Politechnika Wrocławska
Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka (AIR)

PRACA DYPLOMOWA INŻYNIERSKA

TYTUŁ PRACY:
Sterowanie chodem
sześcionożnego robota kroczącego

AUTOR:
Kamil Bogusław Winnicki

PROMOTOR:
Dr inż. Robert Muszyński,
Katedra Cybernetyki i Robotyki

WROCŁAW 2024

Niniejszą pracę dedykuję mojej rodzinie, której wsparcie było nieocenione na każdym etapie mojej edukacji. Szczególne podziękowania kieruję do Pana dr. inż. Roberta Muszyńskiego za poświęcony czas i zaangażowanie.

Spis treści

Do składu pracy wykorzystano system przygotowania dokumentów LaTeX, opracowany przez L. Lamporta [Lam94], będący nakładką systemu TeX, [Knu86a,Knu86b]. Matematyczne czcionki o nazwie AMS Euler, których używamy w tej pracy, zostały przygotowane przez H. Zapfa [KZ86], przy współpracy z D. Knuthem i jego studentami, na zlecenie Amerykańskiego Towarzystwa Matematycznego. Wybrane czcionki składu tekstu, Antykwa Toruńska [Now97] – jeden z nielicznych krojów pisma zaprojektowany specjalnie dla języka polskiego w sposób uwzględniający jego rytm – w odczuciu autora doskonale współgrają z kształtem czcionki AMS Euler, pozwalając na uzyskanie harmonijnej całości. Składu bezszeryfowego tekstu maszynowego dokonano z użyciem opracowanej przez R. Leviena czcionki o nazwie Inconsolata [Lev15].

[Knu86a] D. E. Knuth, The TeXbook, volume A of Computers and Typesetting. Addison-Wesley, Reading, 1986.

[Knu86b] D. E. Knuth, TeX: The Program, volume B of Computers and Typesetting. Addison-Wesley, Reading, 1986.

[KZ86] D. E. Knuth i H. Zapf, AMS Euler — A new typeface for mathematics. Scholary Publishing, 20:131–157, 1986.

[Lam94] L. Lamport, LaTeX: A Document Preparation System. Addison–Wesley, Reading, 1994.

[Lev15] R. Levien, Inconsolata. https://levien.com/type/myfonts/inconsolata.html, 2015.

[Now97] J. Nowacki, Antykwa Toruńska — od początku do końca polska czcionka. *Biuletyn Polskiej Grupy Użytkowników Systemu TeX*, 9:26–27, 1997.

Rozdział 1

Wstęp

Od początków cywilizacji ludzkość czerpała inspirację z przyrody. Już najstarsze przykłady sztuki paleolitu, takie jak malarstwo jaskiniowe, przedstawiają głównie zwierzęta: konie, bizony, żubry a czasem nosorożce czy duże kotowate [Wikj]. Wraz z rozwojem cywilizacji, zachwyt nad mechanizmami świata przyrody ewoluował, znajdując swoje odzwierciedlenie w technologii. Bionika, nazywana też biomimetyką, jest dziedziną zajmującą się projektowaniem technologii inspirowanych organizmami żywymi, która stała się w ostatnich latach jednym z kluczowych obszarów rozwoju robotyki [Wikc]. Dzięki niej możemy tworzyć maszyny, które nie tylko naśladują wygląd zwierząt, ale także adaptują ich sposób poruszania się, sensorykę czy funkcje przystosowawcze. Przykładem zastosowania bioniki są sonary ultradźwiękowe inspirowane sposobem lokalizacji w przestrzeni i zdobywania pożywienia przez nietoperze [Wikk].

Biomimetyka ma swoje korzenie w długiej historii prób odtwarzania zdolności organizmów żywych w maszynach. Już w XVIII wieku Jacques de Vaucanson [Wikh] stworzył mechaniczne urządzenie naśladujące kaczkę. Wynalazek ten, choć prymitywny z perspektywy współczesnej inżynierii, stanowił pionierski przykład dążenia człowieka do odwzorowania natury w technologii. Dziś robotyka korzysta z tej samej idei, tworząc maszyny coraz bardziej zaawansowane, zdolne do rozwiązywania złożonych problemów, które dawniej wydawały się nieosiągalne.

Jednym z obszarów robotyki, który w szczególny sposób czerpie z natury, są roboty kroczące. Ich konstrukcja naśladuje mechanikę poruszania się zwierząt lądowych, które korzystają z odnóży. W porównaniu z klasycznymi, kołowymi robotami mobilnymi, takie rozwiązanie ma wiele zalet. Roboty kroczące są znacznie bardziej wszechstronne – mogą poruszać się po nierównym terenie, pokonywać przeszkody a najbardziej zaawansowane konstrukcje potrafią nawet wchodzić po schodach. Ich zdolność do stabilizacji ruchu w trudnych warunkach, dzięki wielopunktowemu kontaktowi z podłożem, czyni je niezastąpionymi w aplikacjach takich jak eksploracja terenów trudnodostępnych, ratownictwo czy badania środowiskowe.

Roboty kroczące są również dobrym przykładem biomimetyki, gdzie mechanizmy natury – takie jak układ odnóży owadów czy ssaków – stają się podstawą projektowania urządzeń. W niniejszej pracy inspiracją dla autora były polskie chrząszcze, takie jak jelonek rogacz, kozioróg dębosz czy pokazany na rysunku 1.1 próchniaczek czarniawy [Wikm]. Ich sposób poruszania jak i budowa stała się punktem wyjścia do skonstruowania robota kroczącego, który naśladuje prawdziwe owady.

Rysunek 1.1 Próchniaczek czarniawy (łac. Ocypus olens) – gatunek chrząszcza z rodziny kusakowatych

Celem pracy jest zaprojektowanie i wykonanie sześcionożnego robota kroczącego oraz zaprojektowanie i implementacja jego systemu sterowania, bazującego na modelu, który umożliwia uzyskanie płynnego i stabilnego chodu. Układ pracy jest następujący. W rozdziale drugim zaprezentowano matematyczny model kinematyki robota i jego odnóży oraz opisano sposób planowania ruchu. Rozdział trzeci omawia algorytm sterowania oraz oprogramowanie. W rozdziale czwartym przedstawiono budowę robota, z uwzględnieniem konstrukcji mechanicznej i układów elektronicznych. Rozdział piąty zawiera opis przeprowadzonych testów. Na zakończenie, w rozdziale szóstym, podsumowano pracę oraz przedstawiono możliwości dalszego rozwoju projektu.

Rozdział 2

Modelowanie i planowanie chodu

2.1 Kinematyka robota kroczącego

W najprostszym przypadku zakłada się, że robot kroczący porusza się po płaszczyźnie utrzymując korpus równolegle do niej. Wówczas macierz jego kinematyki będzie miała postać

$$K = \begin{bmatrix} \cos\theta_k & -\sin\theta_k & 0 & x_k \\ \sin\theta_k & \cos\theta_k & 0 & y_k \\ 0 & 0 & 1 & z_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1}$$

gdzie współrzędne x_k, y_k, z_k oznaczają położenie środka korpusu robota w globalnym układzie współrzędnych, zaś kąt θ_k jego orientację względem osi prostopadłej do płaszczyzny ruchu (zobacz rysunek 2.1). Dla bardziej zaawansowanego planowania ruchu robota warto uwzględnić także orientację korpusu względem pozostałych dwóch osi. Przyjmując macierz rotacji w postaci

$$R = \mathrm{Rot}(Z, \theta_k)\mathrm{Rot}(Y, \psi_k)\mathrm{Rot}(X, \phi_k) \tag{2.2}$$

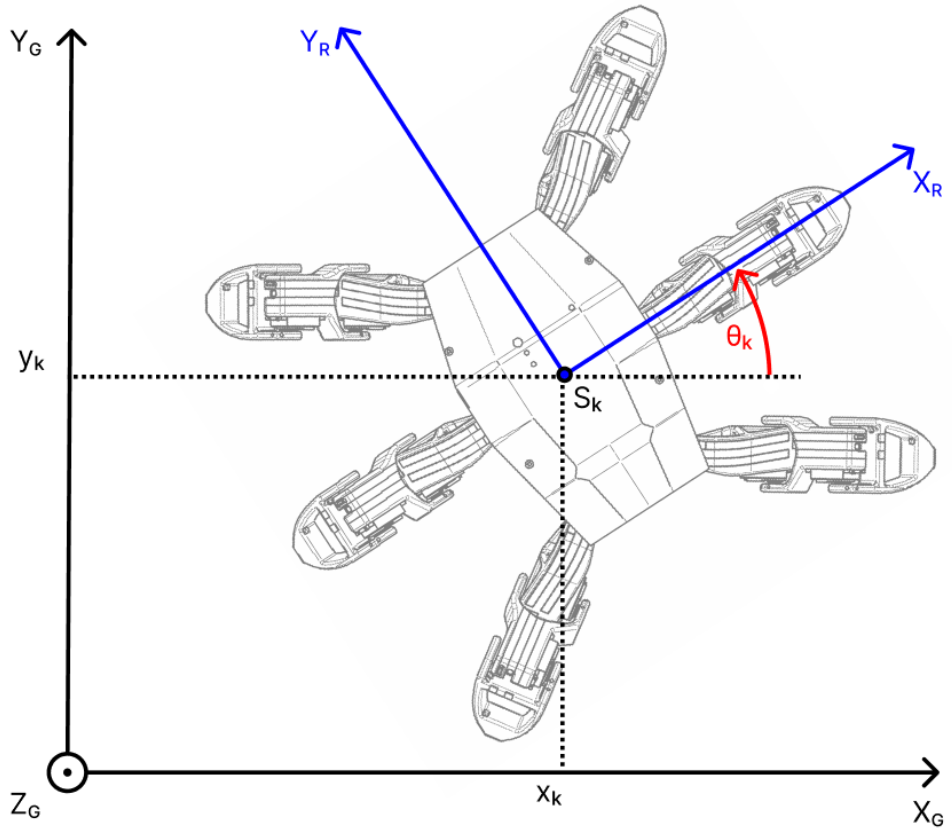
otrzymujemy macierz kinematyki

$$K = \begin{bmatrix} c_{\theta_k}c_{\psi_k} & c_{\theta_k}s_{\psi_k}s_{\phi_k} - s_{\theta_k}c_{\phi_k} & c_{\theta_k}s_{\psi_k}c_{\phi_k} + s_{\theta_k}s_{\phi_k} & x_k \\ s_{\theta_k}c_{\psi_k} & s_{\theta_k}s_{\psi_k}s_{\phi_k} + c_{\theta_k}c_{\phi_k} & s_{\theta_k}s_{\psi_k}c_{\phi_k} - c_{\theta_k}s_{\phi_k} & y_k \\ -s_{\psi_k} & c_{\psi_k}s_{\phi_k} & c_{\psi_k}c_{\phi_k} & z_k \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

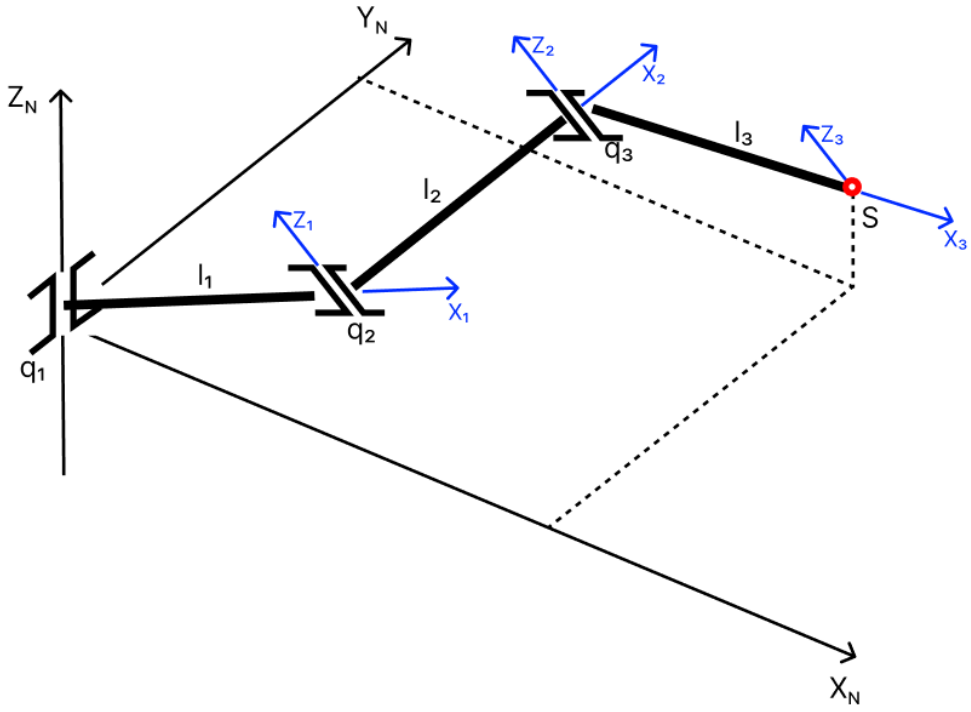
Takie podejście pozwoli na optymalizację ruchu po pochylonym podłożu np. ruchu pod góre, w dół lub wzdłuż skarpy.

2.1.1 Kinematyka prosta nogi robota

Noga robota pokazana schematycznie na rysunku 2.2 stanowi manipulator o trzech członach obrotowych. Kinematyka prosta opisuje pozycję i orientację końca nogi względem korpusu w funkcji kątów w jej przegubach. Wyliczaną kinematykę nogi wyrażamy w jej lokalnym układzie współrzędnych $X_N Y_N Z_N$ o osiach równoległych do odpowiadających im osi układu $X_R Y_R Z_R$ zwiazanego z korpusem robota.



Rysunek 2.1 Położenie robota w globalnym układzie współrzędnych



Rysunek 2.2 Kinematyka nogi robota

i	x_i	y_i
1	d_1	$-d_3$
2	d_2	0
3	d_1	d_3
4	$-d_1$	$-d_3$
5	$-d_2$	0
6	$-d_1$	d_3

Tabela 2.1 Tabela przesunięć dla mocowań nóg

Na podstawie algorytmu Denavita-Hartenberga [Wika] wyznaczamy transformacje układów związanych z kolejnymi członami nogi w postaci

$$\begin{aligned} A_0^1 &= \mathrm{Rot}(Z, q_1)\mathrm{Trans}(X, l_1)\mathrm{Rot}\left(X, -\frac{\pi}{2}\right), \\ A_1^2 &= \mathrm{Rot}(Z, q_2)\mathrm{Trans}(X, l_2), \\ A_2^3 &= \mathrm{Rot}(Z, q_3)\mathrm{Trans}(X, l_3). \end{aligned} \tag{2.4}$$

Macierz kinematyki nogi wyliczamy jako

$$K = A_0^3 = A_0^1 A_1^2 A_2^3 \tag{2.5}$$

otrzymując

$$K = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & c_1(c_{23}l_3 + c_2l_2 + l_1) \\ s_1c_{23} & -s_1s_{23} & c_1 & s_1(c_{23}l_3 + c_2l_2 + l_1) \\ -s_{23} & -c_{23} & 0 & -(s_{23}l_3 + s_2l_2) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

gdzie s_j, c_i oznaczają odpowiednio sinus i cosinus kąta q_i a $s_{23} = \sin(q_2 + q_3)$, $c_{23} = \cos(q_2 + q_3)$.

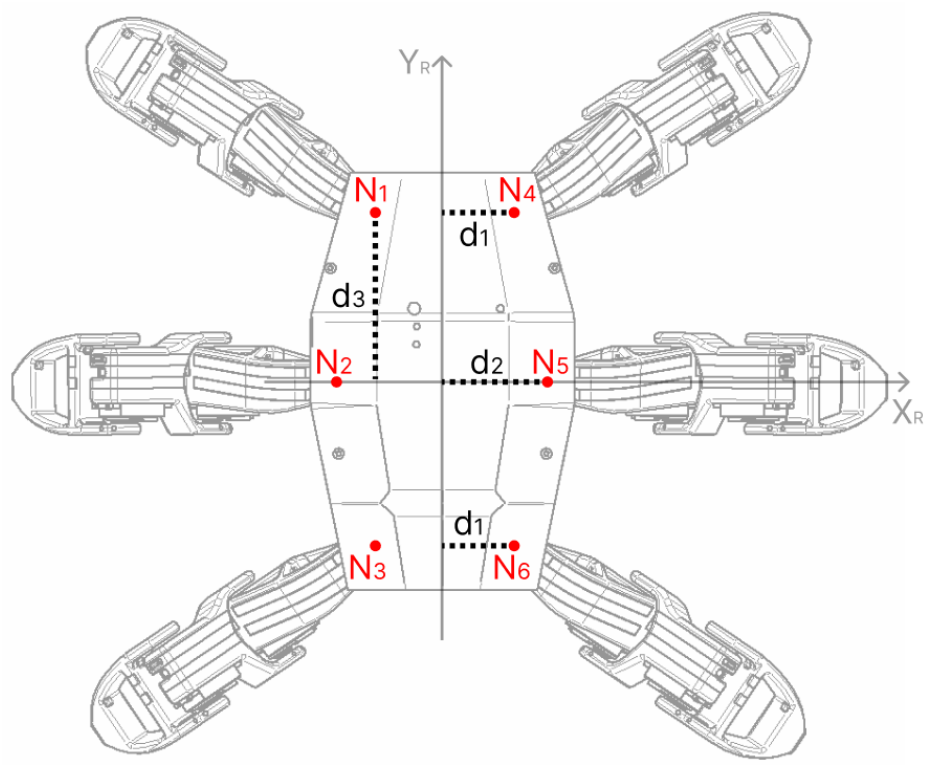
Macierz (2.6) opisuje kinematykę nogi w jej lokalnym układzie współrzędnych $X_N Y_N Z_N$. By wyrazić ją w związanym ze środkiem korpusu robota układem $X_R Y_R Z_R$ należy dokonać odpowiedniego do miejsca zamocowania nogi (widocznego na rysunku 2.3) przesunięcia opisanego transformacją

$$K_i = \begin{bmatrix} I & T_i \\ 0 & 1 \end{bmatrix} K \tag{2.7}$$

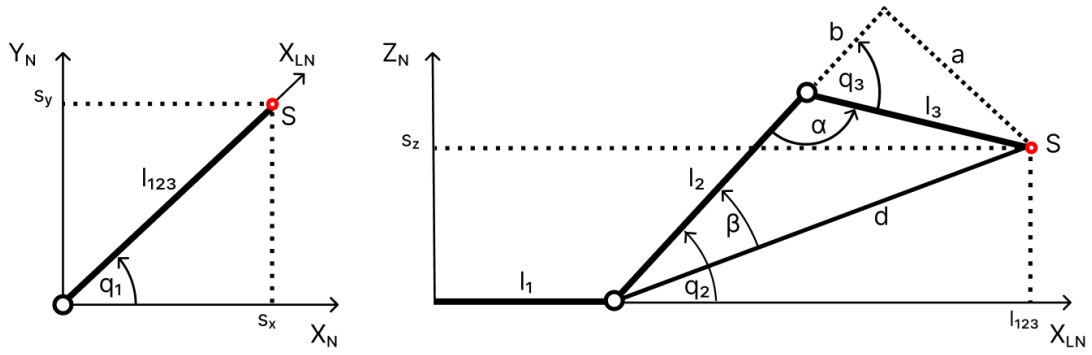
gdzie $T_i = (x_i, y_i, 0)$ to wektor charakteryzujący punkt mocowania i-tej nogi. Wartości przesunięć x_i, y_i dla poszczególnych nóg zebrano w tabeli 2.1.

2.1.2 Kinematyka odwrotna nogi robota

W rozpatrywanym przypadku kinematyka odwrotna pozwala na wyliczenie kątów w przegubach nogi dla zadanego położenia jej końca – stopy. Poniżej przeprowadzimy obliczenia, na podstawie rysunku 2.4, pozwalające na jej wyznaczenie.



Rysunek 2.3 Zamocowania nóg na korpusie robota



Rysunek 2.4 Widok nogi z góry i w jej płaszczyźnie

Załóżmy, że zadajemy położenie końca nogi $s = (s_x, s_y, s_z)^T$. Z twierdzenia Pitagorasa otrzymujemy zależność

$$d^2 = s_z^2 + l_{23}^2, \tag{2.8}$$

gdzie $l_{23} = l_{123} - l_1$ a $l_{123} = \sqrt{s_x^2 + s_y^2}$. Równocześnie z twierdzenia cosinusów mamy

$$d^2 = l_2^2 + l_3^2 - 2l_2 l_3 \cos\alpha, \tag{2.9}$$

co po odpowiednim przekształceniu pozwala na wyliczenie

$$\cos\alpha = \frac{l_2^2 + l_3^2 - d^2}{2l_2 l_3}. \tag{2.10}$$

Z rysunku 2.4 widać, że

$$q_3 = \pi - \alpha, \tag{2.11}$$

co daje

$$\cos q_3 = -\cos\alpha. \tag{2.12}$$

Podstawiając otrzymujemy

$$q_3 = \arccos \frac{s_z^2 + (l_{123} - l_1)^2 - l_2^2 - l_3^2}{2l_2 l_3}. \tag{2.13}$$

Na rysunku 2.4 widać także, że

$$\tan\beta = \frac{a}{l_2 + b}, \tag{2.14}$$

gdzie $a = l_3 \sin q_3$ oraz $b = l_3 \cos q_3$, co prowadzi do

$$\beta = \arctan \frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3}. \tag{2.15}$$

Zauważyć także można, że

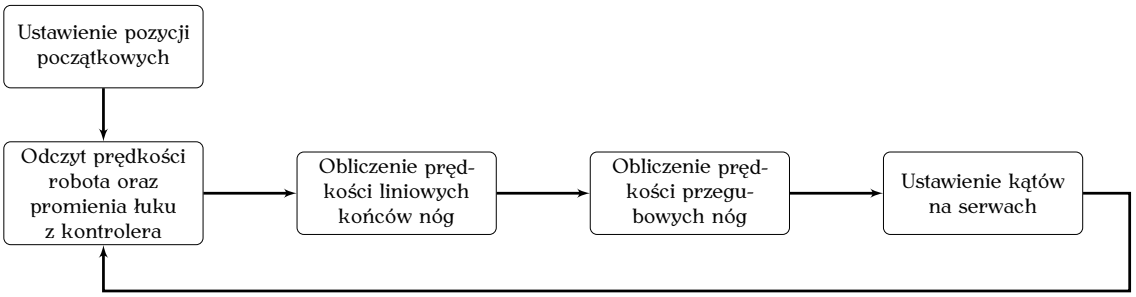
$$\tan(q_2 - \beta) = \frac{s_z}{l_{23}}, \tag{2.16}$$

co po przekształceniu daje

$$q_2 = \beta + \arctan \frac{s_z}{l_{123} - l_1}. \tag{2.17}$$

Rezultatem podstawienia jest

$$q_2 = \arctan \frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3} + \arctan \frac{s_z}{l_{23}}. \tag{2.18}$$



Rysunek 2.5 Diagram planowania ruchu

Z rysunku 2.4 wyznaczamy także

$$\tan q_1 = \frac{s_y}{s_x} \tag{2.19}$$

co daje

$$q_1 = \arctan \frac{s_y}{s_x}. \tag{2.20}$$

Podsumowując obliczenia kinematyki odwrotnej, otrzymujemy kąty $q_N = (q_1, q_2, q_3)^T$ z

$$\begin{aligned} q_1 &= \arctan \frac{s_y}{s_x}, \\ q_2 &= \arctan \frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3} + \arctan \frac{s_z}{l_{23}}, \\ q_3 &= \arccos \frac{s_z^2 + (l_{123} - l_1)^2 - l_2^2 - l_3^2}{2 l_2 l_3}. \end{aligned} \tag{2.21}$$

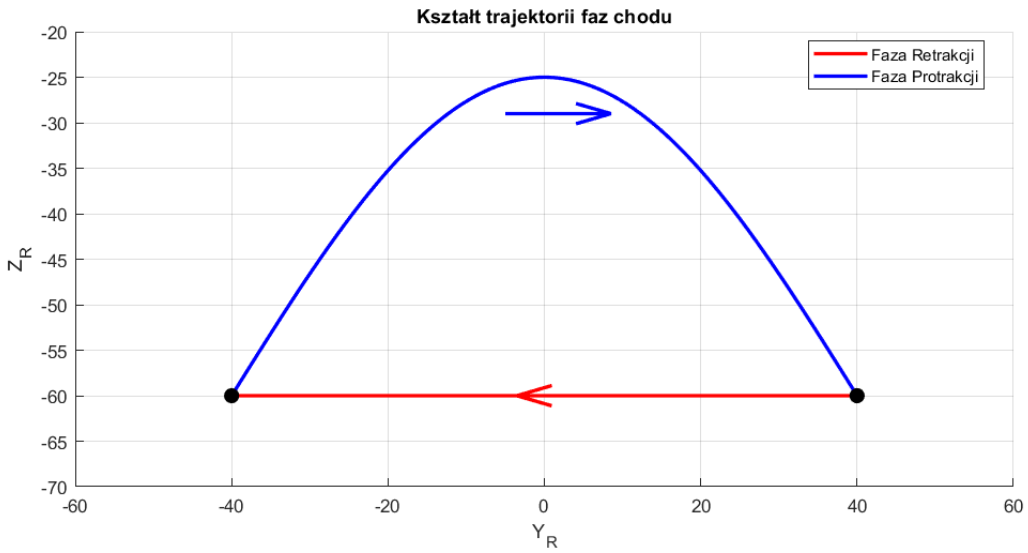
2.2 Planowanie ruchu robota kroczącego

Planowanie ruchu robota kroczącego [Zie14] jest złożonym procesem, który należy rozdzielić na kilka etapów zilustrowanych na rysunku 2.5. W pracy przyjęto, że ruch ten odbywać się będzie wzdłuż trajektorii w kształcie łuku o zadanym promieniu z zadaną prędkością.

2.2.1 Podstawy chodu robotów kroczących

Podstawowym elementem analizy chodu robotów kroczących jest rozróżnienie dwóch naprzemiennych faz ruchu każdej nogi – retrakcji i protrakcji:

- faza retrakcji, zwana także fazą podparcia, to moment, w którym noga pozostaje w kontakcie z podłożem, wprawiając korpus robota w ruch.
- faza protrakcji, to faza przenoszenia, podczas której noga jest unoszona i przemieszczana do nowego punktu podparcia. Przenoszenie końca nogi, patrząc od boku robota, odbywa się po trajektorii w kształcie przykładowo fragmentu okręgu, elipsy, lub sinusoidy co schematycznie, przedstawione zostało na rysunku 2.6.



Rysunek 2.6 Przykładowy kształt trajektorii faz chodu dla końca nogi, względem środka robota

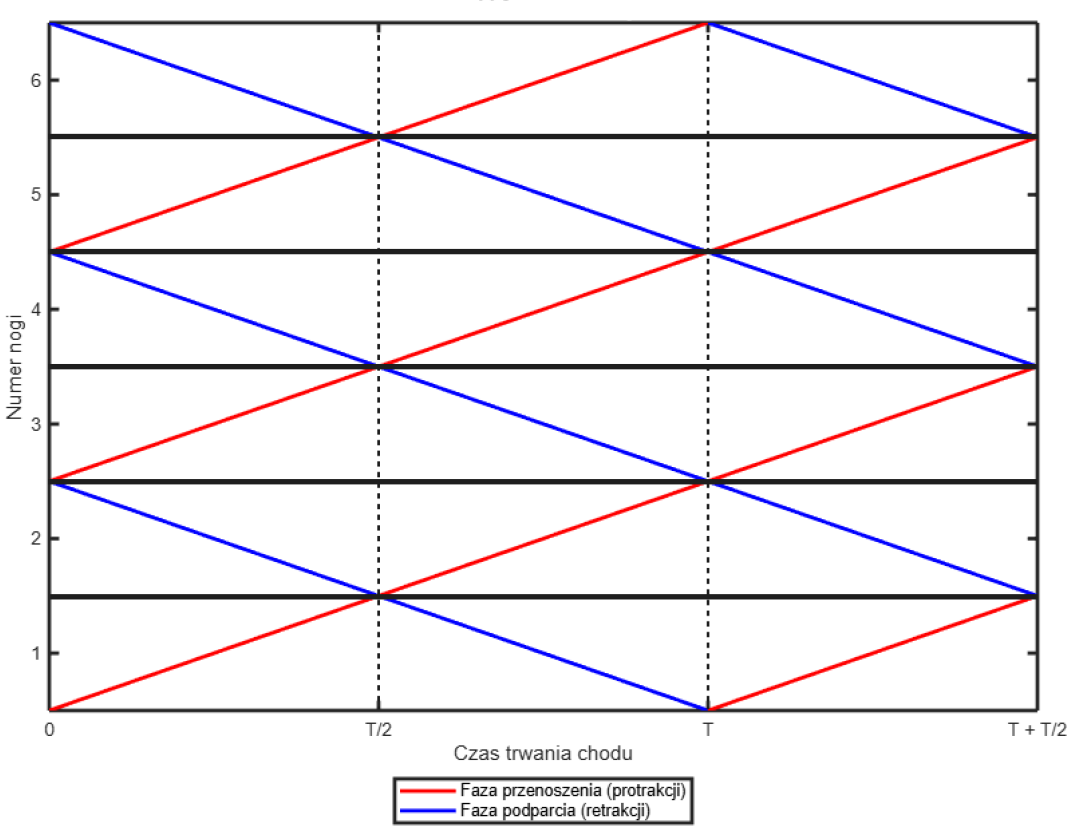
Cały cykl ruchu nogi, zwany okresem T, składa się z czasu trwania obu faz, przy czym każda z nich zajmuje połowę okresu. Sekwencja ruchów nóg robota, określana jako algorytm chodu, definiuje kolejność, w jakiej nogi są przestawiane. Dla sześcionożnych robotów kroczących najbardziej efektywnym algorytmem jest chód trójpodporowy [Gie12], przedstawiony schematycznie na rysunku 2.7. Polega on na tym, że w każdej chwili robot opiera się w trzech tworzących płaszczyznę punktach podparcia. W tym czasie pozostałe trzy nogi przenoszone są do nowych punktów kontaktu z podłożem. Takie podejście sprawia, że przemieszczenie wszystkich sześciu nóg trwa tylko dwie fazy chodu. Wadą tego algorytmu jest to, że w trakcie chodu ciężar robota jest unoszony tylko przez 3 nogi oraz, że zapewnia mniejszą stabilność w stosunku do algorytmów chodu, gdzie punktów podparcia jest więcej.

2.2.2 Pozycja startowa chodu

Przed rozpoczęciem chodu, korzystając z kinematyki odwrotnej (2.1.2), należy zadać pozycje początkowe s_x, s_y, s_z dla wszystkich stóp robota. Przy ich ustawieniu należy mieć na uwadze ograniczenia mechaniczne w postaci maksymalnych kątów w przegubach, które fizycznie jesteśmy w stanie osiągnąć, a także aby poszczególne nogi nie zahaczały wzajemnie o siebie w trakcie ruchu. Zadana pozycja startowa powinna zapewnić odpowiedni rozkład nóg, względem środka ciężkości korpusu robota, aby uniknąć jego opadania.

2.2.3 Wyliczenie prędkości liniowych nóg

Dla każdego końca nogi należy wyznaczyć wektory ich prędkości liniowych, które sprawią że środek korpusu robota będzie poruszał się wzdłuż trajektorii w kształcie łuku. W tym celu wprowadzono dodatkowy układ współrzędnych, powiązany ze środkiem łuku. Relację pomiędzy tym układem a układem współrzędnych zwią-



Rysunek 2.7 Diagram chodu trójpodporowego

zanym ze środkiem korpusu robota przedstawiono schematycznie na rysunku 2.8. Poniżej przeprowadzono odpowiednie obliczenia tych prędkości dla fazy protrakcji.

Jak wspomniano w podrozdziale 2.2, wartościami zadanymi są promień łuku R oraz prędkość liniowa środka korpusu robota v, co daje wartość jego prędkości kątowej

$$\omega = \frac{v}{R}. \tag{2.22}$$

Wektor prędkości kątowej jest prostopadły do płaszczyzny ruchu, co daje

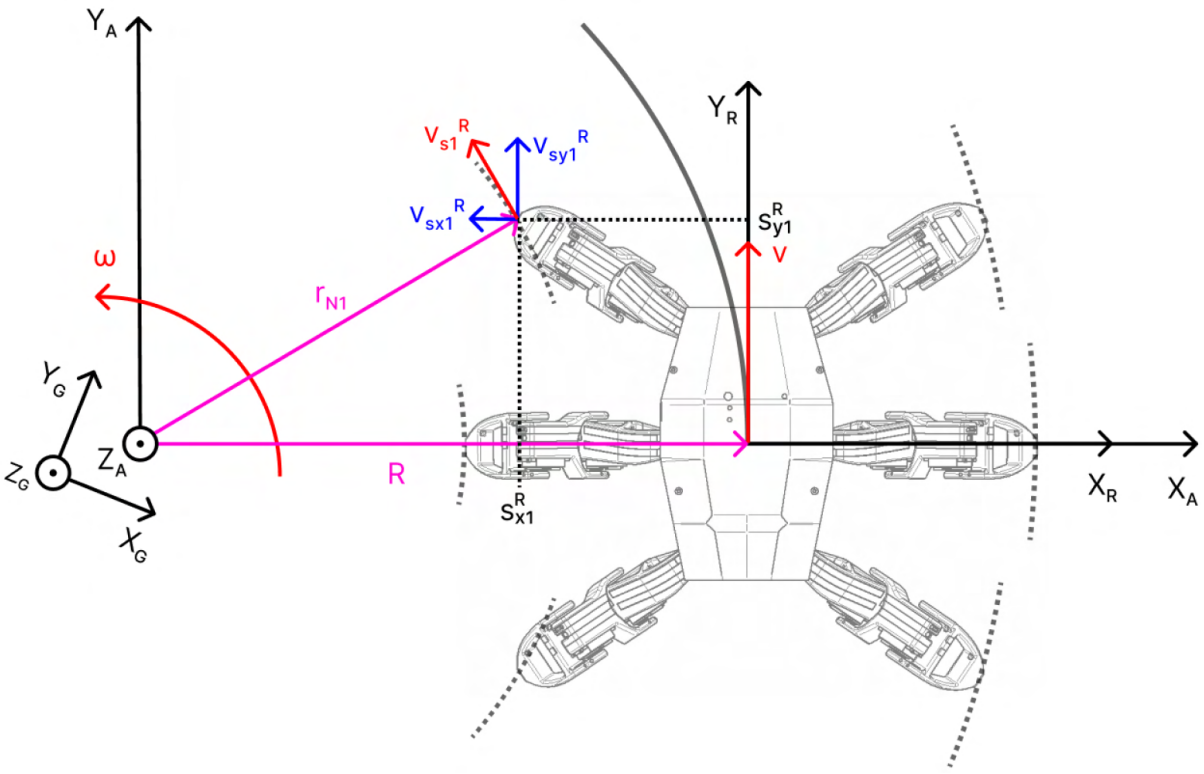
$$\omega^A = \omega^R = \begin{pmatrix} 0 \\ 0 \\ \omega \end{pmatrix}, \tag{2.23}$$

gdzie ω^A, ω^R oznaczają wektor prędkości kątowej, odpowiednio w układzie współrzędnych związanym z łukiem oraz w układzie związanym ze środkiem robota.

Pozycja

$$r^R_{si} = \begin{pmatrix} s^R_{xi} \\ s^R_{yi} \\ 0 \end{pmatrix} \tag{2.24}$$

określa aktualne położenie końca i-tej nogi wyrażonej w układzie związanym ze środkiem robota. Aby wyrazić je w układzie związanym ze środkiem łuku wyliczamy



Rysunek 2.8 Ruch robota po łuku

$$r_{si}^{A} = r_{si}^{R} + \begin{pmatrix} R \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} s_{xi}^{R} + R \\ s_{yi}^{R} \\ 0 \end{pmatrix}. \tag{2.25}$$

Wiedząc, że środek robota ma przemieszczać się z prędkością kątową ω, otrzymujemy prędkość końca i-tej nogi

$$v_{si}^{A} = \omega^{A} \times r_{si}^{A} = \begin{pmatrix} -\omega s_{yi}^{R} \\ \omega(s_{xi}^{R} + R) \\ 0 \end{pmatrix}, \tag{2.26}$$

gdzie v_{si}^{A} to prędkość liniowa końca i-tej nogi w układzie łuku.
Oba przyjęte układy współrzędnych są do siebie równoległe, więc

$$v_{si}^{A} = v_{si}^{R}. \tag{2.27}$$

Ponieważ powyższe obliczenia wektora prędkości liniowych końca nogi zostały przeprowadzone dla fazy protrakcji, dla fazy retrakcji należy przyjąć wektor o przeciwnym zwrocie

$$\begin{aligned} v_{si_p}^{R} &= v_{si}^{R} \\ v_{si_r}^{R} &= -v_{si}^{R} \end{aligned} \tag{2.28}$$

gdzie $v_{si_p}^{R}$ oraz $v_{si_r}^{R}$ oznaczają kolejno prędkość liniową końca nogi w fazie protrakcji oraz retrakcji.

Dodatkowo dla fazy protrakcji należy zadać odpowiednią prędkość w osi Z_R, która spowoduje przemieszczenie końca nogi wzdłuż trajektorii o odpowiednim kształcie, co wspomniane zostało w 2.2.1.

2.2.4 Wyliczenie prędkości przegubowych nogi

Zależność prędkości liniowej końca nogi (stopy) od prędkości kątowych jej przegubów opisuje równanie jakobianowe

$$\dot{S} = J_a(q)\dot{q}, \tag{2.29}$$

gdzie $s = (s_x, s_y, s_z)^T$ – współrzędne położenia stopy, $q = (q_1, q_2, q_3)^T$ – kąty w przegubachnogi a $J_a(q)$ jakobian analityczny nogi we współrzędnych położenia. W celu wyznaczenia zależności prędkości kątowych w przegubach od prędkości liniowych należy skorzystać z równania (2.29) przekształconego do postaci

$$\dot{q} = J_a^{-1}(q)\dot{s}. \tag{2.30}$$

Dla kinematyki prostej (2.6) jakobian analityczny wyrażony we współrzędnych położenia ma postać

$$J_a(q) = \begin{bmatrix} -s_1(c_{23}l_3 + c_2l_2 + l_1) & -c_1(c_{23}l_3 + c_2l_2 + l_1) & -c_1s_{23}l_3 \\ c_1(c_{23}l_3 + c_2l_2 + l_1) & -s_1(c_{23}l_3 + c_2l_2 + l_1) & -s_1s_{23}l_3 \\ 0 & -c_{23}l_3 - c_2l_2 & c_{23}l_3 \end{bmatrix}. \tag{2.31}$$

Jakobian odwrotny do (2.31) wyraża się wzorem

$$J_a^{-1}(q) = \begin{bmatrix} \frac{-s_1}{l_1+l_2c_2+l_3c_{23}} & \frac{c_1}{l_1+l_2c_2+l_3c_{23}} & 0 \\ \frac{c_1c_{23}}{l_2s_3} & \frac{s_1c_{23}}{l_2s_3} & \frac{-s_{23}}{l_2s_3} \\ \frac{-c_1(l_2c_2+l_3c_{23})}{l_2l_3s_3} & \frac{-s_1(l_2c_2+l_3c_{23})}{l_2l_3s_3} & \frac{1}{s_3}\left(\frac{s_2}{l_3} + \frac{s_{23}}{l_2}\right) \end{bmatrix}. \tag{2.32}$$

Rozdział 3

Oprogramowanie

3.1 Algorytm sterowania chodem

Algorytm sterowania chodem, przedstawiony na rysunku 3.1, rozpoczyna się od ustawienia początkowej pozycji nóg. Składa się on z dwóch pętli. Pierwsza z nich jest pętlą nieskończoną, w której odbywa się odczyt danych z kontrolera, na podstawie których wykonywany jest ruch. Dodatkowo, w tej pętli zerowane są zmienne t – aktualny czas trwania drugiej pętli oraz t_s – czas trwania jednej fazy ruchu. Obie te zmienne pełnią funkcję kontrolną w drugiej pętli, w której każda iteracja trwa Δt. Zachowanie stałego odstępu czasu między iteracjami, możliwe jest poprzez użycie funkcji „`millis`" z biblioteki „WiringPi" [mst]. Druga pętla wykonywana jest przez czas T, oznaczający okres w którym wykonywane są dwie fazy ruchu nogi. Głównym elementem tej pętli jest aktualizacja kątów i pozycji nóg, zależnie od wartości zmiennej t_s. W momencie, gdy czas t_s przekroczy połowę okresu T, następuje zmiana fazy ruchu we wszystkich nogach a czas t_s zostaje wyzerowany. Jeśli dana noga rozpoczęła swoją fazę ruchem retrakcji, jej faza zostaje zmieniona na protrakcję i odwrotnie. Gdy pętla ta się zakończy, ustawiane są pozycje startowe co ma zapobiec narastaniu ewentualnych błędów.

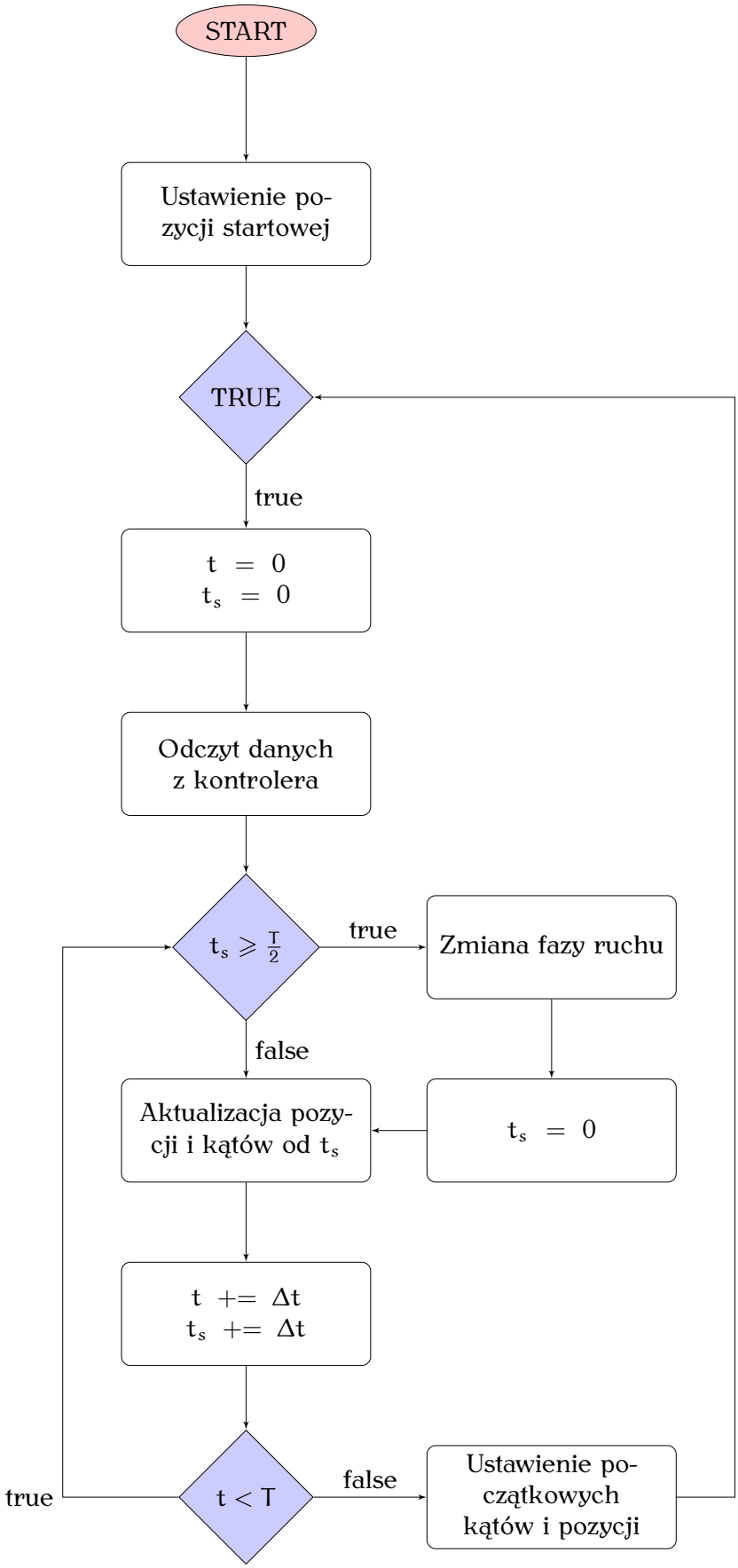
Aktualizacja kątów i pozycji w czasie

W każdej iteracji pętli, korzystając z równań wyznaczonych w podrozdziale 2.2.3 wyliczane są prędkości liniowe końców nóg w zależności od ich aktualnej pozycji oraz fazy w jakiej się znajdują. Następnie wyliczana jest zmiana kątów Δq w przegubach nogi, pozwalająca na wyznaczenie prędkości kątowych z jakimi mają poruszać się przeguby przez czas Δt, aby osiągnąć obliczoną wcześniej prędkość liniową końca nogi. Poniżej opisano sposób wyliczenia Δq,

Pozycja w czasie opisana jest poprzez

$$\Delta p(t) = \begin{pmatrix} v_x t \\ v_y t \\ v_z t \end{pmatrix}, \tag{3.1}$$

gdzie v_x oraz v_y to prędkości liniowe końca nogi wyliczone na podstawie zadanej prędkości środka robota v i promienia R, natomiast prędkość v_z przyjmuje wartości



Rysunek 3.1 Schemat blokowy algorytmu

$$v_z(t) = \begin{cases} \sin\left(\frac{\pi(t-t_{kroku})}{t_{kroku}}\right) h, & \text{dla fazy protrakcji,} \\ 0, & \text{dla fazy retrakcji,} \end{cases} \tag{3.2}$$

gdzie $t_{kroku} = \frac{1}{2}T$ to czas trwania fazy ruchu, natomiast h jest wysokością, na jaką podnoszony jest koniec nogi.

Zmianę położenia w czasie obliczany metodą prostokątów [Wikd]

$$\Delta s(t) = (\Delta p(t + \Delta t) - \Delta p(t)). \tag{3.3}$$

Korzystając z przekształconej formy równania jakobianowego (2.30) obliczamy

$$\Delta q(t) = J_a^{-1}(q_{akt})\Delta s, \tag{3.4}$$

gdzie q_{akt} to aktualne kąty w przegubach. Obliczana co każdą iteracje $\Delta q(t)$ dodawana jest do aktualnych kątów w przegubach, które zadawane są na serwa. Następnie dla nowych wartości kątów obliczane są pozycje końców nóg z wykorzystaniem równań kinematyki prostej.

3.2 Implementacja programu

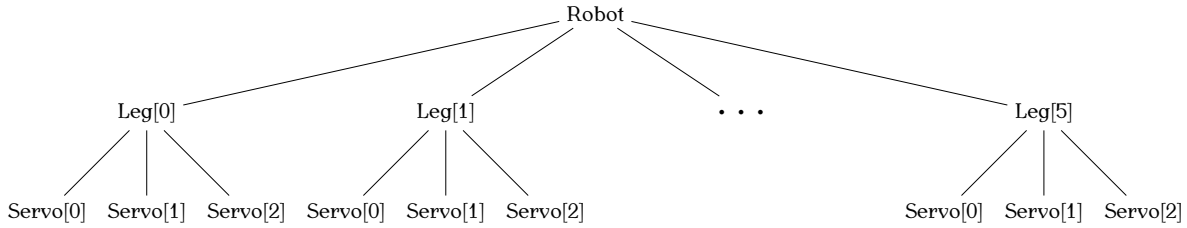
Program sterujący robotem napisany został w języku ANSI C [Ins], na mikrokomputerze Raspberry Pi [Foub]. Do obsługi GPIO [Foua] wykorzystano bibliotekę WiringPi [mst], która znacznie ułatwia obsługę peryferiów takich jak cyfrowe wejścia-wyjścia, czy też wspomaga obsługę protokołu I2C [Wiki]. W trakcie implementacji szczególny nacisk położono na modularność i czytelność kodu. Program został zorganizowany w struktury danych, co pozwoliło na logiczne podzielenie funkcji i łatwiejszą modyfikację. Kod tworzono w środowisku Visual Studio Code (VSCode) [Micb], które oferuje wygodne narzędzia wspierające proces programowania, w tym automatyczne uzupełnianie kodu, debugowanie i integrację z systemami kontroli wersji. Do zdalnego połączenia z Raspberry Pi, korzystano z wtyczki „Remote SSH" [Mica].

3.2.1 Opis struktur programu

Ze względu na złożoność programu oraz dużą liczbę aktuatorów, które muszą być kontrolowane w trakcie działania programu, konieczne było zaprojektowanie odpowiedniego układu struktur danych, który pozwoli na efektywne zarządzanie całym systemem. Zastosowane podejście opiera się na hierarchicznej budowie przedstawionej na rysunku 3.2. Jako główny element hierarchii utworzono strukturę „`Robot`", która pełni rolę kontenera dla sześciu obiektów typu „`Leg`". Każdy z tych obiektów składa się z trzech struktur typu „`Servo`", które służą do sterowania poszczególnymi przegubami w danej nodze.

Struktura Robot

Struktura „`Robot`" oprócz sześciu obiektów typu „`Leg`", przechowuje także aktualną oraz startową pozycję (służącą do korekcji błędów) końca każdej z sześciu



Rysunek 3.2 Schemat blokowy struktur danych

```
typedef struct Robot
{
    Leg _legs[6];
    Vector3 _LegsPositionRobotCenter[6];
    Vector3 _LegsStartPositions[6];
    StepFase _robotStepFase;
    double _robot_velocity;
} Robot;
```

Rysunek 3.3 Definicja struktury „Robot".

nóg, wyrażone w układzie środka korpusu robota $X_R Y_R Z_R$. Struktura ta przechowuje również prędkość środka robota w $[\frac{mm}{s}]$, a także obiekt typu enum „StepFase" zawierający informacje o stanie, w jakim w jakim znajduje się robot. Definicję tej struktury przedstawia rysunek 3.3.

Struktura Leg

Struktura "Leg", której definicję przedstawiono na rysunku 3.4, opisuje jedną nogę robota. Zawiera dane o miejscu zamocowania nogi (przód, środek, tył) oraz o stronie robota (lewa lub prawa). Ponadto, przechowuje indeks kontrolera PCA, który zarządza serwami danej nogi, a także tablicę trzech obiektów „Servo", które reprezentują przeguby (q_1, q_2, q_3). W strukturze znajduje się także informacja o fazie ruchu danej nogi oraz jej aktualnej pozycji w powiązanym z nią układzie współrzędnych $X_N Y_N Z_N$. Dodatkowo struktura przechowuje wartość aktualnej prędkości liniowej końca nogi, krzywych ruchu, zmianę kątów w czasie Δt oraz aktualnych wartości kątów w przegubach, wyrażonych w radianach. Na końcu znajduje się również zmienna przechowująca początkowe kąty, używane do korekcji błędów w ruchu.

Struktura Servo

Struktura „Servo", której definicję przedstawiono na rysunku 3.5, opisuje jeden serwomechanizm. Przechowuje ona aktualną wartość kąta serwa wyrażoną w stopniach, a także zawiera informacje o przypisanym kanale na kontrolerze PCA, który umożliwia sterowanie serwem, oraz o minimalnym i maksymalnym kącie jego pracy, określającym fizyczny zakres ruchu.

```
typedef struct Leg
{
    LegMount _leg_mount;
    RobotSide _side;
    uint8_t _pca;
    Servo _leg_servos[3];
    LegFase _leg_fase;
    Vector3 _leg_position;
    Vector3 _leg_linear_velocity;
    Vector3 _leg_curve_t;
    Vector3 _leg_curve_t_delta_t;
    Vector3 _leg_q_delta;
    Vector3 _leg_actual_q;
    Vector3 _leg_start_q;
} Leg;
```

Rysunek 3.4 Definicja struktury „Leg"

```
typedef struct Servo
{
    uint8_t _pca_channel;
    double _min_angle;
    double _max_angle;
    double _servo_angle;
} Servo;
```

Rysunek 3.5 Definicja struktury „Servo"

```
void writeServo(Servo servo, uint8_t pca)
{
    int pulse_length = SERVO_MIN +(servo._servo_angle * (SERVO_MAX - SERVO_MIN)
    ↪ /270);
    int on_time = 0;
    int off_time = pulse_length;
    int led_on_l = LED0_ON_L + 4 * servo._pca_channel;
    int led_off_l = led_on_l + 2;
    wiringPiI2CWriteReg16(pca, led_on_l, on_time);
    wiringPiI2CWriteReg16(pca, led_off_l, off_time);
}
```

Rysunek 3.6 Funkcja „writeServo"

3.2.2 Przetwarzanie danych z kontrolera

Do odczytywania danych z kontrolera robota wykorzystano bibliotekę Simple DirectMedia Layer 2 (SDL2) [lib], która umożliwia obsługę urządzeń wejściowych, takich jak klawiatury, myszy, joysticki czy inne kontrolery. Pozwala ona w prosty sposób odczytywać sygnały z kontrolera.

Wartości z osi analogowych kontrolera są odczytywane w zakresie od -32767 do 32767. Aby odpowiednio sterować prędkością oraz promieniem skrętu robota, konieczne jest zastosowanie funkcji mapującej, która przekształca te wartości na odpowiadające im minimalne i maksymalne wartości prędkości oraz promienia skrętu. W przypadku, gdy odczyt z osi analogowej kontrolera dla promienia wynosi zero, jest on zastępowany bardzo dużą wartością równą 10^8 [mm], co odpowiada niemal prostoliniowemu ruchowi robota. Dodatkowo, gdy środek łuku znajduje się bardzo blisko środka korpusu robota, prędkości końców nóg gwałtownie rosną. W takich sytuacjach prędkość robota jest odpowiednio ograniczana, aby zapobiec zderzaniu się ze sobą nóg robota.

3.2.3 Sterowanie serwomechanizmami

Serwomechanizmy są sterowane sygnałem PWM o częstotliwości 50 Hz, gdzie czas impulsu w zakresie 0,5 ms do 2,5 ms odpowiada kątowi 0° do 270°. Czas trwania tego impulsu oblicza się ze wzoru

$$\text{pulse\_width} = \text{min\_pulse} + \left(\frac{\alpha}{270^\circ}\right)(\text{max\_pulse} - \text{min\_pulse}), \tag{3.5}$$

gdzie „min_pulse" to czas trwania impulsu odpowiadający kątowi 0°, „max_pulse" to czas trwania impulsu odpowiadający kątowi 270° a α to zadany kąt w stopniach.

Wynik obliczeń zapisywany jest do rejestrów `LEDx_ON_L` oraz `LEDx_OFF_L` kontrolera PCA9685, który odpowiada za generowanie sygnałów PWM. Rejestry te określają moment włączenia i wyłączenia impulsu PWM dla odpowiedniego kanału, oznaczonego jako „x". Nazwy tych rejestrów wynikają z pierwotnego przeznaczenia układu do sterowania diodami led. Funkcja, która ustawia zadany kąt na serwie, przedstawiona jest na rysunku 3.6.

Rozdział 4

Konstrukcja robota

4.1 Konstrukcja mechaniczna

Konstrukcja mechaniczna robota kroczącego powinna łączyć wysoką wytrzymałość i sztywność z niską masą korpusu. Stabilność i płynność poruszania się w dużej mierze zależą od doboru odpowiednich napędów, które muszą utrzymać ciężar robota podczas chodu, a także istotne jest właściwe rozmieszczenie nóg, ich liczba oraz długość poszczególnych członów. Kluczowe jest też zapewnienie szerokiego zakresu ruchu odnóży. W tym podrozdziale przedstawione zostały możliwe rozwiązania konstrukcyjne robotów kroczących, a także opisano budowę mechaniczną sześcionożnego robota kroczącego.

4.1.1 Przegląd rozwiązań

Podział ze względu na posturę

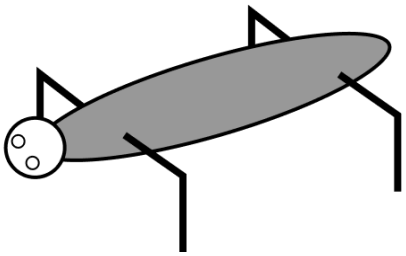
Według [Zie14] roboty kroczące podzielić można ze względu na ich posturę, która wzorowana jest na biologii zwierząt.

Postura gadów (rysunek 4.1) charakteryzuje się tym, że w czasie chodu kąt w stawie kolanowym jest bliski kątowi prostemu a korpus jest „zawieszony" na nogach, zazwyczaj takie roboty posiadają dwie pary odnóży. Przykładem takiego robota jest [ZH02], który jest zdolny do rekonfiguracji nóg do gadziej postury.

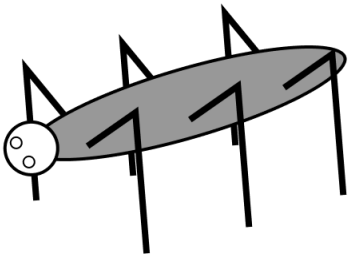
Postura owadów (rysunek 4.2), zastosowana w [Jan07], podobnie jak postura gadów, charakteryzuje się zawieszonym na nogach korpusie, natomiast kąty w kolanach są ostre, co sprawia że sam korpus jest zawieszony stosunkowo niżej. Główną zaletą takiej postury jest wysoka stabilność chodu, dzięki nisko położonemu środkowi ciężkości. Takie roboty posiadają zazwyczaj dwie lub trzy pary odnóży krocznych. Podejście to zapewnia najmniejsze zużycie energii.

Postura czworonogów, (rysunek 4.3) na bazie której został zaprojektowany robot [Dynb], gdzie korpus oparty jest na nogach wysoko nad podłożem, sprawiać może duże problemy z zachowaniem stabilności. Postura ta jest najmniej wydajna energetycznie, natomiast gwarantuje wysokie możliwości chodu po zróżnicowanym terenie, a także zapewnia dużą prędkość przemieszczenia się.

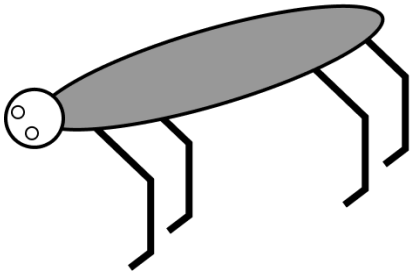
Postura humanoidalna (rysunek 4.4), zastosowana w [Dyna], nawiązuje do postury ludzkiej, w której korpus oparty jest tylko na dwóch nogach. Nogi te muszą



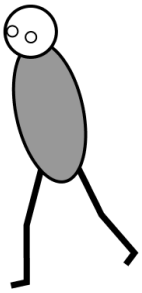
Rysunek 4.1 Postura gadów



Rysunek 4.2 Postura owadów



Rysunek 4.3 Postura czworonogów



Rysunek 4.4 Postura humanoidalna

posiadać więcej stopni swobody aby zapewnić stabilność w trakcie chodu. W momencie przekładania jednej nogi, cały ciężar robota musi zostać przemieszczony na nogę stojącą.

Podział ze względu na rodzaj stabilności chodu

Dla robotów kroczących wyróżnia się także podział ze względu na rodzaj stabilności, który jest ściśle związany z liczbą nóg [For].

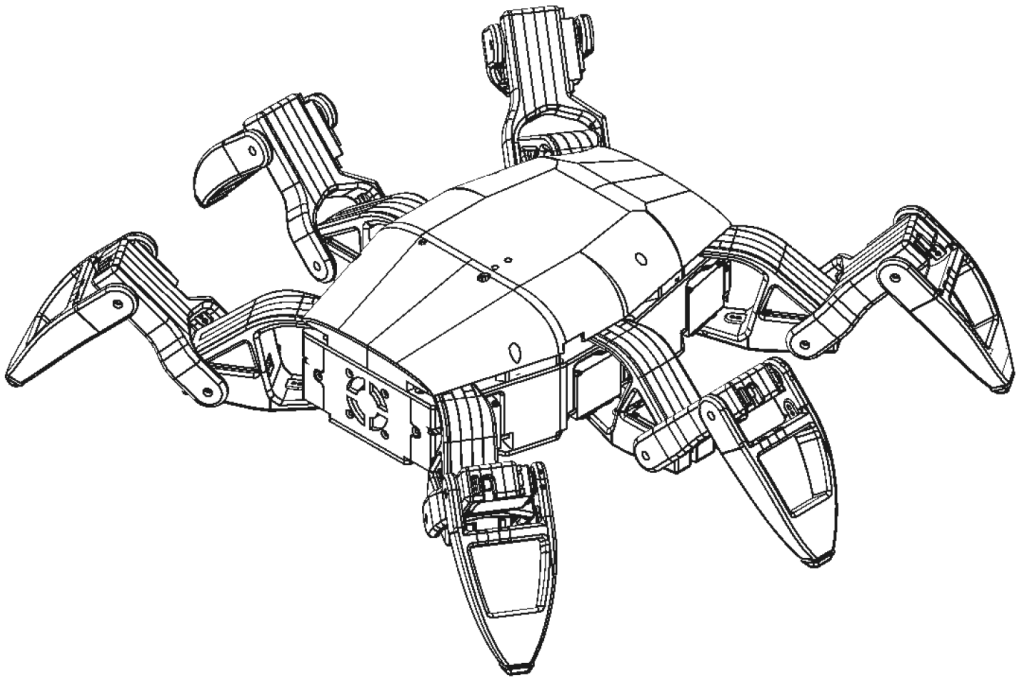
Chód stabilny statycznie, występuje wtedy kiedy robota można zatrzymać w dowolnym momencie, nie tracąc stabilności. Najlepszym tego przykładem jest trójpodporowy chód robotów sześcionożnych.

Chód dynamicznie stabilny charakteryzuje się tym, że stabilność jest utrzymywana dzięki dynamice ruchu a zatrzymanie robota w nieodpowiednim momencie skutkuje jego upadkiem. Przykładem tego ruchu jest szybkie poruszanie się robota ruchem przypominającym galop [Wikf], w którym występują momenty, że nie dotyka on podłoża.

Ruch quasi-statycznie stabilny charakteryzuje się tym, że pomiędzy fazami stabilnymi występują fazy, w których stabilność jest utrzymywana tylko dynamicznie. Przykładem takiego ruchu jest poruszanie się dwunożnego robota, który przewróci się, jeśli zostanie zatrzymany podczas stania na jednej nodze.

4.1.2 Projekt mechaniki

Przy projektowaniu mechaniki robota zdecydowano się na posturę owada, z sześcioma nogami, ze względu na niskie wymagania energetyczne i dużą stabilność. Mechanika robota zaprezentowana na rysunku 4.5, została zaprojektowana w programie Autodesk Inventor [Aut, Wikb], dzięki czemu do wykonania jej fizycznych



Rysunek 4.5 Mechanika sześcionożnego robota kroczącego

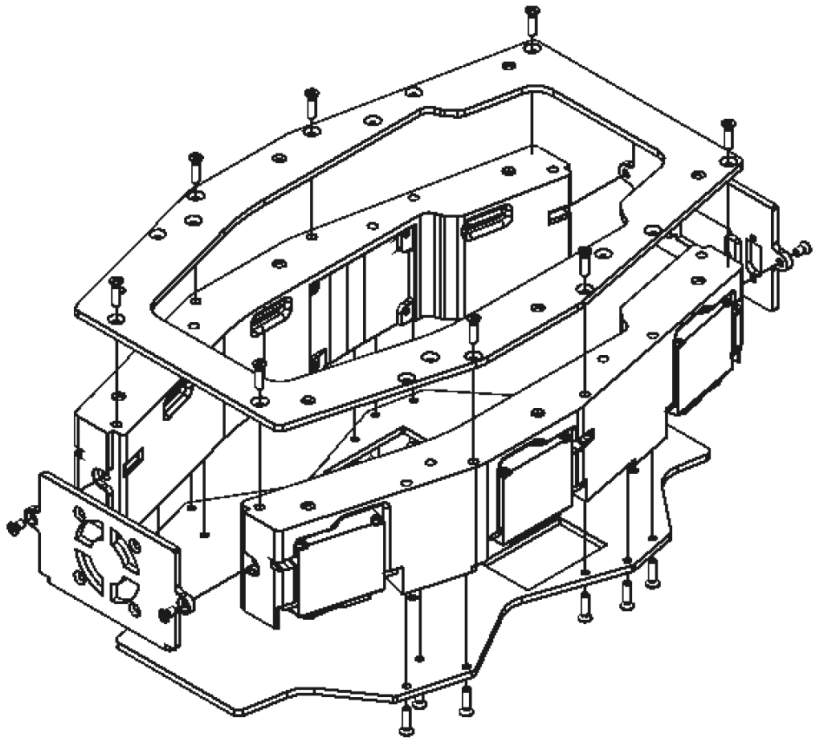
elementów możliwe było zastosowanie druku 3D w technologii FDM [Wike]. Przygotowanie plików GCode [Wikg] dla drukarki 3D odbyło się w programie Bambu Studio [Bam]. Jako materiał druku wykorzystano PLA [Wikl]. Rysunki techniczne poszczególnych elementów konstrukcji zostały opisane w dodatku A.

Konstrukcja nośna

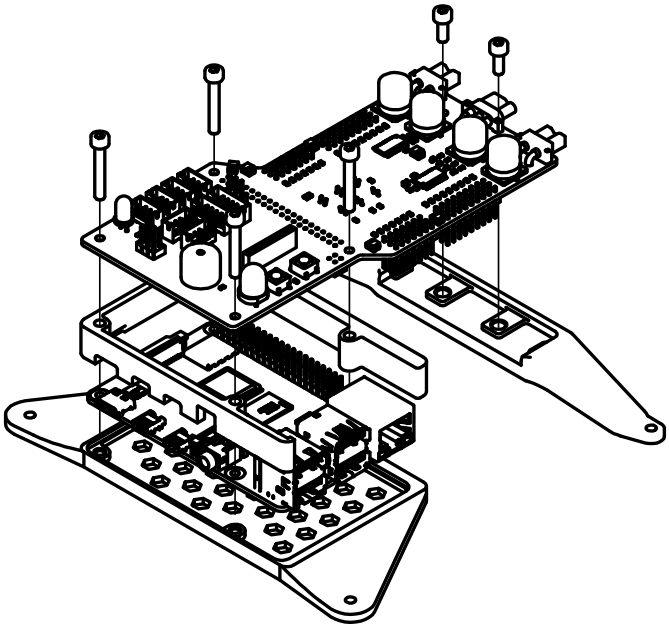
Konstrukcja nośna (rysunek 4.6) stanowi korpus robota do którego przymocowane jest 6 nóg. W jego środku umieszczona została płytka zasilająca, akumulator i przetwornice. Natomiast na górze korpusu zamocowany został uchwyt na płytke sterująca oraz RaspberryPi, którego złożenie przedstawiono na rysunku 4.7. Rozmieszczenie elementów elektronicznych w robocie przedstawiono na rysunku 4.8. Korpus składa się z dwóch ścianek (rysunek 4.9) będących jednocześnie mocowaniami nóg, górnej i dolnej płyty, dodatkowych płyt z przodu oraz z tyłu robota, do których przymocowane są elementy takie jak wentylator, wskaźnik naładowania baterii, włącznik oraz złącze do ładowania baterii.

Projekt nogi robota

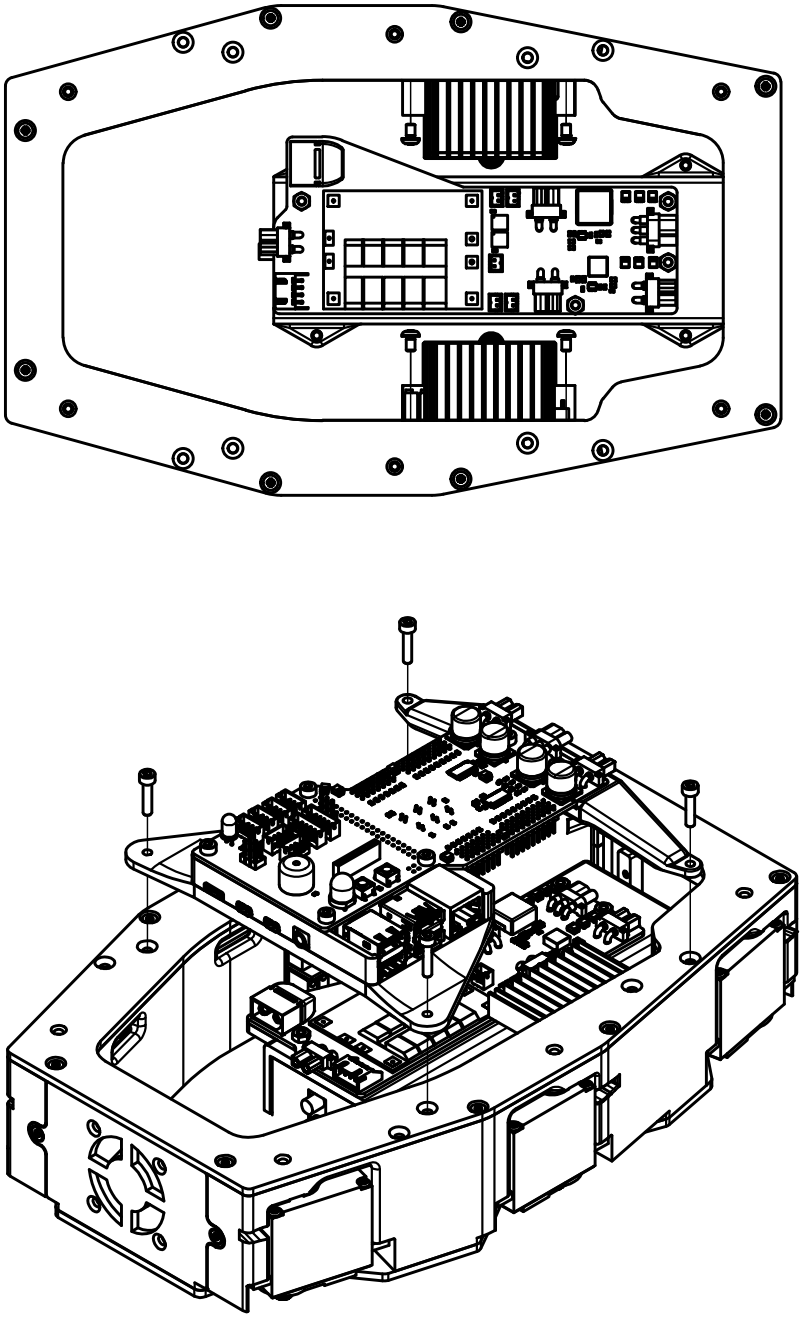
Noga robota składa się z trzech członów w sposób widoczny na rysunku 4.10. Dwa pierwsze człony, których budowa została przedstawiona na rysunkach 4.11 oraz 4.12 mają zamontowane serwomechanizmy. Trzeci człon ma wbudowany czujnik dotyku podłoża a jego konstrukcja przedstawiona została na rysunku 4.13. Stopa, czyli element który stanowi punkt podparcia dla nogi, został wykonany z materiału TPU (ang. Thermoplastic polyurethane) [Wikp]. Posiada on właściwości zbliżone do



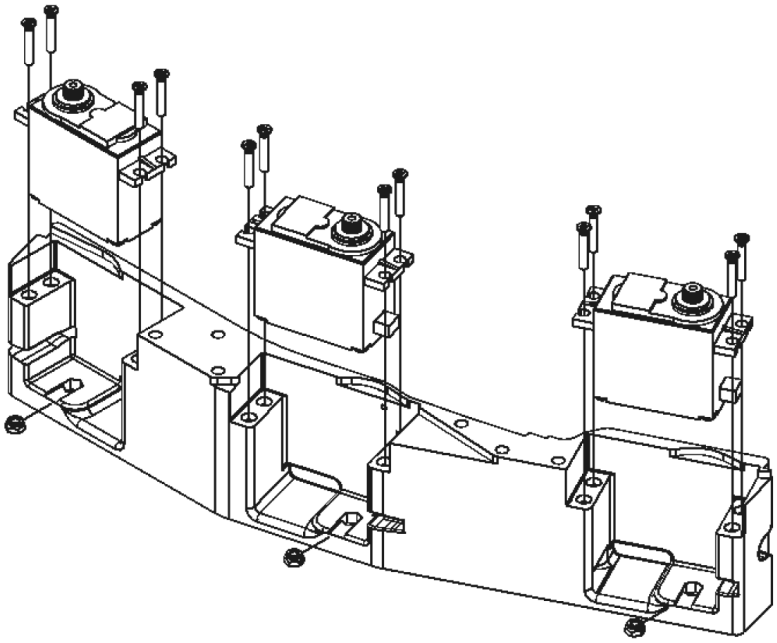
Rysunek 4.6 Złożenie korpusu robota



Rysunek 4.7 Złożenie uchwytu na płytke sterującą i RaspberryPi



Rysunek 4.8 Rozmieszczenie elementów elektronicznych w robocie



Rysunek 4.9 Złożenie ściany bocznej korpusu

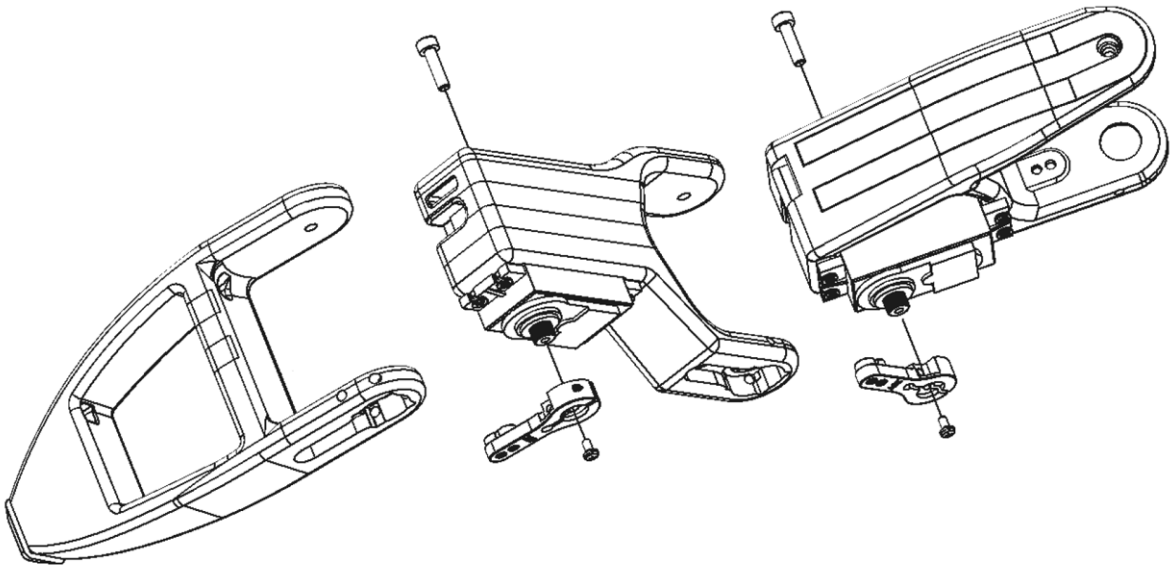
gumy, co ma na celu zwiększenie współczynnika tarcia między podłożem a końcem nogi, aby uniknąć jej ślizgania.

4.1.3 Dobór napędów

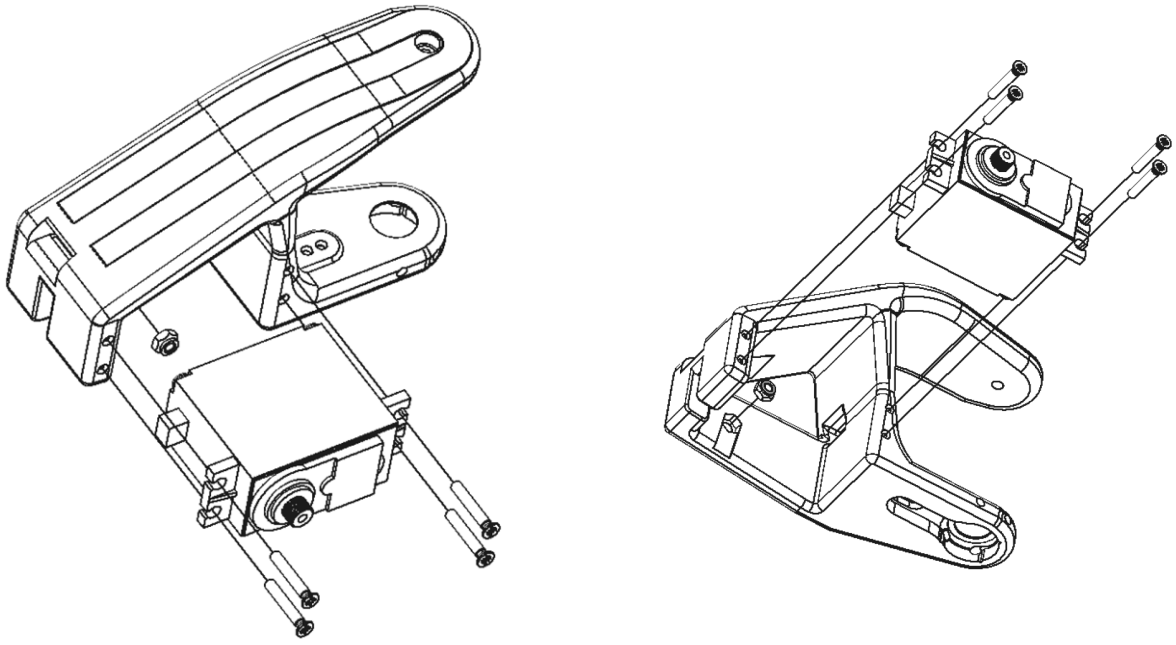
Dobór odpowiednich napędów do sterowania przegubami nóg był kluczowym aspektem projektowania mechaniki robota. Napędy w robocie kroczącym muszą spełniać dwa podstawowe wymagania: zapewniać wystarczający moment obrotowy, aby utrzymać ciężar całego robota i zagwarantować stabilny chód a jednocześnie być na tyle szybkie, by umożliwić płynne ruchy.

W projekcie zdecydowano się na zastosowanie serwomechanizmów, które dzięki swojej dużej przekładni zapewniają wysoki moment obrotowy przy jednocześnie kompaktowej budowie. Wybrano cyfrowe serwomechanizmy. W porównaniu do analogowych serwomechanizmów, rozwiązanie to cechuje się większą stabilnością i odpornością na oscylacje podczas utrzymywania pozycji.

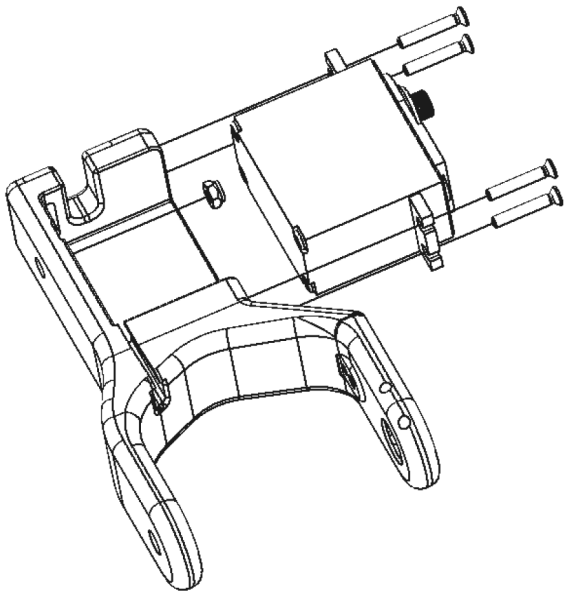
Do przegubów q_2 oraz q_3 zastosowano serwomechanizmy o momencie obrotowym wynoszącym 40 kg*cm^2, ponieważ przeguby te są najbardziej obciążone przy utrzymywaniu korpusu robota w górze. W przegubach q_1 zdecydowano się na serwomechanizmy o momencie obrotowym 20 kg*cm^2, aby zmniejszyć masę oraz koszty konstrukcji, a także zwiększyć prędkość ruchu. Wszystkie serwomechanizmy zastosowane w projekcie są marki DSservo [Teca].



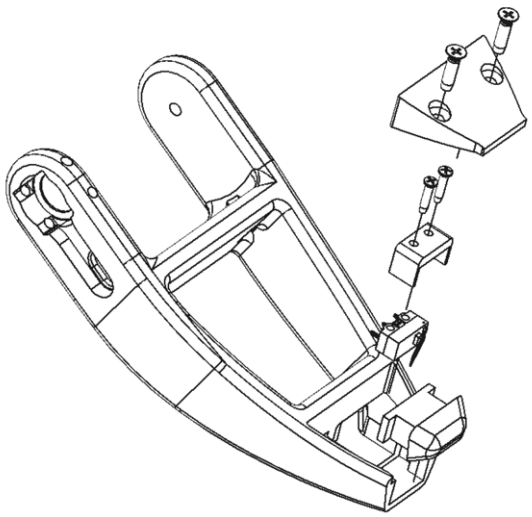
Rysunek 4.10 Złożenie nogi



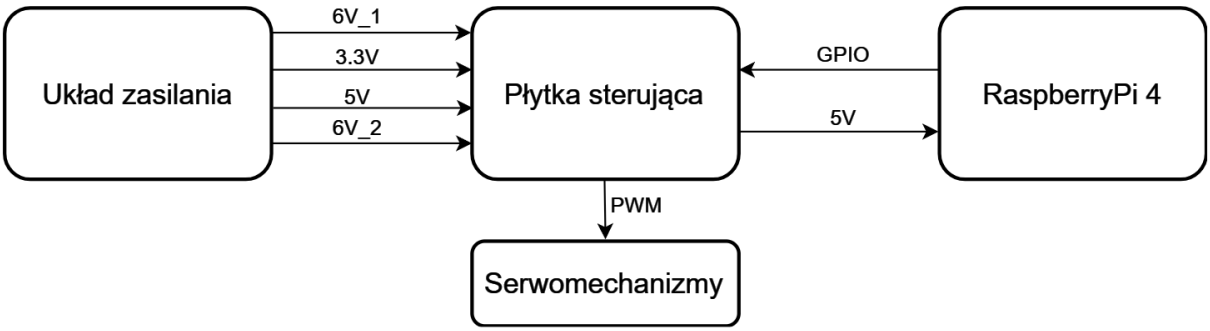
Rysunek 4.11 Złożenie pierwszego członu



Rysunek 4.12 Złożenie drugiego członu



Rysunek 4.13 Złożenie czujnika dotyku w trzecim członie



Rysunek 4.14 Układ elektroniki robota

4.2 Elektronika

Uproszczony schemat elektroniki zastosowanej w robocie kroczącym przedstawiono na rysunku 4.14. Projektowanie schematów oraz płytek PCB zostało zrealizowane przy użyciu programu KiCad [Dev]. W procesie projektowania elektroniki priorytetem była niezawodność, co przełożyło się na zastosowanie prostych rozwiązań. Zdecydowano się również na modułową budowę, co umożliwia łatwą rozbudowę systemu w przyszłości.

W tym podrozdziale, przedstawione zostały wymagania funkcjonalne elektroniki robota kroczącego oraz szczegółowe omówienie projektu i realizacji poszczególnych układów elektronicznych.

4.2.1 Wymagania funkcjonalne

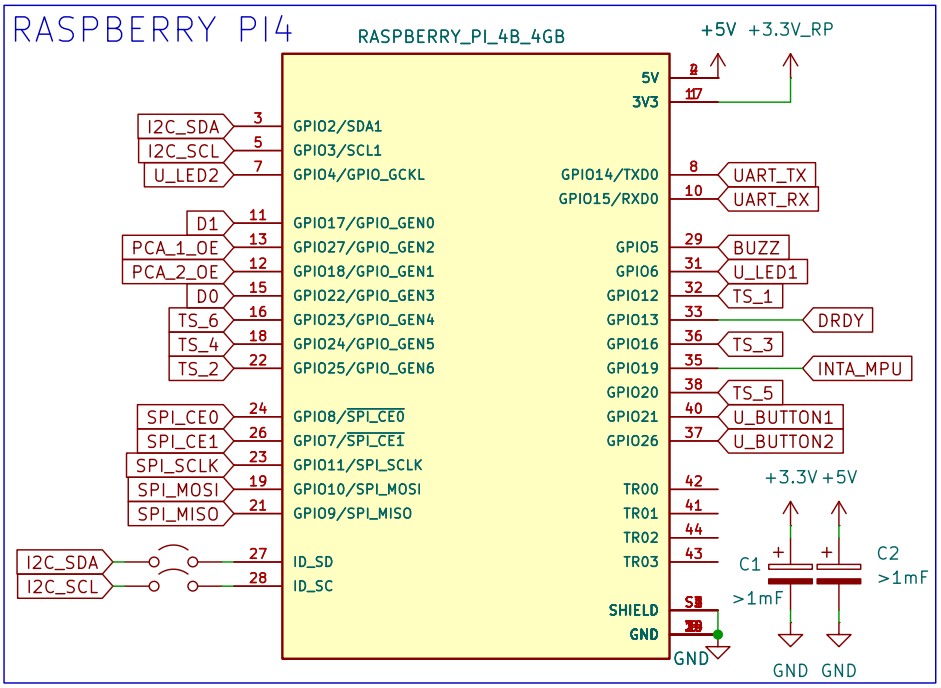
Głównym zadaniem układu elektroniki robota kroczącego jest zapewnienie odpowiedniego zasilania dla serwomechanizmów oraz ich precyzyjne sterowanie, które musi odbywać się z bardzo dużą częstotliwością. Dodatkowo system musi zarządzać zasilaniem pozostałych komponentów robota, zapewniając bezpieczne ładowanie akumulatora oraz ochronę przed nadmiernymi obciążeniami i przepięciami. Układ musi również zapewniać optymalną efektywność energetyczną, by wydłużyć czas pracy robota na jednym ładowaniu.

4.2.2 Elementy składowe

W układzie elektronicznym można wyróżnić elementy składowe takie jak: główny komputer, układ sterowania serwami, układ sensoryczny oraz układ zasilania.

Komputer główny

Jako główny komputer zarządzający całym robotem wykorzystano mikrokomputer RaspberryPi 4b, w wersji z 4GB pamięci RAM, z systemem operacyjnym Raspberry Pi OS [Fouc]. Takie rozwiązanie zapewnia dużą moc obliczeniową przy niewielkich rozmiarach, prostą komunikację oraz duże możliwośći rozbudowy całego systemu. Dużą zaletą RaspberryPi jest to, że pozwala na użycie protokołu SSH [Wikn] w celu połączenia się między robotem a komputerem użytkownika. Dzięki



Rysunek 4.15 Schemat podłączenia RaspberryPi

temu, aby zapewnić komunikację z robotem, wymagane jest jedynie podłączenie go do sieci WiFi. Schemat podłączenia omawianego wyżej urządzenia przedstawiono na rysunku 4.15.

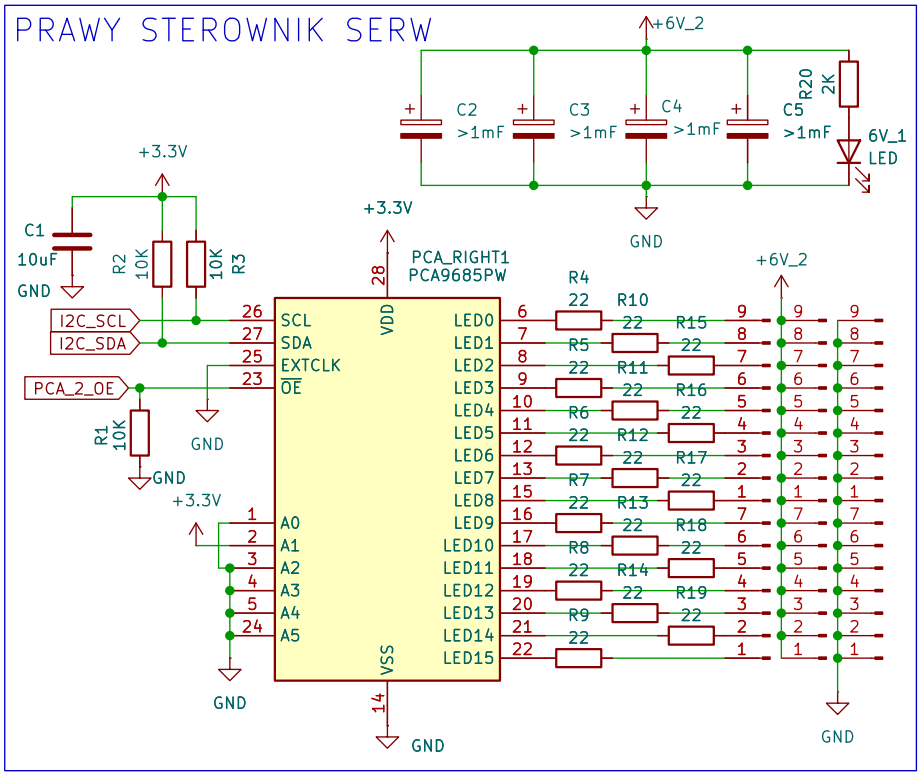
Układ sterowania serwami

Jako układ sterowania serwami zastosowano sterowniki PCA9685 [Sem] pozwalające na kontrolę 16 serwomechanizmów każdy, poprzez generowanie sygnału PWM o odpowiedniej częstotliwości i czasie trwania impulsu. W robocie zastosowano dwa takie układy nadzorujące pracę odnóży po lewej i po prawej stronie robota. Komunikacja z układami PCA9685 odbywa się za pośrednictwem magistrali I2C. Pomiędzy sygnałami sterującymi wychodzącymi z układu a serwomechanizmami zastosowano rezystory o wartości 22 Ω. Ich zadaniem jest ochrona przed wyładowaniami elektrycznymi, które mogłyby powstać na długich przewodach połączeniowych. W celu zabezpieczenia linii zasilającej serwa przed szpilkami napięcia, które mogą wystąpić przy poruszaniu silnikami, oraz aby zapewnić zapas mocy, zastosowano kondensatory elektrolityczne i tantalowe o dużej pojemności. Schemat elektroniczny podłączenia układu PCA9685 przedstawiony został na rysunku 4.16.

Sensoryka

W skład układu sensorycznego robota wchodzą:

- czujniki dotyku, które zostały zbudowane z wykorzystaniem przełączników krańcowych, umieszczonych w końcach nóg. Pozwalają one na wykrycie, czy dana noga robota dotyka podłoża. Wykrywany jest stan logiczny (0 lub 1),



Rysunek 4.16 Schemat podłączenia sterownika PCA9685

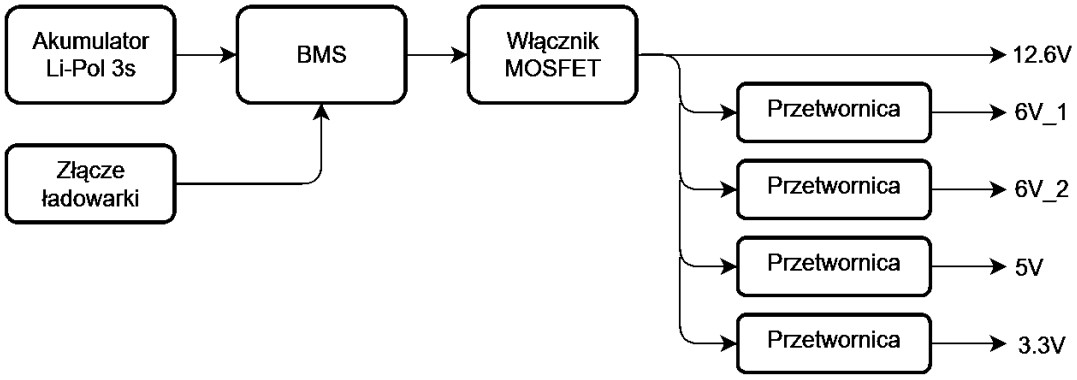
czyli zwarcie do masy lub do zasilania. Złożenie czujnika przedstawiono na rysunku 4.13.

- moduł GY-87 [Sys] zawierający akcelelometr, żyroskop, magnetometr oraz barometr, komunikujący się poprzez magistralę I2C [Wiki],
- złącza pozwalające na podłączenie dodatkowych czujników poprzez magistrale I2C i SPI [Wiko].

Układ zasilania

Układ zasilania dostarcza odpowiednie napięcia zasilające komponenty robota. Składa się on z układu BMS (ang. Battery Management System), włącznika z tranzystorami MOSFET oraz 4 przetwornic. Jako źródło zasilania zastosowano akumulator litowo-polimerowy 3S o pojemności 5000mAh. Schematyczna budowa układu została przedstawiona na rysunku 4.17.

Opisywany układ zasilania prawie w całości umieszczony jest na płytce PCB zaprezentowanej w dodatku B na rysunku B.1. Zastosowany na niej moduł BMS pełni kilka ważnych funkcji, które zabezpieczają akumulator przed uszkodzeniem. Między innymi zabezpiecza akumulator przed zbyt niskim lub zbyt wysokim napięciem na ogniwach oraz wyrównuje ich poziom naładowania. Użyty w układzie model BMS-u posiada też wbudowaną ładowarkę, co sprawia, że w celu naładowania akumulatora wystarczy podłączyć zasilacz o odpowiednim napięciu do wyprowadzonego na zewnątrz obudowy złącza XT30. Na płytce zastosowano dwie przetwornice typu



Rysunek 4.17 Diagram układu zasilania

„step-down", które obniżają napięcie z baterii do 3.3V do zasilania sterowników serw oraz 5V do zasilania komputera głównego. Aby dostarczyć wymagane napięcie 6V do serwomechanizmów zastosowano dwie zewnętrzne przetwornice o maksymalnym prądzie wyjściowym 15A. Układ zasilania został wyposażony w odpowiednie złącza, które pozwalają połączyć linie zasilania do płytki sterującej, zewnętrznych przetwornic, balanser oraz akumulator. W skład układu zasilania wchodzi także moduł [Tecb] z wyświetlaczem diodowym wskazującym użytkownikowi poziom naładowania akumulatora.

4.2.3 Płytki drukowane

Zaprojektowane układy elektroniki robota umieszczono na dwóch płytkach drukowanych PCB. Płytka pierwsza przedstawiona została na rysunku B.1. Znajduje się na niej większość elementów wchodzących w skład układu zasilania robota. Druga płytka, widoczna na rysunku B.4, stanowi rozszerzenie głównego komputera. Umieszczone są na niej sterowniki serwomechanizmów, złącza do zewnętrzych czujników, IMU, buzzer, diody led oraz przyciski.

4.2.4 Kontroler zdalnego sterowania

Do sterowania robotem wykorzystano bezprzewodowy kontroler typu gamepad, wzorowany na konstrukcji kontrolera Xbox 360. Urządzenie wyposażone jest w 17 przycisków oraz posiada 6 osi analogowych.

Komunikacja pomiędzy kontrolerem a głównym komputerem odbywa się za pomocą dedykowanego odbiornika USB, który zapewnia stabilne połączenie bezprzewodowe w paśmie 2,4 GHz o zasięgu do kilkunastu metrów.

Rozdział 5

Testy

W niniejszym rozdziale zebrano wyniki przeprowadzonych w ramach pracy testów w środowisku symulacyjnym oraz na rzeczywistym robocie.

5.1 Testy symulacyjne

Przeprowadzone testy symulacyjne zrealizowane w środowisku „MATLAB" [Mat] miały na celu sprawdzenie przeprowadzanych obliczeń przed ich implementacją w programie robota.

5.1.1 Symulacja prędkości liniowych końców nóg

Przeprowadzono test, którego celem było zweryfikowanie poprawności wzorów wyznaczonych w podrozdziale 2.2.3 oraz graficzne przedstawienie wektorów prędkości liniowych końców nóg w zależności od zadanego promienia łuku R i prędkości środka robota v. Wykonano obliczenia dla trzech wartości promienia:

- mały promień $R = 100mm$,
- średni promień $R = 500mm$,
- bardzo duży promień $R = 10^8 mm$.

We wszystkich przypadkach przyjęto stałą prędkość środka robota $v = 80\frac{mm}{s}$. Wyniki obliczeń dla sześciu nóg zestawiono w tabelach 5.1, 5.2 oraz 5.3. Wizualizacje wektorów prędkości liniowych przedstawiono na rysunkach 5.1, 5.2 oraz 5.3.

Uzyskane wyniki testu doprowadziły do poniższych wniosków.

1. W przypadku małego promienia $R = 100mm$ zaobserwowano, że prędkości liniowe v_{sy} po lewej stronie robota mają przeciwny zwrot w stosunku do prędkości po prawej stronie. Taka zależność pozwala stwierdzić, że im bliżej środka korpusu robota znajduje się środek łuku, tym bardziej ruch robota przypomina obrót w miejscu.

2. Wyniki dla promienia $R = 500mm$ wykazały że prędkości v_{sy} końców nóg po lewej stronie były mniejsze niż po prawej stronie. Pozwala to stwierdzić, że robot poruszałby się po trajektorii w kształcie łuku.

Nr Nogi	s^R_{xi}	s^R_{yi}	v_{sxi}	v_{syi}	v_{si}
1	-220	200	-160.00	-96.00	186.59
2	-250	-50	40.00	-120.00	126.49
3	-220	-200	160.00	-96.00	186.59
4	220	200	-160.00	256.00	301.89
5	250	50	-40.00	280.00	282.84
6	220	-200	160.00	256.00	301.89

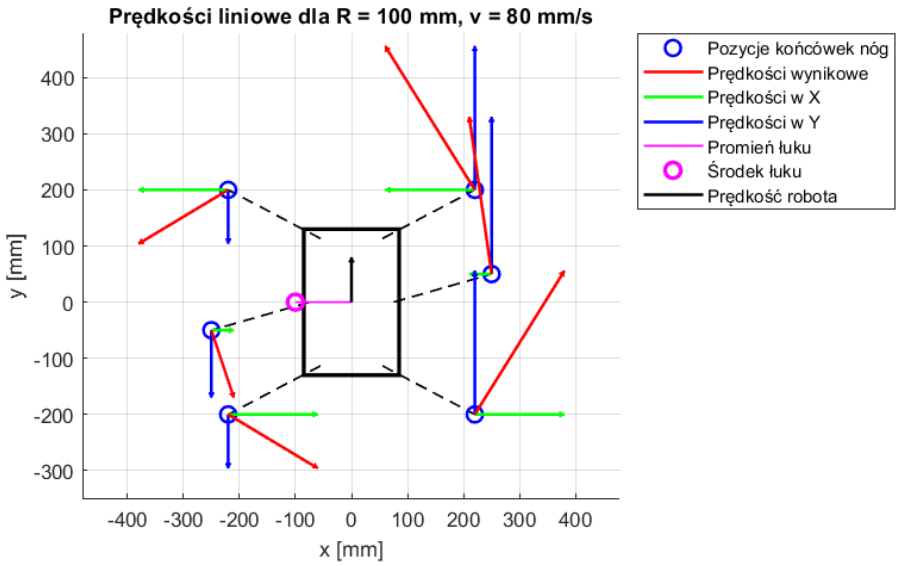
Tabela 5.1 Prędkości liniowe końców nóg dla $R = 100mm$, $v = 80\frac{mm}{s}$

Nr Nogi	s^R_{xi}	s^R_{yi}	v_{sxi}	v_{syi}	v_{si}
1	-220	200	-32.00	44.80	55.05
2	-250	-50	8.00	40.00	40.79
3	-220	-200	32.00	44.80	55.05
4	220	200	-32.00	115.20	119.56
5	250	50	-8.00	120.00	120.27
6	220	-200	32.00	115.20	119.56

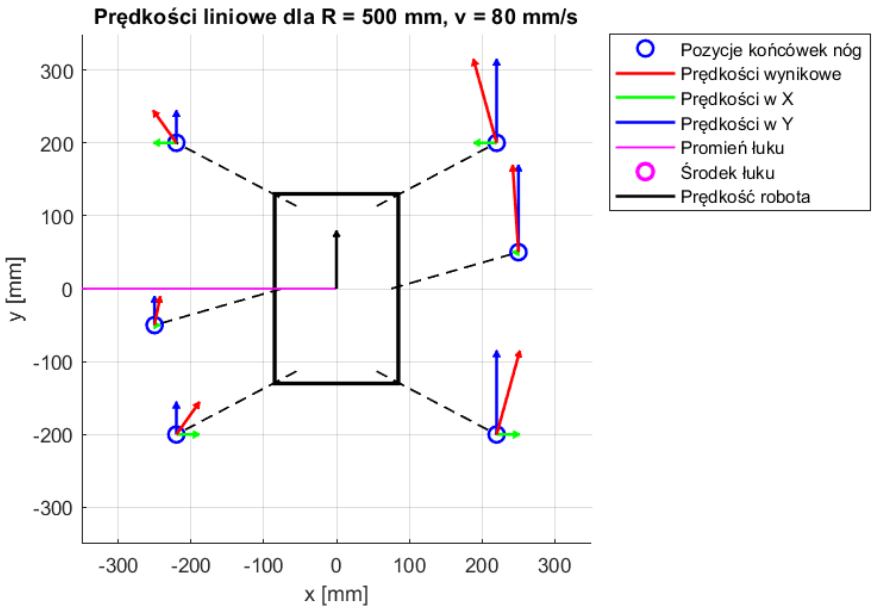
Tabela 5.2 Prędkości liniowe końców nóg dla $R = 500mm$, $v = 80\frac{mm}{s}$

Nr Nogi	s^R_{xi}	s^R_{yi}	v_{sxi}	v_{syi}	v_{si}
1	-220	200	0.00	80.00	80.00
2	-250	-50	0.00	80.00	80.00
3	-220	-200	0.00	80.00	80.00
4	220	200	0.00	80.00	80.00
5	250	50	0.00	80.00	80.00
6	220	-200	0.00	80.00	80.00

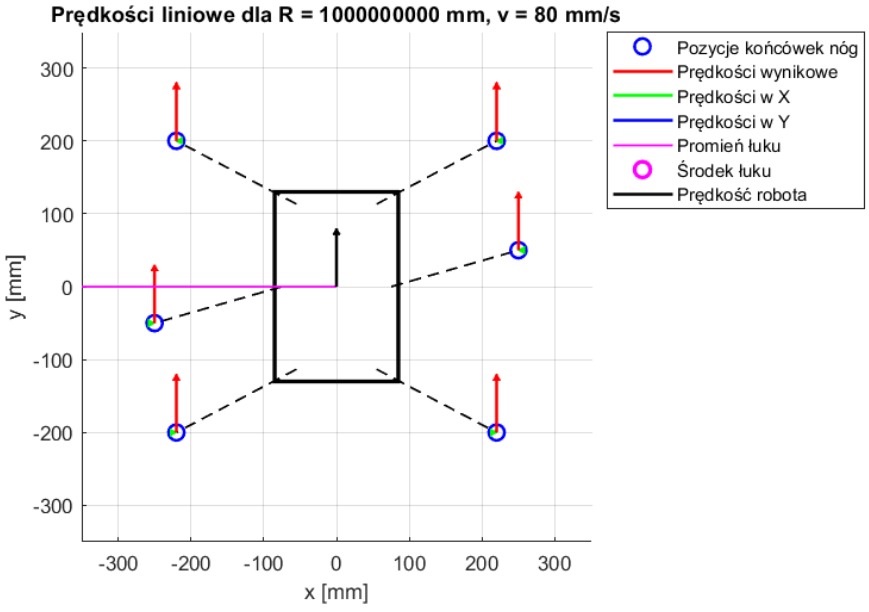
Tabela 5.3 Prędkości liniowe końców nóg dla $R = 10^8mm$, $v = 80\frac{mm}{s}$



Rysunek 5.1 Wizualizacja wektorów prędkości dla $R = 100mm$ oraz $v = 80\frac{mm}{s}$



Rysunek 5.2 Wizualizacja wektorów prędkości dla $R = 500mm$ oraz $v = 80\frac{mm}{s}$



Rysunek 5.3 Wizualizacja wektorów prędkości dla $R = 10^8 mm$ oraz $v = 80\frac{mm}{s}$

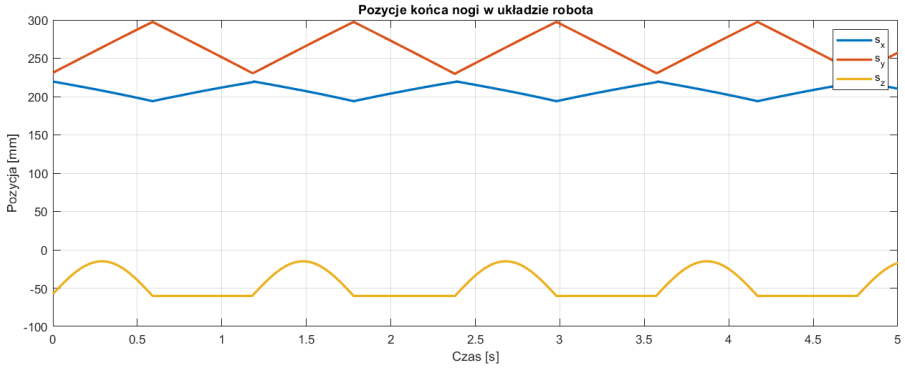
3. Dla bardzo dużego promienia R prędkości v_{sy} po lewej i po prawej stronie robota były takie same a prędkości v_{sx} były równe zero, co sugeruje, że robot poruszałby się do przodu w linii prostej a ewentualne odchylenia od tej trajektorii mogłyby być zauważalne dopiero w bardzo dużej skali.

5.1.2 Test wyliczania prędkości przegubowych

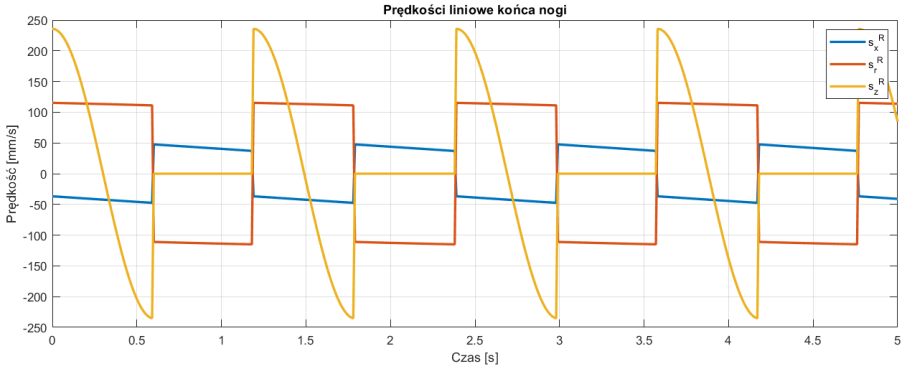
Wykonano szereg symulacji, które miały sprawdzić poprawność wyliczania prędkości przegubowych w nodze dla obliczonych prędkości liniowych jej końca, według metody przedstawionej w podrozdziale 3.1. Wykresy uzyskane w ramach przykładowej symulacji ruchu prawej przedniej nogi, dla prędkości środka korpusu robota $v = 80\frac{mm}{s}$, promienia łuku $R = 500$, okresu chodu $T = 1,2s$ oraz dla czasu symulacji $t = 5s$, przedstawione zostały na rysunku 5.4. Uzyskano także trajektorię tego ruchu widoczną na rysunku 5.5. Na podstawie uzyskanych wyników potwierdzono poprawność metody obliczania prędkości przegubowych.

5.2 Testy na rzeczywistym robocie

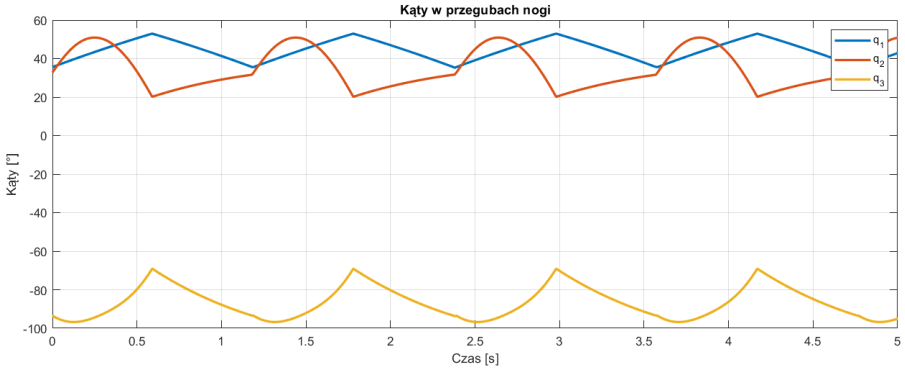
W dalszej części badań przeprowadzono szereg fizycznych testów mających na celu ocenę działania algorytmu sterowania chodem robota. Testy rozpoczęto od weryfikacji poprawności zaimplementowanej kinematyki odwrotnej. Polegały one na zadawaniu określonych pozycji końcówki nogi oraz sprawdzaniu, czy rzeczywista pozycja zgadza się z założeniami. Wyniki potwierdziły, że kinematyka odwrotna została poprawnie wyznaczona. Kolejnym etapem było ustawienie odpowiedniej pozycji startowej, która zapewniała właściwy rozkład środka ciężkości robota na poszczególne nogi. Zostało to zrealizowane w celu zagwarantowania stabilności w trakcie ruchu. Przetestowano również działanie komunikacji pomiędzy głównym komputerem a kontrolerem bezprzewodowym. Podczas testów określono maksymalną prędkość, z jaką robot może się poruszać, zachowując przy tym pełną stabilność. W dalszej części badań analizowano zachowanie robota przy różnych wartościach zadawanych parametrów, takich jak prędkość ruchu środka ciężkości oraz promień łuku trajektorii. Testy wykazały, że robot porusza się płynnie i stabilnie, zarówno podczas ruchu po prostej, jak i po łuku, a także podczas obrotu w miejscu. Uzyskane wyniki potwierdzają skuteczność zaimplementowanego algorytmu sterowania.



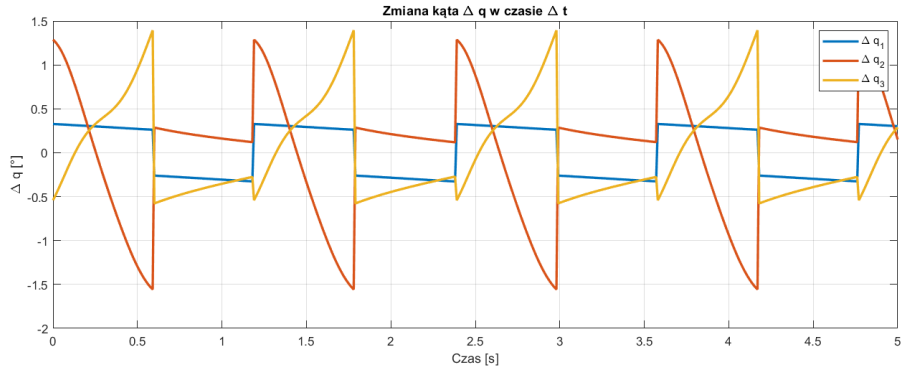
a) Pozycje końca nogi



b) Prędkości liniowe końca nogi

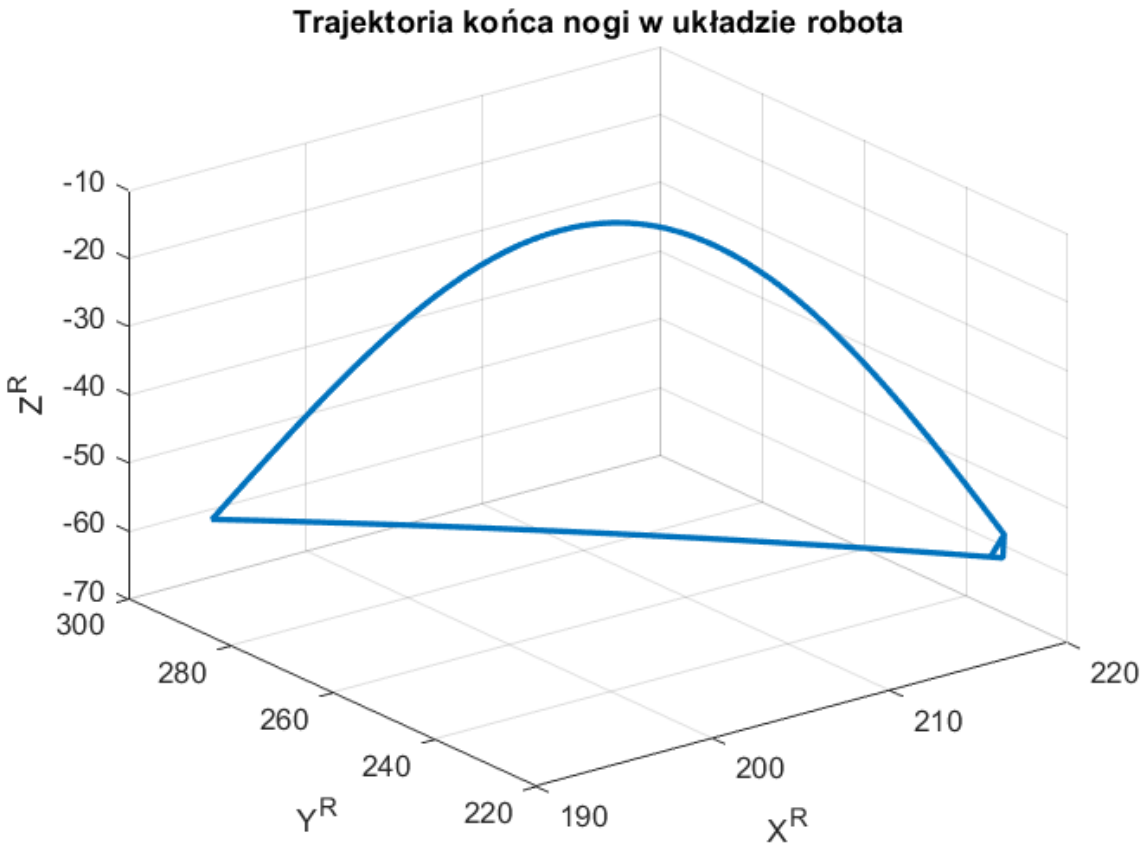


c) Kąty w przegubach nogi



d) Zmiana kąta Δq

Rysunek 5.4 Symulacja ruchu prawej przedniej nogi dla parametrów $v = 80\frac{mm}{s}$, $R = 500mm$, $t = 5s$



Rysunek 5.5 Ścieżka ruchu końca nogi w symulacji dla parametrów $v = 80\frac{mm}{s}$, R = 500mm, t = 5s

Rozdział 6

Podsumowanie

Celem pracy było zaprojektowanie i wykonanie sześcionożnego robota kroczącego, a także zaprojektowanie i implementacja jego systemu sterowania, opartego na modelu, który umożliwi uzyskanie płynnego i stabilnego chodu. Zaprojektowano i wykonano konstrukcję mechaniczną robota oraz dedykowane układy elektroniki, napisano program sterujący chodem na podstawie zadawanych parametrów. Powstały wskutek zrealizowanej pracy robot kroczący potrafi przemieszczać się z zadaną prędkością do przodu i do tyłu, podążając wzdłuż łuku czy też obracać się w miejscu. Jednocześnie jego ruch jest płynny i stabilny. Fakty te pozwalają stwierdzić, że cel pracy został osiągnięty.

Konstrukcja mechaniczna robota spełniła oczekiwania. Jest wytrzymała na obciążenia, zastosowane serwa pozwalają z dużym zapasem mocy utrzymać robota na nogach. Także układ elektroniczny sprostał oczekiwaniom. Zasilacz dostarcza odpowiednią ilość mocy elektrycznej, zapewniając sprawne poruszanie się, główny komputer dysponuje wystarczającą mocą obliczeniową, aby podołać obliczeniom. Zastosowany algorytm sterowania jest wystarczający do tego, aby robot mógł sprawnie poruszać się po płaskiej powierzchni. Wykorzystanie metody jakobianowej do rozwiązania odwrotnego zadania kinematyki celem wyznaczenia pozycji przegubów nóg bardzo ułatwiło implementacje algorytmu sterowania chodem. Zbudowany robot zapewnia ogromne możliwości jego rozwoju w przyszłości.

W przypadku dalszego rozwoju projektu należałoby zwiększyć responsywność algorytmu sterowania, zadawanie parametrów powinno odbywać się w trakcie trwania fazy ruchu a robot powinien odpowiednio na nie reagować zmieniając trajektorię końca nóg. W celu usprawnienia poruszania się po trudnym terenie należałoby dodać do algorytmu pętle sprzężenia zwrotnego z czujników dotyku oraz z modułu IMU, a także wprowadzić możliwość obrotu korpusu wokół dwóch pozostałych osi X_R,Y_R. Warto wprowadzić także możliwość zadawania kształtu ścieżki wzdłuż której robot ma sam podążać. Istnieje również możliwość implementacji prostej autonomii po dodaniu do robota odpowiednich czujników, takich jak czujniki odległości czy kamera. W robocie można także rozważyć umieszczenie chwytaka.

Literatura

[Aut] Autodesk. Autodesk Inventor - strona producenta. https://www.autodesk.com/pl/products/inventor/overview?term=1-YEAR&tab=subscription.

[Bam] BambuLab. Bambu Studio. https://bambulab.com/en/download/studio.

[Dev] KiCad Developers. KiCad. https://kicad.org.

[Dyna] Boston Dynamics. Atlas® and beyond: the world's most dynamic robots. https://bostondynamics.com/atlas/.

[Dynb] Boston Dynamics. Spot® – the agile mobile robot. https://bostondynamics.com/products/spot/.

[For] Forbot. Roboty kroczące – teoria i podstawy projektowania. https://forbot.pl/blog/roboty-kroczace-teoria-podstawy-projektowania-id976.

[Foua] Raspberry Pi Foundation. Gpio pins. https://www.raspberrypi.org/documentation/usage/gpio/.

[Foub] Raspberry Pi Foundation. Raspberry Pi. https://www.raspberrypi.org/.

[Fouc] The Raspberry Pi Foundation. Raspberry Pi OS. https://www.raspberrypi.com/documentation/computers/os.html.

[Gie12] Rafał Gierczak, *Labolatoryjny robot kroczący*. Praca magisterska, Politechnika Wrocławska, 2012.

[Ins] American National Standards Institute. Programming language c. https://pl.wikipedia.org/wiki/J%C4%99zyk_ANSI_C.

[Jan07] Marek Jankiewicz, Przeciwminowe roboty: pojazdy kroczące. *Biuletyn Wojskowej Akademii Technicznej*, strony 273–286, 2007.

[lib] libsdl.org. Simple DirectMedia Layer library. https://www.libsdl.org/license.php.

[Mat] MathWorks. MATLAB. https://www.mathworks.com/products/matlab.html.

[Mica] Microsoft. Remote ssh extension for visual studio code. https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh.

[Micb] Microsoft. Visual studio code. https://code.visualstudio.com/.

[mst] mstroh76. Wiringpi library. https://github.com/WiringPi/WiringPi.

[Sem] NXP Semiconductors. Dokumentacja układu PCA9685. https://www.digikey.be/htmldatasheets/production/1640697/0/0/1/pca9685-datasheet.html.

[Sys] Tempero Systems. Dokumentacja układu GY-87. https://temperosystems.com.au/wp-content/uploads/2024/07/GY-87-10DOF-MPU6050-HMC5883L-BMP180-Infomation-Sheet.pdf.

[Teca] Dongguan DSservo Technology. DSservo. https://www.dsservo.com/en/index.asp.

[Tecb] Handson Technology. Dokumentacja modułu wskaźnika naładowania baterii. https://www.handsontec.com/dataspecs/Instruments/Battery%20Level%20indicator-1~8S.pdf.

[Wika] Wikipedia. Algorytm Denavita-Hartenberga. https://pl.wikipedia.org/wiki/Notacja_Denavita-Hartenberga.

[Wikb] Wikipedia. Autodesk Inventor. https://pl.wikipedia.org/wiki/Autodesk_Inventor.

[Wikc] Wikipedia. Bionika, Biomimetyka. https://pl.wikipedia.org/wiki/Bionika.

[Wikd] Wikipedia. Całkowanie numeryczne. https://pl.wikipedia.org/wiki/Ca%C5%82kowanie_numeryczne.

[Wike] Wikipedia. FDM. https://pl.wikipedia.org/wiki/Osadzanie_topionego_materia%C5%82u.

[Wikf] Wikipedia. Galop. https://pl.wikipedia.org/wiki/Galop_(je%C5%BAdziectwo).

[Wikg] Wikipedia. GCode. https://pl.wikipedia.org/wiki/G-code.

[Wikh] Wikipedia. Jacques de Vaucanson. https://pl.wikipedia.org/wiki/Jacques_de_Vaucanson.

[Wiki] Wikipedia. Komunikacja I2C. https://pl.wikipedia.org/wiki/I%C2%B2C.

[Wikj] Wikipedia. Malarstwo jaskiniowe. https://pl.wikipedia.org/wiki/Malarstwo_jaskiniowe.

[Wikk] Wikipedia. Nietoperze. https://pl.wikipedia.org/wiki/Nietoperze.

[Wikl] Wikipedia. PLA. https://pl.wikipedia.org/wiki/Polilaktyd.

[Wikm] Wikipedia. Próchniaczek czarniawy. https://pl.wikipedia.org/wiki/Pr%C3%B3chniaczek_czarniawy.

[Wikn] Wikipedia. Secure shell. https://pl.wikipedia.org/wiki/Secure_Shell.

[Wiko] Wikipedia. Serial peripheral interface. https://pl.wikipedia.org/wiki/Serial_Peripheral_Interface.

[Wikp] Wikipedia. TPU. https://en.wikipedia.org/wiki/Thermoplastic_polyurethane.

[ZH02] Teresa Zielinska, John Heng, Multifunctional walking quadruped. *Cambridge University Press*, strony 585–593, 2002.

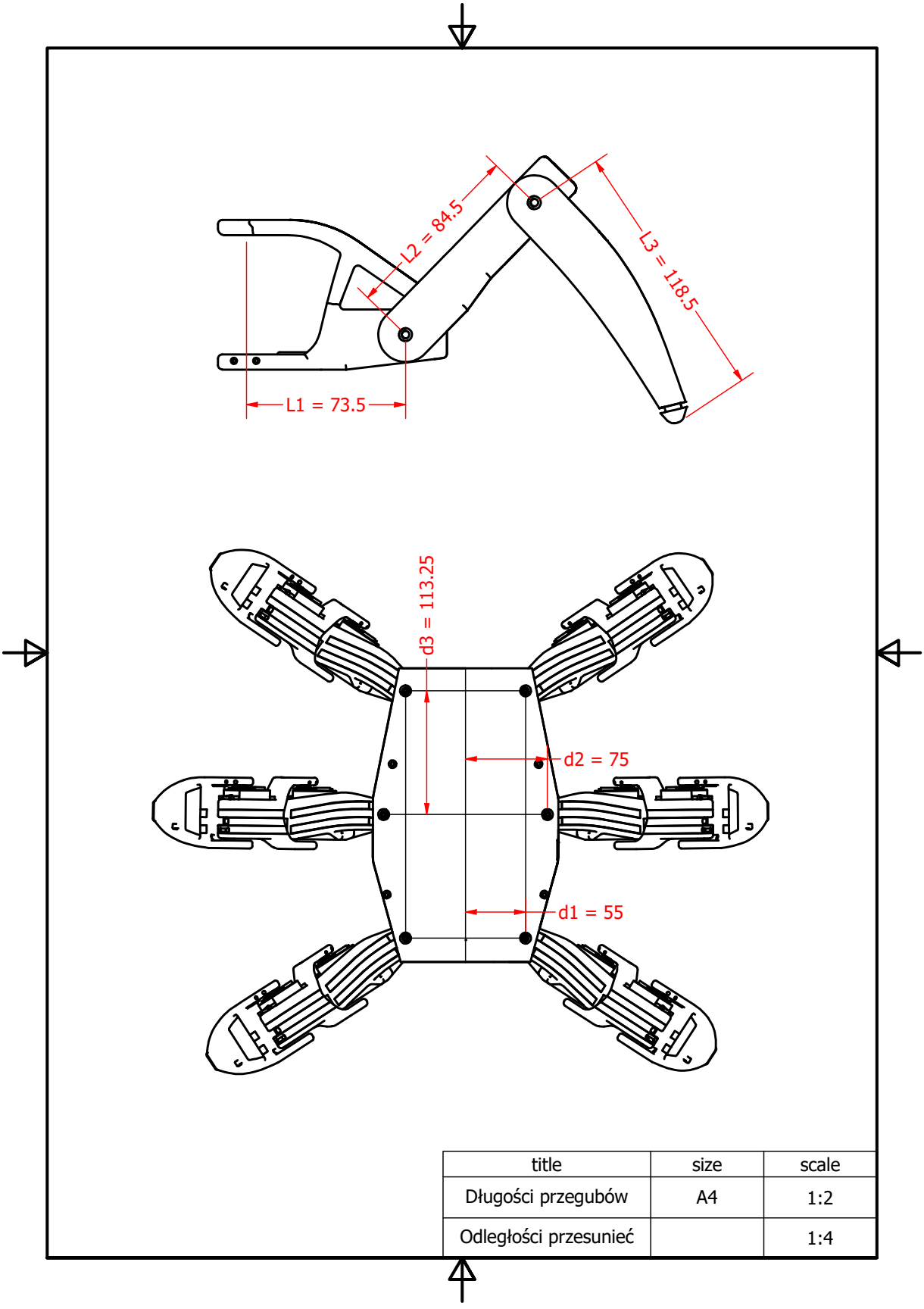
[Zie14] Teresa Zielińska, *Maszyny kroczące*. Wydawnictwo Naukowe PWN, 2014.

Spis rysunków

Dodatek A

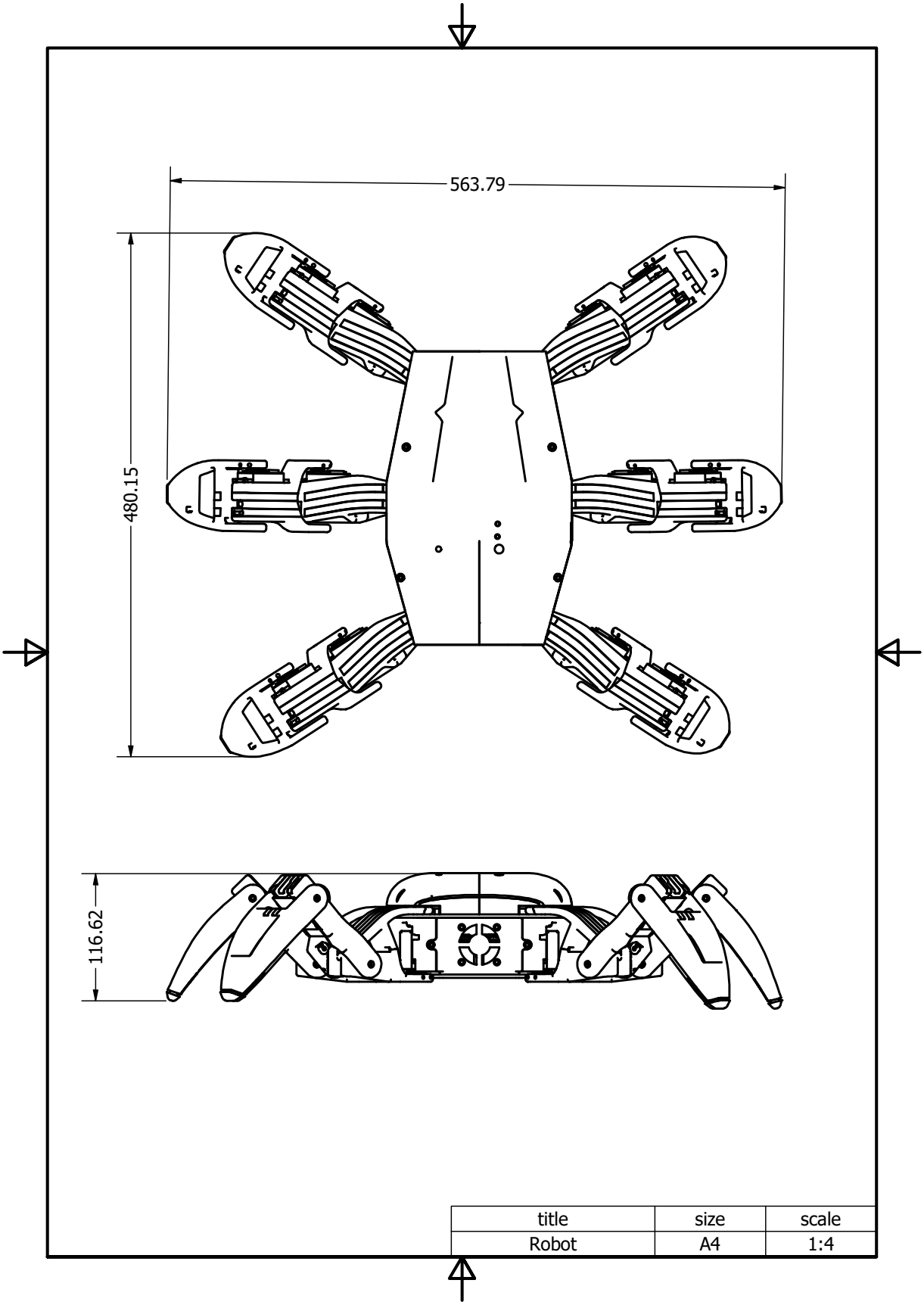
Rysunki techniczne

Dodatek zawiera rysunki techniczne elementów wchodzących w skład konstrukcji mechanicznej robota.

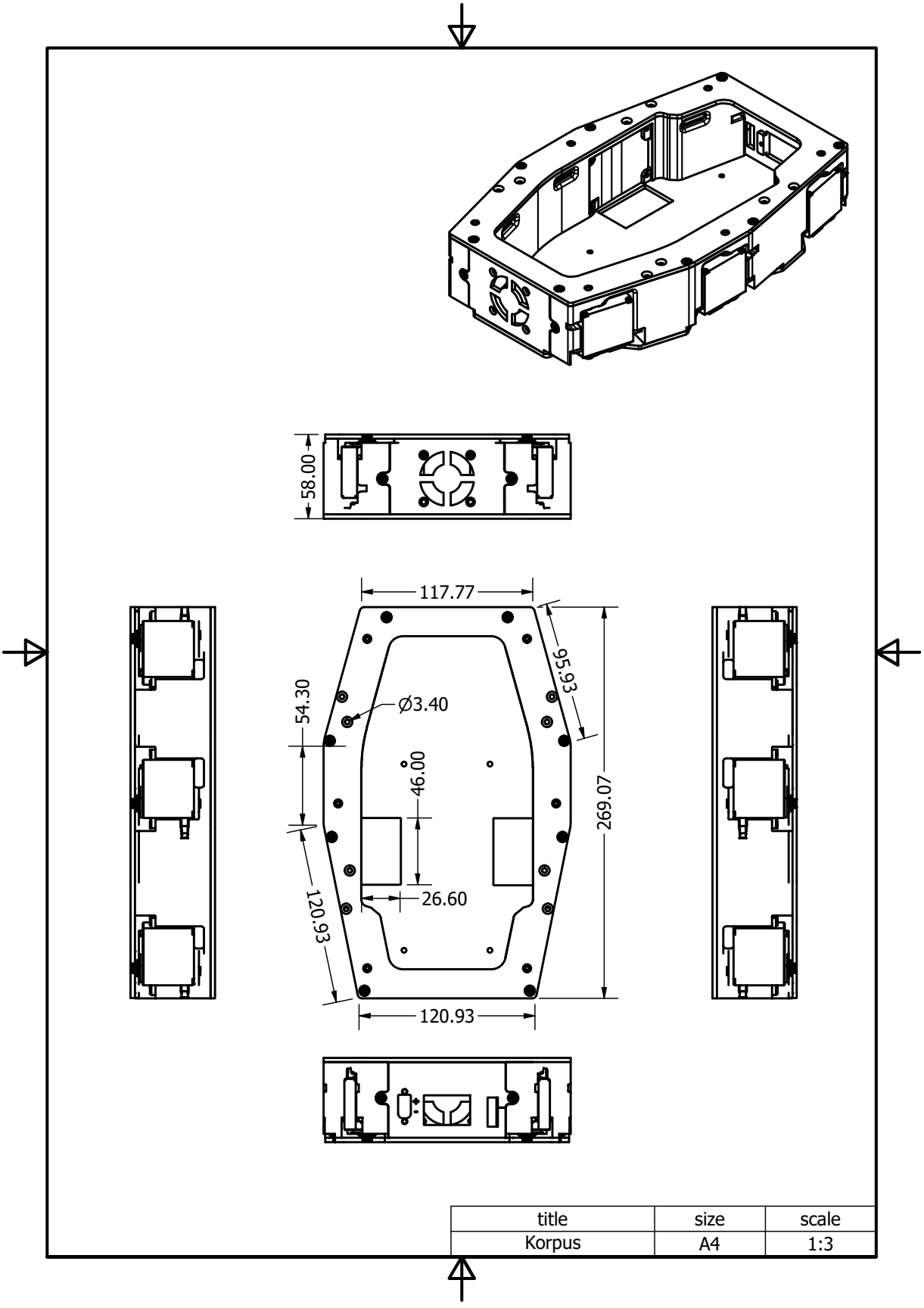


title	size	scale
Długości przegubów	A4	1:2
Odległości przesunieć		1:4

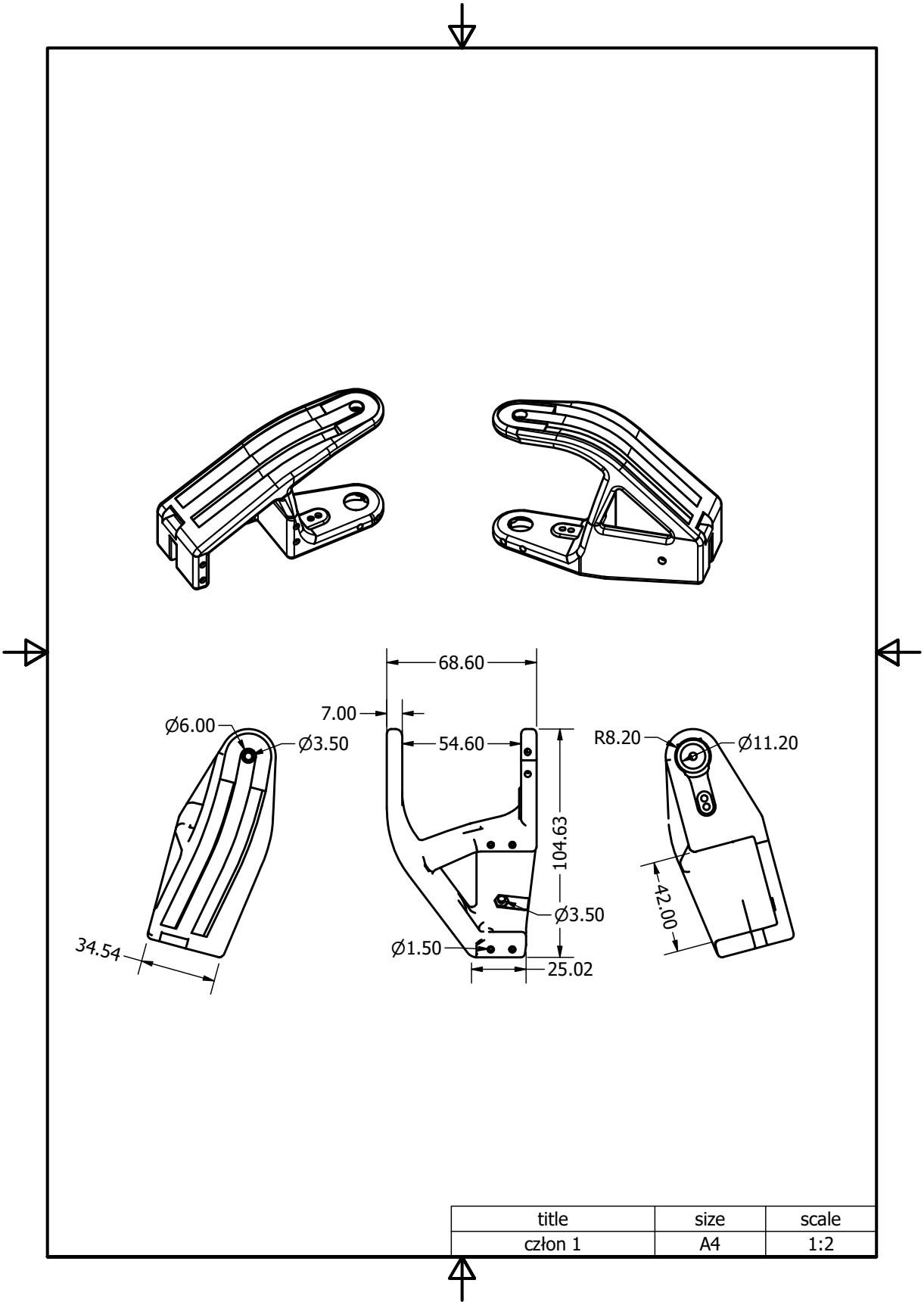
Rysunek A.1 Długości przegubów oraz odległości przesunięć dla nóg



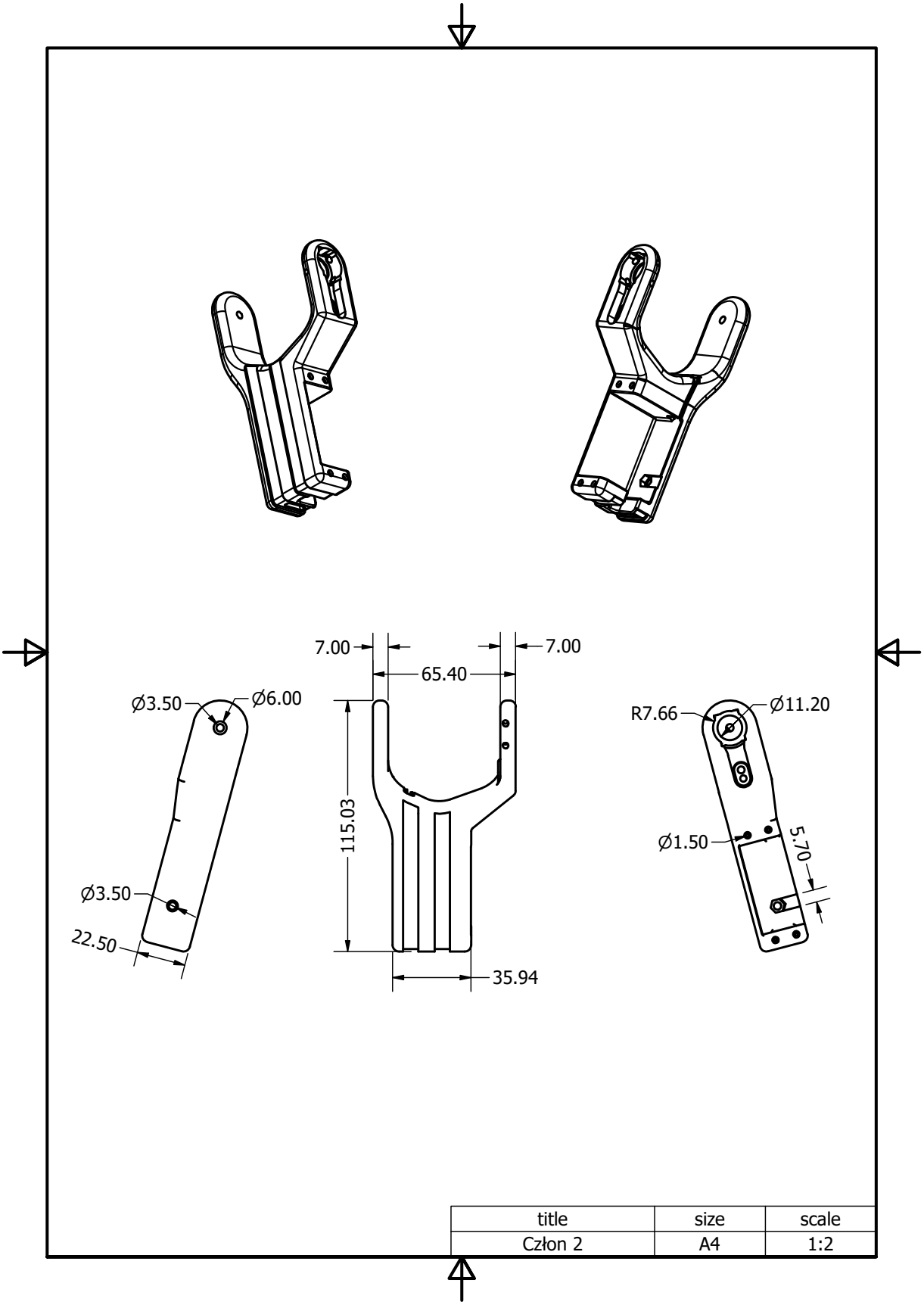
Rysunek A.2 Robot w standardowej pozycji chodu



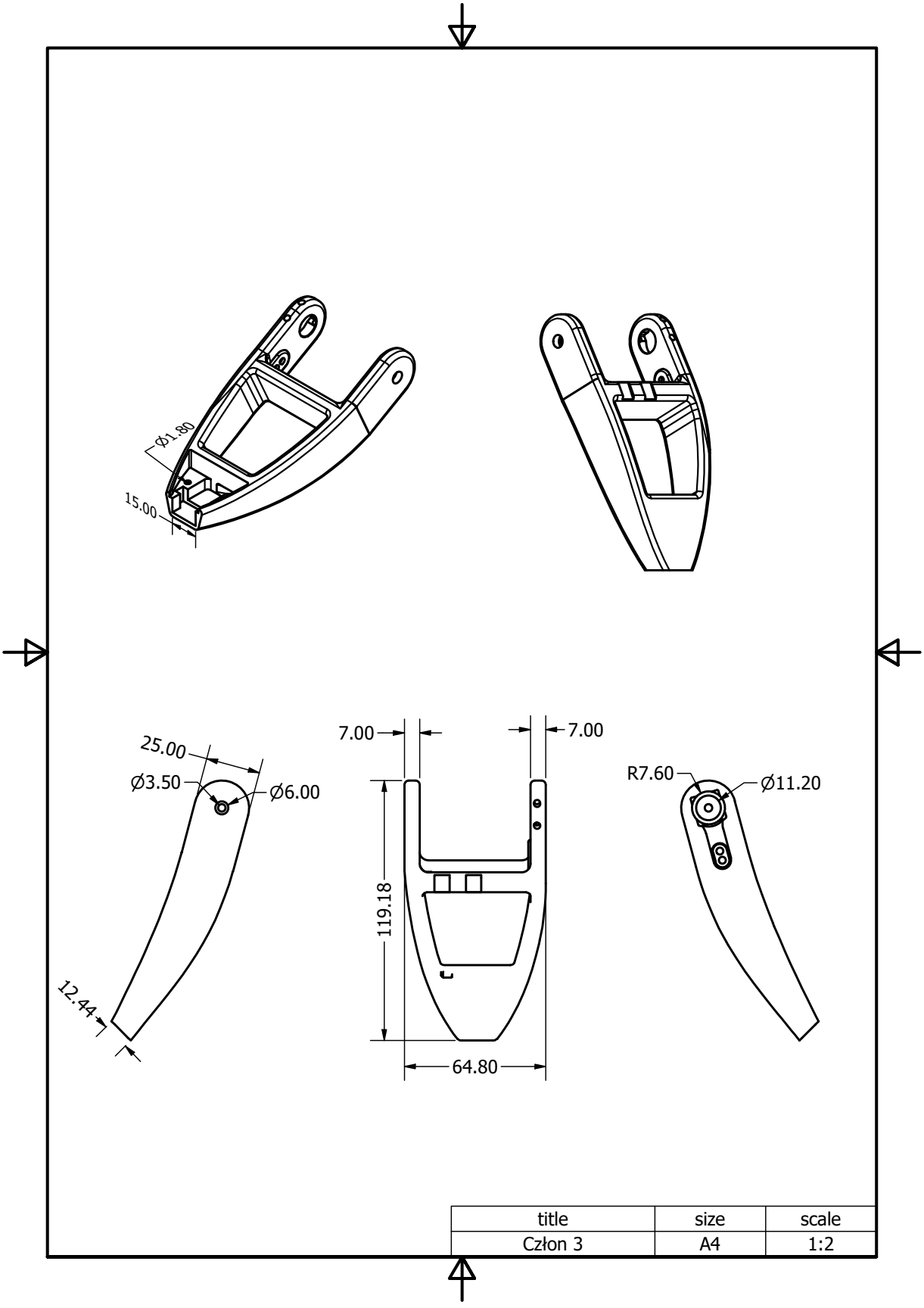
Rysunek A.3 Korpus robota



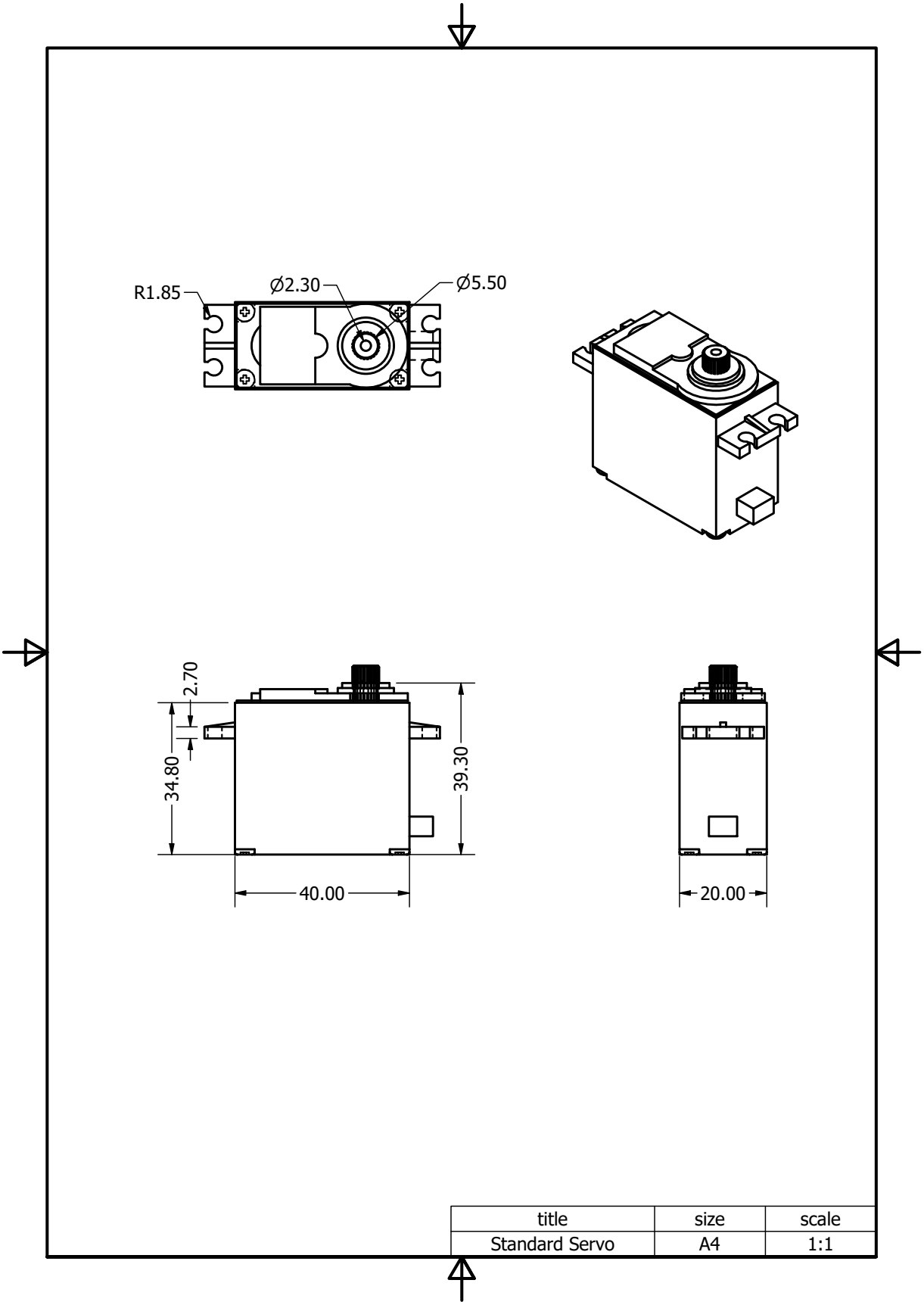
Rysunek A.4 Noga - człon 1



Rysunek A.5 Noga - człon 2



Rysunek A.6 Noga - człon 3

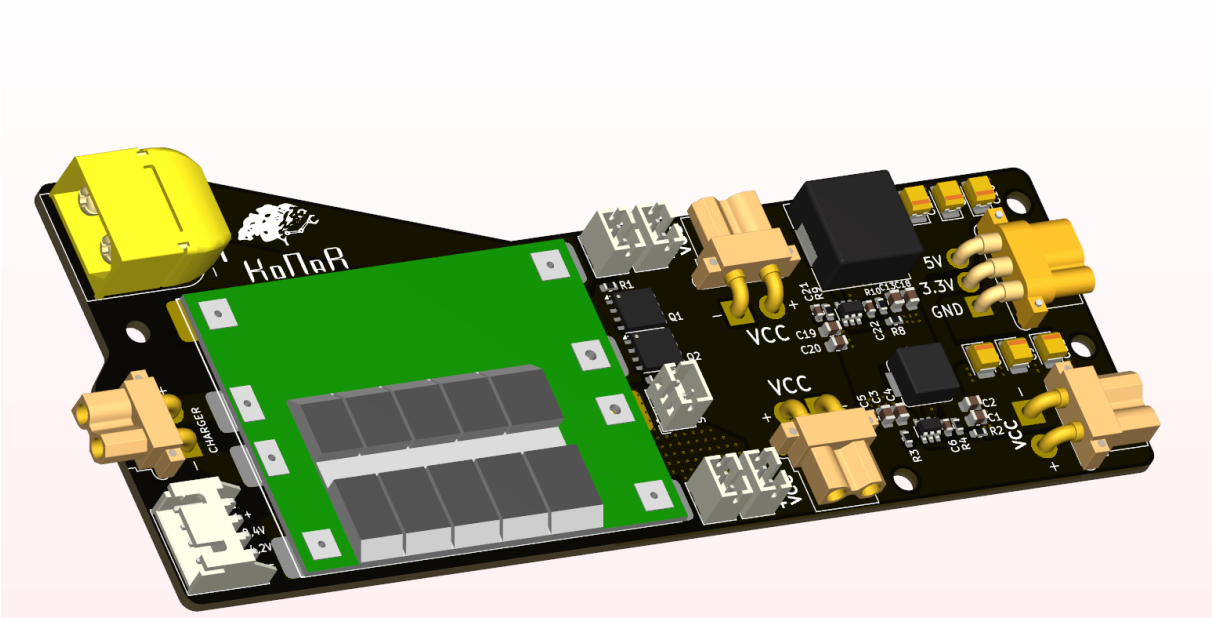


Rysunek A.7 Serwomechanizm w rozmiarze „Standard"

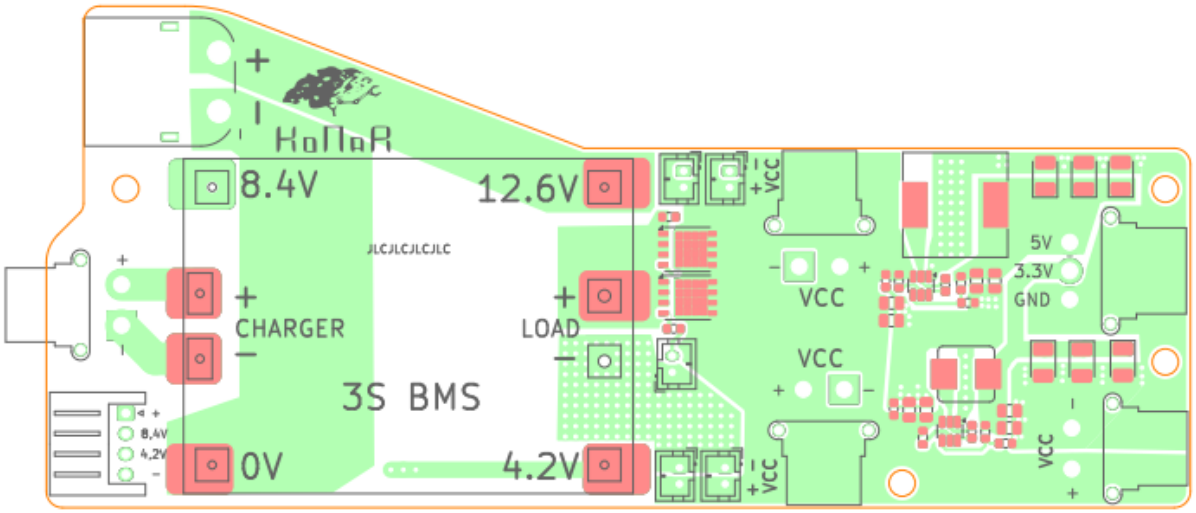
Dodatek B

Płytki PCB

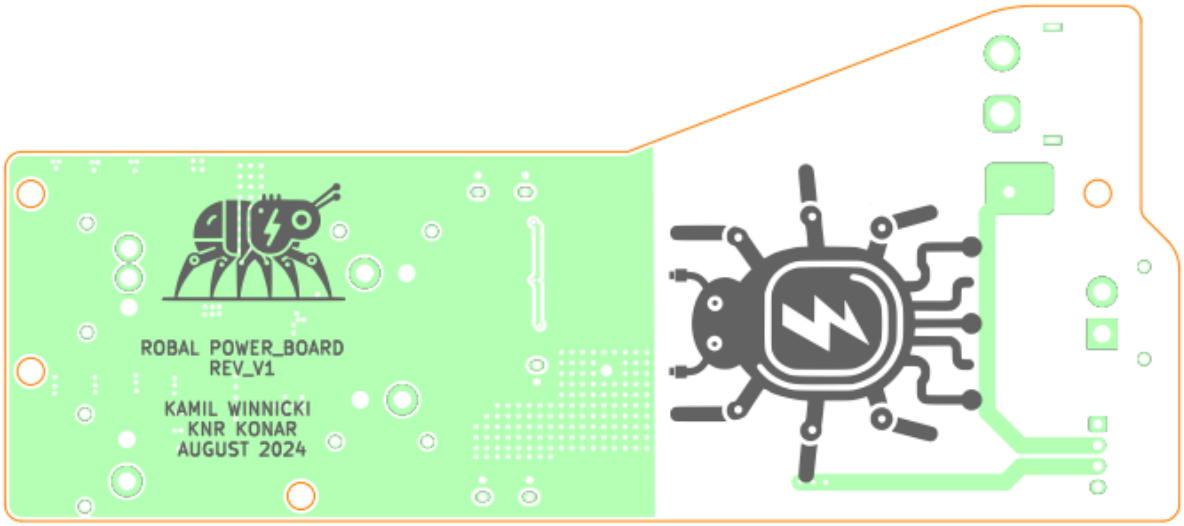
Dodatek zawiera schematy i rendery 3D zaprojektowanych w ramach projektu płytek PCB.



Rysunek B.1 Model 3D płytki zasilającej

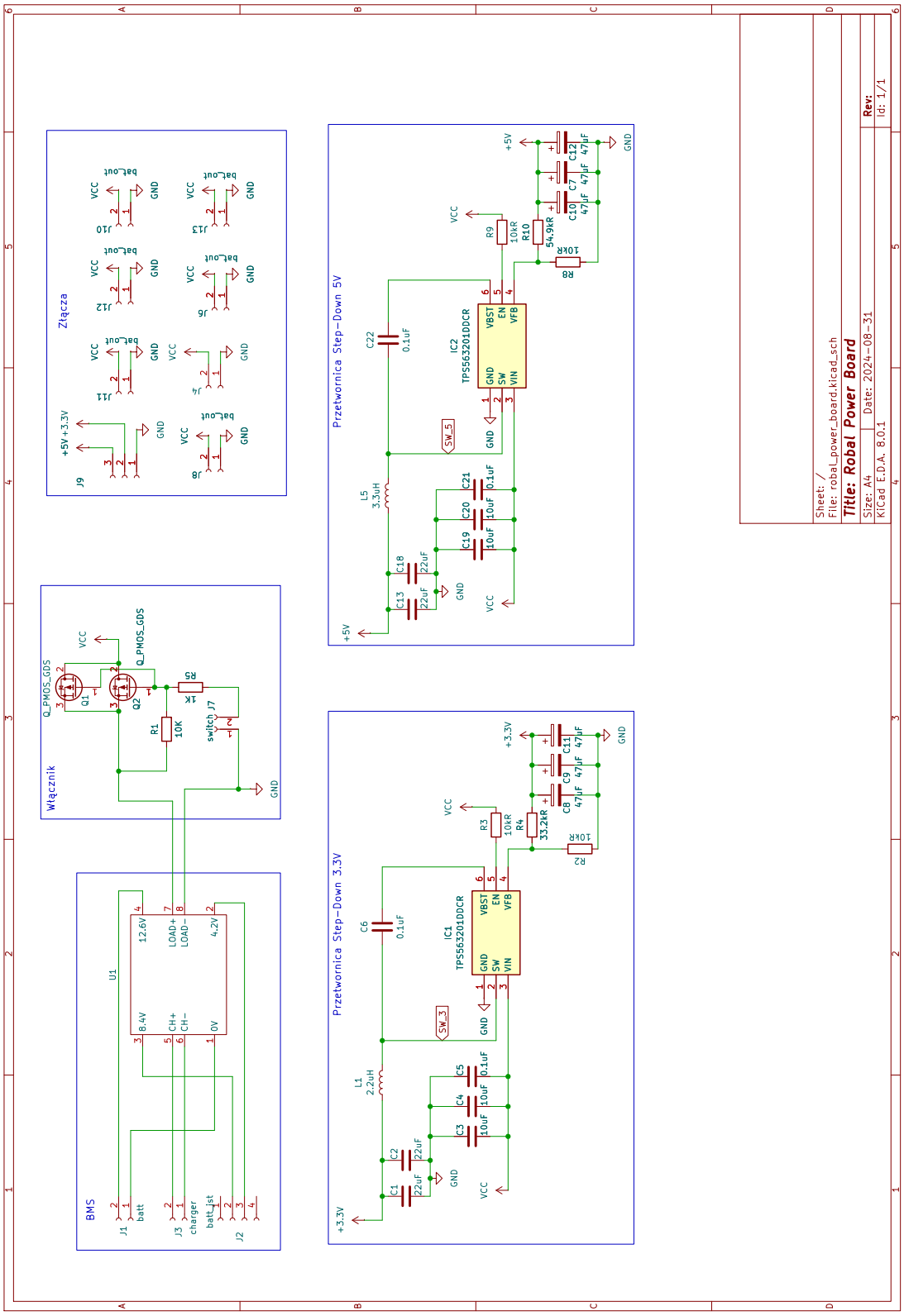


(a) Widok z góry

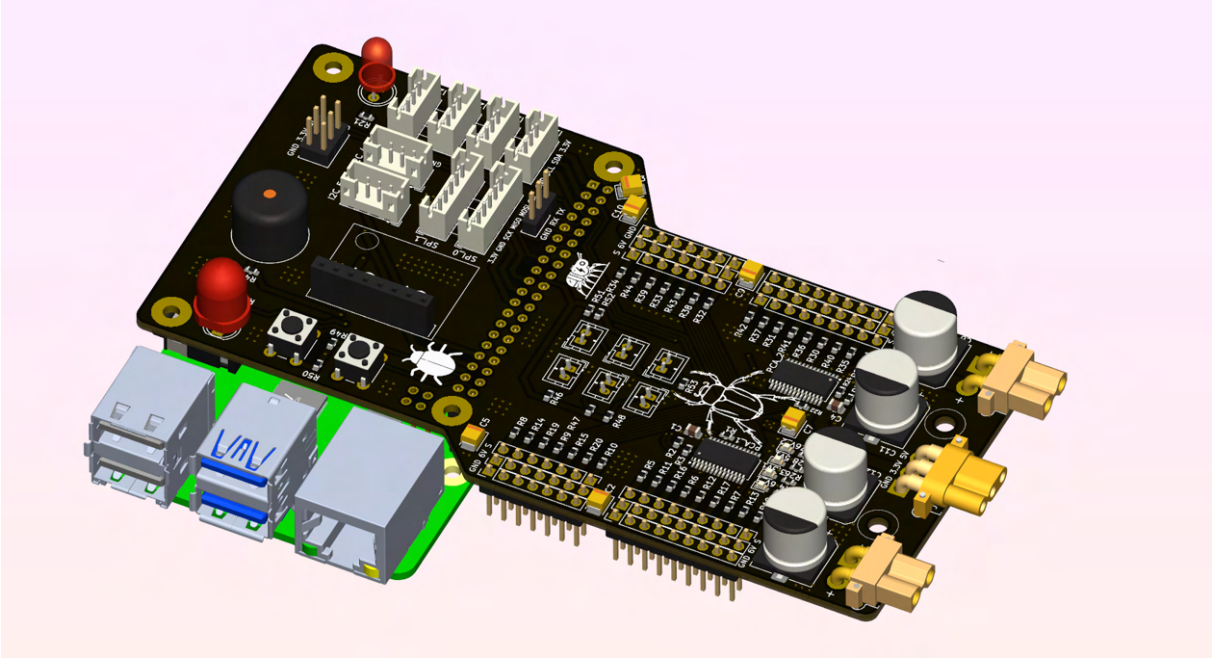


(b) Widok z dołu

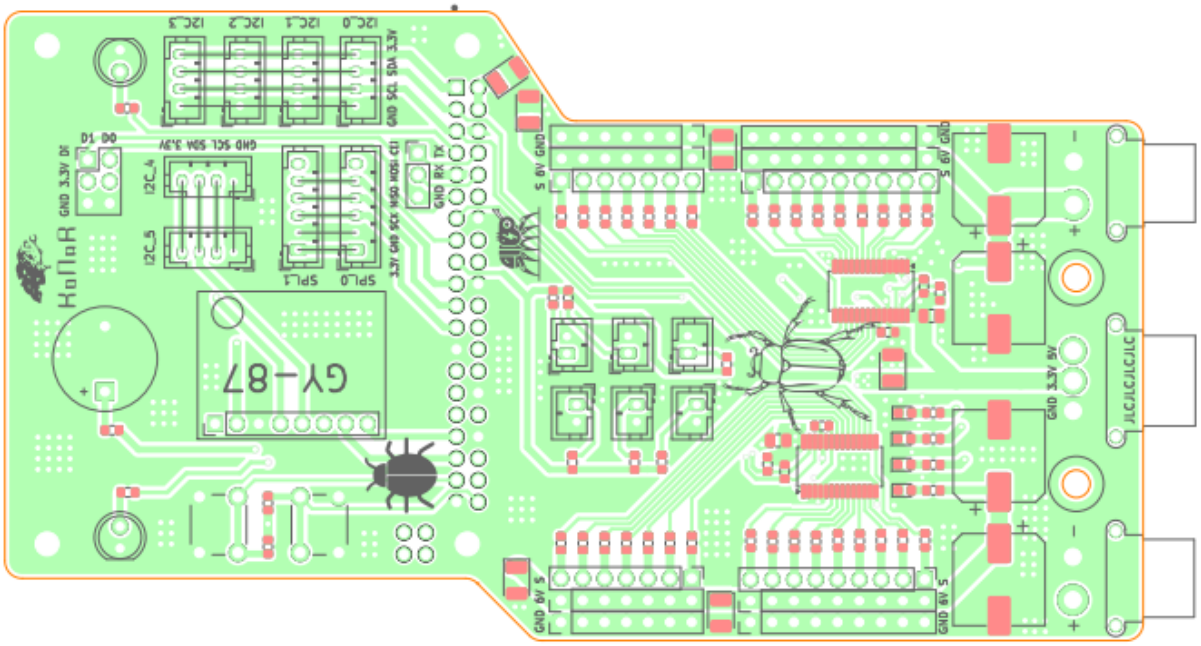
Rysunek B.2 Płytka zasilająca



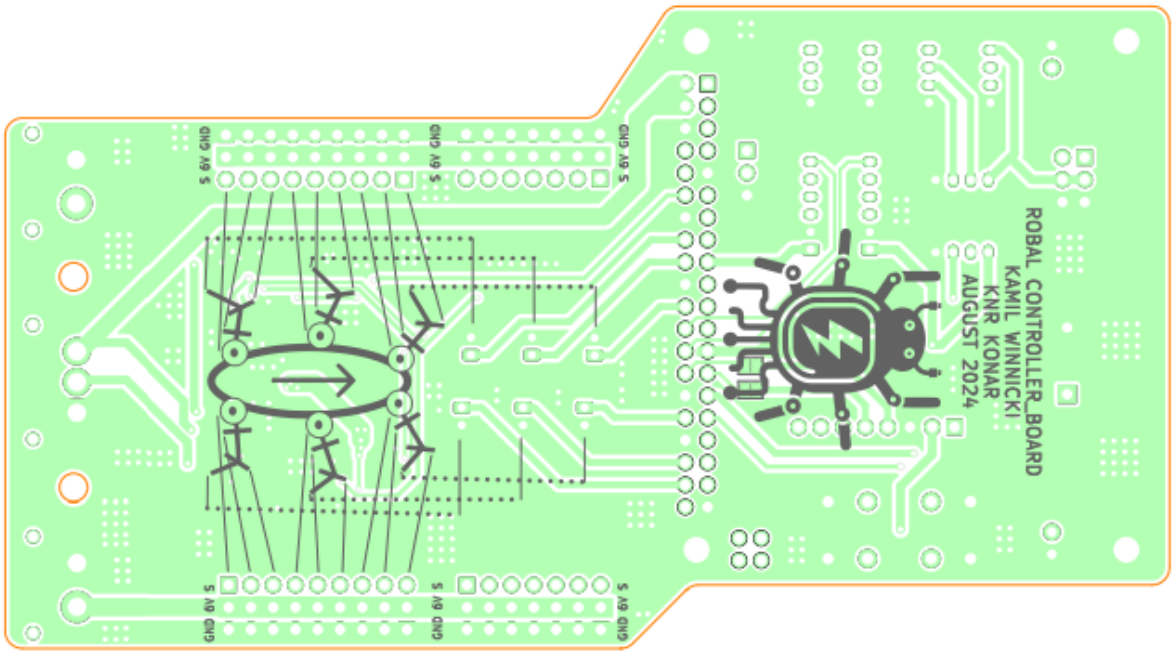
Rysunek B.3 Schemat płytki zasilającej



Rysunek B.4 Model 3D płytki sterującej wraz z RaspberryPi

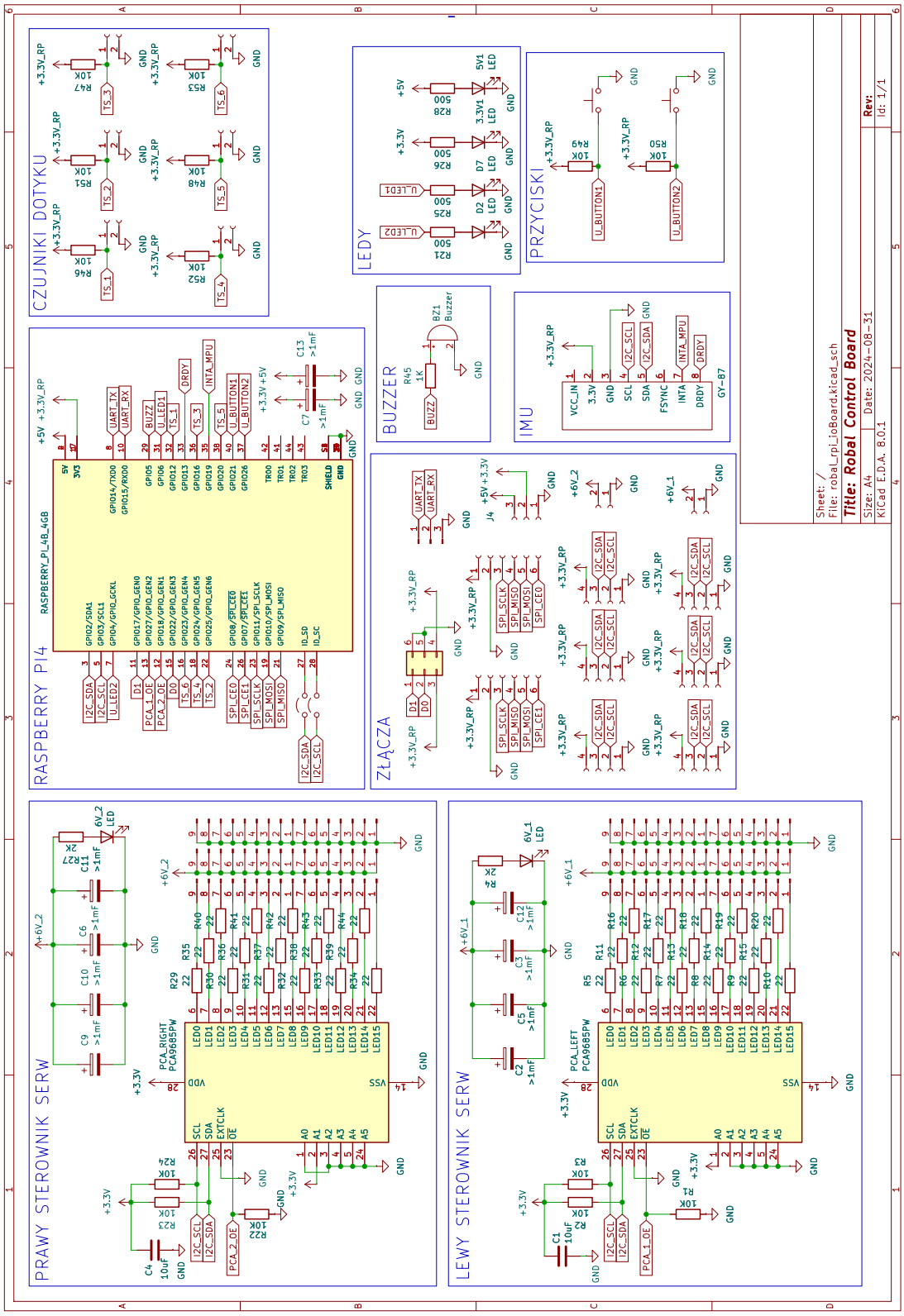


(a) Widok z góry



(b) Widok z dołu

Rysunek B.5 Płytka sterująca



Rysunek B.6 Schemat płytki sterującej

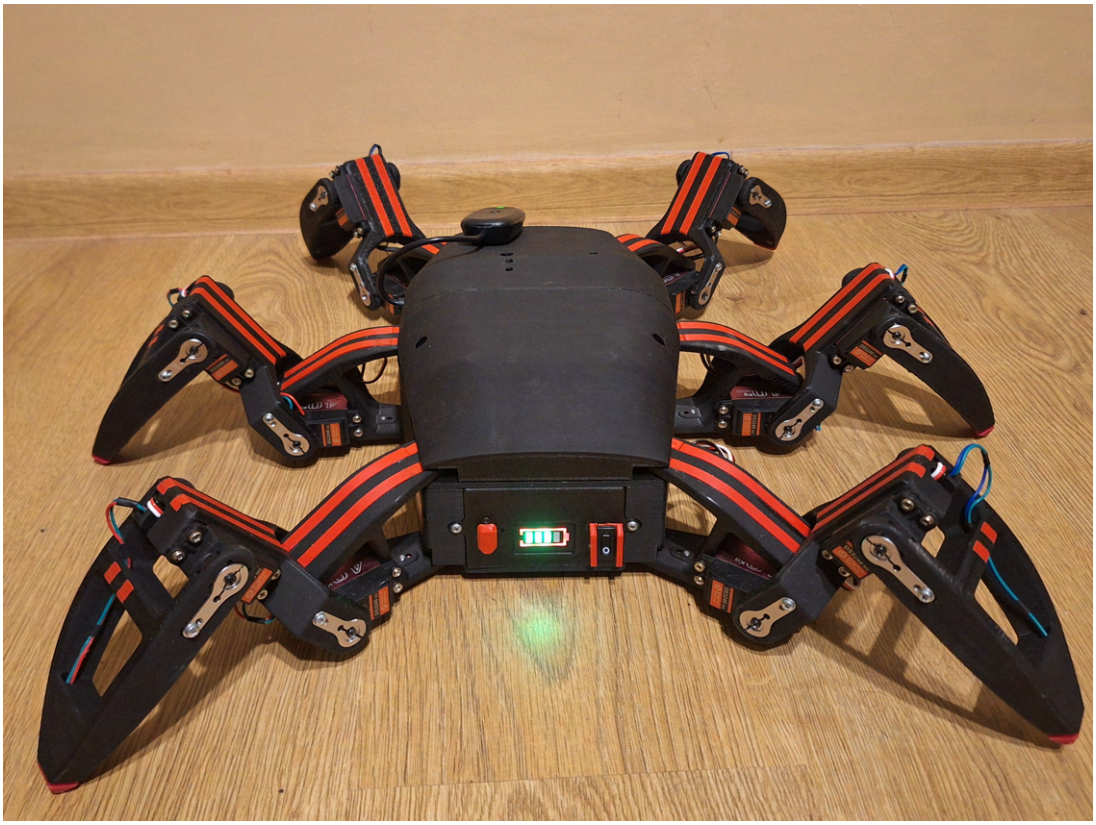
Dodatek C

Zdjęcia robota

Rysunek C.1 Robot w pozycji gotowej do ruchu

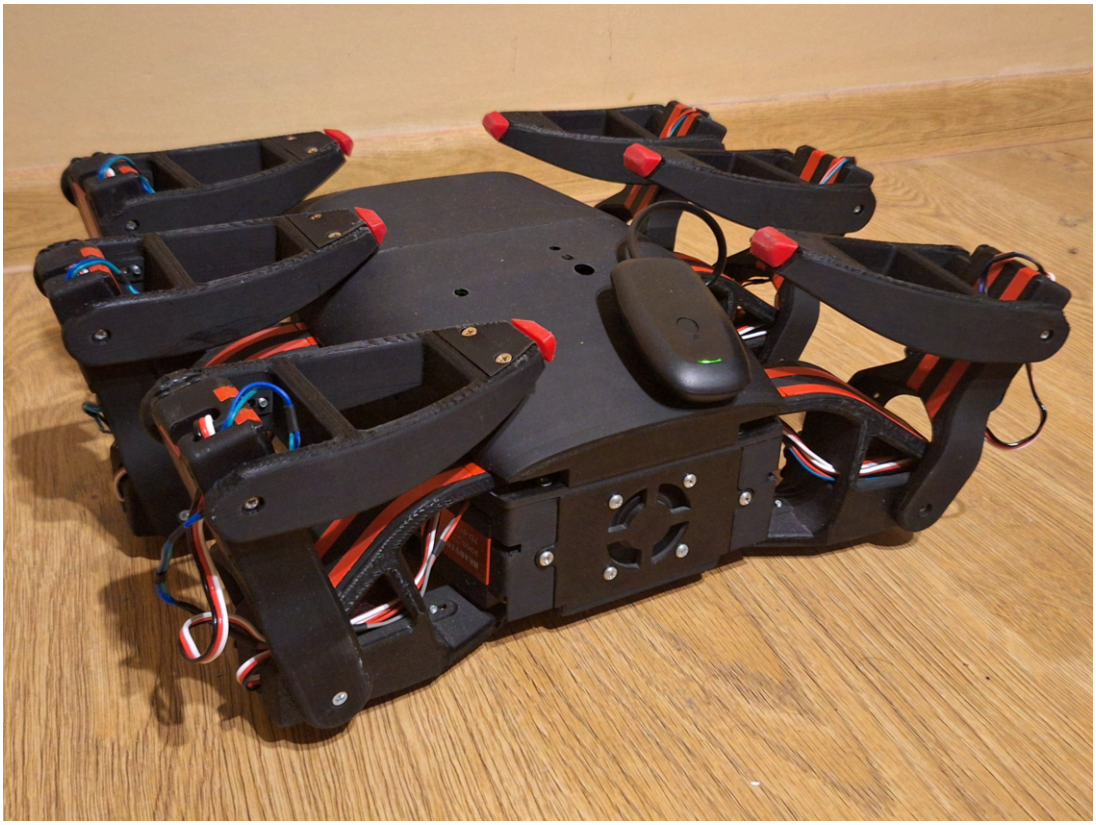
Rysunek C.2 Widok od przodu

Rysunek C.3 Widok z boku

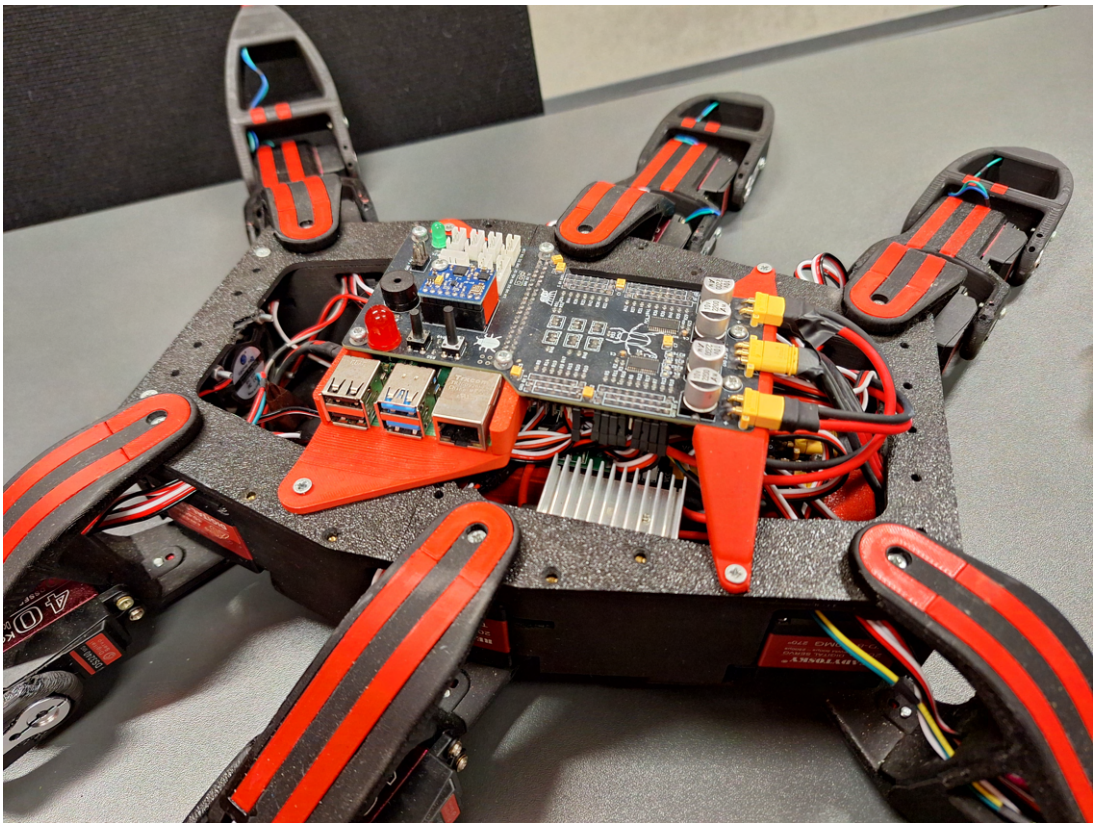


Rysunek C.4 Widok z tyłu

Rysunek C.5 Widok z góry



Rysunek C.6 Robot w pozycji transportowej



Rysunek C.7 Widok bez górnej pokrywy