcji robota.

Konstrukcja została wykonana z wykorzystaniem technologii druku 3D, a do tego celu zastosowano materiał PETG ze względu na jego korzystne właściwości mechaniczne.

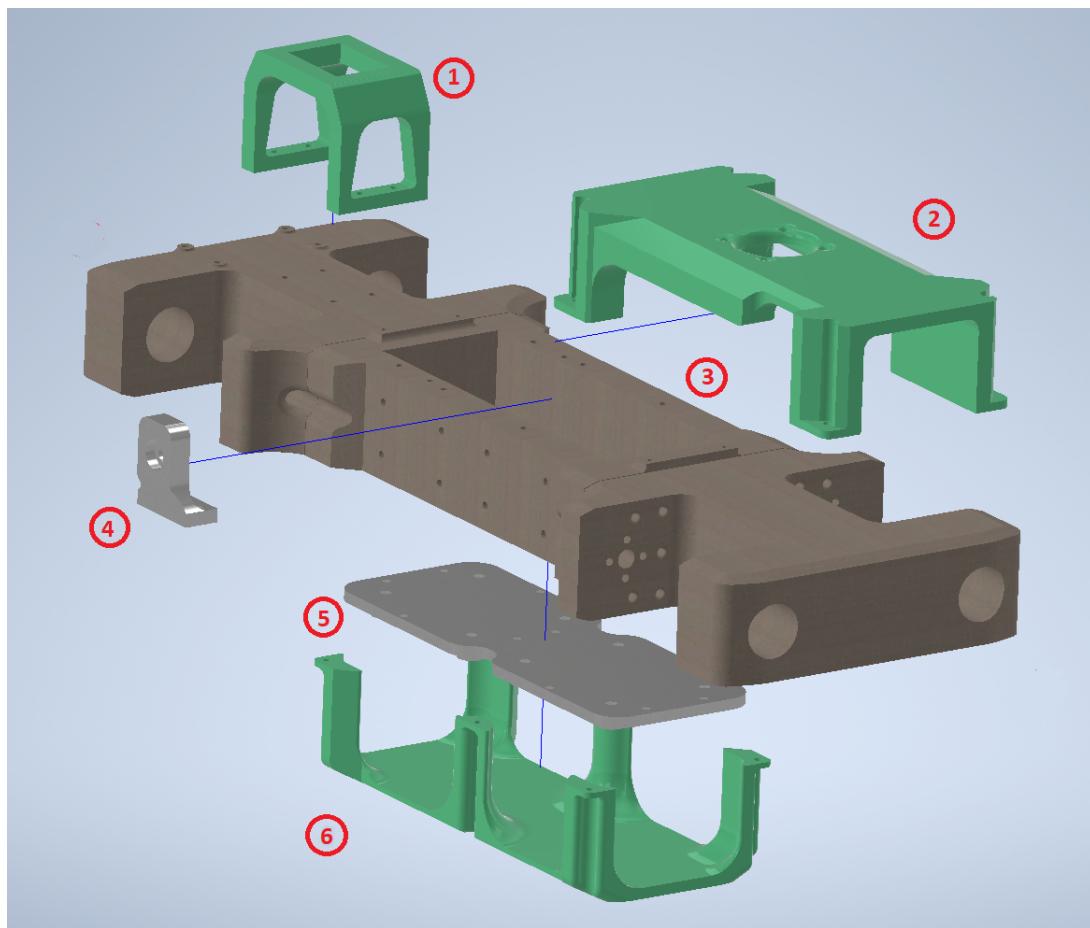
Na rysunku 3.12 przedstawiono poszczególne komponenty korpusu robota. Centralna część korpusu ze względu na znaczne rozmiary, została wydrukowana w trzech oddzielnych elementach, a następnie zmontowana w jedną całość przy użyciu śrub.

W centralnej części robota znajduje się miejsce na baterię, co jest uzasadnione jej znaczną masą. Nad baterią przewidziano przestrzeń przeznaczoną na zamontowanie głównej elektroniki sterującej, pozostały w tym miejscu otwory montażowe dla mosiężnych kołków dystansowych do których następnie przykręcono minikomputer oraz sterownik serwomechanizmów PCA9685.

W dolnej części robota zamontowano podstawę montażową do przykręcenia przetwornic dla części wykonawczej, natomiast w centralnej jej części znajduje się miejsce na przykręcenie modułu akcelerometru. W celu zabezpieczenia komponentów przed uszkodzeniami mechanicznymi wykonano dwie pokrywy. W górnej pokrywie przewidziano również przestrzeń na wentylator dla głównej jednostki sterującej, do której dodatkowo zamontowano radiatory w celu zapewnienia jeszcze skuteczniejszego chłodzenia.

W tylnej części robota został umieszczony uchwyt mocujący przeznaczony dla podwójnego, dwupozycyjnego przełącznika zasilania. Zastosowanie podwójnego przełącznika umożliwia niezależne uruchomienie części logicznej oraz wykonawczej. Takie rozwiązanie jest szczególnie korzystne podczas przeprowadzania testów.

Otwory montażowe przeznaczone dla przetwornicy części logicznej znajdują się poniżej przełącznika zasilania. Na tylnej części robota, za przełącznikiem zasilania, przewidziano miejsce przeznaczone dla płytki obwodu drukowanego (Rys. 3.8), która służy do rozdzielania zasilania. Natomiast płytki rozdzielające linie zasilania od linii sygnałowych serwomechanizmów (Rys. 3.9) zostały zamocowane na końcach obszaru wyznaczonego na baterię w głównej części korpusu robota.



Rys. 3.12. Poszczególne elementy korpusu robota.

Oznaczenia przedstawione na rysunku:

- 1 – Uchwyt mocujący dla dwupozycyjnego przełącznika zasilania,
- 2, 6 – Górná oraz dolna pokrywa,
- 3 – Główny korpus robota,
- 4 – Uchwyt mocujący modułu nRF24L01,
- 5 – Podstawa montażowa przeznaczona dla przetwornic step-down które zasilają część roboczą, wraz z umieszczonymi centralnie na środku otworami do montażu modułu akcelerometru MPU6050.

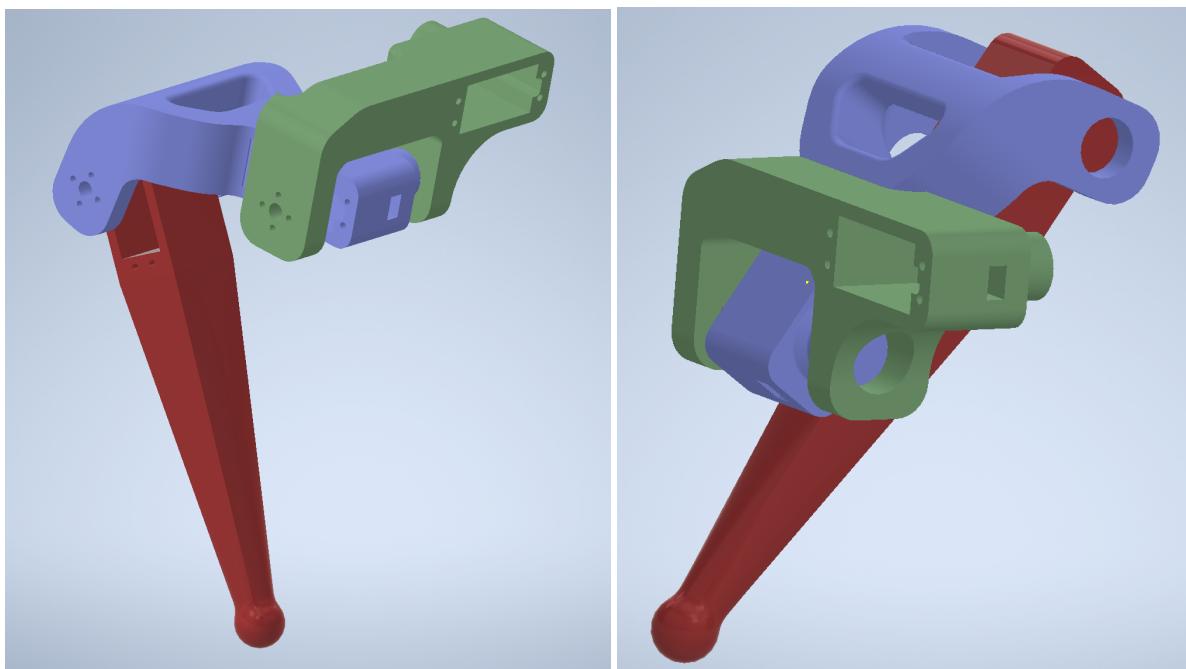
Konstrukcja pojedynczej nogi była projektowana z myślą o późniejszym, pełnym wykorzystaniu zakresu pracy serwomechanizmów oraz umożliwieniu robotowi samodzielnego przyjmowania pozycji pionowej. W tym przypadku wzorowano się na przykładzie konstrukcji nogi robota "Mini Cheetah" (Rys. 2.2). Przyjęte zostały następujące oznaczenia kolorystyczne, widoczne na rysunku 3.13:

- zielony - mocowanie serwomechanizmu realizującego ruch poprzeczny,
- fioletowy - mocowanie serwomechanizmu realizującego ruch stawu biodrowego,
- czerwony - mocowanie serwomechanizmu realizującego ruch stawu kolanowego.

W przypadku robota "Mini Cheetah", z uwagi na większą masę pojedynczego układu silnika, moment obrotowy na stawie kolanowym jest przenoszony za pomocą dodatkowej przekładni. Masa wykorzystanego w projekcie serwomechanizmu jest stosunkowo niewielka, co umożliwia uproszczenie konstrukcji poprzez umieszczenie układu napędowego w miejscu stawu kolanowego.

Elementy nóg zostały zaprojektowane tak, aby osie obrotu serwomechanizmów były równomiernie rozmieszczone pod względem wymiarowym. Kolejne komponenty są ze sobą łączone przez przykręcenie łącznika serwomechanizmów z jednej strony oraz za pomocą wydrukowanego wałka z drugiej strony.

Dla wałka został przewidziany otwór, tworzący rodzaj połączenia w postaci łożyska ślizgowego. W celu zminimalizowania tarcia, elementy połączenia zostały lekko spolerowane drobnym papierem ściernym. Dodatkowo, nałożono niewielką ilość smaru. W rezultacie połączenie to charakteryzuje się płynnym działaniem.

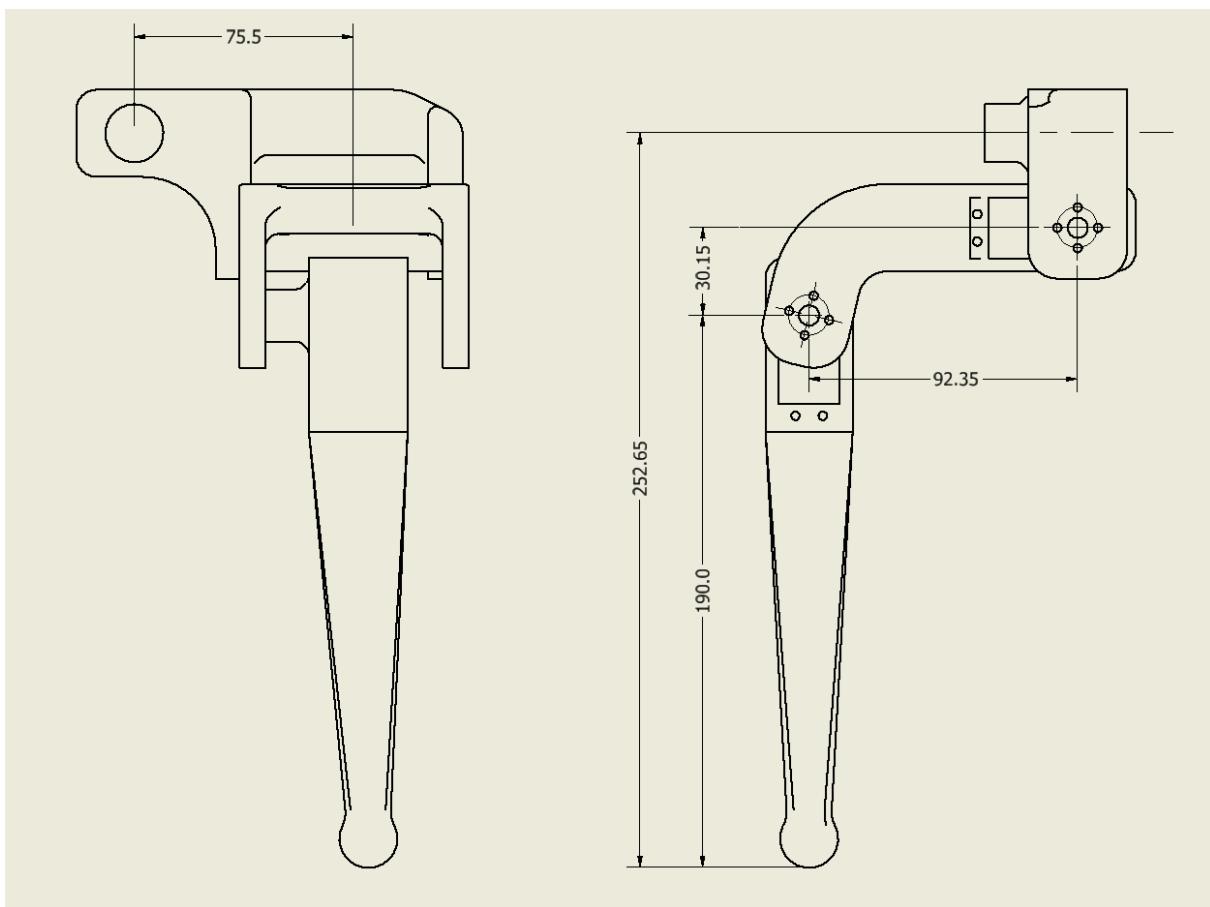


Rys. 3.13. Konstrukcja pojedynczej nogi.

Konfiguracja nogi widoczna po prawej stronie na Rys. 3.13 ilustruje skrajne położenia stawu biodrowego oraz kolanowego, które wynoszą  $0^\circ$  w ustawieniach serwomechanizmów. To umożliwia

pełne wykorzystanie zakresu pracy serwomechanizmów oraz ich mechaniczne odciążenie, ponieważ kolejne elementy opierają się wzajemnie.

Na Rys. 3.14 zaprezentowano zwymiarowany fragment kończyny. W kontekście implementacji programu sterującego, kluczowe jest dokładne określenie wymiarów odległości końca danej części od osi obrotu.



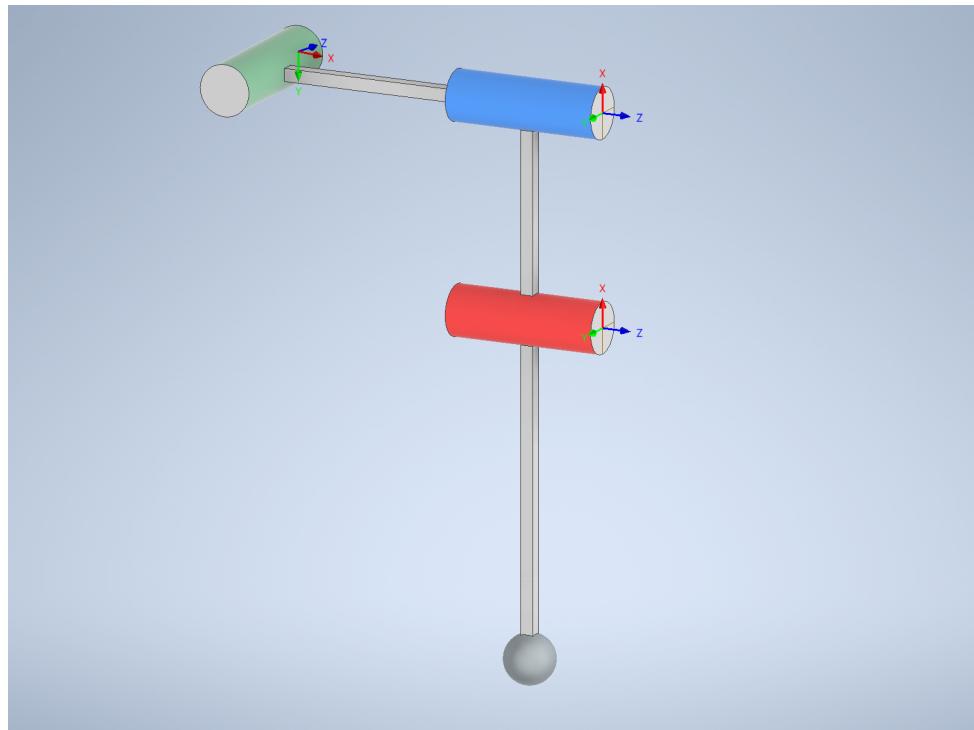
Rys. 3.14. Schemat ilustrujący najistotniejsze wymiary.

Przyjęte zostały oznaczenia dla poszczególnych wymiarów, przy czym wartości zostały podane w jednostce milimetrów (mm):

$$\begin{aligned}
 l_1 &= 75.5 \\
 h_1 &= 32.5 \\
 l_2 &= 92.35 \\
 h_2 &= 30.15 \\
 l_3 &= 190
 \end{aligned} \tag{3.1}$$

### 3.4. Kinematyka prosta

W tym rozdziale przedstawiono opis matematycznego modelu kinematyki pojedynczej nogi robota. Na rysunku 3.15 został zaprezentowany uproszczony model nogi. Bazując na notacji Denavit-Hartenberga, przedstawiono jednoznaczną reprezentację kinematyki manipulatora. Parametry tej notacji, przedstawione w tabeli 3.1, opisują one geometryczne właściwości poszczególnych przegubów.



**Rys. 3.15.** Uproszczony model nogi robota.

**Tabela 3.1.** Tabela parametrów notacji Denavita-Hartenberga

Joint $i$	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	0	$l_1$	0
3	$-\frac{\pi}{2}$	0	0	$-\frac{\pi}{2}$
2	$\theta_2$	0	$-l_2$	0
3	$\theta_3$	0	$-l_3$	0

Występujące oznaczenia:

$\theta_i$  – zmienna przegubowa osi "z",

$d_i$  – przesunięcie w osi "z",

$a_i$  - przesunięcie w osi "x",

$\alpha_i$  - rotacja w osi "x".

W celu usprawnienia obliczeń, napisany został program z wykorzystaniem języka Python, opierający się na równaniach kinematyki manipulatora, zapisanych przy użyciu notacji Denavit-Hartenberga. Przedstawione równanie (3.2) opisuje transformacje jednego ogniw manipulatora umożliwiające wyznaczenie macierzy  $A_i$ , reprezentującej kinematykę  $i$ -tego przegubu.

$$A_i = R_{z,\theta_i} \cdot T_{z,d_i} \cdot T_{x,a_i} \cdot R_{x,\alpha_i} \quad (3.2)$$

gdzie macierze  $R$  i  $T$  są odpowiednio macierzami rotacji i przesunięć w przestrzeni trójwymiarowej, zdefiniowanymi według notacji Denavit-Hartenberga. Równanie (3.3) stanowi formalne przedstawienie szczegółowej postaci poszczególnych macierzy, na podstawie których został stworzony wspomniany program.

$$A_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

Ostateczna macierz reprezentacji kinematyki manipulatora, według notacji Denavit-Hartenberga, powstaje poprzez przemnożenie kolejnych macierzy  $A_i$ , które reprezentują poszczególne przeguby. Ostateczna forma wynikowej macierzy jest postaci:

$$K(\boldsymbol{\theta}) = \begin{bmatrix} R(\boldsymbol{\theta}) & T(\boldsymbol{\theta}) \\ 0 & 1 \end{bmatrix} \quad (3.4)$$

gdzie macierz  $R(\boldsymbol{\theta})$  określa orientację w przestrzeni wyrazoną w bazowym układzie współrzędnych, a wektor  $T(\boldsymbol{\theta})$  określa położenie efektora wybrane w bazowym układzie współrzędnych [15]. W dalszej części pracy wykorzystany został wektor  $T(\boldsymbol{\theta})$  postaci (3.5), gdzie  $\boldsymbol{\theta}$  to wektor zmiennych kątowych kolejnych przegubów.

$$T(\boldsymbol{\theta}) = \begin{bmatrix} X(\theta_1, \theta_2, \theta_3) \\ Y(\theta_1, \theta_2, \theta_3) \\ Z(\theta_1, \theta_2, \theta_3) \end{bmatrix} = \begin{bmatrix} l_1 \cdot \cos(\theta_1) - l_2 \cdot \sin(\theta_1) \cdot \cos(\theta_2) + l_3 \cdot \sin(\theta_1) \cdot \sin(\theta_2) \cdot \sin(\theta_3) - l_3 \cdot \sin(\theta_1) \cdot \cos(\theta_2) \cdot \cos(\theta_3) \\ l_1 \cdot \sin(\theta_1) + l_2 \cdot \cos(\theta_1) \cdot \cos(\theta_2) - l_3 \cdot \sin(\theta_2) \cdot \sin(\theta_3) \cdot \cos(\theta_1) + l_3 \cdot \cos(\theta_1) \cdot \cos(\theta_2) \cdot \cos(\theta_3) \\ l_2 \cdot \sin(\theta_2) + l_3 \cdot \sin(\theta_2) \cdot \cos(\theta_3) + l_3 \cdot \sin(\theta_3) \cdot \cos(\theta_2) \end{bmatrix} \quad (3.5)$$

### Analiza poprawności uzyskanego rozwiązania

Aby zweryfikować poprawność uzyskanych wyników z notacji Denavit-Hartenberga, przeprowadzono serię testów, które polegały na podstawieniu konkretnym wartości zmiennych kątowych  $\theta$  do kolejnych wzorów przedstawionych w równaniu (3.5). Weryfikacja wyników obejmowała kilka konfiguracji wartości kątów  $\theta_1$ ,  $\theta_2$ ,  $\theta_3$ , dla których orientacyjnie można było przewidzieć pozycję efektora manipulatora względem początkowego układu współrzędnych.

Punkt dla kątów  $\theta$  równych  $0^\circ$ :

$$\begin{bmatrix} X(0, 0, 0) \\ Y(0, 0, 0) \\ Z(0, 0, 0) \end{bmatrix} = \begin{bmatrix} 75.5 \\ 282.35 \\ 0 \end{bmatrix} \quad (3.6)$$

W ustawieniach przekazanych jako dane wejściowe, oczekuje się, że układ znajdzie się w pozycji bazowej, zgodnej z modelem przedstawionym na rysunku 3.15. Analizując wyniki, zauważa się, że zgodnie z bazowym układem współrzędnych pozycja efektora manipulatora osiąga punkt na osi  $x$ , którego wartość wynosi 75.5, co jest zgodne z długością  $l_1$ . Ponadto, wartość punktu na osi  $y$  jest równa sumie długości  $l_2$  i  $l_3$  ramion manipulatora, a wartość na osi  $z$  wynosi zero. Uzyskany wynik jest zgodny z ułożeniem manipulatora.

Punkt dla wartości kątów  $\theta_1 = 0^\circ$ ,  $\theta_2 = 90^\circ$ ,  $\theta_3 = 0^\circ$ :

$$\begin{bmatrix} X(0, 90, 0) \\ Y(0, 90, 0) \\ Z(0, 90, 0) \end{bmatrix} = \begin{bmatrix} 75.5 \\ 1.728895118696276e - 14 \\ 282.35 \end{bmatrix} \quad (3.7)$$

Otrzymane wyniki są bardzo zbliżone do tych uzyskanych w pierwszym teście. Jednakże, w tym przypadku, po obrocie drugiego przegubu o  $90^\circ$  zgodnie z regułą prawej dłoni, osiągnięta została wartość 282.35 w osi  $z$ , przy czym wartość  $y$  można w przybliżeniu przyjąć jako zero. Ten przykład dodatkowo potwierdza poprawność równań 3.5.

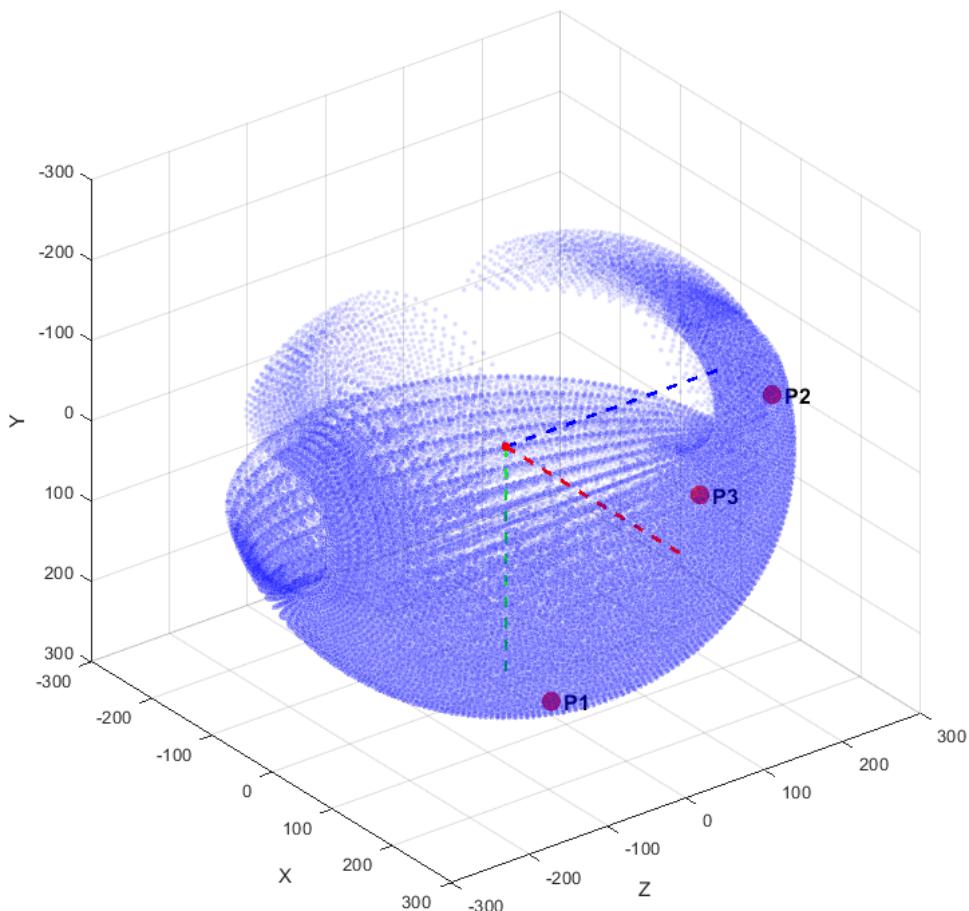
Punkt dla wartości kątów  $\theta_1 = 0^\circ$ ,  $\theta_2 = 0^\circ$ ,  $\theta_3 = 90^\circ$ :

$$\begin{bmatrix} X(0, 0, 90) \\ Y(0, 0, 90) \\ Z(0, 0, 90) \end{bmatrix} = \begin{bmatrix} 75.5 \\ 92.35000000000001 \\ 190.0 \end{bmatrix} \quad (3.8)$$

W tym przykładzie ustalono ostatni z przegubów manipulatora na wartość  $90^\circ$ , uzyskany wynik również jest poprawny. Na podstawie przeanalizowanych przykładów wnioskuje, że wzory uzyskane przy użyciu notacji Denavit-Hartenberga działają poprawnie.

### Wyznaczenie obszaru roboczego

W kontekście planowania trajektorii manipulatora kluczowe jest określenie jego osiągalnej przestrzeni roboczej. Wykorzystując równania (3.5), opisujące kinematykę manipulatora, napisany został skrypt Matlab'a, który umożliwia wygenerowanie zbioru punktów reprezentujących osiągalną przestrzeń manipulatora, oraz wizualizację na ich podstawie obszaru roboczego na wykresie.

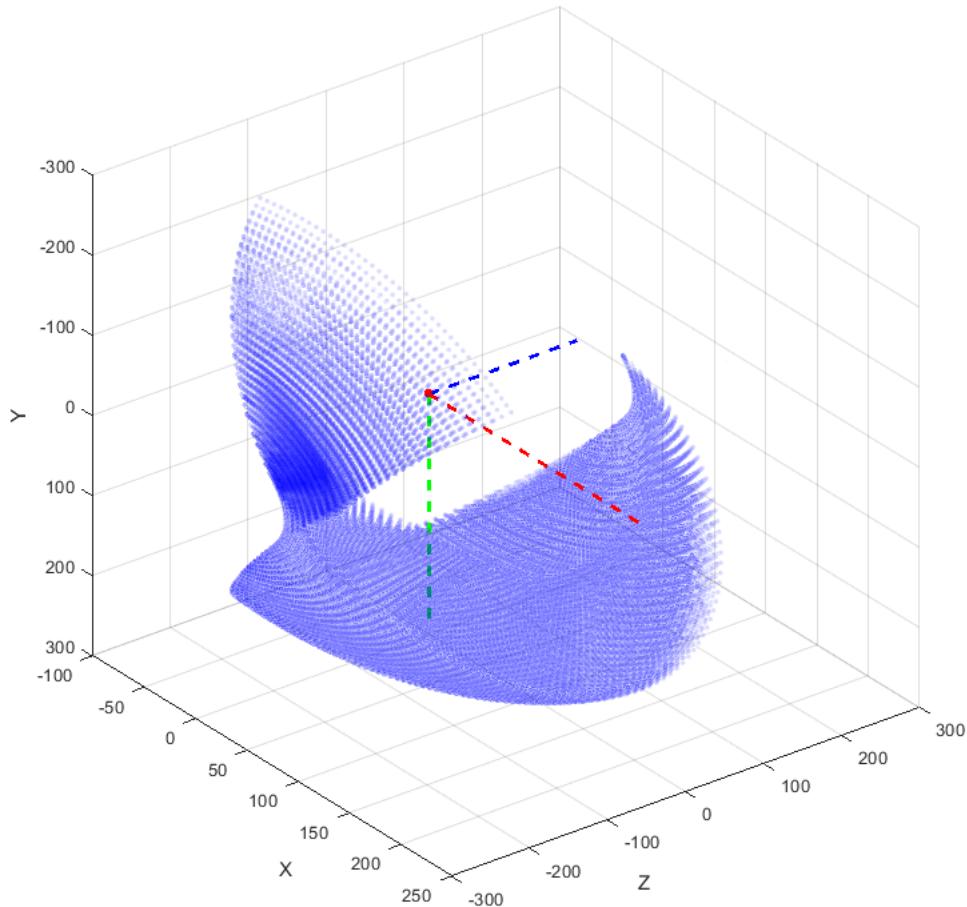


Rys. 3.16. Wykres przestrzeni roboczej manipulatora.

Wykres przedstawiony na rysunku 3.16 obejmuje zakres wartości kątów  $\theta$  w przedziale  $\langle -90^\circ, 90^\circ \rangle$ . W celu zwiększenia czytelności oraz lepszej orientacji, wykres został dostosowany do bazowego układu współrzędnych. Na wykresie zaznaczone zostały punkty  $P_1$ ,  $P_2$ ,  $P_3$ , które reprezentują kolejne wartości, odpowiadające analizowanym parametrom z poprzedniego etapu badawczego. Przy analizie wykresu zaauważalne jest, że punkty  $P_1$  oraz  $P_3$  osiągają skrajne położenia, odpowiadające maksymalnej odległości, jaką efektor manipulatora może osiągnąć. Na wykresie uwidoczniona jest sfera, której środek, w pewnej odległości odpowiadającej długości pierwszego członu nogi  $l_1$ , jest pozbawiony punktów. Rozważenie obszaru, w którym efektor może się poruszać, jest kluczowe w kontekście późniejszego planowania jego ruchu oraz weryfikacji poprawności uzyskanych wartości.

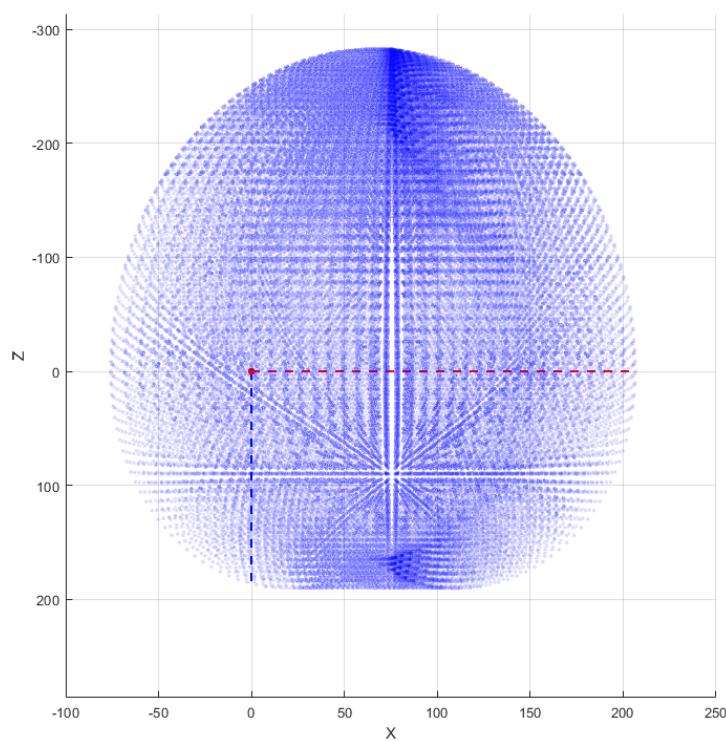
Uproszczony model nogi przedstawiony na rysunku 3.15 został analizowany w określonej konfiguracji, w której rzeczywiste nastawy serwomechanizmów odpowiadały wartościami  $\theta_1 = 90^\circ$ ,  $\theta_2 = 150^\circ$ ,  $\theta_3 = 180^\circ$ . Zakres pracy napędów jest ograniczony do przedziału  $\langle 0^\circ, 180^\circ \rangle$ . W związku z przyjętymi początkowymi nastawami, konieczne jest przedefiniowanie zakresu pracy, uwzględniając rzeczywiste wartości nastaw serwomechanizmów. Istotne jest także określenie podzbioru wartości ustnień, które rzeczywiście będą wykorzystywane podczas pracy robota. Wizualizacja obszaru roboczego dla tych wartości została przedstawiona na rysunku 3.17, co pozwoli uzyskać pełniejszy obraz obszaru ruchu, w którym planujemy operować. W równaniu (3.9) zostały przedstawione dwie rozważane dziedziny wartości.

$$\begin{aligned}\theta_1 &= \langle -90^\circ, 90^\circ \rangle \supseteq \langle -30^\circ, 30^\circ \rangle \\ \theta_2 &= \langle -150^\circ, 30^\circ \rangle \supseteq \langle -150^\circ, 0^\circ \rangle \\ \theta_3 &= \langle 0^\circ, 180^\circ \rangle \supseteq \langle 0^\circ, 120^\circ \rangle\end{aligned}\quad (3.9)$$

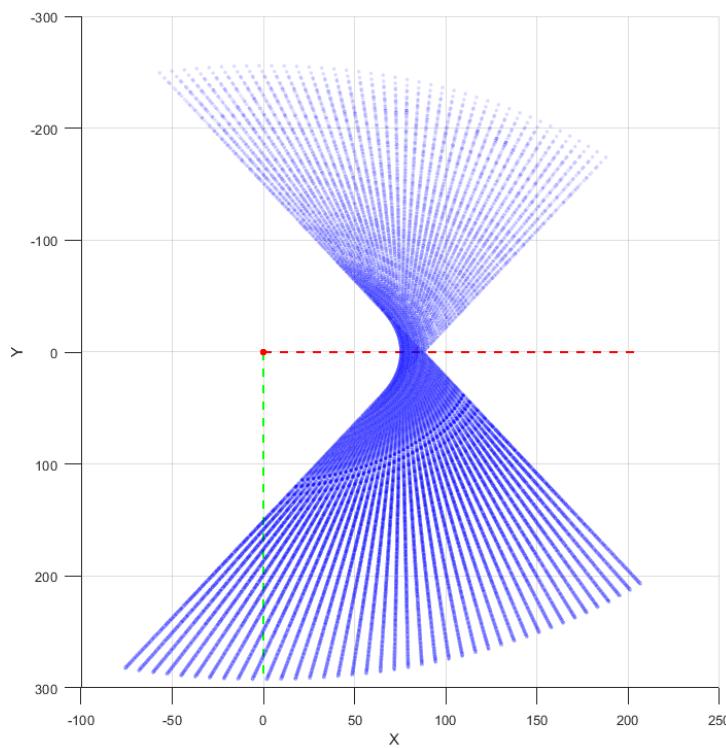


Rys. 3.17. Wykres użytecznej przestrzeni roboczej manipulatora.

Rysunki oznaczone jako 3.18 oraz 3.19 przedstawiają najbardziej istotne rzuty dwuwymiarowe. W porównaniu do orientacyjnego wykresu z rysunku 3.17, granice osiągalnych wartości są na nich znacznie bardziej wyraźne.



Rys. 3.18. Wykres płaszczyzny  $XZ$ .



Rys. 3.19. Wykres płaszczyzny  $XY$ .

### 3.5. Kinematyka odwrotna

Do obliczeń przybliżonych wartości kinematyki odwrotnej zastosowano iteracyjną metodę Newtona-Raphsona [16]. Wzór rekurencyjny algorytmu w przypadku funkcji wielu zmiennych jest wyrażony następująco:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - (F'(\boldsymbol{x}_n))^{-1} F(\boldsymbol{x}_n) \quad (3.10)$$

Wzór dla rozważanego przypadku, w którym poszukiwanymi zmiennymi są kolejne kąty  $\theta_1, \theta_2, \theta_3$ .

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}_{n+1} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}_n - \begin{bmatrix} \frac{\partial F_x}{\partial \theta_1} & \frac{\partial F_x}{\partial \theta_2} & \frac{\partial F_x}{\partial \theta_3} \\ \frac{\partial F_y}{\partial \theta_1} & \frac{\partial F_y}{\partial \theta_2} & \frac{\partial F_y}{\partial \theta_3} \\ \frac{\partial F_z}{\partial \theta_1} & \frac{\partial F_z}{\partial \theta_2} & \frac{\partial F_z}{\partial \theta_3} \end{bmatrix}_n^{-1} \cdot \begin{bmatrix} F_x(\theta_1, \theta_2, \theta_3) - X \\ F_y(\theta_1, \theta_2, \theta_3) - Y \\ F_z(\theta_1, \theta_2, \theta_3) - Z \end{bmatrix}_n \quad (3.11)$$

Przyjęto następujące warunki zakończenia obliczeń :

1. Uzyskanie dostatecznej dokładności:

$$|F(\boldsymbol{x}_n)| \leq \varepsilon \quad (3.12)$$

2. Dostatecznie mała zmiana kolejnych wartości przybliżenia:

$$|\boldsymbol{x}_{n+1} - \boldsymbol{x}_n| \leq \delta \quad (3.13)$$

3. Osiągnięcie maksymalnej liczby iteracji:

$$n \leq N_{\max} \quad (3.14)$$

Algorytm na początku wymaga określenia wartości początkowych. W poszukiwaniu rozwiązania ograniczono się do użytecznego zakresu pracy serwomechanizmów, który przedstawionego w równaniu (3.9). Początkowe wartości przyjęto jako połowa tego zakresu.

$$\theta_1 = 0^\circ, \theta_2 = 60^\circ, \theta_3 = -60^\circ \quad (3.15)$$

Metoda Newtona-Raphsona w ogólnym przypadku jest stosowana do przybliżonego wyznaczania miejsc zerowych funkcji. W kontekście obliczeń kinematyki odwrotnej, konieczna jest jej modyfikacja, w ramach której od wartości uzyskanych z funkcji dla kolejnych współrzędnych odejmowane są kolejno współrzędne kartezjańskie, dostarczane jako dane wejściowe. Opisane działanie widoczne jest w równaniu (3.11).

Algorytm został zaimplementowany przy użyciu języka Python. Wartości parametrów algorytmu zostały ustalone jako:

$$\begin{aligned} \varepsilon &= 0.1 \\ \delta &= 0.01 \\ N_{\max} &= 100 \end{aligned} \quad (3.16)$$

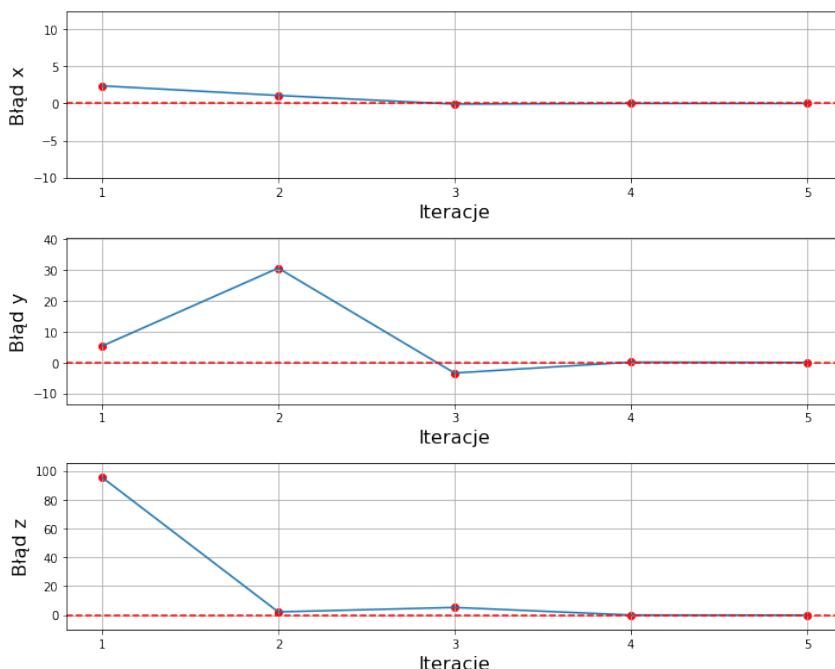
W kontekście implementacji systemu sterowania robotem kroczącym, planowanie trasy będzie realizowane przez określenie dwóch punktów w przestrzeni kartezjańskiej. Na ich podstawie zostanie wyznaczonych osiem punktów pośrednich. Dla każdego z tych punktów będą obliczane konkretne wartości nastaw kątów  $\theta$  przy użyciu przedstawionego wcześniej algorytmu. W celu optymalizacji wydajności obliczeniowej, opracowana trasa zostanie zapisana w formie tabelarycznej struktury danych w kodzie programu.

#### **Wyniki oraz Analiza poprawności działania algorytmu:**

Na podstawie wykresu 3.17, reprezentującego użyteczną przestrzeń roboczą, w celu wygenerowania trasy symulującej pojedynczy krok nogi robota, zostały wybrane kolejne punkty, gdzie operowaną jednostką definiowania współrzędnych jest milimetr. Wartości współrzędnych tych punktów zostały przedstawione w równaniu (3.17):

$$\begin{aligned} P_1 &= (80, 240, -50) \\ P_2 &= (80, 150, 0) \\ P_3 &= (80, 100, 150) \\ P_4 &= (80, 190, 150) \end{aligned} \quad (3.17)$$

Rysunek 3.20 ilustruje wykres przedstawiający przebieg wartości błędu, wyrażonego w milimetrach, podczas obliczania przez algorytm jednego z punktów pośrednich trasy.

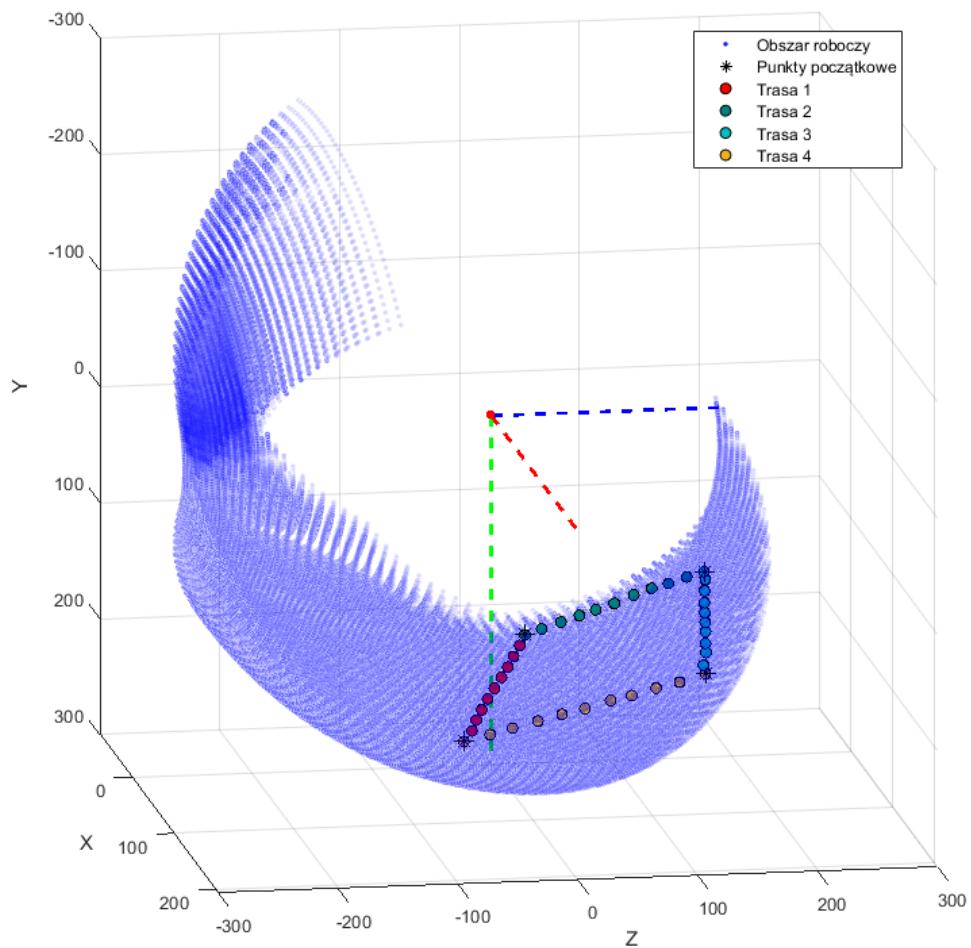


**Rys. 3.20.** Wykres przedstawiający wartość błędu obliczeń.

Jako przykład wybrano jeden z punktów pośrednich o współrzędnych (80, 116, 120), a uzyskany wynik wynosi (80, 116, 119.99). Uzyskane wartości kątów w stopniach:  $\theta_1 = -2.18^\circ$ ,  $\theta_2 = 133.44^\circ$ ,  $\theta_3 = -117.28^\circ$ . W trakcie kolejnych iteracji algorytm przyjmuje jako punkt startowy wartość, która została wcześniej obliczona. Przyspiesza to proces działania algorytmu, ponieważ, znajdując się w dośćczelnie bliskim sąsiedztwie rozwiązania, algorytm szybko zbiega do tego punktu. W wyniku tego podejścia, ilość kolejnych obliczeń ogranicza się do stosunkowo niewielkiej liczby iteracji.

Z analizowanego wykresu przebiegu błędu na rysunku 3.20, jak i również z obserwacji wykresów błędu dla innych punktów, wynika, że do osiągnięcia akceptowalnej wartości błędu, zdefiniowanej w równaniu (3.16), algorytm potrzebuje średnio od 3 do 6 iteracji.

Na rysunku 3.21 została przedstawiona trasa pomiędzy punktami z równania (4.1). Analizując ten wykres, można wysnuć wniosek o poprawności działania algorytmu. Obserwuje się jednakże drobne odchylenia punktów od prostej linii. Tak niewielki błąd można uznać za pomijalny w kontekście rozważanego projektu robota. Pracując z jednostkami milimetra, nawet minimalne odstępstwa od wartości są fizycznie niezauważalne i nie mają praktycznego znaczenia.



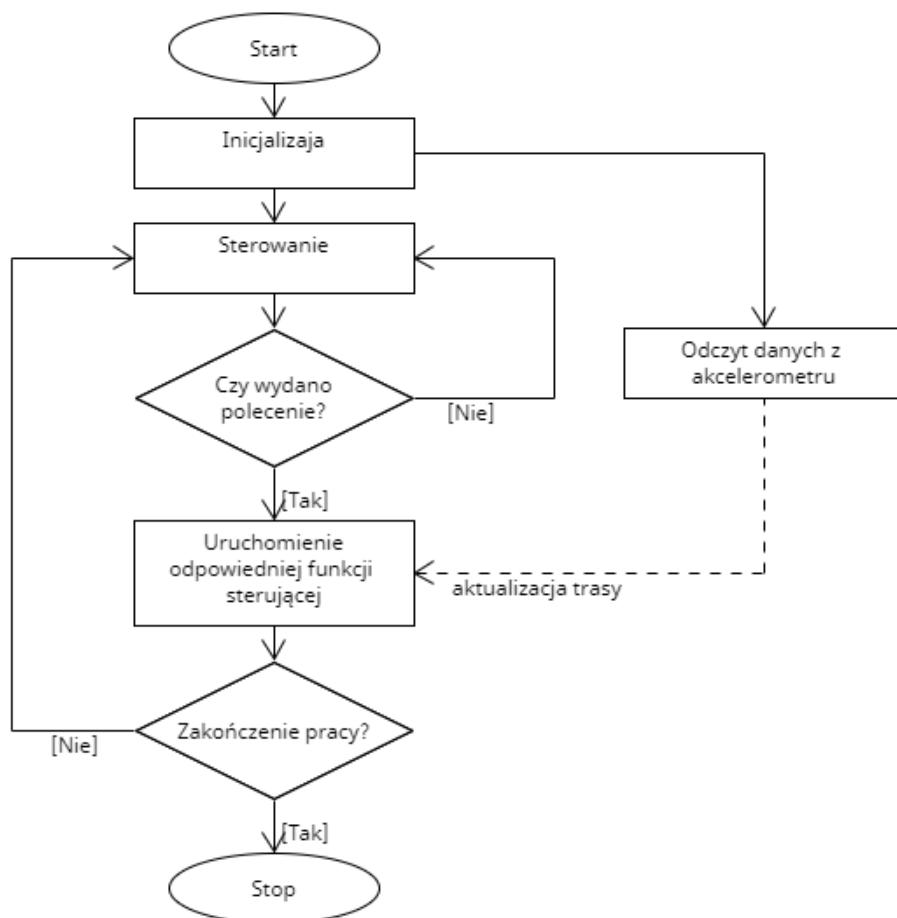
Rys. 3.21. Wykres obszaru roboczego wraz z wygenerowaną trasą punktów.

### 3.6. Implementacja systemu sterowania

W tym rozdziale przedstawiono szczegóły implementacji oprogramowania sterującego robotem kroczącym. Jako jednostkę sterującą wybrano wcześniej wspomniany mini komputer Banana Pi M5, na którym zainstalowano jedną z dostępnych dystrybucji systemu operacyjnego Linux, tj. Ubuntu. Jako język programowania wybrano Python ze względu na jego wszechstronność, obszerny zestaw dostępnych bibliotek oraz prostotę integracji z urządzeniami peryferyjnymi.

Do zdalnej komunikacji z mini komputerem wykorzystano protokół SSH, co umożliwiło programowanie oraz dostęp do plików. W tym celu jako środowisko programistyczne wykorzystano Visual Studio Code z zainstalowanym rozszerzeniem "Remote - SSH".

Na rysunku 3.22 został przedstawiony schemat blokowy wyjaśniający działanie programu sterującego. Poszczególne etapy zostaną szczegółowo omówione w kolejnych podrozdziałach.



Rys. 3.22. Diagram reprezentujący działanie programu.

- Blok inicjalizacji

W trakcie procesu inicjalizacji, centralna jednostka sterująca przeprowadza sekwencję operacji, których celem jest konfiguracja obecnych w projekcie modułów. W ramach tego procesu, dokonywane jest również inicjalizowanie i ustawianie kluczowych parametrów, mających istotne znaczenie dla późniejszych etapów działania programu.

### Konfiguracja modułu nRF24L01

W ramach bloku inicjalizacji przeprowadzona została szczegółowa konfiguracja modułu nRF24L01. Ustalonono odpowiednie parametry transmisji, takie jak częstotliwość, moc sygnału czy tryb pracy, zapewniając optymalne warunki wymiany informacji. Podczas programowania tego modułu, wykorzystano dostępne w sieci kody bibliotek. Dla obsługi urządzenia, stworzono pilota z płytą Blackpill, która została zaprogramowana przy użyciu środowiska STM32Cube IDE wykorzystując moduł nRF24 jako nadajnik.

### Konfiguracja modułu MPU6050

Kolejnym istotnym krokiem w bloku inicjalizacji jest dostosowanie ustawień modułu MPU6050, odpowiedzialnego za pomiar ruchu i orientacji robota. W tym celu wykorzystano gotową bibliotekę napisaną w języku Python, która podczas tworzenia obiektu inicjalizuje niezbędne rejestracje urządzenia oraz zawiera gotowe do wykorzystania funkcje komunikacyjne z modułem. Domyślny program odbierania danych w zupełności sprostał potrzebom projektu.

### Konfiguracja sterownika PCA9685

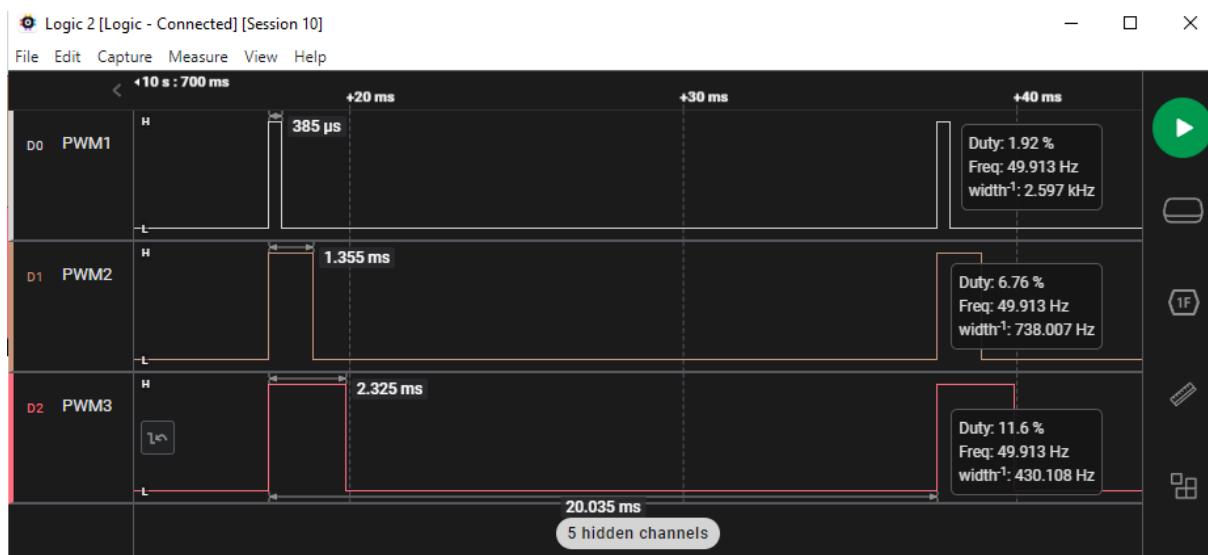
Podczas pierwszego uruchomienia przykładowego programu sterownika z podłączonymi serwomechanizmami do kolejnych portów, nie uzyskano prawidłowych rezultatów, w postaci fizycznie poprawnego zakresu pracy. W celu uzyskania rzeczywistego przebiegu sygnałów, wykorzystano analizator stanów logicznych firmy Saleae, który został podłączony do kilku portów sterownika.

Po uruchomieniu programu na domyślnej konfiguracji stwierdzono, że wyjścia sterownika nie generują wartości zgodnych z oczekiwany. Sygnał sterujący PWM serwomechanizmów, zgodnie z dokumentacją producenta, powinien charakteryzować się częstotliwością  $50\text{Hz}$  oraz wypełnieniem w zakresie  $0 - 2\text{ms}$ . Sterownik PWM w domyślnej konfiguracji nie generował odpowiednich wartości na swoich wyjściach. W wyniku tego dostosowano parametry odpowiedzialne za szerokość generowanego impulsu jak i jego częstotliwość w sposób empiryczny, uzyskując częstotliwość zbliżoną do  $50\text{Hz}$ .

Podczas testowania serwomechanizmu stwierdzono, że minimalna wartość wypełnienia do uzyskania reakcji wynosiła około  $300\mu\text{s}$ . Przy zwiększeniu wypełnienia do  $2.7\text{ms}$ , serwomechanizmy nadal

wykazywały reakcję, osiągając zakres pracy większy niż deklarowany. Korzystając z tych informacji, dostosowano wartości zgadnie z zamierzonym zakresem pracy. Przebiegi czasowe sygnałów PWM sterujących pracą silnika dla finalnej wersji konfiguracji przedstawiono na rysunku 3.23. Oznaczenia portów odpowiadają fizycznym ustawieniom serwomechanizmów:

- PWM1 reprezentuje przebieg odpowiadający pozycji 0°,
- PWM2 reprezentuje przebieg odpowiadający pozycji 90°,
- PWM3 reprezentuje przebieg odpowiadający pozycji 180°.



Rys. 3.23. Fizyczny podgląd sygnałów sterownika PCA9685.

- Blok sterowania

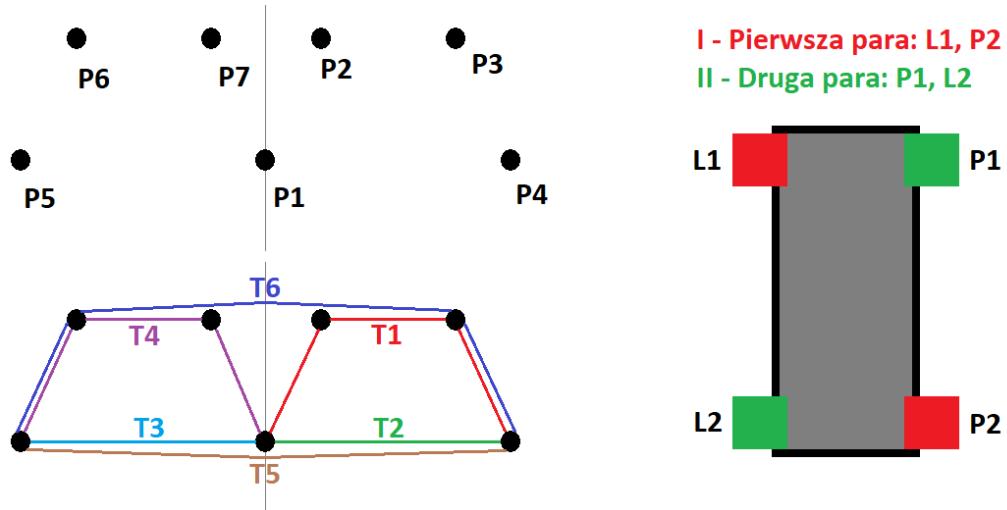
Odbiór danych sterowania odbywa się jednocześnie poprzez komunikację radiową oraz protokół SSH. Główny program odbiornika radiowego wykonuje się równolegle z działającym programem robota, w tym odbieraniu danych sterowania poprzez SSH. Ewentualny jednoczesny odbiór informacji z tych dwóch źródeł rozwiązano przez nadpisanie danych przez funkcję odbiornika radiowego. Dodatkowym zabezpieczeniem programu przed sprzecznymi komunikatami jest flaga gotowości do ruchu robota, która blokuje odebranie kolejnego komunikatu przed zakończeniem realizacji ruchu z poprzedniego.

W celu równoległego wykonywania operacji funkcji modułu nRF24, zastosowano moduł języka Python o nazwie "threading". Zastosowanie wspomnianego modułu oprogramowania umożliwia równoległe wykonywanie wielu operacji w programie poprzez uruchamianie funkcji w oddzielnym wątkach.

- Blok funkcji ruchu

Realizacja ruchu polega na przesuwaniu danej strony robota po podłożu w określonym kierunku. Po odebraniu komendy sterującej, w zależności od niej, wykonywana jest sekwencja ruchu mająca na celu przez określoną liczbę iteracji prowadzić kolejne pary nóg przez wartości kolejnych tras w ustalonej ich kolejności.

W procesie wykonywania pojedynczej sekwencji ruchu nogi zastosowano metodologię opartą na eksperymentalnie zdefiniowanych punktach, które umożliwiają umieszczenie środka ciężkości w centralnej części konstrukcji. Pomiędzy tymi punktami wygenerowano trajektorię, składającą się z równomierne rozmieszczonej tras. Procedura ta, analogicznie do analizy wyników opisanej w rozdziale 3.5, podlegała obliczeniom mającym na celu ustalenie nastaw serwomechanizmów w kolejnych punktach zdefiniowanej trasy ruchu. Na rysunku 3.24 przedstawiono omawiane punkty, zdefiniowane trasy oraz kolorystycznie oznaczone pary nóg.



Rys. 3.24. Oznaczenia punktów, tras oraz kolejnych nóg w realizacji funkcji ruchu.

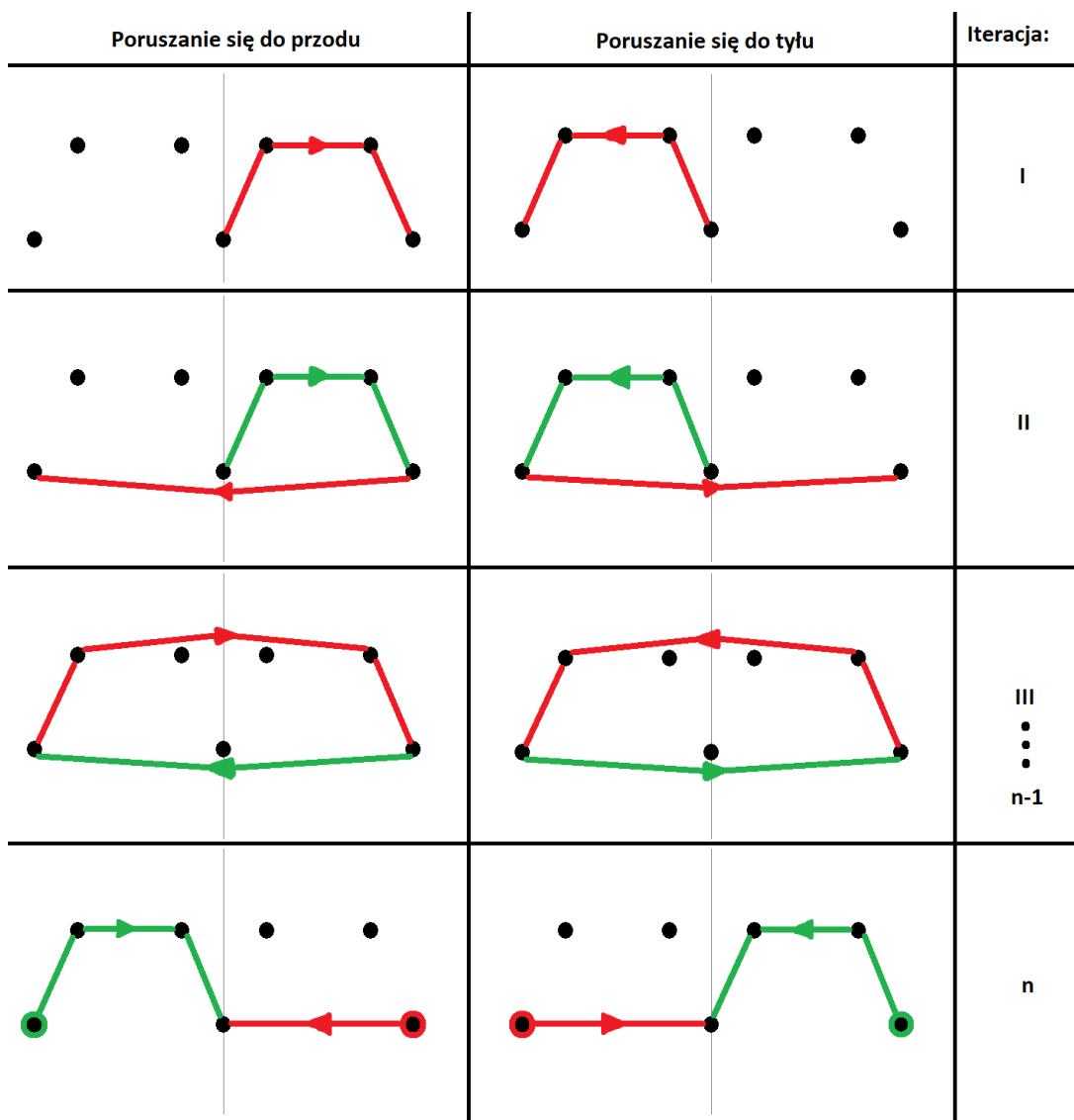
Kolejne trasy złożone są z punktów:

- Trasa T1 złożona z punktów P1, P2, P3 oraz P4.
- Trasa T2 złożona z punktów P1 oraz P4
- Trasa T3 złożona z punktów P1 oraz P5
- Trasa T4 złożona z punktów P5, P6, P7 oraz P1.
- Trasa T5 złożona z punktów P4 oraz P5
- Trasa T6 złożona z punktów P5, P6, P3 oraz P4.

Na rysunku 3.25 został przedstawiony pełny cykl realizacji ruchu przez pary nóg ustalone po przekątnej, zgodnie z oznaczeniami kolorów widocznymi na rysunku 3.24. Program uwzględnia zarówno parzystą, jak i nieparzystą liczbę iteracji.

Kolejne pary nóg działają naprzemiennie w taki sposób, że w trakcie sekwencji kroku jedna para przemieszcza się po górnej trajektorii w powietrzu, podczas gdy druga para równocześnie przesuwa się po powierzchni, powodując przemieszczenie konstrukcji względem podłoża. Przedstawioną sekwencję ruchu cechuje operowanie wokół punktu startowego  $P_1$ . Ruch rozpoczyna się w tym punkcie oraz w końcowym etapie powraca do niego.

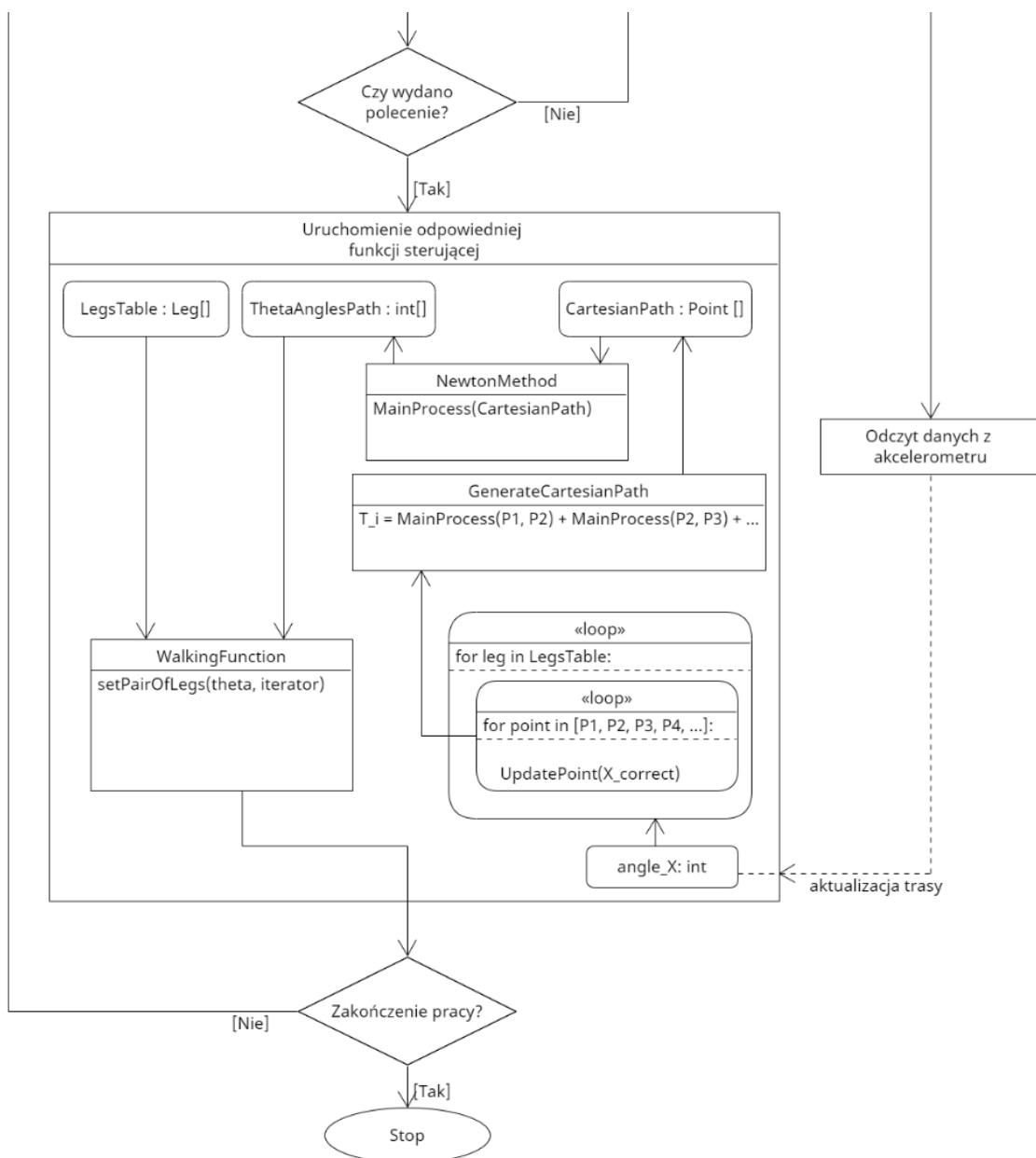
Implementacja funkcji skręcania została również oparta na przedstawionych sekwencjach ruchu do przodu i do tyłu. Podczas skręcania, zależnie od kierunku skrętu, każda para nóg po przekątnej realizuje dwie trasy ruchu, zarówno do przodu, jak i do tyłu, które są dostosowane do położenia konkretnej nogi względem robota. Ten sposób działania zapewnia, że jedna strona robota porusza się w jednym kierunku, a druga w przeciwnym, co prowadzi do skrętu całej konstrukcji.



Rys. 3.25. Ilustracja sekwencji ruchu robota do przodu oraz do tyłu.

Na etapie inicjalizacji działania robota zdefiniowane są jedynie wspomniane punkty. Zarówno trajektoria punktów pomiędzy nimi, jak i wartości kinematyki odwrotnej są pierwotnie obliczane i przechowywane w zmiennych o strukturze tabelarycznej. Przy założeniu, że akcelerometr nie wpływa na wartości punktów oraz po odebraniu polecenia sterującego, kolejne wartości kąta theta są przekazywane do instancji reprezentującej część roboczą.

Opisywany proces został zilustrowany na rysunku 3.26, który przedstawia fragment poprzedniego diagramu z rozszerzeniem omawianej funkcjonalności, prezentując uproszczoną sekwencję wykonywania tej części oprogramowania.



Rys. 3.26. Diagram realizacji funkcji ruchu.

Oznaczenia przyjęte na diagramie:

- **LegsTable : Leg[]** – tablica zawierająca instancje wszystkich nóg,
- **ThetaAnglesPath : int[]** – zapis w postaci tabeli zawierający obliczone wartości nastaw serwomechanizmów dla kolejnych tras,
- **WalkingFunction** – funkcja odpowiedzialna za realizację konkretnej sekwencji ruchu, która w swoim działaniu uruchamia odpowiednią parę nóg dla określonej trasy,
- **GenerateCartesianPath** – funkcja odpowiedzialna za generowanie tras pomiędzy punktami,
- **CartesianPath : Point[]** – zapis w postaci tabeli zawierający wygenerowane punkty równomiernie rozmieszczone pomiędzy zdefiniowanymi punktami,
- **NewtonMethod** – implementacja metody wyliczającej wartości kinematyki odwrotnej dla nastaw serwomechanizmów,
- **angleX : int** – wartość kąta przekazywana przez osobny wątek programu akcelerometru,
- **Xcorrect** – obliczona wartość korekcji współrzędnej X dla punktów.

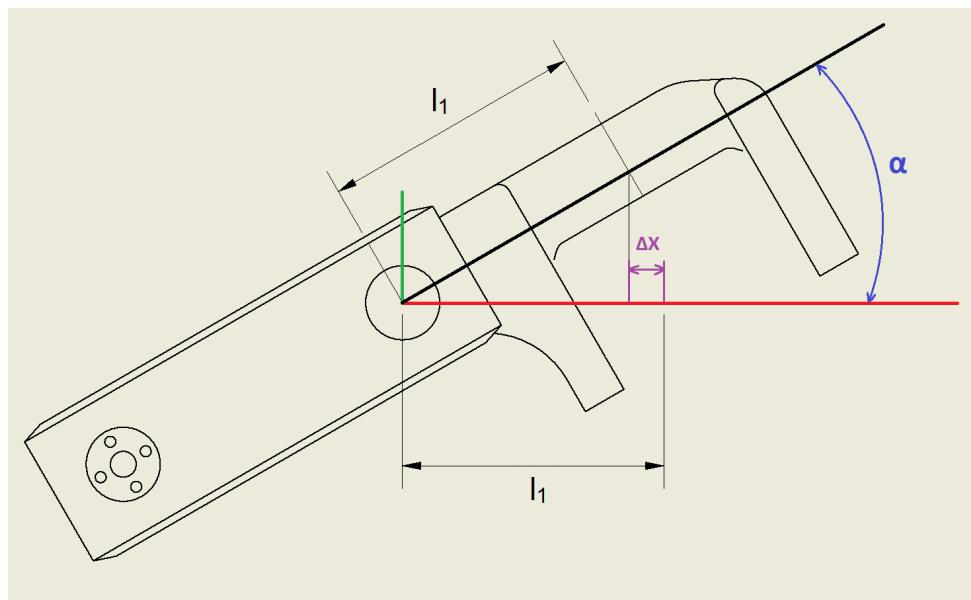
Obliczenia związane z wyznaczaniem punktów pośrednich w układzie kartezjańskim oraz kinematyką odwrotną są przeprowadzane na poziomie pojedynczej instancji nogi. Każda nogą jest wyposażona w zdefiniowane punkty trasy, co umożliwia indywidualne modyfikacje. Instancja nogi posiada parametr określający jej położenie względem robota, co daje możliwość uwzględnienia kierunku, w którym należy dokonywać korekt w przypadku ingerencji akcelerometru.

- Blok odczytu danych z akcelerometru

Dane z akcelerometru są odczytywane równolegle do działającego programu przy użyciu modułu "threading" w języku Python. Ingerencja w początkowo zdefiniowaną trasę zachodzi po przekroczeniu progowej wartości kąta nachylenia. W celu zminimalizowania skomplikowania implementacji, zdecydowano się na korektę tylko jednej osi ruchu, która odpowiada za pochylenie boku robota. Ta oś jest zgodna z osią  $x$  bazowego układu współrzędnych pojedynczej nogi. Celem jest utrzymanie stabilnej pozycji kadłuba robota nawet w przypadku nachylonej powierzchni, na której robot się znajduje. W tym celu, w zależności od kierunku nachylenia, dla każdego zdefiniowanego punktu dokonywana jest korekta współrzędnej odpowiadającej za osiągalną wysokość, którą jest współrzędna  $y$ .

Rysunek 3.27 przedstawia graficzną interpretację równania (3.18), które pozwala obliczyć omawianą korekcję współrzędnej wysokości.

$$\Delta x = l_1 - l_1 \cdot \cos(\alpha) \quad (3.18)$$



**Rys. 3.27.** Schemat ideowy dla równania 3.18 służącego do wyznaczenia korekcji przemieszczenia  $\Delta x$ .

Każda noga robota w zależności od strony, po której się znajduje, dodaje lub odejmowało wartość przemieszczenia  $\Delta x$  od współrzędnej  $y$  dla każdego zdefiniowanego punktu trasy, uwzględniając przy tym znak wartości kąta nachylenia.



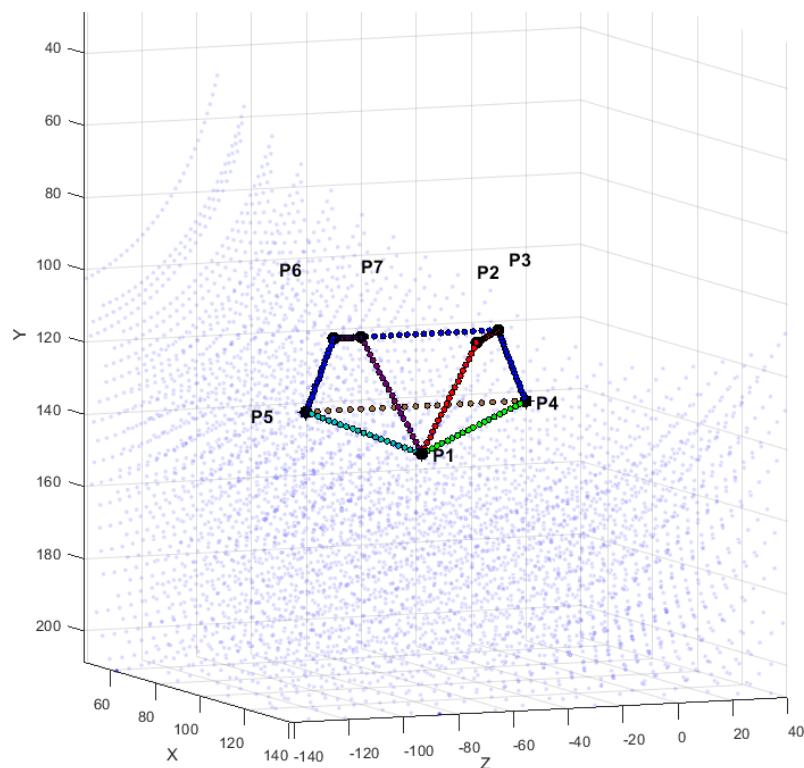
## 4. Prezentacja działania programu

Celem niniejszego rozdziału jest prezentacja wyników obliczeń związanych z kinematyką odwrotną oraz ogólnego zarysu uzyskanych rezultatów w postaci funkcjonującej konstrukcji.

Przedstawione wyniki testu ruchu przeprowadzano w warunkach eksperymentalnych, polegających na uniesieniu robota w powietrzu. Punkty przedstawione na rysunku 3.24 w układzie kartezjańskim mają następujące współrzędne:

$$\begin{aligned} P_1 &= (75, 150, -40), \quad P_2 = (75, 120, -20), \quad P_3 = (60, 120, 0) \\ P_4 &= (60, 140, 10), \quad P_5 = (60, 140, -70), \quad P_6 = (60, 120, -60) \\ P_7 &= (60, 120, -50) \end{aligned} \quad (4.1)$$

Na rysunku 4.1 zostały przedstawione kolejne trasy uzyskane w wyniku obliczeń kinematyki odwrotnej.

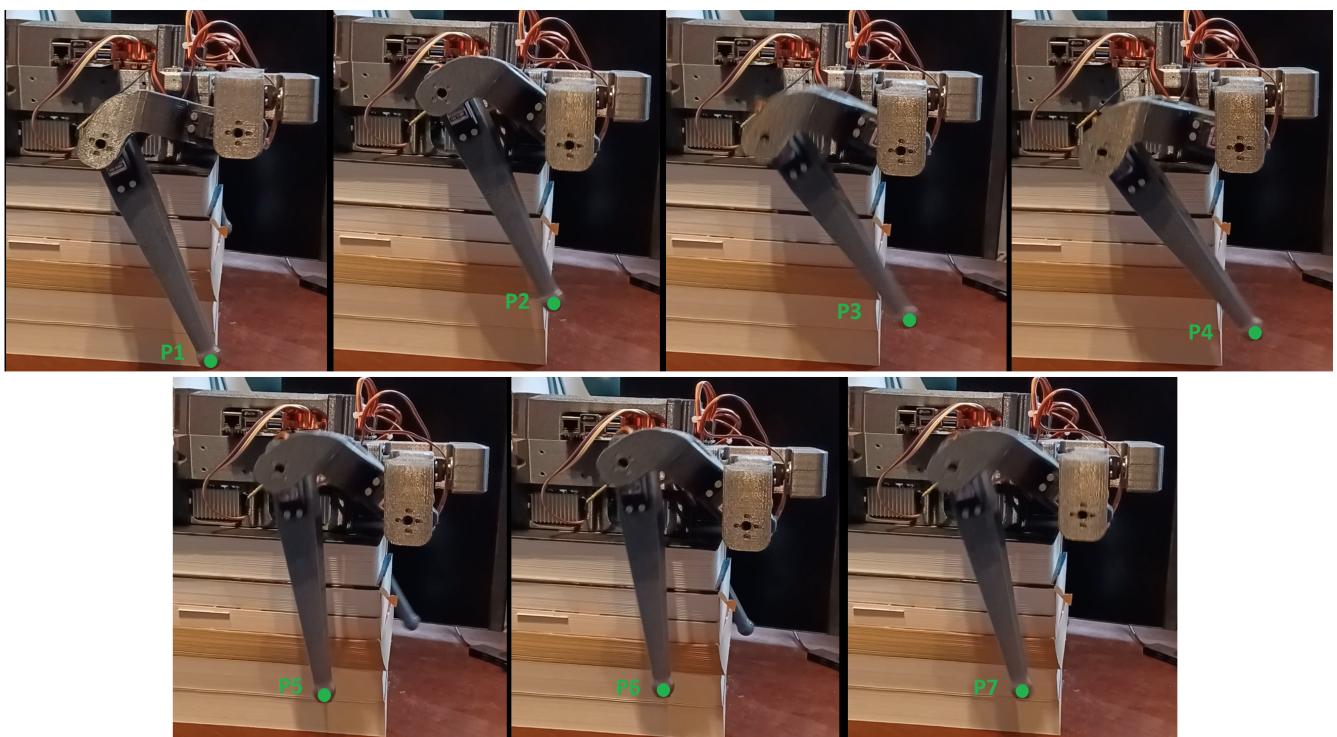


Rys. 4.1. Trasy ruchu dla sekwencji sterowania.

Ewentualne odstępstwa od idealnej linii prostej, które mogłyby być wynikiem błędów numerycznych wynikających z dużej ilości wykonanych operacji obliczeniowych, nie wywierają istotnego wpływu na działanie robota w praktyce, zwłaszcza że nie jest on maszyną precyzyjną. Nawet błędy o wielkości kilku milimetrów nie wywołyły zakłóceń w ogólnym przebiegu ruchu robota.

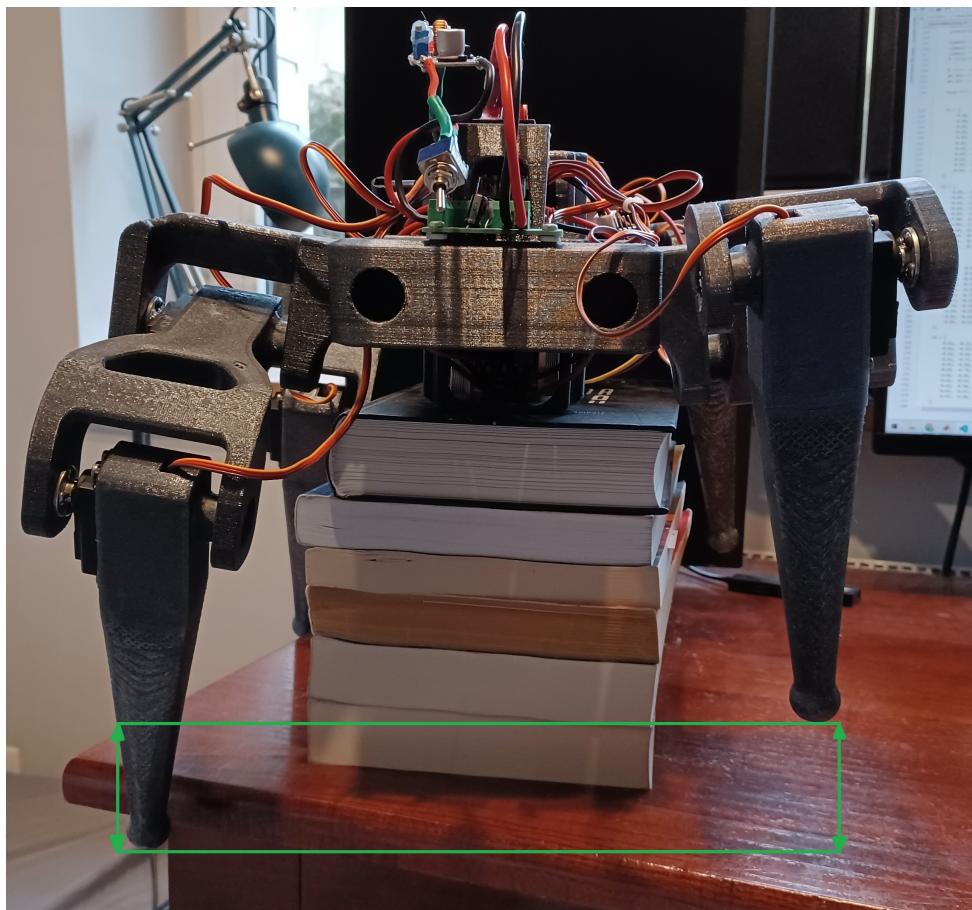
W warunkach ruchu na powierzchni robot osiąga identyczne działanie. Jednakże ze względu na nie-dokładność serwomechanizmów, starających się utrzymać zadaną pozycję pomimo znacznego obciążenia, mogą występować pewne różnice.

Poniżej zaprezentowano zdjęcia przedstawiające fizyczną realizację sekwencji ruchu nogi robota, osiągających wcześniej wspomniane punkty:



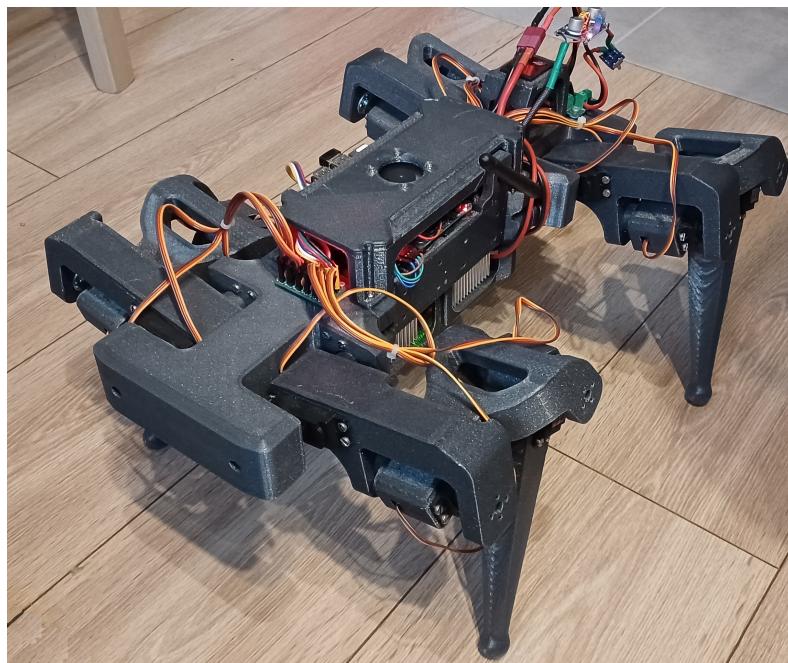
Rys. 4.2. Fizyczny przebieg trajektorii.

Efekt działania akcelerometru został zilustrowany na rysunku 4.3. Po przechyleniu robota na jedną ze stron, program na podstawie kąta przechylenia oblicza wysokość, o którą należy podnieść lub obniżyć daną stronę.

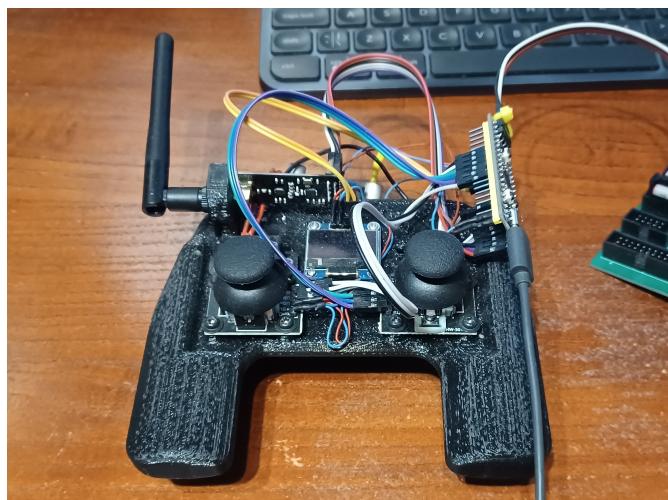


Rys. 4.3. Prezentacja działania akcelerometru.

Na rysunku 4.4 została przedstawiona zrealizowana konstrukcja robota, natomiast rysunek 4.5 prezentuje wykonany pilot sterujący dla operatora.



Rys. 4.4. Konstrukcja robota.



Rys. 4.5. Pilot sterujący dla operatora.

## **5. Podsumowanie oraz wnioski**

Celem pracy było zaprojektowanie robota kroczącego wraz z implementacją jego systemu sterowania. W rezultacie prac osiągnięto wszystkie wymienione założenia przedstawione w rozdziale 3.1.

Zaprojektowana konstrukcja robota umożliwia pełen zakres ruchu serwomechanizmów, co pozwala na realizację różnorodnych funkcji ruchu. Dzięki wyborowi odpowiednio silnych serwomechanizmów robot działa stosunkowo sprawnie.

Dodatkowo, wykorzystanie współrzędnych kartezjańskich celem określenia pozycji końcówki roboczej zapewnia lepszą i bardziej intuicyjną orientację w przestrzeni. Ponadto, umożliwia również większą płynność pracy przy ruchu pomiędzy zdefiniowanymi punktami.

Najtrudniejszym zadaniem okazał się montaż całej konstrukcji oraz kalibracja serwomechanizmów w taki sposób, aby każda nogą posiadała identyczne ustawienia. W tym procesie praca została utrudniona przez fakt stosowania przez producenta serwomechanizmów wału serwonapędu z mocowaniem części roboczej posiadającym nieparzystą liczbę zębów. W rezultacie, w zależności od miejsca, w które trafiło mocowanie, dochodziło do znacznej różnicy w położeniu. Wymagało to wielokrotnego demontażu i ponownego montażu elementów oraz dokładnego ustawienia ich w celu uzyskania satysfakcyjującego efektu, co znaczco zwiększyło nakład czasu.

W kontekście oprogramowania, napotkano kilka drobnych niedogodności, w tym konieczność dokonania szczegółowej konfiguracji sterownika PCA9685, co wymagało dobrania konkretnych wartości parametrów kontrolera, o czym wspomniano w rozdziale 3.6. Modułem, który sprawił większą trudność, był moduł radiowy nRF24. Problematyczna okazała się domyślna biblioteka modułu, napisana pierwotnie dla platformy Raspberry Pi, korzystająca z dedykowanej biblioteki do obsługi pinów ogólnego przeznaczenia, co wymagało gruntownej modyfikacji dla używanego w projekcie minikomputera Banana Pi.

Poza wspomnianymi niedogodnościami, praca przebiegała bez większych problemów.

### **Możliwości dalszego rozwoju:**

Istnieje kilka potencjalnych kierunków, w ramach których można kontynuować rozwój projektu, umożliwiając rozszerzenie funkcjonalności robota. Poniżej przedstawione zostały przykładowe możliwości dalszego rozwoju:

- **Kamera głębi typu Intel RealSense:** Zintegrowanie kamery głębi, przykładowo kamery Intel RealSense, pozwoliłoby robotowi na bardziej precyzyjne zrozumienie otoczenia poprzez pozyskiwanie informacji o przeskalowanej powierzchni, co jest istotne dla precyzyjnej nawigacji robota.
- **Kamera HD:** Jako tańszą alternatywą dla kamery głębi można rozważyć wyposażenie robota w zwykłą kamerę HD, a następnie wykorzystanie technik deep learning do analizy obrazu. Taka implementacja umożliwiłaby robotowi rozpoznawanie i reagowanie na różne obiekty i sytuacje w otoczeniu. To rozszerzenie pozwoliłoby również na implementację bardziej zaawansowanych strategii nawigacyjnych.
- **ROS 2 (Robot Operating System 2):** Wdrożenie systemu ROS 2 stanowiłoby istotny krok w celu zwiększenia efektywności komunikacji między różnymi modułami systemu. ROS 2 wprowadza nowoczesne rozwiązania, co przyczyniałoby się do bardziej stabilnej i skalowej architektury oprogramowania robota.
- **Zastosowanie dodatkowej sensoryki:** Dodanie sensorów siły nacisku na końcach nóg umożliwiłaby precyzyjną ocenę stopnia, w jakim poszczególne kończyny oddziałują z podłożem. Choć taka modyfikacja wymagałaby dostosowania konstrukcji, zapewniłaby cenne informacje zwrotne, umożliwiające wprowadzenie bardziej zaawansowanego algorytmu generującego trasę ruchu robota.
- **Rozwinięcie interfejsu użytkownika:** Stworzenie zaawansowanego interfejsu użytkownika do zdalnego sterowania robotem, umożliwienie zdalnego monitorowania oraz konfiguracji parametrów systemu, a także rozważenie możliwości integracji z chmurą, stanowiłyby potencjalne kroki w kierunku usprawnienia użytkowania robota. Działania te obejmowałyby zdalne monitorowanie danych z otoczenia, analizę danych, implementację uczenia maszynowego, oraz całkowicie zdальną konfigurację parametrów.

Wnioski z przeprowadzonych badań i implementacji stanowią solidną podstawę do dalszego rozwoju projektu. Wprowadzenie wyżej wymienionych możliwości mogłoby uczynić robota bardziej wszechstronnym i skutecznym w różnych scenariuszach zastosowania.

## Bibliografia

- [1] Wikipedia. Legged robot. Dostęp: 8 stycznia 2024. URL: [https://en.wikipedia.org/wiki/Legged\\_robot](https://en.wikipedia.org/wiki/Legged_robot).
- [2] Wikipedia. BostonDynamics. Dostęp: 8 stycznia 2024. URL: [https://en.wikipedia.org/wiki/Boston\\_Dynamics](https://en.wikipedia.org/wiki/Boston_Dynamics).
- [3] Mini cheetah is the first four-legged robot to do a backflip. Dostęp: 8 stycznia 2024. URL: <https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304>.
- [4] Benjamin Katz, Jared Di Carlo i Sangbae Kim. „Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control”. W: International Conference on Robotics and Automation (ICRA) (2019).
- [5] Felix Ruppert Alexander Badri-Spröwitz. Learning plastic matching of robot dynamics in closed-loop central pattern generators. Dostęp: 8 stycznia 2024. URL: <https://www.nature.com/articles/s42256-022-00505-4>.
- [6] Tower Pro. MG996R Servo Motor. Dostęp: 8 stycznia 2024. URL: <https://www.towerpro.com.tw/product/mg996r/>.
- [7] Kamami.pl. Moduł przetwornicy Step Down LM2596S. Dostęp: 8 stycznia 2024. URL: <https://kamami.pl/step-down/543018-modul-przetwornicy-dc-dc-step-down-340v-regulowane-1535v-lm2596.html>.
- [8] Kamami.pl. Moduł przetwornicy Step Down 300W. Dostęp: 8 stycznia 2024. URL: <https://kamami.pl/step-down/1183412-toolkitrc-m4-ladowarka-z-balanserem-do-akumulatorow-lipo.html>.
- [9] Banana Pi. Banana Pi BPI-M5. Dostęp: 8 stycznia 2024. URL: [https://wiki.banana-pi.org/Banana\\_Pi\\_BPI-M5](https://wiki.banana-pi.org/Banana_Pi_BPI-M5).
- [10] NXP Semiconductors. PCA9685 Datasheet. Dostęp: 8 stycznia 2024. URL: [https://download.kamami.pl/p583700-p199856-PCA9685\\_datasheet.pdf](https://download.kamami.pl/p583700-p199856-PCA9685_datasheet.pdf).
- [11] Kamami.pl. Moduł 16-kanałowego sterownika serw. Dostęp: 8 stycznia 2024. URL: <https://kamami.pl/sterowniki-serw/583700-modul-16-kanalowego-sterownika-serw-z-interfejsem-i2c-pca9685.html>.

- [12] abc rc. Moduł akcelerometru MPU6050. Dostęp: 8 stycznia 2024. URL: <https://abc-rc.pl/pl/products/akcelerometr-3-osiodowy-mpu-6050-gy-521-zyroskop-na-i2c-6572.html>.
- [13] Nordic Semiconductor. nRF24L01 Datasheet. Dostęp: 8 stycznia 2024. URL: [https://dl.btc.pl/kamami\\_wa/nrf24l01\\_ds.pdf](https://dl.btc.pl/kamami_wa/nrf24l01_ds.pdf).
- [14] abc rc. Moduł bezprzewodowy nRF24L01. Dostęp: 8 stycznia 2024. URL: <https://abc-rc.pl/pl/products/modul-bezprzewodowy-nrf24l01-wzmacniacz-i-antena-zasieg-do-1-1km-7669.html>.
- [15] Wikipedia. Notacja Denavita-Hartenberga. data dostępu. URL: [https://pl.wikipedia.org/wiki/Notacja\\_Denavita-Hartenberga](https://pl.wikipedia.org/wiki/Notacja_Denavita-Hartenberga).
- [16] Wikipedia. Newton's method. Dostęp: 8 stycznia 2024. URL: [https://en.wikipedia.org/wiki/Newton%27s\\_method](https://en.wikipedia.org/wiki/Newton%27s_method).