POLITECHNIKA POZNAŃSKA

WYDZIAŁ INZYNIERII MECHANICZNEJ

PRACA DYPLOMOWA INŻYNIERSKA

KONSTRUKCJA ZDALNIE STEROWANEGO ROBOTA KROCZĄCEGO

KRZYSZTOF PUZIO

Promotor pracy:

prof. dr hab. inż. Michał Wieczorowski

Poznań 2022 rok

Spis treści

1. WSTĘP ... 3

2. PRZEGLĄD STANU TECHNIKI W ZAKRESIE ROBOTÓW KROCZĄCYCH ... 5

A. KLASYFIKACJA ROBOTÓW KROCZĄCYCH ... 5

B. RODZAJE I DIAGRAMY CHODU ... 6

C. NAPĘDY ROBOTÓW KROCZĄCYCH I PRZYKŁADY KONSTRUKCJI ... 8

3. ALGORYTM CHODU ... 12

A. TYPY CHODÓW STABILNYCH STATYCZNIE ROBOTÓW SZEŚCIONOŻNYCH ... 12

B. SEKWENCJA PORUSZANIA NOGAMI ... 13

C. DOPASOWANIE KĄTA MIĘDZY CZŁONAMI NOGI W CELU OSIĄGNIĘCIA STAŁEJ WYSOKOŚCI ROBOTA ... 14

D. ANALIZA DŁUGOŚCI KROKU ... 17

4. KONSTRUKCJA MECHANICZNA ... 19

5. PROJEKT ELEKTRONIKI ... 21

A. SCHEMAT ELEKTRONIKI ... 21

B. SPIS ELEMENTÓW ... 21

C. PŁYTKA PCB(ANG. PRINTED CIRCUIT BOARD) ... 24

D. OBLICZENIA WARTOŚCI ELEMENTÓW ... 25

6. PROGRAM ... 32

A. DZIAŁANIE PWM I OBLICZENIE WARTOŚCI POTRZEBNYCH DO INICJALIZACJI SERWOMECHANIZMU ... 32

B. KLASA SERVO I SLAVESERVO ... 33

C. KLASA LEG ... 36

D. KLASA BODY I MASZYNA STANÓW ODPOWIADAJĄCA ZA PORUSZANIE SIĘ ROBOTA ... 38

E. KOMUNIKACJA ZA POMOCĄ UART-U ... 39

F. SPRAWDZENIE NAPIĘCIA NA BATERII ... 40

G. FUNKCJA MAIN ... 41

H. KALIBRACJA NÓG ... 41

7. KOMUNIKACJA ... 43

A. APLIKACJA NA URZĄDZENIE MOBILNE ... 43

B. POŁĄCZENIE Z ROBOTEM ... 44

8. PODSUMOWANIE ... 45

LITERATURA ... 50

Streszczenie

Praca miała na celu zaprojektowanie zdalnie sterowanego robota kroczącego. Przeprowadzono analizę skonstruowanych już maszyn kroczących i oceniono jak poszczególne parametry wpływają na działanie gotowej konstrukcji. Następnie omówiony jest sposób poruszania się projektowanego robota. W kolejnych rozdziałach opisana jest mechaniczna konstrukcja, projekt elektroniki i napisany program wraz ze sposobem łączności. Na podstawie obliczeń dobrano też odpowiednie elementy elektroniczne. Tworząc pracę wykonano również prototyp robota, dzięki któremu można były wyciągnąć praktyczne wnioski z pracy maszyny.

Summary

The main aim of the study is to construct a remotely controlled walking robot. In the thesis a few robots were described including the impact of certain properties of the analysed machines on their final performance. Then the walking control algorithm is introduced followed by the mechanical construction. In the next chapter the electrical schematics are shown with the choice of specific elements based on the calculations. Then the program, which controls the robot and the remote control system is described. The prototype of the described robot was created, so in the last chapter are included some practical conclusions.

1. Wstęp

Dzięki rozwojowi techniki dostęp do różnych jej zdobyczy takich jak np. do smartphone ‘ów i komputerów osobistych stał się niesamowicie ułatwiony. Coraz częściej można zauważyć też wzrost zainteresowania robotami nie tylko w przemyśle, ale też w gospodarstwach domowych, w których mogą one odkurzać, kosić trawę, czy zmywać podłogi.[1]

W XX. wieku o robotach pisali głównie autorzy science fiction, lecz dzisiejszy stan techniki pozwala na zbudowanie wielu urządzeń, o których ludzie na początku poprzedniego wieku mogli tylko pomarzyć. Powstają urządzenia kontrolowane za pomocą fal elektromagnetycznych generowanych przez mózg, roboty latające, autonomiczne samochody, czy roboty kroczące odzwierciedlające ruchy zwierząt.[1]

Z odkryć archeologicznych wiemy, że ludzie od dawna próbowali tworzyć mechanizmy które naśladowałyby istoty żywe. Jednak dzięki rozwojowi technologii jest możliwe nie tylko robotów poruszających się samodzielnie za pomocą silników elektrycznych, ale coraz częściej są one wyposażane w inteligentne systemy, dzięki którym mogą swoje zadania wykonywać autonomicznie.[1]

Roboty kroczące są coraz częściej wykorzystywane w inspekcjach trudno dostępnych miejsc niedostępnych lub niebezpiecznych dla człowieka. Dodatkowo łatwiej im poruszać się w niesprzyjających warunkach otoczenia niż klasycznym robotom wykorzystującym do przemieszczania się koła, ponieważ nierówne podłoże i przeszkody występujące na drodze nie zatrzymają ich.

Ta praca przedstawi podstawowe informacje o robotach kroczących oraz opisze konstrukcję robota sześcionożnego – hexapoda.

Rozdział drugi poruszy teoretyczne zagadnienia związane z robotami kroczącymi oraz przedstawi kilka przykładów konstrukcji maszyn kroczących.

Rozwinięcie rozdziału drugiego o konkretne rodzaje chodów dostępnych dla konstruowanego w pracy robota nastąpi w rozdziale 3. Zawiera on również opis wybranego chodu dla opracowywanej maszyny oraz obliczeniowa analiza ruchu nogi, które przydadzą się w dalszej części pracy.

W rozdziale 4. rozpoczęta będzie praktyczna część pracy poprzez opis konstrukcji mechanicznej robota.

Rozdział 5. skupia się na elektronicznej części projektu przedstawiając odpowiednie schematy, obliczenia i dobór elementów.

Po skonstruowaniu części mechanicznej i elektronicznej należy oprogramować robota w celu jego poprawnego działania. Proces ten jest przedstawiony w rozdziale 6. Opisane są kolejne części programu sterującego, a na końcu opisany jest proces kalibracji, którego program bazował na utworzonych w tym rozdziale elementach.

Aby możliwa była komunikacja z robotem należy wybrać odpowiedni układ łączności. W tym przypadku jest to połączenie bluetooth, a sterowanie następuje z poziomu aplikacji na urządzenie mobilne. Jej działanie jest opisane w rozdziale 7.

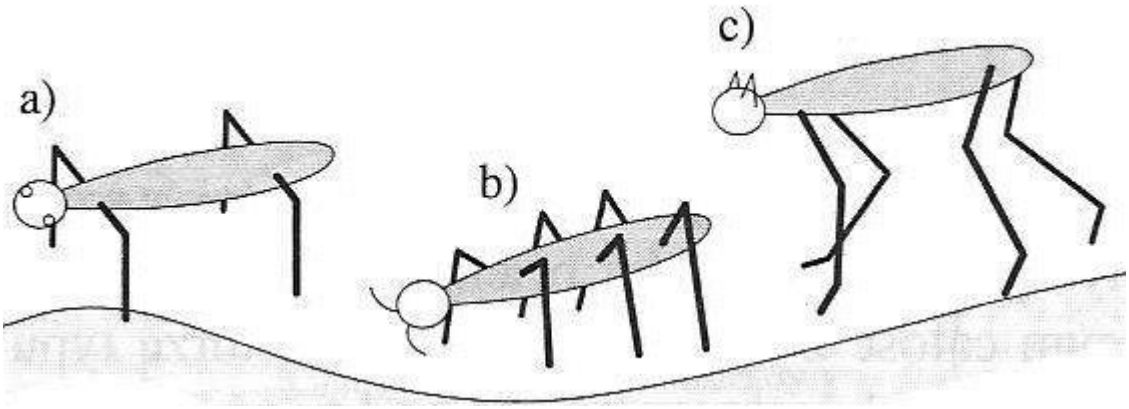
Zakończeniem jest rozdział 8, w którym praca została podsumowana. Uwzględniono w tym rozdziale proces rozwoju robota podczas jego tworzenia w celu usprawnienia jego działania i ogólne parametry konstrukcji.

2. Przegląd stanu techniki w zakresie robotów kroczących

a. Klasyfikacja robotów kroczących

Według definicji robot kroczący to urządzenie techniczne, które przemieszcza się podobnie jak większość zwierząt, czyli używając kończyn.[2]

Ponieważ roboty kroczące są zazwyczaj wzorowane na zwierzętach, naśladowane jest ich ułożenie nóg i w ten sposób można wyodrębnić 3 typowe ułożenia nóg**: gada, owada i ssaka**(rys. 2.1).[3]



Rysunek 2.1 Typowe ułożenie nóg zwierząt[1]: a)gada, b)owada, c) ssaka

To co można od razu zauważyć, to ułożenie nóg robota determinuje jego wysokość. Dla przykładu maszyna krocząca z ułożeniem nóg typowym dla ssaka ma dużo wyżej położony środek ciężkości niż np. typowym dla gada. Położenie środka ciężkości ma duży wpływ na stabilność robota i im znajduje się on wyżej tym trudniej utrzymać równowagę maszyny kroczącej.[3]

Roboty mogą się poruszać różnymi rodzajami chodu ze względu na ich stabilność.

Pierwszy rodzaj to **chód stabilny statycznie**. Robot w każdym momencie swojego ruchu może być zatrzymany i nie straci on równowagi.[3]

Jego przeciwieństwem jest **chód dynamicznie stabilny**, który może być zatrzymany tylko w niektórych momentach. W pozycjach, przy których robot nie jest stabilny statycznie utrzymuje on równowagę dzięki dynamice ruchu.[3]

W konstrukcjach w których stabilność chodu zapewnia konstrukcja nogi występuje **chód quasi-statycznie stabilny**. Aby osiągnąć stabilność tym sposobem, stopy robotów (najczęściej bipedów) są duże i ciężkie.[3]

Kolejnym parametrem robota umożliwiającym jego klasyfikację jest ilość nóg, która jest związana z jego sposobem poruszania się.

Roboty jednonożne mogą się poruszać jedynie za pomocą skoków. Są one dynamicznie stabilne, a kierunek ruchu mogą wybrać za pomocą wychylenia ciała.[3]

Roboty dwunożne (bipedy) poruszają się ruchem stabilnym dynamicznie lub quasi-statycznie stabilnym. Ich wzorcem biologicznym jest przeważnie człowiek.[3]

Roboty czworonożne (Quadropedy) potrafią się poruszać zarówno chodem statycznie stabilnym, jak i stabilnym statycznie. Ich wzorcem biologicznym są zazwyczaj ssaki lub gady.[3]

Roboty sześcionożne (Hexapody) również mogą się poruszać ruchem stabilnym statycznie lub dynamicznie. W przypadku tej grupy można jednak wyróżnić więcej niż jeden rodzaj chodu stabilnie statycznego.[3]

Roboty ośminoożne (Octapody) są wzorowane na pajakach i zwykle poruszają się chodem statycznie stabilnym.[1]

Roboty wielonożne obejmują grupę wszystkich robotów o ilości nóg większej niż 6. Roboty te poruszają się chodem stabilnym statycznie i ich ruch przypomina zwykle ruch stonogi.[1]

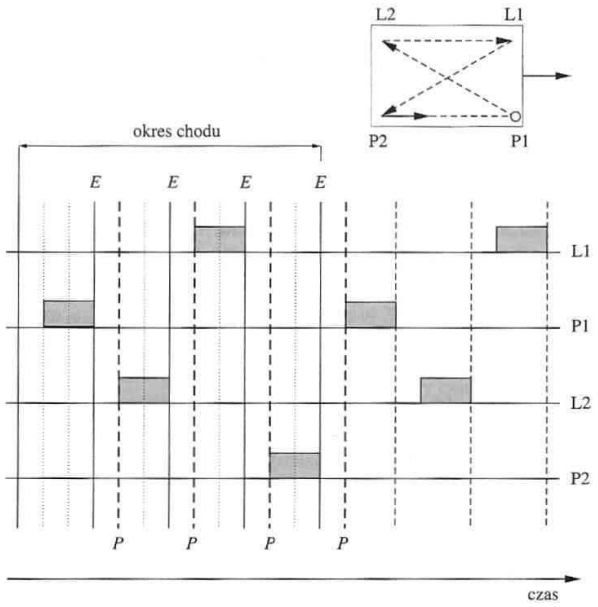
b. Rodzaje i diagramy chodu

Pierwsze analizy chodów były związane z końmi. W 1870 r. Leyland Stanford, zarządca stanu Kalifornia był przekonany, że w pewnych momentach nogi konia podczas truchtu nie stykają się z podłożem. Założył się więc ze swoim przyjacielem o 25 tys. dolarów amerykańskich (odpowiednik dzisiejszych 500 tys. dolarów) o ten fakt. Zlecono więc lokalnemu fotografowi, Edwardowi Muybridgowi zebranie dowodów w celu rozstrzygnięcia zakładu. Artysta ustawił kilkanaście aparatów fotograficznych wzdłuż trasy konia. Zwierzę biegnąc zrywało nitki ułożone na trasie, które uwalniały spusty migawek. Powstała seria zdjęć pozwoliła dowieść racji Stanforda, który wygrał zakład, a Muybridge rozwijając swój pomysł wniósł duży wkład w powstanie kamery i był jednym z pionierów kinematografii.[1] [4]

Chód to sposób poruszania się za pomocą nóg. Chody można podzielić na różne kategorie w zależności od kolejności stawianych kończyn i czasu trwania **fazy podparcia**

(inaczej retrakcji - noga dotyka podłoża i powoduje ruch robota) oraz czasu trwania **fazy przenoszenia** (inaczej protrakcji - noga przemieszcza się nad ziemią do nowej pozycji).[1]

Prostym ruchem możliwym do zaimplementowania i często używanym jest **ruch falowy**, w którym występuje sekwencja ruchów uruchomionych w pętli. Każde kolejne wykonanie cyklu ruchów jest takie same, więc możliwe jest przedstawienie takiego ruchu na diagramie.[3]

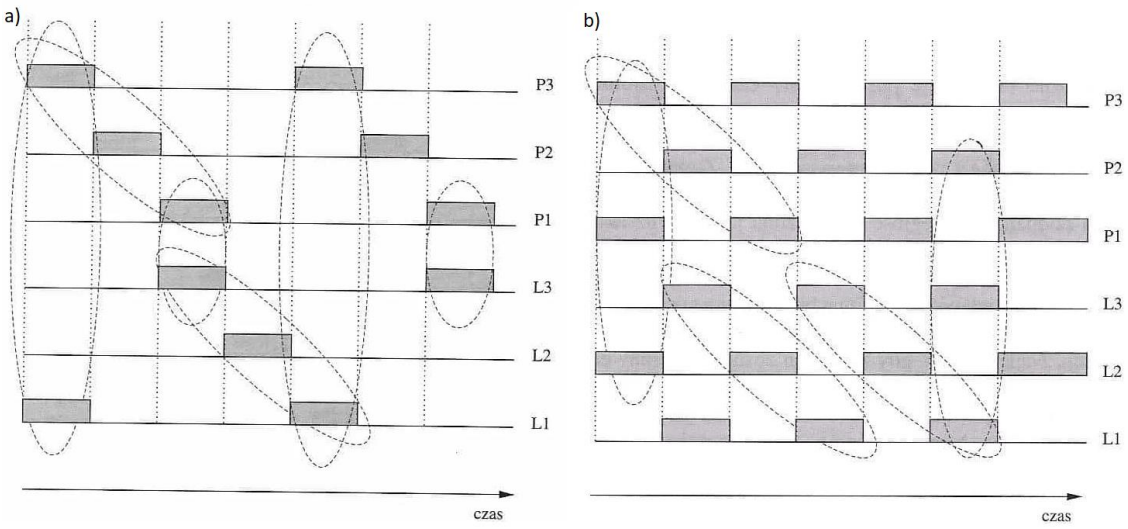


Rysunek 2.2 Diagram chodu Quatropoda - prostokąty symbolizują fazę przenoszenia konkretnej nogi[1]

Dzięki diagramom jesteśmy w stanie w prosty sposób zarówno odczytać sekwencję ruchów, jak i znaleźć **współczynnik obciążenia nogi**. Jej wzór można przedstawić jako:

$$współczynnik\ obciążenia\ nogi = \frac{czas\ przemieszczenia\ (protrakcji)}{okres\ chodu}$$

Poniżej przedstawiono diagramy hexapoda dla chodu o współczynniku obciążenia nogi równym 2/3 oraz 1/2 (rys. 2.3).



Rysunek 2.3 Diagram chodu hexapoda o współczynniku obciążenia 1/4 (a) i 1/2(b)[1]

c. Napędy robotów kroczących i przykłady konstrukcji

Temat robotów kroczących jest podejmowany przez zarówno hobbystów robotyki jak np. James Burton, jak i firmy z branży robotyki takie jak np. Festo i Boston Dynamics, których robot „Spot" był nawet inspiracją do odcinka „Metalhead" serialu „Black Mirror".

Rysunek 2.4 Po lewej stronie robot Spot[5], a po prawej jego mroczna wersja z serialu "Black Mirror"[6]

Pierwsza różnica, którą widać to sterowanie, które dla najprostszych robotów jest realizowane za pomocą różnego rodzaju kontrolerów, dzięki którym zadaje się konkretne ruchy lub pozycje robota. W zaawansowanych rozwiązaniach można np. podać tylko punkt docelowy, do którego robot sam będzie się kierował omijając różne przeszkody na swojej drodze.

Bardziej zaawansowane roboty korzystają również z Systemu **ROS** (*ang. Robot Operating System*), który umożliwia im symulowanie ruchów i ich konsekwencji zanim dojdzie do nich w rzeczywistości.

Poza układami sterowania, które są zwykle bardziej zaawansowane w przypadku profesjonalnych robotów różnice występują w używanych napędach.

W prostszych projektach używa się **serwomechanizmów modelarskich**. Są stosunkowo tanie i bardzo prosto jest nimi sterować. W tym celu wystarczy na przewód sygnałowy (standardowo żółty) wysłać sygnał PWM (ang. Pulse-Width modulation) o częstotliwości ok. 50Hz (okresie 20ms) i odpowiednim wypełnieniu (zwykle stan wysoki trwa od 540ms-2400ms). Sterownik w środku serwomechanizmu interpretuje zadaną wartość i za pomocą potencjometru porównuje z aktualnym położeniem wału, a następnie przemieszcza za pomocą silnika DC i systemu przekładni do żądanego położenia. [7]

Mają one jednak kilka mankamentów. Po pierwsze, można sterować jedynie ich położeniem, ale już nie np. prędkością, czy momentem. Po drugie mają w środku przekładnię, która jest wykonana zwykle (zwłaszcza w przypadku tańszych modeli) z tworzywa, przez co koła zębate przy większym obciążeniu mogą się zwyczajnie zniszczyć. [7]

Dodatkowo samo pozycjonowanie nie jest bardzo dokładne. Przykładowo serwomechanizm MG995 posiada histerezę otrzymanego sygnału równą 0,005ms, co odpowiada wartości ok. 0,5°.[8]

Wiele z tych problemów da się obejść stosując np. dodatkowe przekładnie i sprężyny, czy dodatkowe czujniki jak np. James Burton w swoim projekcie „Mini Robot Dog", przez co mogą one znacznie poprawić swoje parametry.[9]

W bardziej profesjonalnych projektach stosuje się jednak zwykle silniki **BLDC** (ang. *brushless direct-current motor)*, które pozwalają na dużo większą kontrolę nad ruchem stawów robota. Ich cena jednak jest zdecydowanie wyższa, a sterowanie bardziej skomplikowane. Jeżeli tworzenie własnego kontrolera silników pochłonie za dużo pracy można skorzystać z gotowych modułów sterowników, jednak ich cena również jest dosyć wysoka.

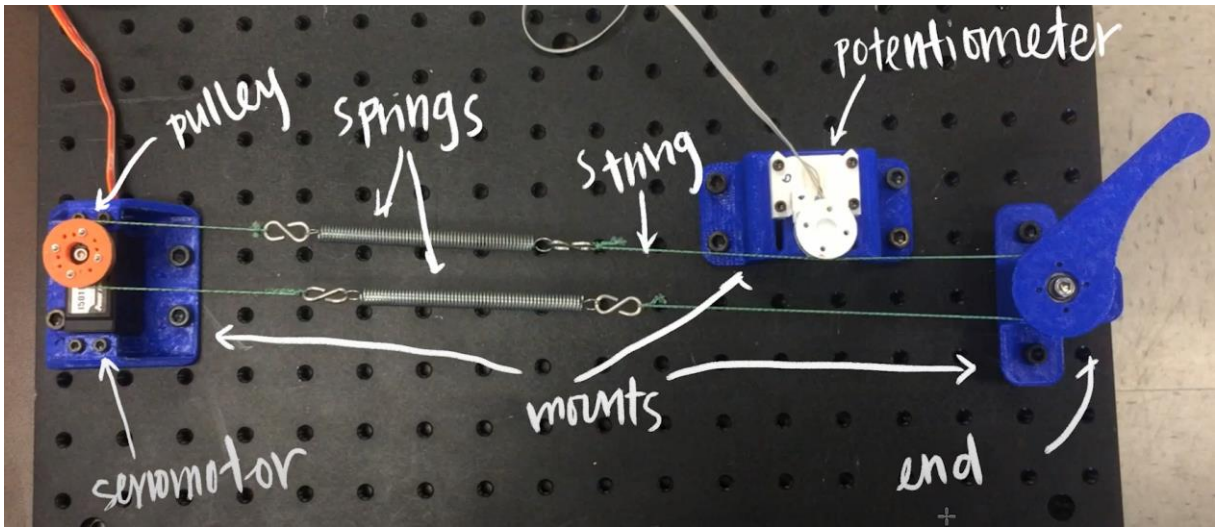
Ich użycie może mieć następujące formy:

DD – (ang. *Direct Drive*) nie stosuje się żadnych przekładni w celu zwiększenia momentu obrotowego. Zaletami tego rozwiązania są prostota, mała masa, bardzo małe opory mechaniczne i niezawodność układu. Dodatkowo brak przekładni pozwala na zrzucenie robota z wysokości 2m bez obawy o zniszczenie zębatek. Wadą jest mały moment siły w stosunku do następnych rozwiązań, ponieważ w całości pochodzi z silnika elektrycznego, przez co zwłaszcza przy jego dynamicznym używaniu przez dłuższy czas może się on silnie nagrzewać.[10]

Przy zastosowaniu przekładni redukcyjnej (o przełożeniu zwykle do 10:1) otrzymujemy **QDD** (ang. *Quasi-Direct Drive*). Dzięki temu napędy mają dużo wyższy moment siły przy braku zwiększenia masy silnika. Napędy takie są jednak nieco bardziej skomplikowane i wymagają wykonania wysokiej jakości części. Ze względu na ich własności stają się one najczęstszym typem stosowanym do napędów robotów kroczących, szczególnie w mniejszych konstrukcjach.[10]

Rysunek 2.5 Przykładem bardziej zaawansowanego robota jest maszyna zbudowana przez startup MAB Robotics, założony przez studentów Politechniki Poznańskiej.[10]

SEA (ang. *series elastic actuator*) to układ, w którym przekładnia ma wyższe przełożenie (zwykle 100:1). Dodatkowo stosuje się zestaw sprężyn, który pozwala na dokładne sterowanie momentem z którym napęd działa (nie trzeba go już przybliżać mierząc pobór prądu) oraz zwiększa tolerancję na wstrząsy.[11]



Rysunek 2.6 Zasada działania napędów SEA - mierząc długość sprężyn i znając ich stałą sprężystości można wyznaczyć siły działające w układzie z prawa Hooke'a: $F = k * x$, gdzie F to siła, k to współczynnik sprężystości, a x to wydłużenie sprężyny[12]

W rzeczywistości układy tego typu potrafią być bardzo złożone i przez to są dosyć drogie. Mają one jednak naprawdę wiele zalet, przez co roboty bardzo dobrze dostosowują się do terenu, w którym pracują.[10]

Siłowniki hydrauliczne również są dobrym wyborem, jeżeli wymagane są większe wartości momentów siły. Robot Atlas stworzony przez Boston Dynamics korzysta z nich i pozwalają one mu poruszać się z dużą precyzją. Dzięki temu może on biegać, skakać, czy nawet robić salta. [10]

Robot HyQReal z Istituto Italiano di Tecnologia Dynamic Legged Systems, w którym również zastosowano napęd hydrauliczny jest w stanie nawet przeciągnąć mały samolot pasażerski Piaggio P180 Avanti, ważący 3300kg i o rozpiętości skrzydeł 14m, sam ważąc przy tym 140 kg.[13]

Ten rodzaj napędów ma jednak również wady. Należy dokonywać częstych przeglądów, a cały układ musi być odporny na wysokie ciśnienie cieczy, która w nim krąży. Sama kontrola cieczy, w celu wykonywania dokładnych ruchów jest skomplikowana, przez co układy te są bardzo drogie. Dodatkowo należy stosować szczególne środki bezpieczeństwa, szczególnie przy pracy urządzenia w obszarze przebywania ludzi ze względu na duże wysokie ciśnienie w układzie.[10]

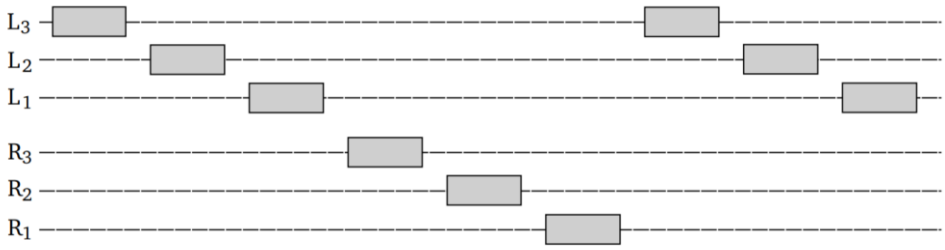
3. Algorytm chodu

a. Typy chodów stabilnych statycznie robotów sześcionożnych

W zależności od konkretnego robota, algorytmy chodu mogą być różne. Przy doborze należy przede wszystkim zwrócić uwagę rodzaj chodu, ilość i rozmieszczenie nóg i moc użytych napędów.

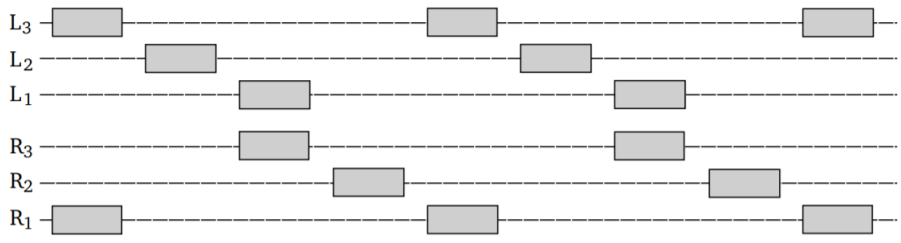
Dla hexapoda, który poruszałby się chodem stabilnym statycznie istnieje kilka możliwości poruszania się robota:

Chód metachroniczny (pełzający) - jednocześnie przenoszona jest jedna noga robota, a współczynnik obciążenia każdej z nóg wynosi 1/6. Jest to najwolniejszy sposób przemieszczania się robota, jednak jest on bardzo stabilny i w przypadku nierówności terenu, czy różnych przeszkód robot nie powinien stracić równowagi.[14]



Rysunek 3.1 Wykres kolejności przestawiania nóg w chodzie metachronicznym[14]

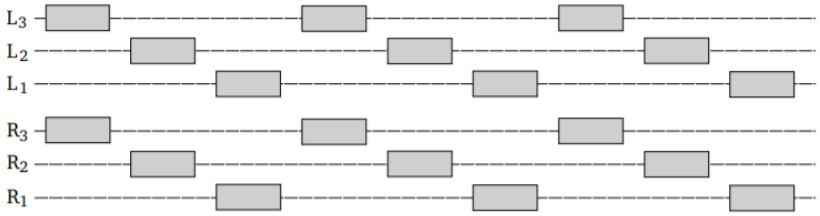
Chód czterotaktowy – jednocześnie przenoszone są jedna lub dwie nogi, ich współczynnik obciążenia wynosi 1/4. Chód ten charakteryzuje się nieco większą prędkością od chodu pełzającego, przy czym nadal jest bardzo stabilny.[14]



Rysunek 3.2 Wykres kolejności przestawiania nóg w chodzie czterotaktowym[14]

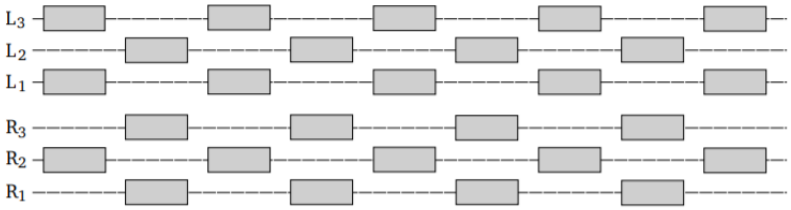
Chód gąsienicowy – nogi przenoszone są parami, od tyłu do przodu (na podobieństwo gąsienicy). Współczynnik obciążenia nóg w przypadku tego chodu wynosi 1/3. Ten typ chodu nie jest już tak stabilny jak poprzednie i jego implementacja wymaga dobrania odpowiednich wartości wykroku i zakroku, a w przypadku złych warunków

otoczenia (nierównego podłoża czy przeszkód) może dojść do utraty stabilności. Chód ten dobrze sprawdza się jednak w przypadku regularnych kształtów przeszkód (np. schody).[14]



Rysunek 3.3 Wykres kolejności przestawiania nóg w chodzie gąsienicowym[14]

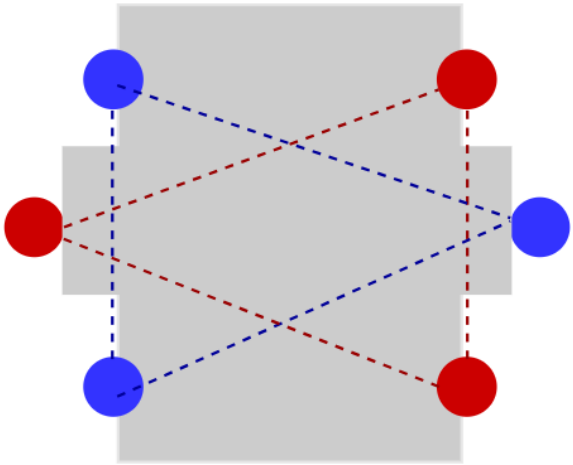
Chód trójpodporowy – jest to najszybszy chód, jakim mogą się poruszać sześcionogi. Współczynnik obciążenia nóg w tym przypadku wynosi 1/2. Jest to najczęściej stosowany chód przez owady sześcionożne i sprawdza się bardzo dobrze na płaskich odcinkach. Posiada najmniej zapasu stabilności z wymienionych chodów i w celu jej zwiększenia owady które korzystają z chodu trójpodporowego mają zwykle środkowe nogi rozstawione szerzej, posiadając dodatkowo nisko środek ciężkości wynikający z ułożenia nóg typowego dla owadów (przedstawiony w poprzednim rozdziale). [14]



Rysunek 3.4 Wykres kolejności przestawiania nóg w chodzie trójpodporowym[14]

b. Sekwencja poruszania nogami

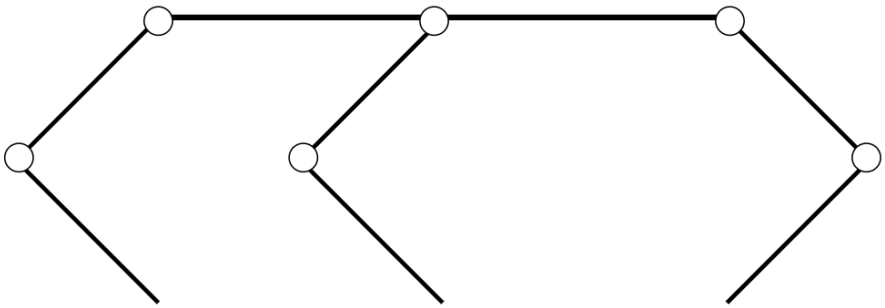
Wybrano najszybszy możliwy chód dla hexapoda, czyli chód trójpodporowy. Może on zatem jednocześnie przesuwać się na 3 nogach stojących na ziemi (faza retrakcji), a pozostałe 3 są w tym czasie przemieszczane nad ziemią (faza protrakcji), aby po opuszczeniu znajdowały się w odpowiedniej pozycji.



Rysunek 3.5 Zespoły współpracujących ze sobą nóg nr 1 (niebieski) i 2 (czerwony)

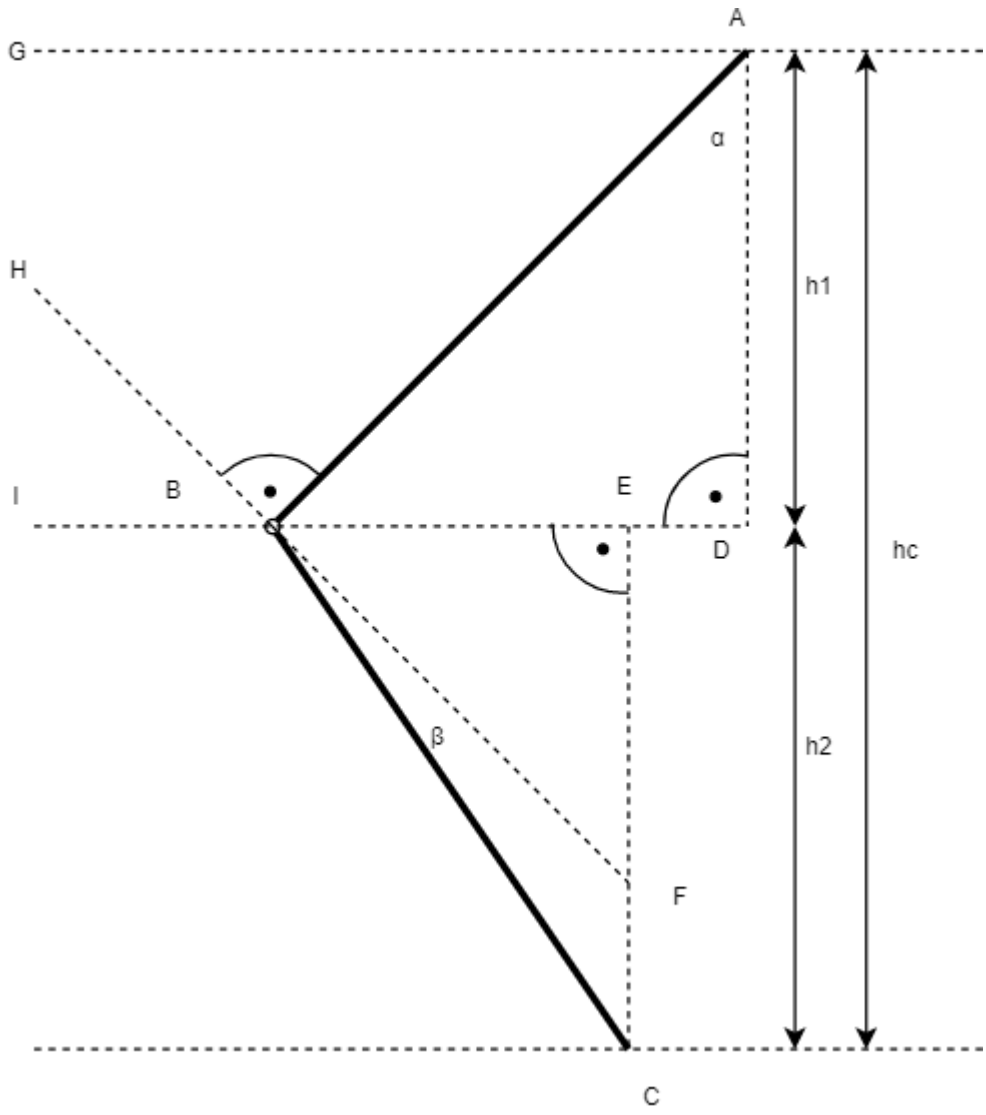
Nogi poruszają się w zespołach (jak na rysunku 3.5), przez co środek ciężkości robota znajduje się zawsze w środku trójkąta o wierzchołkach będących nogami robota z jednego zespołu.

Szczególnie przy cięższych nogach zmiana środka ciężkości robota przy ruchu nóg ma duże znaczenie, więc w celu zmniejszenia wpływu tego czynnika nie wszystkie nogi zginają się w tą samą stronę (rys.3.6).



Rysunek 3.6 Schemat kinematyczny przedstawiający ułożenie i kierunek zginania nóg robota - rysunek od boku

c. Dopasowanie kąta między członami nogi w celu osiągnięcia stałej wysokości robota



Rysunek 3.7 geometryczny schemat nogi robota

Na rysunku powyżej (3.7) pogrubione odcinki AB i BC symbolizują dwa segmenty nogi robota, a hc to wysokość jego nogi. Serwomechanizmy są umieszczone w punktach A oraz B.

Z zależności geometrycznych otrzymujemy:

$\sphericalangle ABD = 90° - \alpha$ (suma kątów w trójkącie ABD = 180°)

$\sphericalangle DBF = \alpha$ ($\sphericalangle ABF - \sphericalangle ABD = 90° - (90° - \alpha) = \alpha$)

$\sphericalangle HBI = \alpha$ ($\sphericalangle HBI$ i $\sphericalangle DBF$ są wierzchołkowe)

Kąt zadawany serwomechanizmowi w punkcie A to $\sphericalangle GAB$, a serwomechanizmowi w punkcie B $\sphericalangle HBC$.

Program zmienia położenie serwomechanizmu w punkcie A w zadanym zakresie, a silnik w punkcie B ma za zadanie znajdować taką pozycję nogi, przy której hc (wysokość całkowita) nie zmienia się.

Ponieważ długości AB i BC są równe, zakładamy dla łatwiejszych rachunków, że są równe 1, więc hc<2 i odpowiada ona sumie cosinusa α i sinusa (α+β).

Program ma więc dany kąt GAB, długości nóg AB i BC (AB=BC) oraz hc i na tej podstawie ma za zadanie obliczyć kąt HBC.

Aby to zrobić program korzysta następujących wzorów:

$$\alpha = 90° - \sphericalangle GAB$$

$$h1 = cos\alpha$$

$$\beta = \mathrm{asin}(hc - \mathrm{h1}) - \alpha$$

$$\sphericalangle HBC = 180° - \beta$$

W przypadku nogi postawionej w drugą stronę analogicznie (rys. 3.8):

Kąt zadawany serwomechanizmowi w punkcie A' to ∢J'A'B', a serwomechanizmowi w punkcie B' ∢D'B'C'.

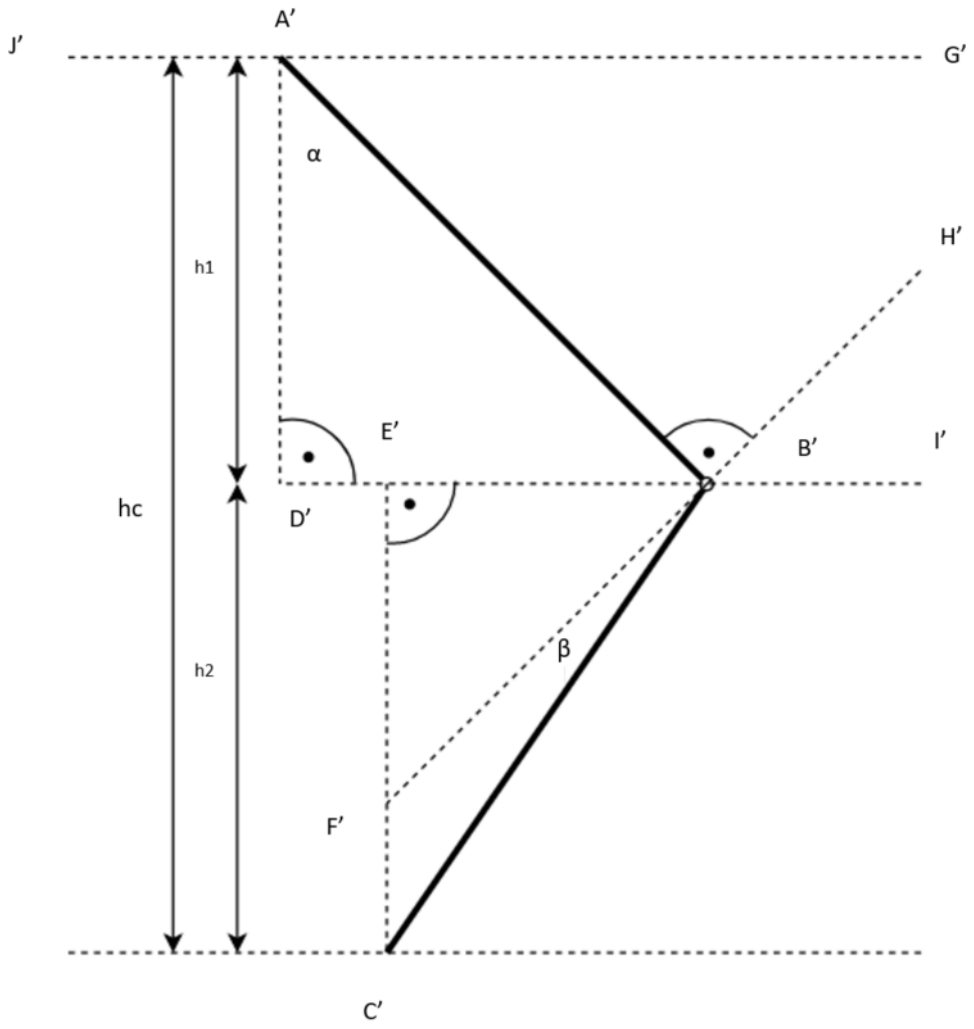
Program ma więc tak jak w poprzednim przypadku dany jeden kąt J'A'B', długości nóg AB i BC (AB=BC) oraz hc i na tej podstawie ma za zadanie obliczyć kąt D'B'C'.

Aby to zrobić program korzysta następujących wzorów:

$$\alpha = 90° - \sphericalangle J'A'B'$$

$$h1 = cos\alpha * 1$$

$$\sphericalangle D'B'C' = \mathrm{asin}(hc - \mathrm{h1})$$



Rysunek 3.8 Analiza geometryczna nogi ugiętej w drugą stronę

d. Analiza długości kroku

Dla każdego robota kroczącego można przeprowadzić analizę chodu. W tym celu analizuje się długość kroku, który jest odległością między dwoma kolejnymi śladami tej samej nogi.[1]

W naszym przypadku można go policzyć korzystając z rysunku 3.8 z poprzedniego podpunktu. Zakładając, że zadana wartość na serwomechanizm górny zmienia się od 100° do 130°, a długość każdego segmentu nogi wynosi 70mm, można policzyć położenie końca nogi w skrajnych położeniach w osi poziomej względem górnego stawu nogi.

W tym celu skorzystano z zależności trygonometrycznych:

$$położenie\ stawu\ dolnego(\alpha) = sin\alpha * l$$

Gdzie l to długość człony nogi (obydwa człony są jednakowe i mają długość 70mm)

Zatem:

$$położenie\ stawu\ dolnego(100° - 90°) = \sin(10°) * 70 \approx 12mm$$

$$położenie\ stawu\ dolnego(130° - 90°) = \sin(40°) * 70 \approx 45mm$$

Natomiast przesunięcie końca stopy względem dolnego stawu można przedstawić następująco:

$$położenie\ stopy(\alpha + \beta) = -\cos(\alpha + \beta) * l + położenie\ stawu\ dolnego(\alpha)$$

Z rozdziału 3. wiemy, że:

$$h1 = cos\alpha$$

$$\beta = \text{asin}(hc - \text{h1}) - \alpha$$

Zatem po podstawieniu odpowiedniej wartości pod β otrzymamy:

$$położenie\ stopy(\alpha) = -\cos(\text{asin}(\text{hc} - \cos\alpha)) * l + sin\alpha * l$$

Zatem po podstawieniu odpowiednich wartości otrzymujemy:

$$położenie\ stopy(100° - 90°) = -\cos(\text{asin}(1.7 - \cos 10°)) * 70 + sin10° * 70 \approx -37mm$$

$$położenie\ stopy(130° - 90°) = -\cos(\text{asin}(1.7 - \cos 40°)) * 70 + sin40° * 70 \approx 20mm$$

Krok wynosi więc w tym wypadku różnicę tych wartości:

$$Krok = 20mm - (-37mm) = 57mm$$

4. Konstrukcja Mechaniczna

Do zaprojektowania robota skorzystano z programu Fusion 360.

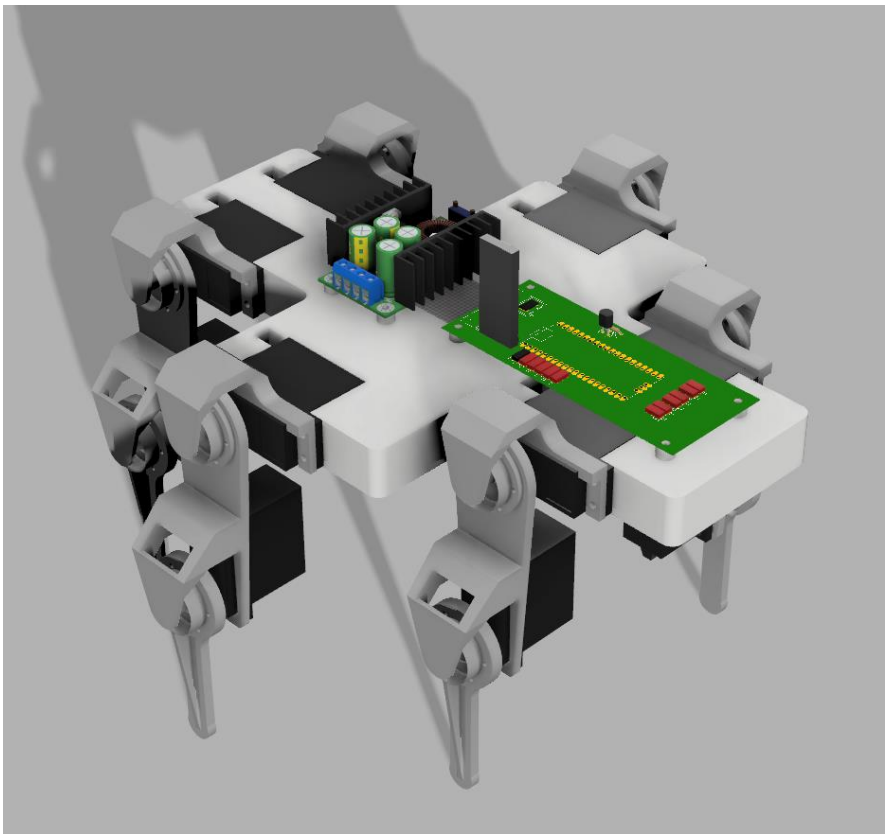
Środkowe nogi zostały oddalone od środka ciężkości robota w celu zwiększenia zakresu ruchu nóg robota oraz w celu zmniejszenia sił działających na te nogi, ponieważ w chodzie trójpodporowym są one jedynymi punktami podpierającymi robota ze swojej strony. Dodatkowo zabieg ten pozwala na zwiększenie stabilności robota.

Po zaprojektowaniu płytki w programie Eagle można z niej było również wygenerować model CAD, który został dodany do projektu.

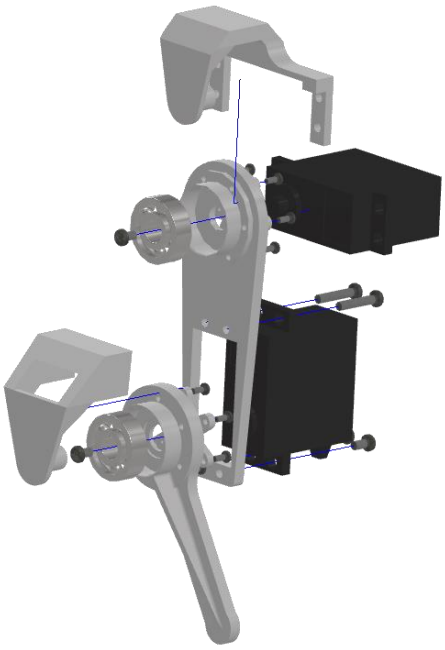
Zastosowana bateria została zmierzona i zamodelowana za pomocą prostopadłościanu z naddatkiem wynikającym z wychodzących z niej przewodów, a model przetwornicy został pobrany z ze strony grabcad. [15]

Do napędu nóg zastosowano 12 serwomechanizmów Mg995, których model pobrano ze strony grabcad.[16] Okazało się jednak, że orczyk z pobranego modelu nie ma odpowiednich wymiarów, w związku z czym należało go zamodelować. W celu odciążenia wałów silników zastosowano łożyska kulkowe promieniowe typu 608ZZ (o wymiarach 8x22x7).

Obydwa człony nóg mają tą samą długość, aby uprościć obliczenia, które będzie wykonywał mikrokontroler podczas ruchu robota.



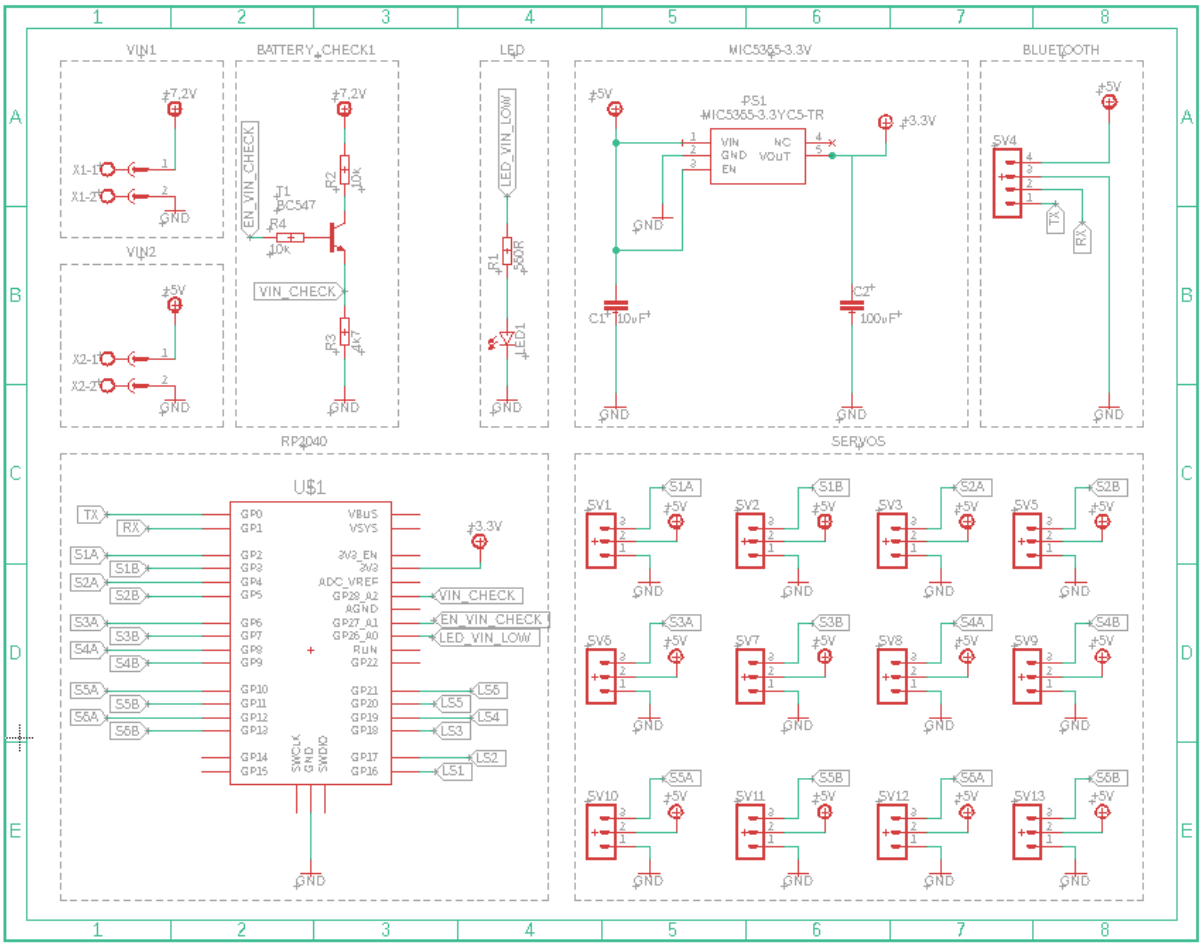
Rysunek 4.1 Zdjęcie modelu robota



Rysunek 4.2 Zdjęcie złożenia nogi robota

5. Projekt Elektroniki

a. Schemat elektroniki



Rysunek 5.1 Schemat elektroniczny płytki stworzonej do sterowania robotem

Większość użytych elementów zostało wziętych ze standardowych bibliotek programu Eagle, a elementy niewystępujące w bibliotekach podstawowych pobrano z Internetu.[17]

b. Spis elementów

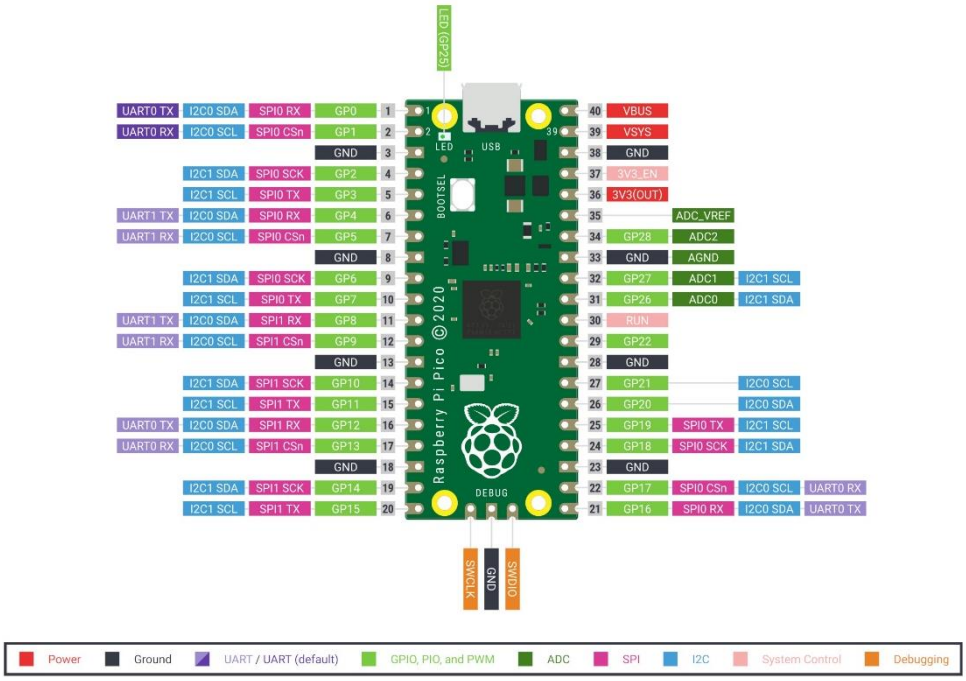
i. Mikrokontroler Raspberry Pi Pico (RP2040) – do sterowania robotem potrzebowano mikrokontrolera, który pozwalałby na:

- Kontrolę 12 serwomechanizmów przez PWM
- Pozwalałby na komunikację za pomocą UARTu (ang. Unsynchronised asynchronous receiver-transmitter)
- Miałby wejście analogowe do czytania napięcia baterii, aby robot nie pracował, kiedy napięcie baterii osiągnie zbyt niską wartość

- Miałby jeszcze co najmniej 2 wolne piny cyfrowe wejścia/wyjścia (GPIO – ang. *general-purpose input/output*)
- Miałby wystarczającą pamięć do wgrania napisanego programu

Wybrany mikrokontroler spełnia wszystkie założenia posiadając:

- 16 osobnych kanałów PWM
- 2 interfejsy UART
- 3 wejścia analogowe
- 26 pinów cyfrowych
- 2048 kB pamięci flash[18]



Rysunek 5.2 Pinout Raspberry pi pico i funkcje konkrektnych pinów[18]

ii. Serwomechanizmy MG995

Rysunek 5.3 Zdjęcie serwomechanizmu MG995[8]

iii. Moduł bluetooth HC-06

Rysunek 5.4 Zdjęcie modułu bluetooth HC-06[19]

W celu połączenia przez bluetooth aplikacji na urządzenie mobilne z robotem skorzystano z modułu HC-06, który zamienia informację przesłaną przez bluetooth na UART i na odwrót.

Ponieważ moduł HC-06 działa pod napięciem od 3,6V do 6V został on podłączony do 5V. Działa on jednak na tym samym standardzie co mikrokontroler, czyli CMOS (stan wysoki - 3,3V). [19]

iv. Stabilizator MIC5365 -3.3YC5-TR – stabilizator obniżający napięcie do 3,3V, w celu zasilenia mikrokontrolera raspberry pi pico
v. Przetwornica xl4016, która obniża napięcie baterii na 5V.

Rysunek 5.5 Zdjęcie przetwornicy XL4016[20]

vi. Pakiet Li-Pol firmy Dualsky o pojemności 800mAh, prądzie rozładowania 25C i dwóch celach połączonych szeregowo (o napięciu łącznie 7,4V)

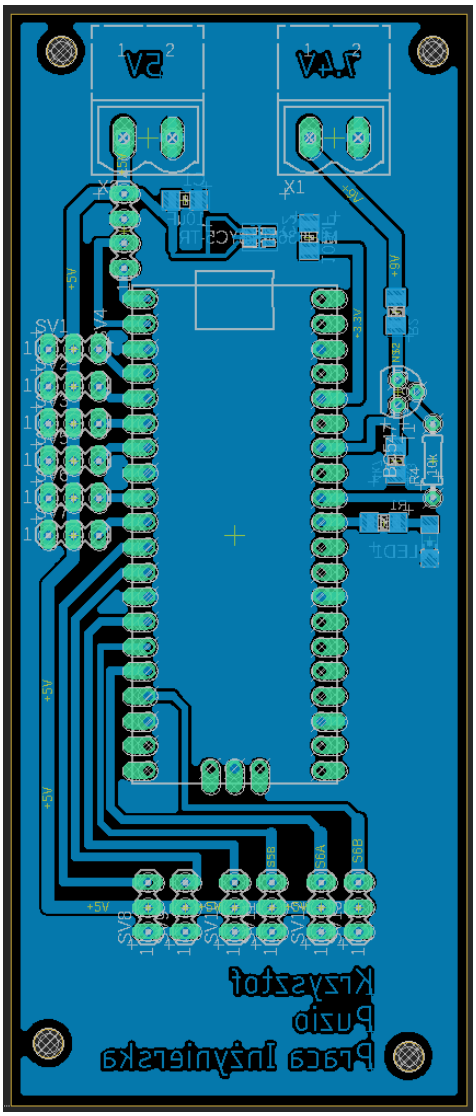
Rysunek 5.6 Zdjęcie użytego akumulatora li-pol[21]

vii. Wejścia ARK
viii. Tranzystor BC547B wchodzi w stan nasycenia w celu zmierzenia napięcia baterii, a następnie w stan zatkania w celu ograniczenia pobieranego prądu
ix. LED o kolorze czerwonym
x. Oporniki
xi. Kondensatory

c. Płytka PCB(ang. Printed Circuit Board)

Tworząc płytkę drukowaną część elementów takie jak goldpiny, złącza ARK, czy użyty tranzystor zostały zamontowane na płytce za pomocą technologii THT (ang.*Through-Hole Technology*), a część takie jak rezystory i kondensatory za pomocą metody SMD (ang. *surface-mount technology*).

Szczególnie należało uważać lutując stabilizator, ze względu na jego bardzo małe wymiary, aby po pierwsze nie zewrzeć ze sobą nóżek pinów wychodzących z obudowy i aby go umieścić w odpowiedniej pozycji na płytce. Dodatkowo projektując płytkę jednowarstwową pojawił się mały problem z połączeniem standardowej grubości ścieżkami pinu GND (ang. *Ground*) z masą całej płytki. Połączono więc go trzema bardzo cienkimi ścieżkami z masą wylaną na płytce, co jak się okazało po zbudowaniu prototypu nie sprawiło żadnego problemu zarówno w działaniu jak i w montażu układu.



Rysunek 5.7 Projekt płytki PCB

d. Obliczenia wartości elementów

i. Przetwornica

Z pomiarów wyznaczono średni prąd pracy (podczas ruchu robota) jako 500mA przy napięciu baterii 8,2V, więc średnia moc wynosi 4,1W.

Maksymalny prąd, jaki teoretycznie mógłby pobierać nasz układ wynosi:

I_m – prąd pobierany przez mikrokontroler = 100mA

12 * I_s – prąd pobierany przez serwomechanizmy = 12 * 350mA = 4,2 A

I_{rest} – maksymalny prąd pobierany przez moduł HC-06 oraz diody inne elementy = 100mA

Moc maksymalna wynosi:

$$p_{max} = I_c * U$$

Gdzie:

p – moc pobierana przez robota

I – całkowity prąd maksymalny = 4,4A

U – napięcie przy którym działa układ = 5V

$$p_{max} = 4{,}4\ A * 5V = 22W$$

Dobrano przetwornicę XL4016 o parametrach[20]:

I_{max} = 9A

p = 300W

spełnia ona zatem wymagane warunki.

ii. Bateria Li-Pol

W celu dobrania odpowiedniej baterii należy najpierw oszacować prąd pobierany przez układ, napięcie potrzebne do zasilania i czas przez który urządzenie ma działać. W tym konkretnym przypadku należy również szczególnie uwzględnić masę baterii, ponieważ będą ją musiały dźwigać nogi robota.

Z pomiarów wykonanych przy poprzednim podpunkcie wyznaczono średnią moc pobieraną przez robota równą 4,1W. Zakładając średnie napięcie baterii równe 7,4V średni prąd pobierany przez baterię wynosi ok. 600mA.

Dobrano baterię o pojemności 800mAh, wydajności prądowej 25C[21].

Można więc obliczyć maksymalny prąd, jaki można pobrać z baterii[22]:

$$I_{max} = C_p * x$$

Gdzie:

I_{max} - maksymalne natężenie prądu

C_p – pojemność pakietu

x - wydajność prądowa

$$I_{max} = 25C * 0{,}8\ Ah = 20A$$

Czas w jakim robot może działać na jednym naładowaniu[22]:

$$t_p = C_p / I_ś$$

Gdzie:

t_p - wydajność prądowa

$I_ś$ - średnie natężenie prądu

C_p – pojemność pakietu

$$t_p = \frac{800mAh}{600mA} = 1h\ 18min$$

iii. Stabilizator MIC5365-3.3YC5-TR

Stabilizator ma za zadanie obniżyć napięcie z 5V na 3,3V, aby dostosować je do mikrokontrolera. Maksymalny prąd jaki pobiera raspberry pi pico to 100mA[23].

Moc która wydzieli się na stabilizatorze wyniesie zatem:

$$p = I * U_{drop}$$

p – moc wydzielona na stabilizatorze

I – natężenie prądu płynącego przez stabilizator

U_{drop} – różnica napięć między napięciem wejściowym a wyjściowym

$$p = 0{,}1\ A * 1{,}7\ V = 0{,}17W$$

Wzór na temperaturę stabilizatora:

$$T = T_A + p * R_{JA}$$

T - temperatura stabilizatora

T_A – temperatura otoczenia

p – moc wydzielona na stabilizatorze

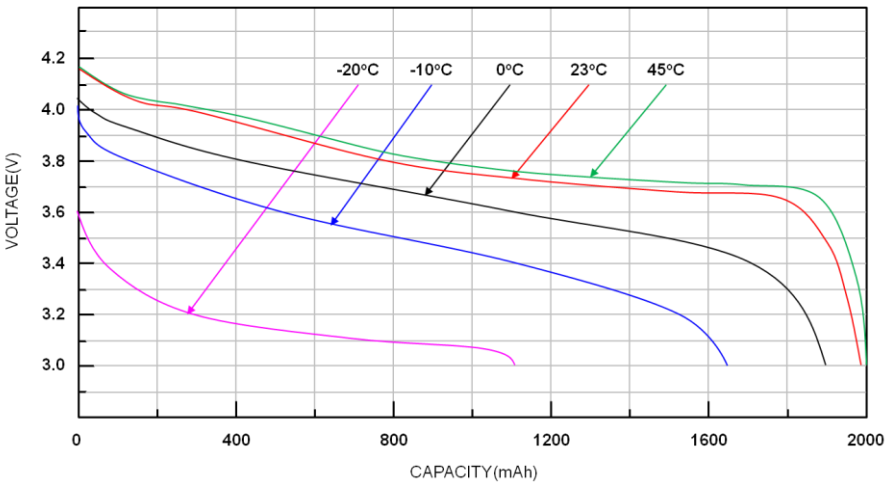
R_{JA} - Rezystancja cieplna równa 256,5 °C/W[24]

$$T = 20°C + 0{,}17W * 256{,}5\ °C/W = 63{,}6°C$$

Temperatura pracy stabilizatora przy pokojowej temperaturze otoczenia będzie zatem się znajdować w odpowiednim zakresie i nie będzie za gorąca.

iv. Mierzenie napięcia baterii

Układ jest zasilany z baterii LiPol składającej się z dwóch ogniw połączonych szeregowo. W celu dobrego działania baterii pojedyncze ogniwo powinno pracować w przedziale od 3,4V (poniżej tego napięcia bateria rozładowuje się bardzo szybko) do 4,2V (maksymalne napięcie baterii).[25]



Rysunek 5.8 Wykres przedstawia nieliniowość rozładowywania się baterii Li-Pol[26]

Posiadając w akumulatorze dwie cele połączone szeregowo mnożymy więc te wartości razy 2 i otrzymujemy przedział od 6,8 V do 8,4 V, przy czym zwiększono dolną granicę do 7 V.

Zastosowany dzielnik napięcia powinien obniżać napięcie baterii do maksymalnie do napięcia zasilania mikrokontrolera, które wynosi 3,3V. [23]

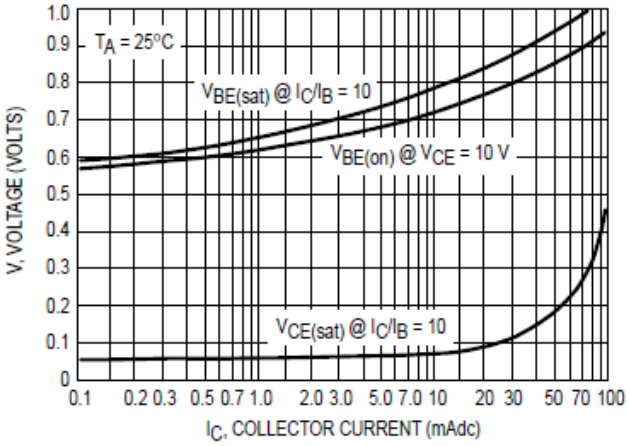
Spadek napięcia na tranzystorze w stanie nasycenia przy prądzie kolektora 10 mA wynosi typowo 0.09V, a maksymalnie 0.25V. W układzie może płynąć jeszcze mniejszy prąd, ponieważ nie potrzebujemy z niego nic zasilać, a po prostu należy sprawdzić napięcie baterii. Może więc on wynosić np. 0,5mA.[27]

Aby policzyć rezystancją obydwóch tranzystorów korzystamy ze wzoru:

$$R = U/I$$

Po podstawieniu wartości 7,4V i 0,5mA otrzymujemy 14,8 kΩ.

Z charakterystyk tranzystora widać, że dla mniejszego natężenia prądu napięcie kolektor-emiter osiąga nieco niższe wartości, jednak niewiele, w związku z czym można założyć spadek 0,09V.[27]



Rysunek 5.9 Wykres zależności Napięcia od prądu kolektora[27]

Wzór na napięcie na dzielniku napięcia będzie więc miał następującą postać:

$$U_{ADC} = V_{CE} + (U_B - V_{CE}) * \frac{R_3}{R_3 + R_2}$$

Gdzie:

U_{ADC} - Napięcie na przetworniku analogowo cyfrowym

V_{CE} – Napięcie kolektor-emiter

U_b – napięcie baterii

Ponieważ napięcie kolektor-emiter w stosunku do napięcia baterii jest znikome (ok.1%) można je pominąć i otrzymujemy klasyczny wzór dla dzielnika napięcia:

$$U_{ADC} = U_B * \frac{R_3}{R_3 + R_2}$$

Dobrano rezystory o wartościach 4,7kΩ i 10 kΩ, zatem maksymalne napięcie na mikrokontrolerze będzie wynosić ok.2,69V, zatem poniżej 3,3V, w związku z czym nawet, gdy bateria byłaby w pełni naładowana układ będzie działał całkowicie poprawnie i bezpiecznie.

Maksymalne napięcie, którym można by zasilać układ, żeby nie przekroczyć napięcia zasilania mikrokontrolera przy dzielniku napięcia można prosto policzyć po przekształceniu wzoru na napięcie na wejściu mikrokontrolera i wynosi ono 10,3V.

Napięcie na wejściu mikrokontrolera, które będzie uznane za zbyt niskie do działania układu (bateria osiągnie poziom poniżej 7V) wynosi natomiast 2,238V.

Do tranzystora dobrano również rezystor pozwalający na osiągnięcia saturacji dla stanu wysokiego pinu mikrokontrolera.

Minimalny prąd bazy, aby osiągnąć napięcie saturacji powinien wynosić:

$$I_{Bmin} = Ic/\beta$$

Gdzie:

I_{Bmin} – minimalny prąd bazy

I_c – prąd kolektora

β – współczynnik wzmocnienia tranzystora

Napięcie baza-emiter wynosi 0,7V, w związku z czym na oporniku napięcie będzie spadać o 2,6V.[27]

Wzmocnienie przy prądzie kolektora wynosi ok. 75% wartości referencyjnej (z wykresu), która wynosi typowo 290, w związku z czym wzmocnienie wyniesie 217. [27]

Po podstawieniu odpowiednich wartości otrzymamy 2,3uA.

Rezystor zatem powinien mieć maksymalnie wartość 1MΩ, zastosowano jednak mniejszy rezystor 10kΩ.

v. LED

W celu poprawnego działania na użytej diodzie napięcie powinno spadać o 1,8V przy natężeniu prądu 20mA. W związku z tym napięcie, które spadnie na oporniku powinno wynosić:[28]

$$3{,}3V - 1{,}8V = 1{,}5V$$

Przy 20mA LED będzie świecił dosyć mocno, a ponieważ jest on potrzebny tylko do powiadomienia użytkownika, który znajduje się obok można użyć dużo mniejszej wartości natężenia prądu. Poza tym, aby diodę z rezystorem podpiąć bezpośrednio do mikrokontrolera należy zobaczyć jakie natężenie prądu jest on w stanie zapewnić na wyjściach. Dla Raspberry pi pico jest to 50mA na wszystkie wyjścia, więc na zaświecenie diodą trzeba by było użyć prawie połowę dostępnego dla wyjść prądu. Przyjęto zatem, że prąd płynący przez diodę będzie równy ok. 2,5mA.[23]

Aby prąd płynący przez opornik miał wartość 2,5mA, dobrany rezystor powinien mieć wartość:

$$\frac{1{,}5V}{2{,}5mA} = 600\Omega$$

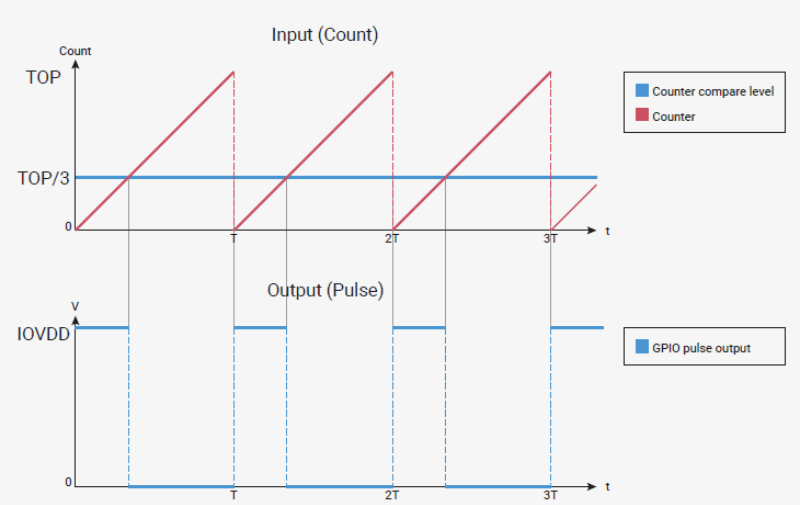
W projekcie zastosowano nieco mniejszy rezystor o wartości 560 Ω, dla którego natężenie prądu wynosi ok. 2,7mA.

6. Program

a. Działanie PWM i obliczenie wartości potrzebnych do inicjalizacji serwomechanizmu

Do sterowania serwomechanizmami należy użyć sygnału PWM. Wg. Noty katalogowej jego okres powinien wynosić ok. 20ms, dla położenia 0° stan wysoki powinien trwać 0,5ms, a dla położenia 180° 2,4ms. Ponieważ odpowiednim położeniom odpowiadają konkretne wartości czasu stanu wysokiego, w celu najprostszego sterowania serwomechanizmem okres 20 ms najlepiej podzielić na 20 000 części.

Funkcji PWM dla użytego mikrokontrolera ustawia się w następujący sposób: najpierw dobiera się wartość, do której mikrokontroler będzie liczył np. 4 i czas w jakim do niej doliczy (okres PWMa) np. 40ms. Następnie ustawia się liczbę porównywaną z wartością licznika, przy osiągnięciu której mikrokontroler zmieni stan konkretnego pinu. Wyjścia PWM są dwukanałowe, więc nastawiając jeden taki zegar można uczynić go wrażliwym na dwie liczby (np. na pierwszym wyjściu wartość ta wynosi 1, a na drugim 2). Po doliczeniu do 1 (czyli 1*40ms/4 = 10 ms) pierwszy pin zmieni stan z wysokiego na niski, a po doliczeniu do dwóch (czyli 2*40ms/4 = 20ms) zmieni się stan drugiego pinu. Wypełnienie w pierwszym przypadku wynosi 1/4, a w drugim 2/4, czyli 1/2.



Rysunek 6.1 Zasada działania PWM [23] w tym przypadku wartością porównywaną z wartością licznika jest liczba TOP/3, co przekłada się na wypełnienie = 1/3

W celu obliczenia wartości potrzebnych do ustawienia odpowiedniej częstotliwości PWMa skorzystano z następującego wzoru z noty katalogowej producenta:

$$f_{PWM} = \frac{f_{sys}}{(TOP + 1) * (CSR_{PH_{CORRECT}} + 1) * (DIV_{INT} + \frac{DIV_{FRAC}}{16})}$$

Gdzie:

f_{PWM} – częstotliwość PWM (50Hz)

TOP – Wartość rejestru TOP, po osiągnięciu której licznik jest zerowany (20 000)

$CSR_{PH_{CORRECT}}$ – Wartość konfiguracyjna wpływająca na sposób zachowania się PWMa po osiągnięciu wartości TOP. W naszym przypadku jest równa 0

DIV_{INT} - 8-bitowa (max. 255) wartość całkowita praskalera

$\frac{DIV_{FRAC}}{16}$ – 4-bitowa (max 15) wartość całkowita podzielona przez 16 w celu osiągnięcia wartości niecałkowitej

Ponieważ znana jest docelowa wartość częstotliwości PWM (1/20ms =50Hz) oraz wartość TOP (20 000) przekształcamy wzór w celu obliczenia wartości preskalera:

$$DIV_{INT} + \frac{DIV_{FRAC}}{16} = \frac{f_{sys}}{f_{PWM} * (TOP + 1)}$$

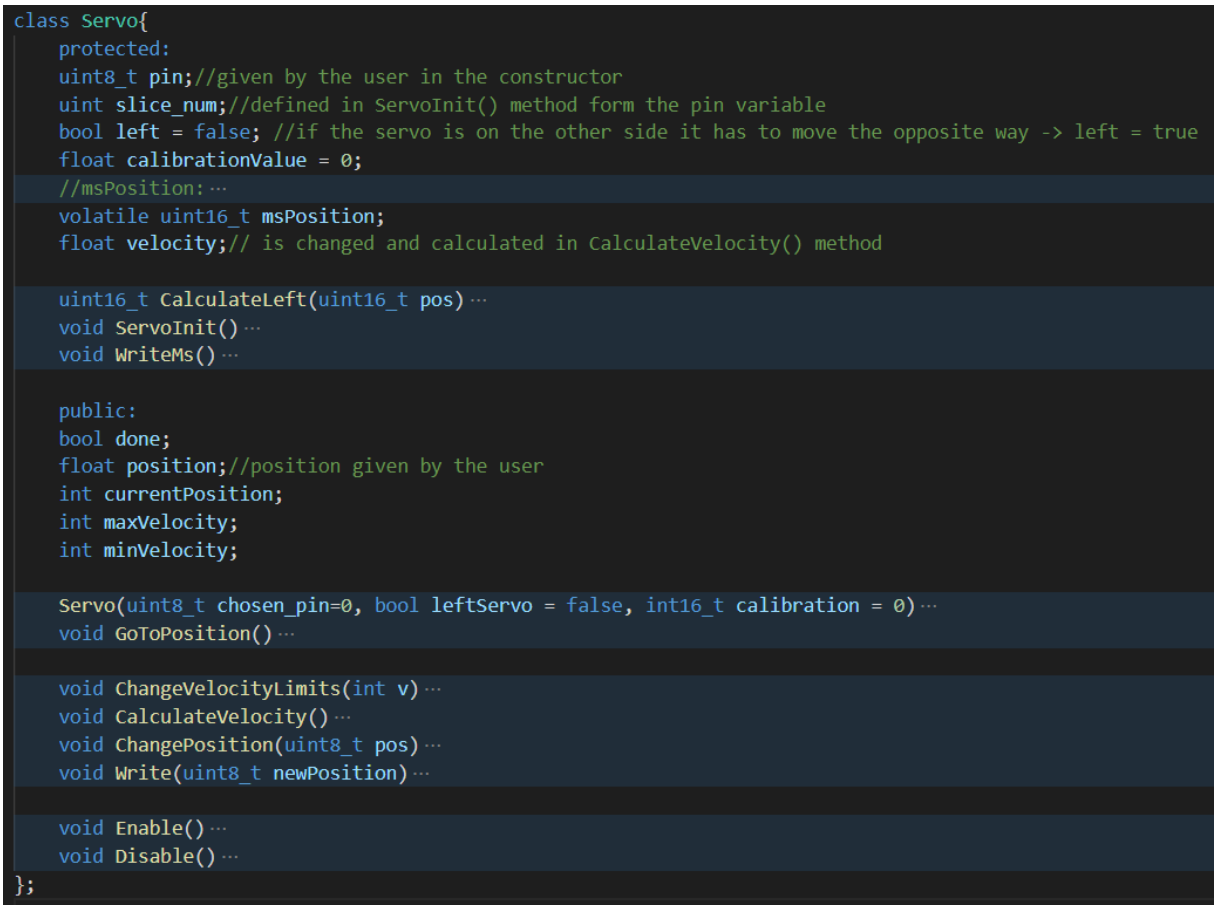
Po podstawieniu odpowiednich wartości otrzymamy:

$$DIV_{INT} + \frac{DIV_{FRAC}}{16} \approx 125{,}0$$

Zatem z obliczeń wynika, że wartość DIV_{INT} wynosi 125, a wartość $\frac{DIV_{FRAC}}{16}$ jest równa 0.

b. Klasa `Servo` i `SlaveServo`

Aby w prosty sposób zmieniać pozycje poszczególnych serwomechanizmów stworzono obiekt `Servo`, który pozwala na prostą kontrolę nad nimi.



```
class Servo{
    protected:
    uint8_t pin;//given by the user in the constructor
    uint slice_num;//defined in ServoInit() method form the pin variable
    bool left = false; //if the servo is on the other side it has to move the opposite way -> left = true
    float calibrationValue = 0;
    //msPosition: ···
    volatile uint16_t msPosition;
    float velocity;// is changed and calculated in CalculateVelocity() method

    uint16_t CalculateLeft(uint16_t pos) ···
    void ServoInit() ···
    void WriteMs() ···

    public:
    bool done;
    float position;//position given by the user
    int currentPosition;
    int maxVelocity;
    int minVelocity;

    Servo(uint8_t chosen_pin=0, bool leftServo = false, int16_t calibration = 0) ···
    void GoToPosition() ···

    void ChangeVelocityLimits(int v) ···
    void CalculateVelocity() ···
    void ChangePosition(uint8_t pos) ···
    void Write(uint8_t newPosition) ···

    void Enable() ···
    void Disable() ···
};
```

Rysunek 6.2Kod przedstawiający klasę Servo

W celu utworzenia nowego obiektu musimy podać numer pinu, do którego jest podłączone serwo, jego stronę (lewa/prawa) oraz wartość kalibracji, którą program użyje do zadawania konkretnych pozycji. Program następnie wykonuje metodę `ServoInit()`, która uruchamia PWM na konkretnym pinie z częstotliwością (około) 20 ms.

Każdy serwomechanizm można uruchomić za pomocą metody `Enable()` i wyłączyć za pomocą funkcji `Disable()`.

Serwomechanizmem steruje się zadając odpowiednie nasycenie PWMa przy częstotliwości ok. 50Hz. W nocie katalogowej jest napisane, że położenie serwomechanizmu można regulować przy zmianie czasu trwania stanu wysokiego na pinie (dla 0° 0,5 ms, a dla 180° 2,4 ms), przy częstotliwości 50Hz (i okresie 20ms). W tym celu została stworzona metoda `WriteMs()`, która zmienia wypełnienie PWMa na konkretnym kanale (wybranym przy deklaracji obiektu) na wartość pola `currentPosition`. W przypadku, kiedy obiekt `Servo` był zadeklarowany jako „lewy” wartość ta jest zmieniana tak, aby była ona symetryczna względem kąta 90° za pomocą

metody `CalculateLeft()`. Dodatkowo do zadawanych wartości dodawana jest wartość `calibrationValue` w celu osiągnięcia dokładnej pozycji serwomechanizmu.

Aby zadać konkretną pozycję w stopniach należy użyć metody `Write()`, do której w argumencie należy podać pozycję w stopniach (między 0° a 180°). Następnie jest ona skalowana między skrajnymi możliwymi wartościami czasu trwania stanu wysokiego wyjścia PWM sterującego serwomechanizmem (0,5ms – 2,400ms) za pomocą funkcji `map()`. Po otrzymaniu wyniku jest on wpisywany do pól `msPosition` oraz `currentPosition` i wywoływana jest funkcja `WriteMs()`;

Poza trybem zadającym serwomechanizmowi konkretną pozycję została również dodana możliwość zadania oczekiwanej pozycji (metoda `ChangePosition()`), a następnie zbliżanie się do niej przy kolejnych uruchomieniach metody `GoToPosition()`.

Pierwsza metoda przyjmuje wartość żądaną w stopniach, przelicza ją na wartość czasu trwania stanu wysokiego i wpisuje ją do pola `msPosition.` Jest to docelowa wartość pozycji. Dodatkowo flaga `done`, która mówi o tym, czy już ruch został wykonany zmienia wartość na `false`.

Metoda `GoToPosition()` kolejno zmienia wartość pola `currentPosition`, które zachowuje aktualną pozycję serwomechanizmu tak, aby zbliżała się ona do pozycji zadanej przechowywanej w polu `msPosition`. Zmiana ta następuje o wartość `velocity`, która jak można się domyślić odpowiada prędkości przemieszczania się serwomechanizmu. Jest ona obliczana przy każdym uruchomieniu metody `GoToPosition()` za pomocą metody `CalculateVelocity()`, która pozwala na przyspieszanie i zwalnianie silnika. Kiedy zmienna `currentPosition` osiągnie wartość oczekiwaną, znaczy to, że ruch jest wykonany i flaga `done` jest zmieniana na wartość `true`.

```
class SlaveServo: public Servo{
    private:
    bool slaveBack = true;
    public :
    bool enableSlave = false;
    SlaveServo(uint8_t chosen_pin=0, bool leftServo = false, int16_t calibration = 0, bool sBack = false)···
    uint8_t Calculate(int pos)···
    void SlavePosition(float pos)···
};
```

Rysunek 6.3 Kod przedstawiający klasę SlaveServo

Klasa `SlaveServo` dziedziczy od klasy `Servo` i zawiera dodatkowe metody takie jak `Calculate()` i `SlavePosition()` oraz pola `slaveBack` i `enableSlave`.

Metoda `Calculate()` wykorzystuje wzory wyprowadzone w rozdziale 3.c do obliczenia pozycji drugiego członu przy podanej pozycji pierwszego serwomechanizmu. Korzysta ona z pola `slaveBack`, do wykorzystania jednego z dwóch możliwych wariantów położenia nogi.

Metoda `SlavePostion` oblicza za pomocą metody `Calculate()` pozycję, którą następnie za pomocą dziedziczonej funkcji `WriteMs()` podaje poprzez PWM na serwomechanizm. Instrukcje te wykonują się tylko, jeżeli ustawiono flagę `enableSlave` na `true`.

c. Klasa `Leg`

```
class Leg
{
    public:
    Servo master;
    SlaveServo slave;
    int maxPos = MASTER_SERVO_MAX_POS;
    int minPos = MASTER_SERVO_MIN_POS;
    int upPos = SLAVE_UP_POSITION;

    void initLeg()···
    public:
    Leg(int pinMaster = 2, int pinSlave = 3, bool leftLeg = false, bool sBack = true, int16_t calibrationMaster = 0,int16_t calibrationSlave = 0)···
    void ChangeLegVelocityLimits(int v)···
    void WriteMaster(int position, bool slaveEnabled)···
    void ChangePosition(uint8_t pos, bool slaveEnabled)···
    void GoToPositionMaster()···
    void ChangePositionSlave(uint8_t pos)···
    void GoToPosition()···

    void ChooseMove(State s, bool enableSlave)···
    bool Move()···
    bool MoveDone()···
    void DisableLeg()···
};
```

Rysunek 6.4 Klasa Leg

Na jedną nogę składają się dwa serwomechanizmy – `master` na górze nogi oraz `slave` między dwoma jej członami. Konstruktor klasy tworzy te obiekty z argumentów podanych przy jego wywołaniu.

Mechanizm sterowania serwomechanizmami jest tu taki sam jak w klasie `Servo`: można zdefiniować konkretne położenie, do którego silniki mają się przesunąć od razu lub oczekiwaną pozycję końcową, do której serwomechanizmy będą dążyć przy kolejnym wywoływaniu odpowiedniej metody (`GoToPosition()` lub `GoToPositionSlave()`). Jedyna różnica polega na tym, że serwomechanizmy są w funkcjach powiązanie, co oznacza, że po zadaniu konkretnej wartości masterowi (serwomechanizmowi na górze) drugi może automatycznie dostosować swoje położenie.

W celu łatwiejszego programowania ruchu stworzono również dodatkowe metody:

`ChooseMove()` zadaje oczekiwane położenie korzystając z jednej z czterech pozycji nogi – przód, tył, góra i dół, przy czym należy również podać w argumencie metody zmienną `enableSlave` odpowiadającą za możliwość dostosowania położenia drugiego członu do pierwszego w celu osiągnięcia stałej wysokości nogi.

Metoda `Move()` wykonuje ruch serwomechanizmów aż do osiągnięcia żądanej pozycji i zwraca wartość `true`, jeżeli ruch został całkowicie wykonany (osiągnięto zadaną pozycję).

Metoda `MoveDone()` sprawdza, czy wszystkie człony osiągnęły żądaną pozycję i jeżeli tak, to zwraca wartość `true`.

d. Klasa `Body` i maszyna stanów odpowiadająca za poruszanie się robota

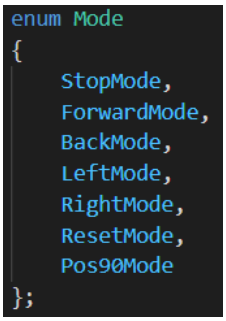
```
class Body
{
    private:
    uint8_t legTeam1[3] = {0,3,4};
    uint8_t legTeam2[3] = {1,2,5};
    State movingStates[ARRAY_SIZE(forwardStates)];
    Leg legs[6];
    void ChangeToForward()···
    void ChangeToBack()···
    void ChangeToLeft()···
    void ChangeToRight()···
    void ChangeToStop()···
    void ChangeToReset()···
    void ChangeToResetTemp()···
    void ChangeTo90()···

    public:
    Mode modeType = StopMode;
    int step = 0;
    bool reset = false;
    Body(uint8_t masterPins[6], uint8_t slavePins[6], const int16_t calibration[12])···
    void ChangeBodyVelocityLimits(int v)···
    void Move()···
    bool MovesDone()···
    void ModeChanged(Mode s)···
    void DisableLegs()···
};
```

Rysunek 6.5 Klasa Body

Klasa Body została stworzona w celu ułatwienia sterowania całym robotem. Składa się na nią 6 obiektów klasy `Leg`, które są przez nią sterowane. Pracują one w zespołach (zgodnie z założeniem z rozdziału 3), które są zdefiniowane w tablicach `legTeam1` i `legTeam2`.

Konstruktor klasy odpowiednio konfiguruje serwomechanizmy dla każdej nogi, a następnie je inicjalizuje (ustawia w pozycji 90°).

Pole `modeType` przechowuje wartość typu enumerowanego `Mode`, która odpowiada obecnemu stanowi robota (rys. 6.4).



Rysunek 6.6 Typ Mode

Tablica `movingStates` zawiera kolejność kroków, które robot powinien wykonać w danym trybie (np. dla ruchu na wprost najpierw noga się podnosi, następnie przemieszcza do przodu, później jest stawiana na ziemi i przesuwana do tyłu). Jest ona używana w metodzie `Move()`, która działa na zasadzie maszyny stanów. Wartość aktualnego miejsca w tablicy jest przechowywana w zmiennej step. Po znalezieniu odpowiedniej wartości (np. `Foreward`) program wykonuje konkretną instrukcję. Przykładowo może być to nadanie nowej pozycji każdej z nóg, a następnie przejście do kolejnego kroku. W kolejnym wywołaniu metody `Move()` program wchodzi do nowego kroku (np. `LoopMaster`). W tym przypadku noga jest zbliżana do zadanej pozycji (za pomocą metody `GoToPosition()`) i dopiero po jej osiągnięciu (metoda `MovesDone()` zwraca wartość `true`) pole step jest inkrementowane (przechodzimy do następnego kroku).

W celu zmiany sposobu poruszania się należałoby zmienić wartości kolejnych kroków w tablicy `movingStates`. W tym celu stworzono metody `ChangeToForeward()`, `ChangeToBack()` itd. Korzystają one z globalnych tablic `forwardStates`, `backStates` itd., które przechowują kolejność kroków dla odpowiadających im trybów. Poza tym metody te zmieniają również zmienną step na odpowiednią wartość oraz mogą na początku nowego ruchu zresetować pozycję robota.

Zmiana trybu jest wykonywana przez metodę `ModeChanged()`, która dla podanego w argumencie trybu wywołuje odpowiednią metodę do zmiany tablicy `movingStates`.

e. Komunikacja za pomocą UART-u

Aby sterować robotem zdalnie komendy przekazywane są za pomocą UARTu.

UART inicjalizowany jest za pomocą funkcji `UART_INIT()`, która korzysta z wcześniej zdefiniowanych prędkości transmisji oraz numerów pinów.

Po otrzymaniu danych jest wywoływana funkcja `OnUartRx()`. Wykonuje się ona w przerwaniu. Po odczytaniu danych wstawia odpowiednią wartość pod zmienną globalną `state` przechowującą tryb, w którym działa robot.

f. Sprawdzenie Napięcia na baterii

Aby zmierzyć napięcie baterii najpierw musi być uruchomiona opcja ADC (ang. *analog to digital converter*) na konkretnym pinie.

Służy do tego funkcja `ADC_INIT(),` która używa wcześniej zdefiniowanych pinów.

Do zmierzenia napięcia baterii służy funkcja `MeasureBattery().` Najpierw włącza ona tranzystor, a następnie sprawdza napięcie na wejściu ADC, po czym wyłącza tranzystor. Do obliczenia konkretnej wartości napięcia używa ona obliczonego granicznego napięcia odczytanego przez mikrokontroler z rozdziału 4, które wynosi 2,238V.

Należy też przeliczyć wartość napięcia na wartość odczytaną przez mikrokontroler na wejściu ADC. Posiadając 12 bitowy przetwornik maksymalna wartość wynosi:

$$1 \ll 12 - 1 = 4095$$

Napięcie odniesienia wynosi 3,3V, w związku z tym cyfrowa wartość graniczna wynosi:

$$ADC = \frac{U_g}{U_{ref}} * 4095$$

Po podstawieniu wartości otrzymujemy 2777.

Program po odczytaniu wartości z przetwornika porównuje wynik pomiaru z wartością graniczną. Jeżeli jest ona mniejsza, to mruga 3 razy czerwoną diodą i zmienia globalną zmienną `enableProgram` na wartość `false`, a następnie funkcja zwraca

wartość `false`. W przeciwnym wypadku program zmienia wartość zmiennej `enableProgram` na `true` i również zwraca tę wartość.

g. Funkcja `main`

Na początku inicjalizowane są używane przez program piny od diod, uart i przetwornik analogowo-cyfrowy.

Następnie mierzone jest napięcie baterii i jeżeli mieści się w dobrym zakresie tworzony jest obiekt `body`, gdzie argument konstruktora podawane są odpowiednie piny. Wywoływana jest jeszcze metoda `ModeChanged()`, gdzie jako argument jest podawana wartość zmiennej globalnej `mode`, która ma wartość `POS_90_MODE`.

W nieskończonej pętli `while` są wykonywane następujące kroki:

Sprawdzenie napięcia baterii, jeżeli jest ono za niskie wszystkie serwomechanizmy są dezaktywowane i program zatrzymuje się na 10s.

Jeżeli napięcie było odpowiednie wykonywana jest następująca sekwencja:

- Sprawdzenie, czy przez aplikacją mobilną została zadana nowa prędkość lub tryb działania robota. Jeżeli nastąpiła zmiana wywoływane są odpowiednie metody na obiekcie body w celu zaktualizowania wartości używanych przez obiekt.
- Zapalenie kontrolnej diody na płytce raspberry pi pico.
- Wykonanie jednego ruchu – w pętli wywoływana jest metoda `Move()` na obiekcie `body` aż do osiągnięcia docelowej pozycji zdefiniowanej w wykonywanym kroku. Po każdym wykonaniu metody program jest zatrzymywany na 20ms, aby ruch miał odpowiednią dynamikę
- Zgaszenie diody kontrolnej

Zatrzymanie programu na 100ms???

h. Kalibracja nóg

Samo założenie serwomechanizmów nie gwarantuje niestety takiego samego położenia wszystkich serwomechanizmów (idealnych 90°). Wał przenosi moment siły na

orczyk za pomocą połączenia kształtowego, przez co przy zakładaniu orczyka możemy wybrać tylko dyskretne wartości pozycji najbliżej tej pożądanej, a nie dowolną.

W związku z tym problemem postanowiono odpowiednio skalibrować serwomechanizmy w programie. W ten sposób po dobraniu odpowiednich wartości dla każdego silnika istnieje możliwość zwiększenia dokładności pozycji.

Do kalibracji nóg użyto klasy `Servo` z głównego programu i trochę zmodyfikowanej wersji aplikacji. Po kliknięciu przycisków prawo/lewo następuje inkrementacja/dekrementacja o wartość 0,005ms, a po kliknięciu przycisków góra/dół zmiana następuje o 0,001ms czasu trwania stanu wysokiego. Te zmiany czasu trwania stanu wysokiego odpowiadają wartościom 0,5° i 0,1°.

Po sprawdzeniu o jakie wartości należy skalibrować każdy napęd, wartości zostały wpisane do głównego programu sterującego ruchem robota.

7. Komunikacja

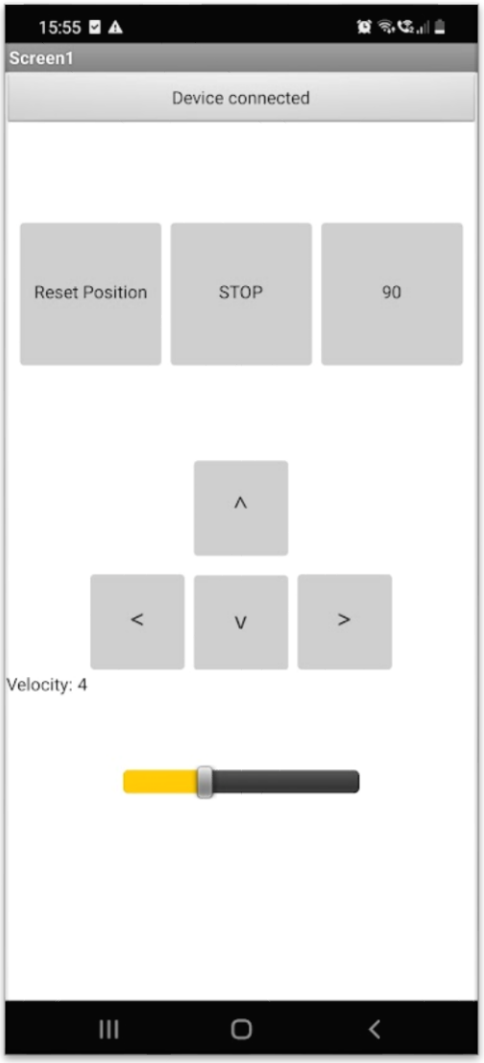
a. Aplikacja na urządzenie mobilne

W celu zdalnego kontrolowania robota stworzono aplikację za pomocą narzędzia MIT App Inventor. Pozwala ono na tworzenie aplikacji za pomocą prostych komend i zamiast np. konfiguracji całego połączenia przez bluetooth wystarczy użyć odpowiedniej komendy do wykonania tego zadania.

Najpierw należało zdefiniować zadania aplikacji:

- Połączenie z modułem bluetooth
- Po wciśnięciu odpowiedniego przycisku (np. strzałka do przodu) aplikacja wysyła przez bluetooth odpowiedni znak
- Po zmianie prędkości wysyłana jest odpowiednia sekwencja znaków. Następnie utworzono widok użytkownika z wszystkimi potrzebnymi przyciskami

Po zdefiniowaniu zadań należało stworzyć odpowiedni interfejs użytkownika, a następnie wystarczyło wybrać odpowiednie komendy w celu zaprogramowania aplikacji.



Rysunek 7.1 Widok użytkownika

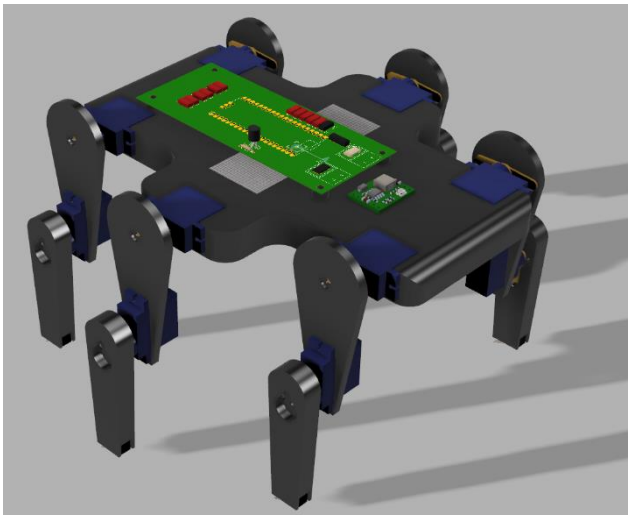
b. Połączenie z robotem

Po przesłaniu przez bluetooth danych z aplikacji, moduł HC-06 odbiera je, a następnie przesyła za pomocą UARTu do mikrokontrolera. W celu odbioru danych wywołuje się funkcja `OnUartRx()`, która zmienia wartości odpowiednich zmiennych w celu zmiany parametrów pracy robota.

8. Podsumowanie

Na początku projektu zdecydowano się na mikrokontroler raspberry pi pico ze względu na jego parametry takie jak np. duża liczba kanałów PWM. Dodatkowo jest to nowy mikrokontroler (z początku 2021r.) o wielu innych funkcjach takich jak np. możliwość obsługi dwóch rdzeni, więc była to ciekawa propozycja do użycia w projekcie. Producent zadbał o bardzo dobrą dokumentację, a społeczność użytkowników szybko zaczęła tworzyć poradniki, w związku z czym rozpoczęcie projektu z tą płytką, mimo że było bardziej skomplikowane niż np. praca z arduino, nie niosło ze sobą problemów, których rozwiązania byłoby trudno znaleźć.

Przy projektowaniu robota parokrotnie doszło do zmian jego koncepcji, szczególnie mechanicznej jego części.



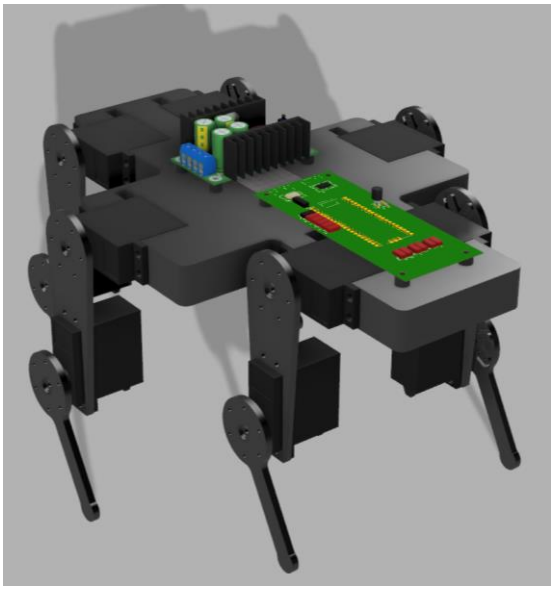
Rysunek 8.1 Pierwsza konstrukcja robota

Pierwsza wersja opierała się o serwomechanizmy sg90 i mimo że robot dał radę chodzić, jeżeli jego kroki nie były zbyt duże, to silniki miały bardzo mały zapas mocy. Dodatkowo cała konstrukcja była mało sztywna, co również było jej wadą. Na końcach nóg początkowo przewidziane były krańcówki w celu sygnalizowania kontaktu nogi z ziemią. Należałoby wtedy jednak skonstruować stopę robota, która zawsze dostosowywałaby się do podłoża, żeby krańcówka była w poprawnym położeniu i stykając się z ziemią zawsze się załączała.

Zmieniono więc serwomechanizmy na model MG995. Są one w stanie działać z dużo wyższym momentem siły (13kg*cm zamiast 1,8kg*cm w przypadku sg90). Dzięki temu można było zwiększyć długość nogi ze 100mm do 140mm. Zwiększono również korpus robota.[8] [29]

Napędy te są jednak dużo cięższe. Każdy waży 55g (dla porównania sg90 waży 9g). Wymagają one również większego natężenia prądu do zasilania, przez co przetwornica i bateria, które były odpowiednie dla pierwszej wersji robota musiały zostać

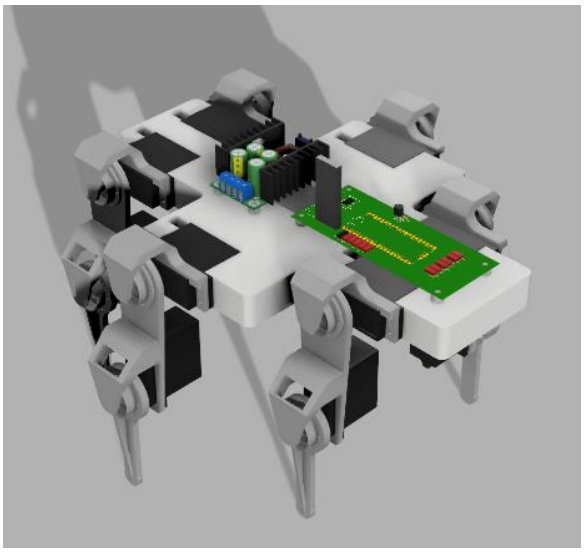
na nowo dobrane. Sam układ sterowania i płytka mogły pozostać bez zmian, ponieważ nowe serwomechanizmy steruje się tak samo jak te poprzednio użyte. Wszystkie te czynniki złożyły się na zwiększenie masy robota z ok. 300g do nieco ponad kilograma (głównie przyczyniła się do tego zwiększenie łącznej masy silników o ok. 550g).[8] [29]



Rysunek 8.2 Druga wersja robota

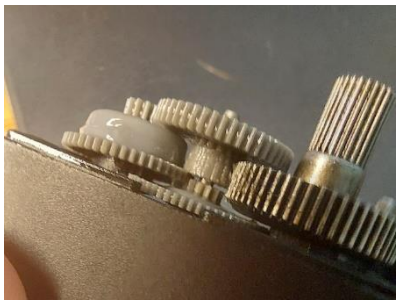
Nowa konstrukcja pozwoliła na większą dynamikę, a luzy przy wałach silników były minimalne. Cały czas istniał jeszcze jednak problem ze sztywnością, w związku z czym zdecydowano się trzeci raz usprawnić konstrukcję mechaniczną robota.

W tym celu zastosowano łożyskowanie w miejscach stawów robota, aby odciążyć wały silnika. Dodano również żebro na dolnym członie nogi w celu jej usztywnienia. Końcowa konstrukcja waży ok. 1,2 kg.



Rysunek 8.3 Zdjęcie modelu ostatecznej wersji robota

Zastosowane serwomechanizmy posiadają co prawda na wyjściu metalowy wał, jednak przekładnia jest wykonana z plastiku, co pozwoliło prawdopodobnie obniżyć cenę napędu. Niestety przy większych obciążeniach plastikowe zęby nie wytrzymują kontaktu z metalowymi i ulegają zniszczeniu (rys.8.4).



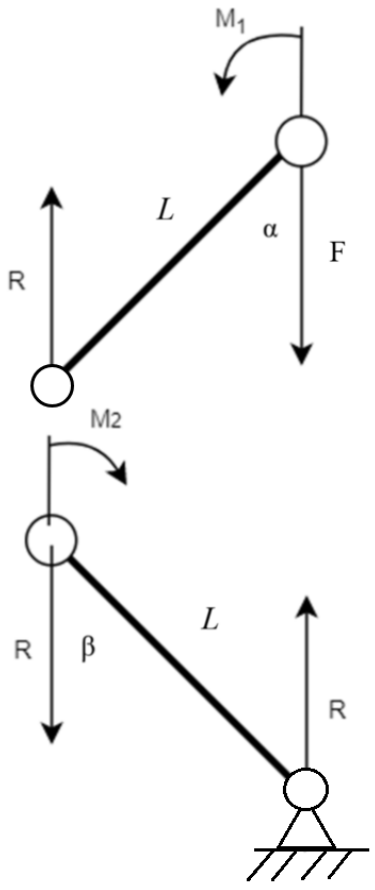
Rysunek 8.4 Zdjęcie zniszczonej przekładni

Na cenę konstrukcji złożyło się głównie 12 serwomechanizmów MG995 (łącznie ok. 240zł z plastikową przekładnią lub ok. 360zł z metalową). Poza tym użyty mikrokontroler, moduł bluetooth i przetwornica kosztowały po ok. 20 zł, bateria 30 zł, a łączny koszt filamentu użytego do wydruku konstrukcji wyniósł ok. 20 zł. Elementy mechaniczne takie jak łożyska i śruby kosztowały łącznie ok. 20 zł. Podstawowe części elektroniczne były bardzo tanie (maksymalnie 10 zł łącznie), w związku z czym w sumie koszt budowy robota wynosi w zależności od zastosowanej przekładni od 380zł do 500zł.

Można wykonać obliczenia jaki teoretyczny ciężar jest w stanie przenieść robot. Najlepiej byłoby wykonać testy użytych serwomechanizmów, ponieważ dane podane przez producenta nie zawsze są zgodne z rzeczywistością. Pomiary wykonane przez

Roberta Smorgasborda wskazały, że użyte przez niego serwomechanizmy miały prawie dwukrotnie mniejszy moment niż deklarowany.[30] Z jego badań wynikało, że przy napięciu zasilania 4,8 V moment generowany na wale wyjściowym wynosi 5,5 kg*cm, czyli ok. 55 N*cm.

W obliczeniach przyjęto mocno uproszczony model zakładając nieważkość i brak odkształceń nóg. Wzięto pod uwagę tylko siły działające w pionie (tym samym kierunku co siła grawitacji), a w miejscu styku nogi z ziemią uznano, że jest ona utwierdzona za pomocą podpory przegubowej nieprzesuwnej (dla której siła reakcji w osi poziomej wynoszą 0). Dodatkowo są to obliczenia statyczne, a nie dynamiczne w związku z czym nie dotyczą sytuacji w której robot się porusza. Pozwalają one jednak mimo tych uproszczeń ukazać rząd wielkości obciążenia, które robot jest w stanie unieść.



Rysunek 8.5 Siły i momenty działające na człony nóg

$$R = F$$

$$M_1 \geq R * sin\alpha * L$$

$$M_2 \geq R * sin\beta * L$$

Ponieważ największe obciążenia pojawiają się przy największych kątach (sinus w przedziale <0°,90°> rośnie) do obliczeń wybrano kąt 45°, który jest maksymalnym kątem zadawanym w programie. Maksymalne momenty wyjściowe serwomechanizmów (M_1 i M_2) są takie same i przyjęto wartość wyznaczoną doświadczalnie równą 550 N*mm.[30] Silniki mają za zadanie utrzymać nogę w określonej pozycji więc w ramach możliwości działają z momentem utrzymującym układ w równowadze. Długość każdego członu nogi wynosi 70 mm, w związku z czym:

$$F_{max} = R_{max} = \frac{M}{sin\alpha * L} = \frac{550}{\sin 45° * 70} \approx 11N$$

Ponieważ robot stoi co najmniej na 3 nogach (jak przedstawiono w rozdziale 3), a ciężar jest na nie równo rozłożony, mnożymy otrzymany wynik razy 3 i otrzymujemy wartość 33N. Jest to maksymalny ciężar robota z obciążeniem, przy czym sama maszyna waży ok. 1,2kg, w związku z czym jest w stanie teoretycznie przenieść masę około 2 kg. Należy jednak pamiętać, że ważne jest odpowiednie rozłożenie masy w celu zachowania przez robota równowagi oraz równomiernego obciążenia silników. Obliczone wartości sił są teoretycznie i nie biorą pod uwagę wszystkich czynników, w związku z czym może się okazać, że w rzeczywistości jest on w stanie przenieść mniejszą masę. Obliczenia pokazują jednak, że zamontowanie kamery, czy manipulatora przenoszącego lekkie przedmioty, jeżeli tylko nie wpłyną one silnie na zmianę środka masy, nie będą stanowić za dużego obciążenia dla robota. Jeżeli zaistniałaby potrzeba zainstalowania na robocie większego ładunku (2-3kg) mógłby on również dać radę, jednak należałoby zmienić algorytm chodu z trójpodporowego na np. metachroniczny. Poprawiłoby to również stabilność maszyny, co może być szczególnie ważne, jeżeli dodane obciążenie zmieniłoby położenie środka ciężkości.

Stworzona konstrukcja jest platformą, do której można by dodać kamerę, czy manipulator w celu dostosowania jej do konkretnego zadania. Nie ma ona zbyt wielkich rozmiarów, przez co robot mógłby wchodzić w małe, trudno dostępne miejsca. Może on dosyć długi czas pracować na baterii (ponad godzinę), mimo że jej pojemność wynosi tylko 800 mAh. W przypadku upadków z wysokości niestety może on przestać poprawnie działać ze względu na plastikową przekładnię wewnątrz użytych serwomechanizmów, która przy większych obciążeniach może ulec uszkodzeniu.

Literatura

[1] T. Zielińska, *Maszyny kroczące*, 2. wyd. Warszawa: Wydawnictwo naukowe PWN, 2014.

[2] Morecki A. i Knapczk J., *Basics of Robotics*. Wien: Springer Verlag, 1999.

[3] Zuk, „Roboty kroczące - teoria i podstawy projektowania”, *FORBOT*, 7 sierpień 2009. https://forbot.pl/blog/roboty-kroczace-teoria-podstawy-projektowania-id976 (dostęp 14 styczeń 2022).

[4] „Eadweard Muybridge – Wikipedia, wolna encyklopedia”. https://pl.wikipedia.org/wiki/Eadweard_Muybridge (dostęp 16 styczeń 2022).

[5] „Spot® - The Agile Mobile Robot”, *Boston Dynamics*. https://www.bostondynamics.com/products/spot (dostęp 23 styczeń 2022).

[6] „Sloan Science & Film”. http://www.scienceandfilm.org/articles/3044/the-dog-in-black-mirrors-metalhead (dostęp 23 styczeń 2022).

[7] Tomasz „Motylasty” Motyl, „Serwomechanizmy - podstawy teorii na temat serwy - sklep Modelmotor”. https://www.modelmotor.pl/webpage/serwomechanizmy-czesc-i.html (dostęp 17 styczeń 2022).

[8] „Nota katalogowa serwomechanizmu MG995”.

[9] J. Burton, „Mini Robot Dog - YouTube”. https://www.youtube.com/playlist?list=PLpwJoq86vov_qzmadB1zj6aVKObH3WbeH (dostęp 17 styczeń 2022).

[10] Ł. Antczak, „How to choose the proper actuator for a legged robot?”, *Medium*, 23 kwiecień 2021. https://mab-robotics.medium.com/how-to-choose-the-proper-actuator-for-a-legged-robot-664e341ea308 (dostęp 17 styczeń 2022).

[11] Ł. Antczak, „Legged robots: Keeping legs moving”, *Medium*, 16 grudzień 2021. https://mab-robotics.medium.com/legged-robots-keeping-legs-moving-f2ea10f9b5f0 (dostęp 14 styczeń 2022).

[12] *Series Elastic Actuators*. Dostęp: 18 styczeń 2022. [Online]. Dostępne na: https://www.youtube.com/watch?v=gZLO2Am0Zk8

[13] „HyQReal - Dynamic Legged Systems - IIT”. https://www.iit.it/web/dynamic-legged-systems/hyqreal (dostęp 18 styczeń 2022).

[14] M. Piątek, „Problemy sterowania robotami kroczącymi - generowanie chodu hexapoda”. [Online]. Dostępne na: https://winntbg.bg.agh.edu.pl/rozprawy2/10537/full10537.pdf

[15] „DC to DC Buck Converter 8A XL4016E1 | 3D CAD Model Library | GrabCAD”. https://grabcad.com/library/dc-to-dc-buck-converter-8a-xl4016e1-1 (dostęp 18 styczeń 2022).

[16] „Servo MG995 | 3D CAD Model Library | GrabCAD”. https://grabcad.com/library/servo-mg995-3 (dostęp 21 styczeń 2022).

[17] „Component Search Engine: Free Symbols, footprints, & 3D models”. https://componentsearchengine.com/ (dostęp 22 styczeń 2022).

[18] „Raspberry Pi Pico - RP2040 ARM Cortex M0+ Botland - Sklep dla robotyków”. https://botland.com.pl/moduly-i-zestawy-do-raspberry-pi-pico/18767-raspberry-pi-pico-rp2040-arm-cortex-m0-0617588405587.html (dostęp 18 styczeń 2022).

[19] „Moduł Bluetooth HC-06 ZS-040”, *BOTLAND*. https://botland.com.pl/moduly-bluetooth/6818-modul-bluetooth-hc-06-zs-040.html (dostęp 18 styczeń 2022).

[20] S. A.-R. na bazie IdoSell (www.idosell.com/shop), „Przetwornica obniżająca napięcie 300W 1,2V-32V 9A - XL4016 - STEP-DOWN”, *Sklep ABC-RC*. https://abc-rc.pl/product-pol-8578-Przetwornica-obnizajaca-napiecie-300W-1-2V-32V-9A-XL4016-STEP-DOWN.html (dostęp 18 styczeń 2022).

[21] „Pakiet Li-Pol Dualsky 800mAh 25C 2S 7.4V ECO-S”, *BOTLAND*. https://botland.com.pl/akumulatory-li-pol-2s-74v-/2393-pakiet-li-pol-dualsky-800mah-25c-2s-74v-eco-s-6941047104655.html (dostęp 18 styczeń 2022).

[22] KD93, „Akumulatory litowo-polimerowe, Li-po - kompendium cz.1”, *FORBOT*, 21 styczeń 2012. https://forbot.pl/blog/akumulatory-litowo-polimerowe-li-po-kompendium-cz-1-id291 (dostęp 14 styczeń 2022).

[23] „Nota katalogowa mikrokontrolera RP2040”.
[24] „Nota katalogowa stabilizatora liniowego MIC5365 -3.3YC5-TR”.
[25] KD93, „Akumulatory litowo-polimerowe, Li-po - kompendium cz.2”, *FORBOT*, 21 styczeń 2012. https://forbot.pl/blog/akumulatory-litowo-polimerowe-li-po-kompendium-cz-2-id417 (dostęp 14 styczeń 2022).
[26] D. Power, „Complete Guide for Lithium Polymer(Lipo) Battery: History, Charging,Safety, Storage and Care”, *Lithium ion Battery Manufacturer and Supplier in China-DNK Power*, 9 styczeń 2019. https://www.dnkpower.com/lithium-polymer-battery-guide/ (dostęp 18 styczeń 2022).
[27] „Nota katalogowa tranzystora BC547B”.
[28] „Nota katalogowa czerwonej diody 3mm”.
[29] „Nota katalogowa serwomechanizmu sg90”.
[30] „Tower Pro MG995 RC Servo: Full Size, 180°, Metal Gear and Cheap – Review & Test - YouTube”. https://www.youtube.com/watch?v=rgkJVHD-ROg&t=696s (dostęp 22 styczeń 2022).