POLITECHNIKA WARSZAWSKA

Wydział Mechatroniki

Praca dyplomowa inżynierska

Jan Czarnowski
Jan Fraś

Opracowanie, budowa i oprogramowanie sześcionożnego robota kroczącego

Opiekun pracy:
prof. nzw. dr hab. Barbara Siemiątkowska

Instytut Automatyki i Robotyki

Warszawa 2013

Kierunek: Automatyka i Robotyka

Specjalność: Robotyka

Data urodzenia: 26 grudnia 1990

Data rozpoczęcia studiów: 1 października 2009

Urodziłem się 26 grudnia 1990 roku w Norymberdze. Po ukończeniu szkoły podstawowej kontynuowałem naukę w Publicznym Gimnazjum nr 3 im. Stanisława Staszica w Starogardzie Gdańskim, a następnie w I Liceum Ogólnokształcącym im. Marii Skłodowskiej-Curie w Starogardzie Gdańskim, w klasie o profilu matematyczno - fizycznym. W październiku 2009 rozpocząłem studia dzienne na Wydziale Mechatroniki Politechniki Warszawskiej. Po drugim roku nauki na tym wydziale wybrałem kierunek Automatyka i Robotyka, specjalność robotyka. W 2011 roku równolegle podjąłem naukę na Wydziale Matematyki i Nauk Informacyjnych Politechniki Warszawskiej, na kierunku Informatyka. Latem 2012 roku odbyłem praktyki w Przemysłowym Instytucie Automatyki i Pomiarów, gdzie później przyjęto mnie na staż, w ciągu którego stworzony został robot przedstawiony w tej pracy.

…....
podpis studenta

Kierunek: Automatyka i Robotyka

Specjalność: Robotyka

Data urodzenia: 21 września 1990

Data rozpoczęcia studiów: 1 października 2009

Ukończyłem Szkołę Podstawową i Gimnazjum im. J. H. Dąbrowskiego w Słupi Wielkiej oraz Liceum Ogólnokształcące Św. Marii Magdaleny w Poznaniu. W 2009 roku rozpocząłem studia na Wydziale Mechatroniki Politechniki Warszawskiej. W 2010 roku podjąłem drugi kierunek studiów na Wydziale Fizyki Uniwersytetu Warszawskiego, który w 2011 roku przerwałem na rzecz kierunku Informatyka na Wydziale Matematyki i Nauk Informacyjnych Politechniki Warszawskiej. Od roku 2002 do roku 2008 wyczynowo uprawiałem lekkoatletykę. W latach 2004 – 2006 zajmowałem się modelarstwem lotniczym a w 2010 roku wziąłem udział w zawodach zespołowych SAE Aero Design w Stanach Zjednoczonych. W roku 2012 odbyłem praktyki oraz staż w Przemysłowym Instytucie Automatyki i Pomiarów, efektem których jest prezentowany w niniejszej pracy robot.

…....
podpis studenta

Prowadząca pracę: prof. nzw. dr hab. Barbara SIEMIĄTKOWSKA
Konsultant pracy: mgr inż. Jakub BARTOSZEK,
Przemysłowy Instytut Automatyki i Pomiarów

Jan Czarnowski
Jan Fraś

Praca dyplomowa

„Opracowanie, budowa i oprogramowanie sześcionożnego robota kroczącego”

Założenia pracy:

a. zastosowanie płytki systemowej PandaBoard,
b. zastosowanie mikrokontrolera z rodziny STM32,
c. zastosowanie serwomechanizmów elektrycznych,
d. wykorzystanie systemu operacyjnego UNIX,
e. wykorzystanie pakietu do programowania robotów ROS,

Zakres pracy:

1. badanie i porównanie różnych metod ruchu,
2. opracowanie konstrukcji platformy,
3. opracowanie algorytmu ruchu,
4. oprogramowanie mikrokontrolera,
5. opis systemu sterowania w robocie,
6. instalacja i konfiguracja systemu operacyjnego Linux na systemie wbudowanym,
7. kompilacja i instalacja pakietu ROS na płytkę systemową o architekturze ARM,
8. napisanie programów wysokiego poziomu sterujących robotem,
9. opracowanie wniosków końcowych.

Dyrektor instytutu
prof. dr hab. inż. Jan Maciej Kościelny

STRESZCZENIE

W niniejszej pracy przedstawiona została konstrukcja i oprogramowanie sześcionożnego robota kroczącego, który porusza się autonomicznie lub może być sterowany bezprzewodowo. Celem projektu jest opracowanie robota kroczącego, który potrafi sprawnie przemieszczać się w różnych typach terenu. W tym celu przeanalizowane zostały modele ruchu zwierząt pod kątem zastosowania w robocie mobilnym. Do wybranego modelu ruchu opracowano kinematykę oraz odpowiadającą jej konstrukcję. Przedstawiono również system sterowania, polegający na skomunikowaniu trzech urządzeń – mikrokontrolera ARM, systemu wbudowanego PandaBoard oraz komputera PC. Aby to osiągnąć, urządzenia te musiały zostać przystosowane i odpowiednio oprogramowane, co również stanowi przedmiot rozważań tej pracy.

ABSTRACT

This paper presents construction and programming of a hexapod robot, which can be autonomic or controlled wirelessly. The aim of this project is to develop a machine which can easily move through various types of challenging terrain. To achieve this, different models of animal walk have been analysed for suitability to use in mobile robotics. Kinematics and corresponding schematics were developed. The robots control system, which consists of communicating three devices together: an ARM microcontroller, PandaBoard system on chip (SoC) and PC class computer is also presented. All these devices had to be customised and adequatly programmed, what is also an object of this papers study.

Spis treści

1. Wstęp....7

2. Porównanie różnych metod ruchu....9

2.1. Wprowadzenie....9

2.2. Roboty jeżdżące....10

2.2.1.Koła....10

2.2.2 Gąsienice....11

2.3. Chód....11

2.3.1 Podział chodu ze względu na liczbę nóg....11

2.3.2. Podział chodu ze względu na ułożenie nóg....12

2.3.3 Podział chodu ze względu na dynamikę....12

3. Konstrukcja robota....14

3.1. Założenia ogólne....14

3.2. Przykładowe mechanizmy ruchu....14

3.3. Proponowana kinematyka....14

3.4. Proste i odwrotne zagadnienie kinematyki nogi....15

3.5. Model nogi....18

3.6. Dobór napędów:....19

3.6.1. Siły reakcji podłoża oraz momenty obciążające serwomechanizmy....20

3.6.2. Środek masy....21

3.6.3. Siły reakcji....22

3.7. Wyniki obliczeń....24

4. Opracowanie algorytmów ruchu....25

4.1. Realizacja ruchu prostoliniowego....25

4.2. Realizacja ruchu prostoliniowego w dowolnym kierunku:....29

4.3. Realizacja ruchu obrotowego....30

4.4. Realizacja ruchu w dowolnym kierunku po dowolnej krzywej z jednoczesnym obrotem robota....31

4.5. Schemat kroków....34

4.6. Implementacja w robocie....35

5. Oprogramowanie mikrokontrolera....39

5.1. Opis urządzenia....39

5.2. System sterowania nogami....40

5.3. Komunikacja....41

5.4. Sterowanie serwami....42
5.5. Program sterujący....43
5.5.1. Struktura programu....43
5.5.2 Działanie programu....44
6. System sterowania....45
6.1. Wprowadzenie....45
6.2. System sterowania zastosowany w robocie....46
6.2.1. Opis systemu....46
6.2.2. Zastosowane urządzenia....47
6.2.3. Komunikacja pomiędzy PC a komputerem pokładowym....47
6.2.4. Komunikacja pomiędzy komputerem pokładowym a mikrokontrolerem....48
6.3. Podsumowanie....50
7. Konfiguracja komputera pokładowego....51
7.1. Wprowadzenie....51
7.2. Urządzenie zastosowane w pracy....52
7.3. Wybór systemu operacyjnego....52
7.4. Instalacja systemu operacyjnego....53
7.5. Konfiguracja systemu operacyjnego....54
7.5.1. Konfiguracja środowiska pracy....54
7.5.2. Sterowniki do układu OMAP oraz procesora graficznego....54
7.5.4. Przesyłanie obrazu z kamery....55
8. Platforma ROS....58
8.1. Platformy programistyczne....58
8.2. Działanie systemu ROS....59
8.3. Instalacja platformy ROS na komputerze pokładowym....61
8.4. Podsumowanie....61
9. Programy sterujące....62
9.1. Wprowadzenie....62
9.2. Koncepcja programu....63
9.3. Problem współdzielenia kamery....64
9.4. Sterowanie zdalne....65
9.5. Sterowanie autonomiczne....67
9.6. Podsumowanie....73
10. Podsumowanie....74
Bibliografia....76

Rozdział 1

Wstęp

Robotyka mobilna wraz z rozwojem cieszy się coraz większym zainteresowaniem na świecie. Niestety, jest to wciąż dziedzina słabo rozwinięta. Tempo i stopień zaawansowania drugiej gałęzi robotyki, robotyki przemysłowej, jest wyższy, ponieważ przynosi ona na chwilę obecną większe korzyści. Manipulatory stosowane w zakładach przemysłowych pozwalają na zwiększenie efektywności produkcji, a systemy wizyjne umożliwiające sortowanie wyrobów zwiększają ich jakość. Z tego powodu zainteresowanie firm badaniami nad tą gałęzią jest duże. Naukowcy i inżynierowie, którzy zajmują się robotami mobilnymi stawiani są przed problemami dużo bardziej złożonymi, co powoduje mniej dynamiczny rozwój tej dziedziny.

Robotyka mobilna bada możliwość zastosowania autonomicznych lub sterowanych robotów zdolnych do samodzielnego przemieszczania się. Użyteczność robotów jest zazwyczaj silnie ograniczona przez ich zdolność do pokonywania trudnego terenu. Obecny stan technologii (wizji maszynowej, budowy chwytaków, manipulatorów) pozwala na wyręczanie człowieka w żmudnych lub niebezpiecznych pracach, lecz tylko w stacjonarny sposób. Problem stanowi dostarczenie opracowanych usług do użytkownika lub do określonego miejsca. Przykładem może być robot przenoszący ciężkie ładunki, który ma podążać za człowiekiem, lub robot patrolujący granice państw lub posiadłości, przemieszczający się po wyznaczonej trasie. Z tego powodu odnajdywanie rozwiązań pozwalających robotom mobilnym na sprawniejsze poruszanie się w trudnym terenie znacznie zwiększa ich użyteczność i może doprowadzić do ich przejścia do użytku codziennego.

Celem autorów tej pracy jest opracowanie działającego robota mobilnego, który jest w stanie poruszać się sprawnie w trudnym terenie oraz pokonywać przeszkody takie jak schody. W trakcie realizacji tego celu ma powstać trzystopniowy system sterowania, który jest na tyle uniwersalny, że można go zastosować w dowolnym robocie mobilnym. Badania przeprowadzone w ramach niniejszej pracy mogą zostać wykorzystane do stworzenia robota znajdującego zastosowanie w realnych scenariuszach, takich jak poszukiwanie osób

zaginionych w gruzowiskach po trzęsieniu ziemi lub penetracji trudno dostępnych miejsc.

Jednym z założeń dotyczących budowanego robota jest możliwość sterowania nim bezprzewodowo, za pomocą kontrolera do gier podłączonego do komputera klasy PC. Robot ma również posiadać prosty tryb autonomiczny, w którym porusza się samodzielnie. Zamontowanie kamery pozwala na osiągnięcie tego celu za pomocą wizji maszynowej. Ponadto możliwe jest wtedy przesyłanie obrazu do sterującego komputera. Sterowanie robota ma w przyszłości również zawierać sprzężenie zwrotne od czujników, które umożliwi inteligentne poruszanie się robota oraz utrzymywanie jego korpusu pod określonym przez użytkownika kątem.

Niniejsza praca została napisana przez dwóch autorów. Podział prac wygląda następująco:

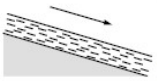
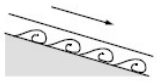
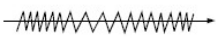
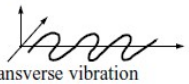
Wstęp i podsumowanie	Jan Czarnowski, Jan Fraś
Rozdziały 2 – 5	Jan Fraś
Rozdziały 6 – 9	Jan Czarnowski

Rozdział 2

Porównanie różnych metod ruchu

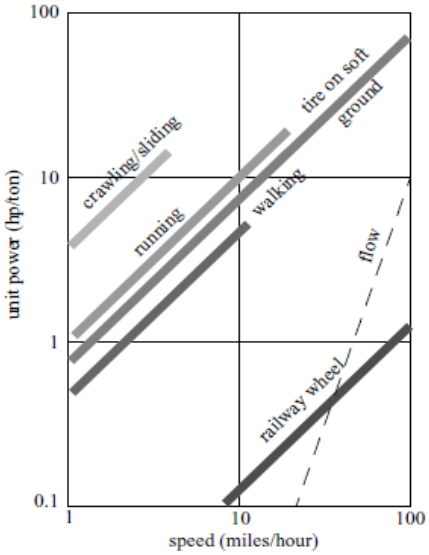
2.1. Wprowadzenie

W robotyce mobilnej najbardziej podstawowym problemem jest lokomocja. Istnieje wiele sposobów poruszania się i wybór jednego z nich jest istotnym czynnikiem determinującym możliwość późniejszego zastosowania konstrukcji oraz podstawowym ograniczeniem pod kątem obszaru jej działania. W laboratoriach na całym świecie rozwijane są roboty kroczące, skaczące, biegające, pełzające, jeżdżące, pływające oraz latające. Większość z przytoczonych tutaj metod ruchu zaczerpnięta została z obserwacji przyrody. Wyjątek stanowią maszyny jeżdżące – nie istnieje biologiczny odpowiednik tego sposoby realizacji ruchu. Żadne stworzenie nie wyewoluowało w sposób pozwalający na realizację połączenia obrotowego potrzebnego do toczenia. Poruszanie się na kołach jest wyjątkowo efektywne, i sprawdza się na płaskich powierzchniach. Pomimo braku kół w przyrodzie, ten sposób poruszania się nie jest całkowicie obcy naturze. Warto zauważyć, że ruch człowieka można przybliżać toczeniem wielokąta o długości boku równej długości kroku.

Type of motion	Resistance to motion	Basic kinematics of motion
Flow in a Channel 	Hydrodynamic forces	Eddies 
Crawl 	Friction forces	Longitudinal vibration 
Sliding 	Friction forces	Transverse vibration 
Running 	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum 
Jumping 	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum 
Walking 	Gravitational forces	Rolling of a polygon (see figure 2.2) 

Rys. 1: Zestawienie sposobów lokomocji spotykanych w przyrodzie[6]

Wraz ze zmniejszaniem długości kroku wielokąt dąży do okręgu.[6] Na rysunku 2 przedstawione zostało zestawienie różnych typów lokomocji pod kątem wydajności energetycznej w zależności od prędkości.

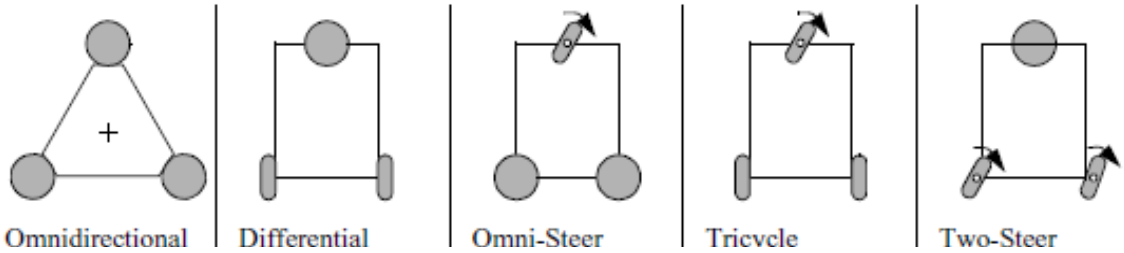


Rys. 2: Zapotrzebowanie na moc w zależności od prędkości dla różnych metod ruchu.[6]

2.2. Roboty jeżdżące

2.2.1.Koła

Lokomocja kołowa opiera się na zastosowaniu (w ogromnie przeważającej większości) trzech lub więcej kół, z których przeważnie co najmniej dwa są czynne. Konstrukcja takiego rozwiązania jest prosta i tania. Możliwych jest kilka rozwiązań kinematyki, w tym rozwiązania holonomiczne i nieholonomiczne. Rozwiązania z kołami biernymi i z napędem na wszystkie koła, z kołami skrętnymi i z kołami zamocowanymi pod stałym kątem do korpusu (rys. 3).



Rys. 3: Pięć podstawowych realizacji napędu trójkołowego.[6]

Ze względu na prostotę implementacji i względnie wysoką efektywność energetyczną rozwiązania bazującego na kołach, jest to najczęściej stosowane rozwiązanie w robotyce mobilnej. Zastosowanie nóg wiąże się z wieloma ograniczeniami oraz daleko większą złożonością mechanizmu. Jednak w obliczu konieczności poruszania się po nieregularnym gruncie konieczne staje się poszukiwanie innych rozwiązań.

2.2.2 Gąsienice

Realizacja napędu gąsienicowego jest ostatecznie bardzo podobna do napędu kołowego w przypadku gęstego ułożenia kół w jednej linii. Podstawową wadą takiego rozwiązania jest niekomfortowa zmiana kierunku. Wiąże się ona z przemieszczaniem skrajnych kół (skrajnych fragmentów gąsienic) po gruncie w kierunku nierównoległym do normalnie realizowanego. Gąsienice umożliwiają pokonywanie stosunkowo nieregularnego terenu.

2.3. Chód

Chód jest sposobem lokomocji wykorzystującym do tego celu kończyny. Cechą charakterystyczną chodu jest kontakt z podłożem w dyskretnych punktach. W przypadku pozostałych sposobów przemieszczania się kontakt ten jest ciągły. Pozwala to na przystosowanie schematu kroczenia do terenu, selekcję punktów podparcia oraz zmienne obciążenie danych fragmentów terenu w zależności od warunków. Chód umożliwia też pokonywanie wysokich przeszkód (w przypadku kół ograniczonych przez rozmiar kół) oraz zmianę rozmiarów robota w zależności od potrzeb.

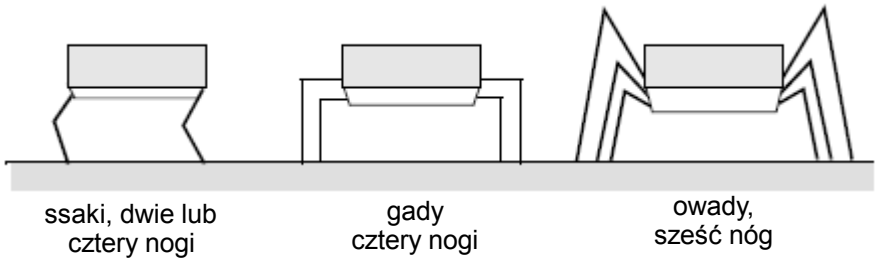
Chody można sklasyfikować ze względu na liczbę zaangażowanych w ten proces kończyn, ułożenie nóg, liczbę nóg mających kontakt z podłożem w danej chwili, bądź schemat według którego nogi są przenoszone.

2.3.1 Podział chodu ze względu na liczbę nóg

W skrajnym przypadku w chodzie może brać udział jedna noga, mamy wówczas do czynienia ze skakaniem. Możliwy jest chód dwunożny (taki jak ten spotykany u człowieka) oraz chód czworonożny, który dość znacznie różni się od poprzednich, a także chód sześcionożny, również mocno odbiegający od pozostałych. Używanie większej liczby nóg nie wnosi już znacznych zmian pod kątem mechaniki ruchu.

2.3.2. Podział chodu ze względu na ułożenie nóg

Rozróżnia się trzy podstawowe ustawienia nóg:



Rys. 4: Ustawienie nóg spotykane w przyrodzie[6]

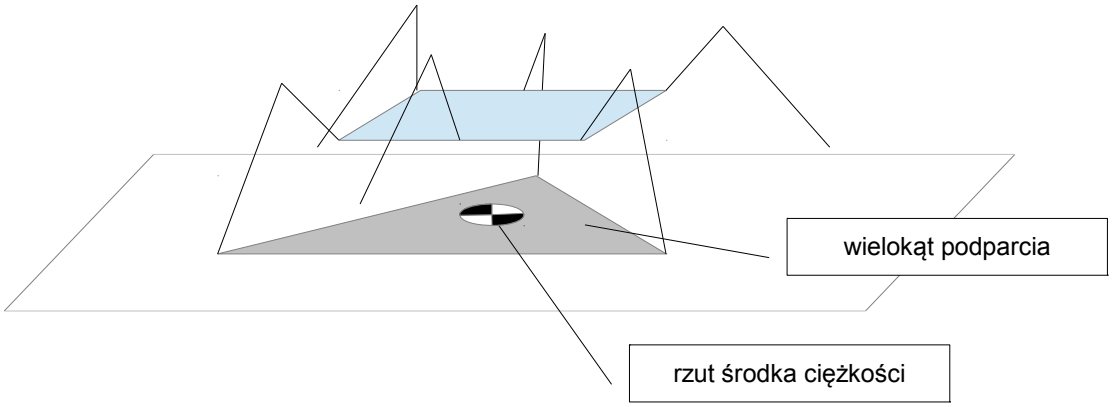
Przeprowadzone badania wskazują, że najefektywniejsze pod kątem konsumpcji energii jest ułożenie wzorowane na owadach.[9]

2.3.3 Podział chodu ze względu na dynamikę

Oprócz zewnętrznych wyróżników wyróżniamy chody stabilne statycznie i dynamicznie. W pierwszym przypadku, w dowolnym momencie, rzut środka ciężkości maszyny znajduje się wewnątrz wielokąta podparcia. Robot zatrzymany w dowolnej pozycji nie przewróci się. Chód stabilny dynamicznie nie spełnia tego warunku - środek ciężkości wykracza poza wielokąt podparcia, lub często w ogóle nie można o takim wielokącie mówić (jeżeli mniej niż trzy nogi mają kontakt z podłożem). Równowaga w chodzie stabilnym dynamicznie utrzymywana jest dzięki dynamice ruchu. Robot zatrzymany w miejscu przewróci się.

Do realizacji chodu stabilnego statycznie potrzeba przynajmniej czterech nóg, chociaż konfiguracja taka jest dość uciążliwa. W trybie takim możliwe jest przenoszenie tylko jednej nogi naraz, co wymusza wykonanie czterech cykli na jedną długość kroku. Konstrukcja sześcionożna daje większe możliwości pod kątem generacji chodu stabilnego statycznie. Do dyspozycji mamy sześć nóg, a do uzyskania stabilności potrzebne są trzy. W konsekwencji trzy mogą być przenoszone. Co za tym idzie, do pokonania dystansu jednego kroku potrzeba dwóch cykli. Poza tym możliwe jest uzyskanie zdecydowanie stabilniejszego rozstawienia nóg niż w przypadku czterech kończyn. Chód statycznie stabilny jest zdecydowanie łatwiejszy w kontroli, nie wymaga bowiem dynamicznego utrzymywania równowagi. Do realizacji chodu stabilnego dynamicznie z kolei konieczna jest mniejsza złożoność kinematyki (mniejsza liczba kończyn).

Na rysunku 5 została przedstawiona graficzna ilustracja wielokąta podparcia dla chodu trójpodporowego. Rzut środka ciężkości mieści się w obszarze wielokąta, chód jest stabilny statycznie.



Rys. 5: Graficzna ilustracja wielokąta podparcia

Rozdział 3

Konstrukcja robota

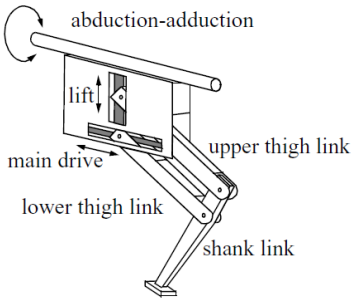
3.1. Założenia ogólne

Założeniem pracy jest opracowanie sześcionożnej, mobilnej platformy kroczącej zdolnej do przemieszczania się w nieregularnym terenie. Kinematyka mechanizmu jest ważnym czynnikiem determinującym możliwości robota. Na przestrzeni lat zaproponowanych zostało wiele rozwiązań kinematyki kończyn zdolnych do przemieszczania się, nie wszystkie jednak umożliwiają ruch w nieregularnym terenie.

3.2. Przykładowe mechanizmy ruchu

Istnieją różne konstrukcje umożliwiające realizację przemieszczenia za pomocą chodu. Między innymi mechanizmy:

- czteroczłonowy,
- pantografu,
- o jednym stopniu przesuwnym i dwóch obrotowych.[8]

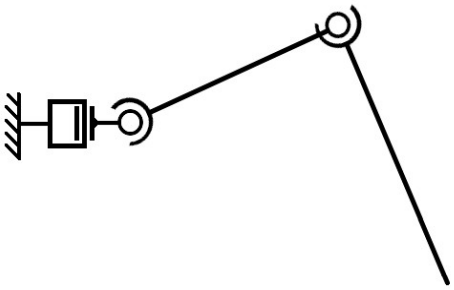


Rys. 6: Konstrukcja pantorgafu [6]

3.3. Proponowana kinematyka

Zaproponowana przez nas konstrukcja kończyny ma trzy stopnie ruchliwości, składają się na nią trzy człony oraz trzy ruchome połączenia obrotowe klasy piątej (rys. 8). Można w tej kinematyce wyróżnić połączania będące odpowiednikiem biodra (pierwsze dwie pary

kinematyczne, dwa stopnie swobody), oraz kolana (ostatnia para kinematyczna, jeden stopień swobody). Kinematyka taka spotykana jest w wielu, zarówno amatorskich jak i profesjonalnych, opracowanych do tej pory konstrukcjach, takich jak robot LAURON z 2009 roku widoczny na zdjęciu (rys. 7 [1]) .



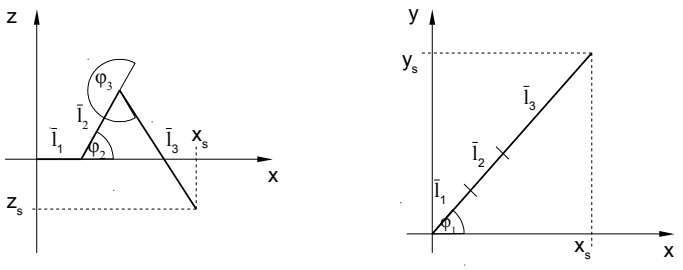
Rys. 8: Proponowana kinematyka nogi

Rys. 9: Robot LAURON [1]

Proponowana przez nas kinematyka pozwala na osiągnięcie dowolnego punktu z przestrzeni ograniczonej długością członów oraz na realizację dowolnej trajektorii w tej przestrzeni. Pozwoli to na realizację dowolnej trasy oraz adaptację do nieregularnego otoczenia.

3.4. Proste i odwrotne zagadnienie kinematyki nogi

Na rys. 10 przedstawiono rzuty kinematyki nogi na płaszczyzny OXY oraz OXZ w kartezjańskim układzie współrzędnych. Osie w tym układzie są równoległe do odpowiednich osi w układzie współrzędnych robota (rys. 11).



$\bar{l}_1$ – człon pierwszy
$\bar{l}_2$ – człon drugi
$\bar{l}_3$ – człon trzeci
φ_1 – kąt pomiędzy osią x a członem 1
φ_2 – kąt pomiędzy członem 1 a członem 2
φ_3 – kąt pomiędzy członem 2 a członem 3
x_s,y_s,z_s, – współrzędna stopy w układzie fundamentu nogi

Rys. 10: Rzuty kinematyki nogi na płaszczyzny OXY i OXZ układu wsp. fundamentu nogi.

1 http://en.wikipedia.org/wiki/LAURON

Zagadnienie proste kinematyki takiego mechanizmu opisują równania:

$$\begin{aligned} x_s &= \left(l_1 + l_2\cos(\varphi_2) + l_3\cos(\varphi_3+\varphi_2)\right)\cos(\varphi_1) \\ y_s &= x_s\tan(\varphi_1) \\ z_s &= l_2\sin(\varphi_2) + l_3\sin(\varphi_3+\varphi_2) \end{aligned} \tag{1}$$

przy czym l_1, l_2, l_3, oznaczają długości kolejnych członów.

Rozwiązanie zagadnienia odwrotnego jest skomplikowane i wymaga rozwiązania układu równań nielinowych. Rozwiązanie to jest potrzebne do określenia współrzędnych maszynowych koniecznych do osiągnięcia zadanych współrzędnych stopy w przestrzeni kartezjańskiej. W ogólności, dla omawianej kinematyki, istnieją cztery konfiguracje spełniające zadanie odwrotne.[8]

Ze względów praktycznych wszystkie kąty zostaną wyrażone przy pomocy funkcji sinus i cosinus, pozwoli to na zastosowanie jednej funkcji do wyłuskiwania wartości kąta w przyszłości, co uprości konstrukcję oprogramowania. Przyjęto oznaczenia $S_n = \sin(\theta_n)$, $C_n = \cos(\theta_n)$.

Wyznaczenie współrzędnej φ_1 nie jest trudne:

$$\begin{aligned} S_1 &= \pm\frac{y_s}{\sqrt{(x_s^2+y_s^2)}}, \\ C_1 &= \pm\frac{x_s}{\sqrt{(x_s^2+y_s^2)}}, \end{aligned} \tag{2}$$

przy czym znak +/- przyjmuje taką samą wartość we wszystkich trzech zworach i w dalszych rozważaniach będzie oznaczany przez symbol δ_1.

Następnie, wprowadzając zmienne pomocnicze:

$$\begin{aligned} a &= \delta_1\sqrt{x_s^2+y_s^2} - l_1, \\ b &= \frac{a^2+z_s^2+l_2^2-l_3^2}{2l_2}, \end{aligned} \tag{3}$$

otrzymujemy:

$$S_2 = \frac{z_s b \pm a\sqrt{a^2 + z_s^2 - b^2}}{a^2 + z_s^2},$$
$$C_2 = \frac{a b \mp z_s\sqrt{a^2 + z_s^2 - b^2}}{a^2 + z_s^2}, \tag{4}$$

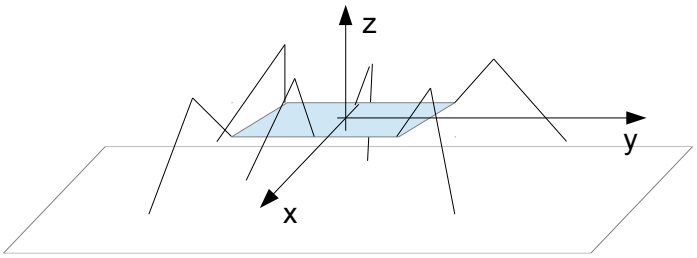
gdzie +/- i -/+ oznaczają przeciwne znaki i są niezależne od δ_1 i będą dalej oznaczane przez δ_2 oraz $-\delta_2$.

Trzecia współrzędna definiowana jest przez:

$$S_3 = \frac{z_s C_2 - a S_2}{l_3},$$
$$C_3 = \frac{a C_2 + z_s S_2 - l_2}{l_3}. \tag{5}$$

Ustalenie wartości

$$\delta_1 = 1,$$
$$\delta_2 = 1, \tag{6}$$

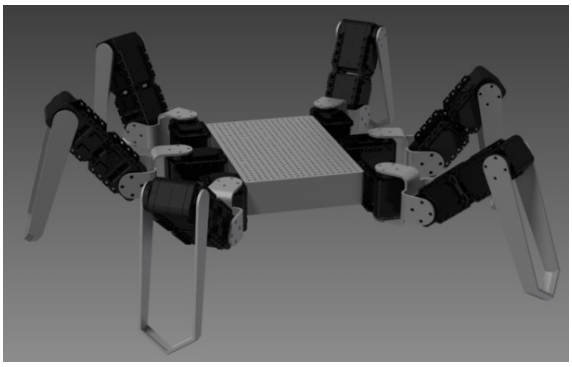


Rys. 11: Orientacja wewnętrznego układu współrzędnych robota

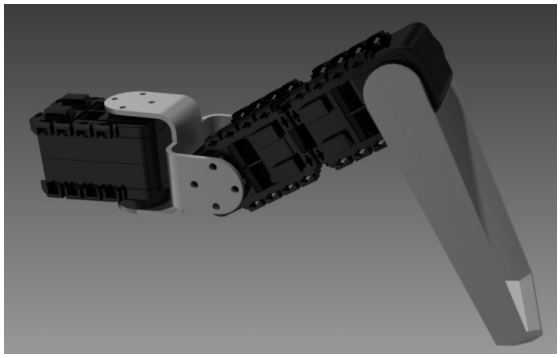
jednoznacznie określa φ_1, φ_2 oraz φ_3 względem współrzędnych x_s, y_s, z_s oraz zapewnia ułożenia nogi jak na rys 8. Środek układu współrzędnych robota (rys. 11) jest umiejscowiony w środku geometrycznym korpusu. Oś Y jest skierowana wzdłuż normalnego kierunku ruchu, oś Z jest osią pionową o zwrocie skierowanym do góry. Orientacja osi X wynika z ułożenia pozostałych osi.

3.5. Model nogi

Zaproponowany przez nas mechanizm realizujący powyższą kinematykę zaprojektowany przy użyciu programu Inventor przedstawiono na rysunkach 12 i 13.

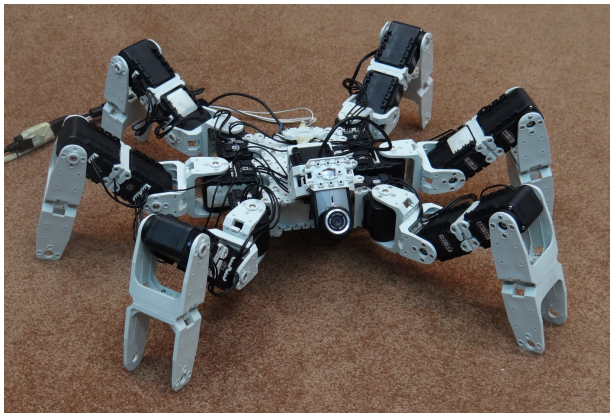


Rys. 12: Model mechaniki robota w programie Inventor.

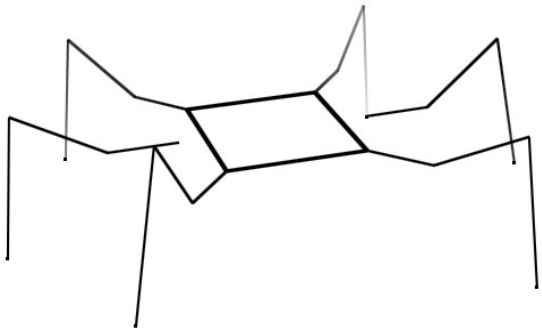


Rys 13: Model mechanizmu nogi w programie Inventor.

Bazując na zestawie do szybkiego prototypowania firmy Dynamixel zbudowany został prototyp realizujący proponowaną kinematykę. Prototyp napędzany jest serwami Dynamixel AX-12 (rys. 17).

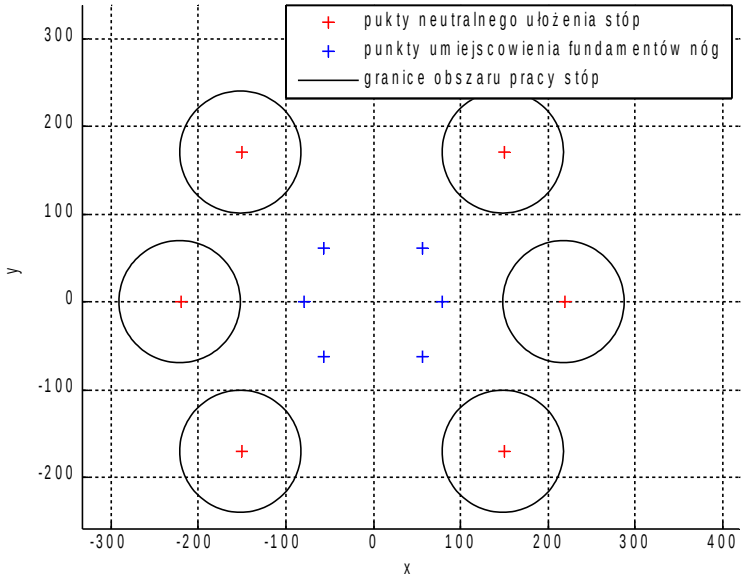


Rys. 14: Konstrukcja prototypowa



Rys. 15: Schematycznie zaprezentowana kinematyka platformy.

Każda para kinematyczna napędzana jest całkowicie niezależnie przy pomocy oddzielnych aktuatorów. Dzięki temu możliwe jest osiągnięcie dowolnego ułożenia nogi. Na rysunku 16 zaprezentowano jedno z możliwych rozstawień stóp, wraz z narzuconymi programowo ograniczeniami ruchu w płaszczyźnie OXY. Ograniczenia takie są konieczne, aby zapobiec kolizji nóg w trakcie kroczenia.



Rys. 16: Schemat rozstawienia nóg robota

3.6. Dobór napędów

Jako napęd robota zastosowane zostały cyfrowe serwomechanizmy elektryczne Dynamixel AX-12. Serwa te zasilane są napięciem z przedziału 7 – 10 V i rozwijają w tym zakresie moment od 12 do 16,5 kgfcm, przy masie własnej wynoszącej 55g. Zakres pracy wynosi 300˚.

W tabeli 1 przedstawiono podstawowe parametry serwa podawane przez producenta[2].

Tabela 1: Parametry serwomechanizmu AX-12[2]

	AX-12	
Weight (g)	55	
Gear Reduction Ratio	1/254	
Input Voltage (V)	at 7V	at 10V
Final Max Holding Torque(kgf.cm)	12	16.5
Sec/60degree	0.269	0.196

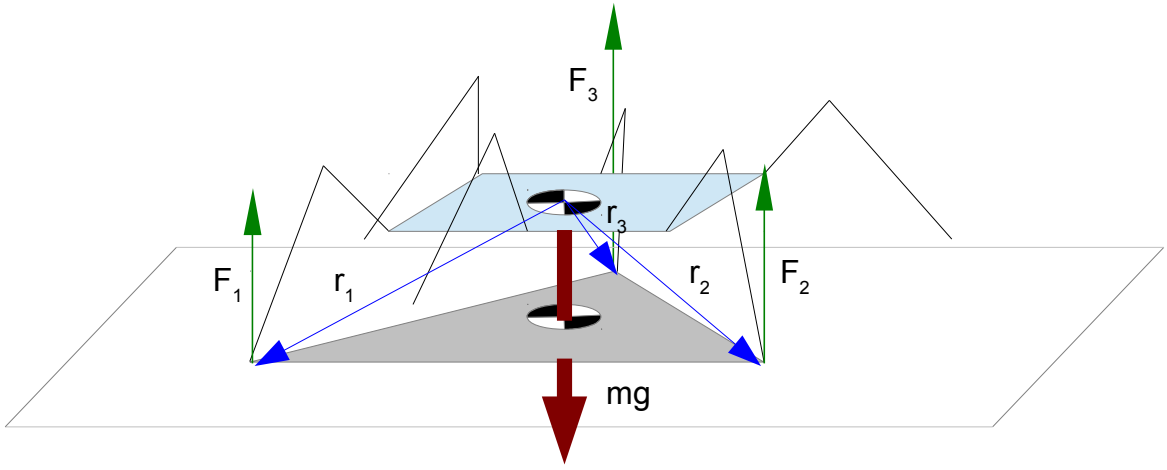
Rys. 17: Serwomechanizm AX-12[2]

2 http://support.robotis.com/en/product/dynamixel/ax_series/dxl_ax_actuator.htm

Biorąc pod uwagę fakt, że rozważania dynamicznych obciążeń jest bardzo skomplikowane, oraz że maszyna krocząca nie porusza się z dużymi przyspieszeniami, mogą zostać one z małą stratą dokładności pominięte w poniższych obliczeniach.[8]

3.6.1. Siły reakcji podłoża oraz momenty obciążające serwomechanizmy

Istotną wartością pod kątem przydatności zastosowania proponowanych serwomechanizmów jest moment siły jaki są one w stanie pokonać. Aby móc określić wymaganą, minimalną wartość momentu, którą serwomechanizm musi być w stanie pokonać, trzeba znać wartość siły obciążającej poszczególne nogi. Robot porusza się chodem trójpodporowym, co oznacza, że w danej chwili tylko trzy nogi dźwigają całą masę robota.



Rys. 18: Siły działające na robota. Zielone strzałki reprezentują siły reakcji podłoża, niebieskie wektory łączące środek masy z punktami przyłożeni sił, czerwona strzałka przedstawia siłę ciążenia.

Znając ułożenie nóg, oraz masę robota, przy założeniu statycznej równowagi całego mechanizmu, jesteśmy w stanie obliczyć siły reakcji podłoża (zielone strzałki na rys. 18).

Prostą i łatwą w implementacji metodą jest rozwiązanie układu równań wynikających z pierwszej zasady dynamiki Newtona. Wymaga ona rozwiązania układu równań 7 i 8.
Równanie równowagi sił względem każdej z osi:

$$\begin{cases} F_{x1}+F_{x2}+F_{x3}+m\cdot g_x=0, \\ F_{y1}+F_{y2}+F_{y3}+m\cdot g_y=0, \\ F_{z1}+F_{z2}+F_{z3}+m\cdot g_z=0, \end{cases} \tag{7}$$

równanie równowagi momentów sił względem każdej z osi:

$$\begin{cases} F_{z1}r_{y1}+F_{z2}r_{y2}+F_{z3}r_{y3}+F_{y1}r_{z1}+F_{y2}r_{z2}+F_{y3}r_{z3}+m\,g_z r_{y0}+m\,g_y r_{z0}=M_x=0 \\ F_{z1}r_{x1}+F_{z2}r_{x2}+F_{z3}r_{x3}+F_{x1}r_{z1}+F_{x2}r_{z2}+F_{y3}r_{z3}+m\,g_z r_{x0}+m\,g_x r_{z0}=M_y=0 \\ F_{x1}r_{y1}+F_{x2}r_{y2}+F_{x3}r_{y3}+F_{y1}r_{x1}+F_{y2}r_{x2}+F_{y3}r_{x3}+m\,g_x r_{y0}+m\,g_y r_{x0}=M_z=0 \end{cases} \tag{8}$$

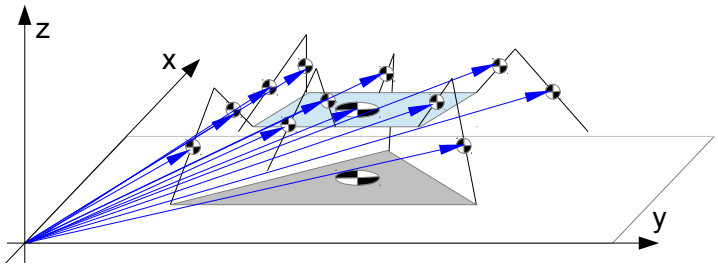
gdzie r_{an} oznacza składową *a* ramienia działania siły *n* dla punktu względem którego momenty są sumowane. Na rysunku 18 przyjęto, że punktem tym jest środek masy, dzięki czemu moment od siły ciężkości jest zerowy ($r_{x0} = 0$, $r_{y0} = 0$, $r_{z0} = 0$).

3.6.2. Środek masy

Aby powyższe równania rozwiązać, konieczna jest znajomość położenia środka masy. Można go wyznaczyć posługując się poniższym wzorem:

$$\vec{r}_{srodkamasy} = \frac{\vec{r}_1 m_1 + \vec{r}_2 m_2 + \cdots + \vec{r}_n m_n}{m_1 + m_2 + \cdots + m_n} \tag{9}$$

Biorąc pod uwagę, że kończyny robota obarczone są dość znaczną masą, położenie środka ciężkości maszyny może zależeć od ułożenia mechanizmu, co z kolei wpływa na rozkład sił. Można założyć z dużą dokładnością, że środki mas poszczególnych członów znajdują się w ich geometrycznych środkach, środek masy korpusu, także umiejscowiony jest w jego geometrycznym środku:



Rys. 19: Graficzna ilustracja środków masz poszczególnych członów i wektorów łączących je ze środkiem układu współrzędnych

Zważywszy, że początek układu współrzędnych umiejscowiony jest w w naszym przypadku w geometrycznym środku korpusu, oraz masy poszczególnych członów wynoszą odpowiednio:

$$m_{uda} = 119[g]$$
$$m_{lydki} = 37[g]$$
$$m_{korpusu} = 840[g]$$
$$m_{akumulatora} = 250[g]$$
$$m_{elektroniki} = 200[g]$$

oraz symetryczne rozstawienie nóg, środek rzut środka ciężkości znajduje się w punkcie [0,0] w pozycji neutralnej (kiedy wszystkie nogi ustawione są symetrycznie) i (jak pokazała symulacja w programie Matlab) waha się w przedziale $[-2,2]\times[-3,3]$ milimetra na płaszczyźnie OXY podczas chodu.

3.6.3. Siły reakcji

Zakładamy brak tarcia pomiędzy stopami i podłożem, oraz równoległość podłoża do płaszczyzny OXY. Kierunek osi Z jest równoległy do kierunku przyspieszenia ziemskiego $\overline{g}$. W konsekwencji siły $\overline{F}_1$, $\overline{F}_2$ i $\overline{F}_3$ oraz przyspieszenie $\overline{g}$ mają zerowe składowe x i y oraz niezerową składową z. Przyjmując za punkt względem którego sumujemy momenty środek masy robota ($r_0 = 0$) oraz pomijając zerowe składniki otrzymujemy z równań (8) układ równań (10).

$$\begin{cases} F_{z1} \cdot r_{y1} + F_{z2} \cdot r_{y2} + F_{z3} \cdot r_{y3} = M_x = 0 \\ F_{z1} \cdot r_{x1} + F_{z2} \cdot r_{x2} + F_{z3} \cdot r_{x3} = M_y = 0 \end{cases}, \tag{10}$$

Siły reakcji nie mają składowych nierównoległych do osi Z, więc M_z jest zawsze równe zero. Postępując podobnie w przypadku równań dotyczących sił otrzymujemy tylko jedno równanie opisujące siły względem osi z (pozostałe składowe są zerowe) i dokładając otrzymane równanie do układu (10) otrzymujemy układ którego zapis macierzowy przedstawia równanie (11)

$$\begin{bmatrix} 1 & 1 & 1 \\ r_{x1} & r_{x2} & r_{x3} \\ r_{y1} & r_{y2} & r_{y3} \end{bmatrix} \cdot \begin{bmatrix} F_{z1} \\ F_{z2} \\ F_{z3} \end{bmatrix} = \begin{bmatrix} -m g_z \\ 0 \\ 0 \end{bmatrix}, \tag{11}$$

co jest jednoznaczne równowadze sił oraz momentów względem osi OX i osi OY.
Rozwiązaniem powyższego równania jest:

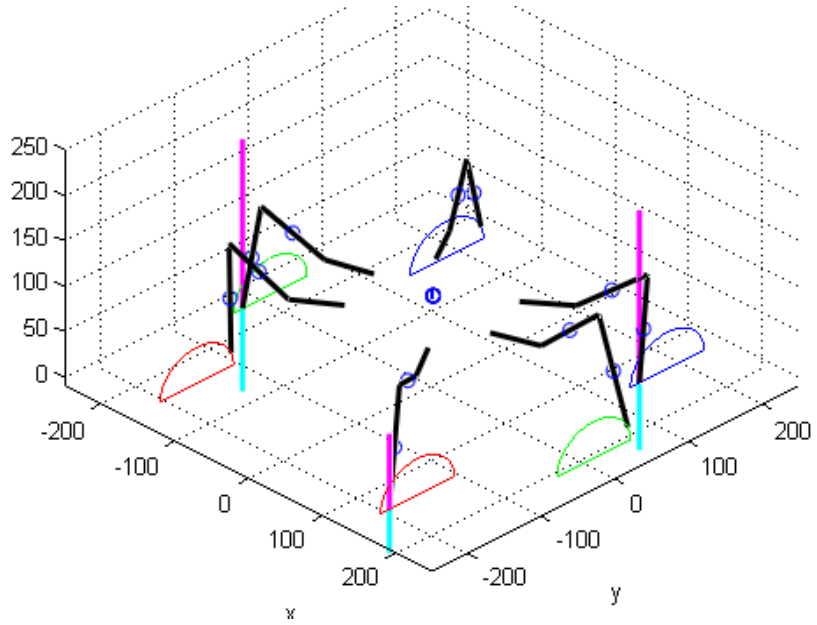
$$\begin{bmatrix} F_{z1} \\ F_{z2} \\ F_{z3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ r_{x1} & r_{x2} & r_{x3} \\ r_{y1} & r_{y2} & r_{y3} \end{bmatrix}^{-1} \cdot \begin{bmatrix} -m g_z \\ 0 \\ 0 \end{bmatrix}. \tag{12}$$

Na podstawie sił działających na stopę możliwe jest określenie wartości momentu obciążającego serwomechanizm napędzający daną parę kinematyczną. Biorąc pod uwagę że składowa M_z momentów statycznych jest zerowa, moment statyczny obciążający pierwsze połączenie jest równy zero. W połączeniu tym występują momenty dynamiczne związane z dynamiką ruchu nogi oraz całej platformy. Pozostałe połączenia obciążone są momentami, których wartości (ze względu na prostopadłość sił do płaszczyzny OXY) równe są iloczynowi wartości siły i długości rzutu promienia łączącego punkt przyłożenia siły z osią działania momentu, na płaszczyznę OXY. Długość tego rzutu, ze względu na ułożenie nogi,

jest zawsze większa dla połączenia drugiego (pomiędzy pierwszym i drugim członem) niż dla połączenia trzeciego (drugi i trzeci człon), co za tym idzie, najmocniej obciążane jest serwo drugie. Z powyższych względów analiza momentów dla połączenia pierwszego i trzeciego została w dalszych rozważaniach pominięta.

Połączenia kinematyczne są realizowane poprzez bezpośrednie osadzenie kolejnych członów na osiach serwomechanizmów, w związku z tym moment obciążający daną oś jest tożsamy z momentem obciążenia serwomechanizmu.

W celu zbadania zmienności momentów podczas chodu, zależności (9) i (11) zostały zaimplementowane oraz zwizualizowane w środowisku Matlab (rys. 23).

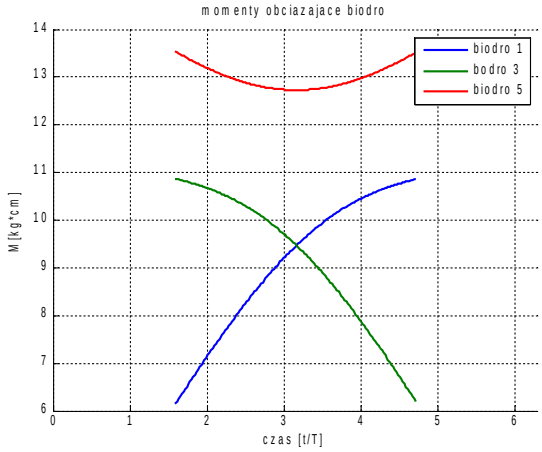


Rys. 20: Wizualizacja chodu robota wraz z naniesionymi siłami reakcji (różowe słupki), oraz momentami obciążającymi stawy biodrowe (błękitne) dla nóg będących w fazie podporowej. Niebieskie kółka oznaczają środki mas poszczególnych członów nóg.

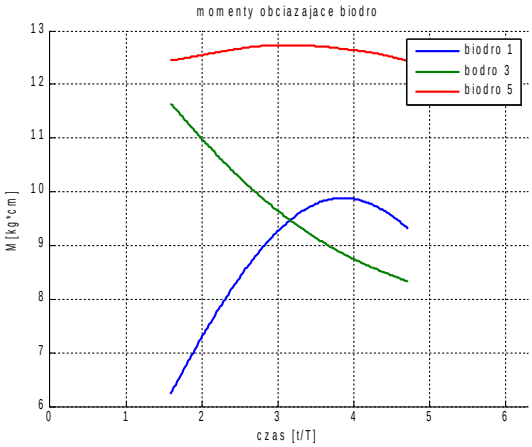
Warto zauważyć, że z równań (10) wynika, iż im bardziej oddalony jest punkt kontaktu stopy z podłożem od rzutu środka ciężkości przy niezmienionym ułożeniu pozostałych nóg, tym mniejsza siła działa na daną nogę (równowaga momentów), jednak na tym większym działa ramieniu – co powoduje wzrost momentu obciążającego biodro. Jednoczesne odsuwanie wszystkich nóg podporowych od środka ciężkości niweluje efekt zmniejszania siły reakcji (suma reakcji musi równoważyć siłę ciążenia), co za tym idzie skutkuje zwiększeniem momentów obciążających serwa. Kolejnym istotnym elementem jest fakt, że w trójpodporowym schemacie ruchu po jednej stronie masa robota obciąża dwie nogi, po drugiej natomiast – jedną. Konsekwencją jest wyraźnie mocniejsze obciążenie środkowej nogi w stosunku do nogi przedniej i nogi tylnej.

3.7. Wyniki obliczeń

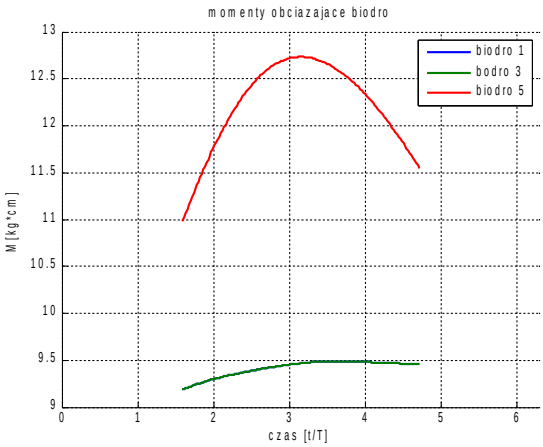
Poniżej przedstawiono wykresy symulowanych obciążeń dla stawów biodrowych przy różnych parametrach chodu. Symulację ze względu na symetrię zjawiska przeprowadzono tylko dla jednej trójki nóg.



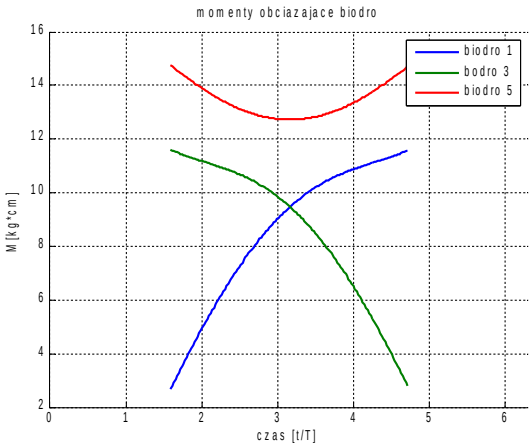
Rys. 21: Ruch prostoliniowy w kierunku 0 rad, długośc kroku 100 [mm]



Rys. 22: Ruch prostoliniowy, kierunek 45 st. długość kroku 100 [mm]



Rys. 23: Ruch prostoliniowy, kierunek 90 st. długość kroku 100 [mm]. Obciążenia biodra 1 i 3 są identyczne.



Rys. 24: Ruch prostoliniowy, kierunek 0 st. Długoś kroku 160 [mm]

Z analizy wynika, że przy obciążenie statyczne pojedynczego serwomechanizmu nie powinno przekraczać 14 kgfcm dla długości kroku 100 mm, co daje zapas w przypadku zastosowanych urządzeń.

W symulacji zbadane zostały obciążenia podczas realizacji chodu według algorytmu, którego opracowanie przedstawione zostało w kolejnym rozdziale.

Rozdział 4

Opracowanie algorytmów ruchu

4.1. Realizacja ruchu prostoliniowego

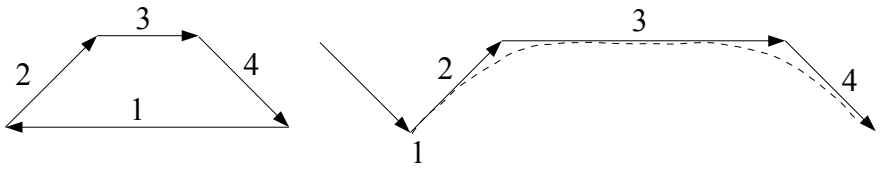
Podczas ruchu, nogi platformy kroczącej mogą znajdować się w dwóch fazach:

- w fazie podporowej,
- w fazie przenoszenia.

W fazie podporowej końcówka nogi realizuje trajektorię konieczną do planowanego przemieszczenia maszyny. Końcówka nogi ma niezmienną pozycję w zewnętrznym układzie współrzędnych związanych z podłożem (przy założeniu braku poślizgów), trajektoria realizowana jest w układzie wewnętrznym robota.

W fazie przenoszenia noga przemieszcza się w ogólności w obu wspomnianych wyżej układach, pomiędzy kolejnymi punktami podparcia. Podstawowym czynnikiem determinującym kształt trajektorii w tej fazie jest założenie braku kontaktu z podłożem. Trajektoria stopy w tej fazie nie ma większego znaczenia z punktu widzenia realizowanego ruchu, jednak istnieją ograniczenia wynikające z bezwładności członów, kinematyki mechanizmu oraz zużycia energii wpływające na krzywą po której można przemieszczać stopę w tej fazie ruchu. Faza przenoszenia odpowiada za przemieszczanie stopy pomiędzy kolejnymi punktami podparcia.

Złożenie tych dwóch faz daje linię ciągłą w przestrzeni w obu układach współrzędnych.



Rys. 25: Przykładowa trajektoria stopy w układzie wsp. Robota i w zewnętrznym układzie wsp.1. faza podparcia, 2,3,4 faza przenoszenia

O ile w fazie podparcia trajektoria stopy jest bardzo ściśle zdefiniowana przez pożądany ruch platformy, o tyle w fazie przenoszenia możliwe jest zastosowanie wielu różnych trajektorii.

Podstawowym ruchem realizowanym przez platformę mobilną jest ruch prostoliniowy. W ruchu tym środek układu współrzędnych robota przemieszcza się po linii prostej względem zewnętrznego układu odniesienia. Zakładamy brak obrotów jednego układu względem drugiego – robot cały czas jest zwrócony tą samą stroną w kierunku ruchu.

Układ współrzędnych robota porusza się liniowo względem układu odniesienia. W fazie podporowej końcówki nóg nie przesuwają się względem podłoża (mają stałą pozycję względem zewnętrznego układu współrzędnych), a co za tym idzie – muszą poruszać się po liniach prostych w układzie robota. Parametryczna postać prostej w kartezjańskim układzie współrzędnych:

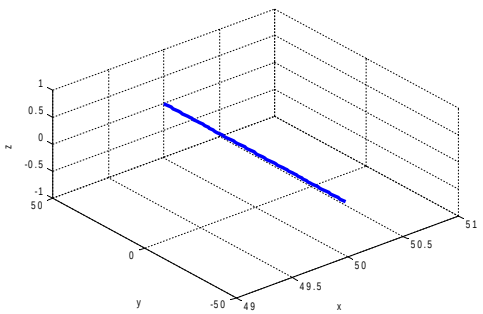
$$\begin{cases} x = x_0 + ar \\ y = y_0 + br \\ z = z_0 + cr \end{cases} \tag{13}$$

gdzie r jest parametrem, a a, b i c są pewnymi stałymi .

Jeżeli założymy, że oś z jest prostopadła do powierzchni po której przemieszcza się robot, a powierzchnia ta jest płaska, można założyć że w układzie robota w fazie podporowej współrzędna z jest stała:

$$\begin{cases} x = x_0 + a\,r \\ y = y_0 + b\,r \\ z = z_0 \end{cases} \tag{14}$$

Umieszczając środek układu robota w jego geometrycznym środku, w taki sposób, że oś z jest skierowana do dołu, a oś y wskazuje przód robota, aby uzyskać ruch wzdłuż osi y (rys. 26) należy przyjąć że $x = x_0$.



Rys. 26: odcinek prostej w przestrzeni kartezjańskiej

Na potrzeby ruchu kończyny załóżmy, że parametr *r* zmienia się w pewnym zakresie i przyjmijmy stałą *w,* opisującą długość wykroku i zakroku (połowa długości kroku), oraz stałą *h* odpowiadającą za wysokość podnoszenia nogi. Ponadto parametr *r* w fazie przenoszenia różni się od parametru w fazie podporowej (r_1) i jest inny dla współrzędnej *y* i współrzędnej *z* (r_2, r_3). Składając odcinek prostej wynikający z fazy podporowej, z fragmentem trajektorii odpowiadającym za przenoszenie stopy (którego rzut na płaszczyznę OXY jest linią prostą, jednak sam fragment w ogólności prostą nie jest - ze względu na możliwą zmienność r_2 względem r_3) otrzymujemy poniższą trajektorię:

$$\begin{cases} x(t) = x_0 \\ y(t) = y_0 + r_1 w & , t \in \textit{faza podporowa}, \\ z(t) = z_0 \\ x(t) = x_0 \\ y(t) = y_0 + r_2 w & , t \in \textit{faza przenoszenia}, \\ z(t) = z_0 + r_3 h \end{cases} \tag{15}$$

gdzie:

r_1 – parametr zmieniający się w zakresie [-1,1], od którego uzależniona jest współrzędna *y* stopy w fazie podporowej,
r_2 – parametr zmieniający się w zakresie [-1,1], od którego uzależniona współrzędna y w fazie przenoszenia,
r_3 – parametr zmieniający się w zakresie [0,1], od którego uzależniona współrzędna z w fazie przenoszenia,
w – połowa długości kroku (długość zakroku / wykroku),
h – maksymalna wysokość podnoszenia stopy w fazie przenoszenia.

Aby uzależnić położenie stóp od czasu zdefiniujmy parametry *r* w następujący sposób:

$$\begin{aligned} r_1 &= \frac{2}{\pi}\, asin(\sin(p)), \\ r_2 &= \sin(p), \\ r_3 &= \cos(p), \end{aligned} \tag{16}$$

gdzie wartość parametru *p* zdefiniowana jest następująco (17):

$$p(t) = p(t - \Delta t) + \Delta t \cdot v, \tag{17}$$

przy czym *p(t)* to wartość p w danej chwili czasu, *p(t-Δt)* to wartość w poprzedniej chwili próbkowania *t-Δt*, a *v* jest zmienną opisującą szybkość działania mechanizmu. Dla $v = 0$, wartość parametru *p* nie zmienia się w czasie, pozycja stóp jest stała.

Fazę podporową definiujemy jako zakres p spełniający warunek (18):

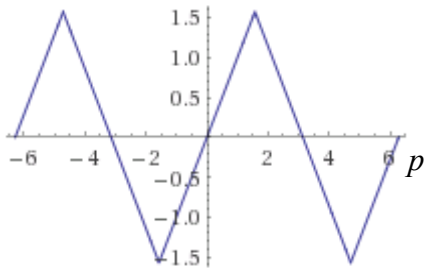
$$p \in \left[k\pi - \pi, k\pi - \frac{\pi}{2} \right) \cup \left[k\pi + \frac{\pi}{2}, k\pi + \pi \right), \quad k \in \mathbb{Z}, \tag{18}$$

i fazę przenoszenia jako (19):

$$p \in \left[k\pi - \frac{\pi}{2}, k\pi + \frac{\pi}{2} \right), \quad k \in \mathbb{Z}, \tag{19}$$

co łącznie zapewnia zmienność poszczególnych parametrów w zakładanym zakresie.

W fazie podporowej współrzędna y zależy od złożenia funkcji arcus sinus z funkcją sinus. Dzięki temu zabiegowi otrzymujemy przebieg, który w przedziałach ma stałą wartość pochodnej i zmienia się w zakresie $\left[-\frac{\pi}{2}, \frac{\pi}{2} \right]$ (rys. 27). Noga porusza się w fazie podporowej ze stałą prędkością, a funkcja opisująca jej ruch jest okresowa. Czynnik $2/\pi$ został wprowadzony w celu przeskalowania zakresu zmienności do [-1,1].



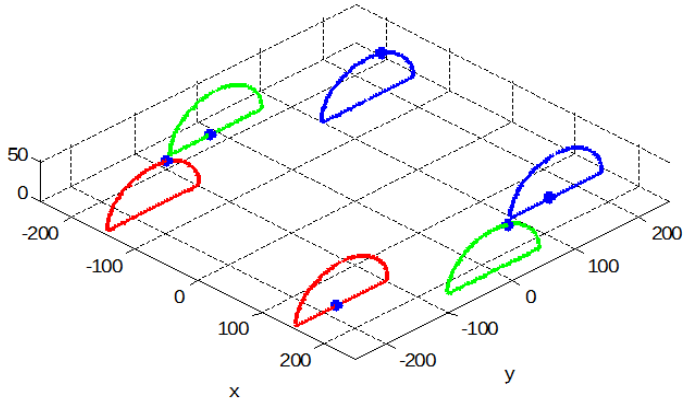
Rys. 27: Przebieg funkcji *arcsin*(*sin*(*p*)).

Zaproponowany przez nas kształt trajektorii realizowanej w celu generowania chodu prostoliniowego został opisany równaniami (20) i przedstawiony na rysunku 28.

$$\begin{cases} x_i = x_{0_i} \\ y_i = y_{0_i} + w \cdot \frac{2}{\pi} \cdot asin(\sin(p)) & p \in \left[k\pi - \pi, k\pi - \frac{\pi}{2} \right) \cup \left[k\pi + \frac{\pi}{2}, k\pi + \pi \right), \textit{faza podporowa} \\ z_i = z_{0_i} \\ x_i = x_{0_i} \\ y_i = y_{0_i} + w \cdot \sin(p) & p \in \left[k\pi - \frac{\pi}{2}, k\pi + \frac{\pi}{2} \right), \textit{faza przenoszenia}, \\ z_i = z_{0_i} + h \cdot \cos(p) \end{cases} \tag{20}$$

gdzie:

w – połowa długości kroku (długość zakorku/wykroku),
h – wysokość podnoszenia stóp w fazie przenoszenia,
v – parametr odpowiadający za szybkość przemieszczania się po trajektorii.



Rys. 28: przebieg trajektorii stóp robota. Niebieskie kropki oznaczają położenie stopy w danej chwili.

Dla poszczególnych nóg otrzymujemy takie same krzywe przesunięte o współrzędne odpowiadające neutralnym położeniom danych nóg. Jak widać, w fazie podporowej jest to odcinek równoległy do osi *y*, w fazie przenoszenia fragment elipsy o jednej półosi pokrywającej się z odcinkiem fazy podporowej. Funkcje użyte w zaproponowanej trajektorii są okresowe co pozwala na uzależnienie ich w prosty sposób od czasu.

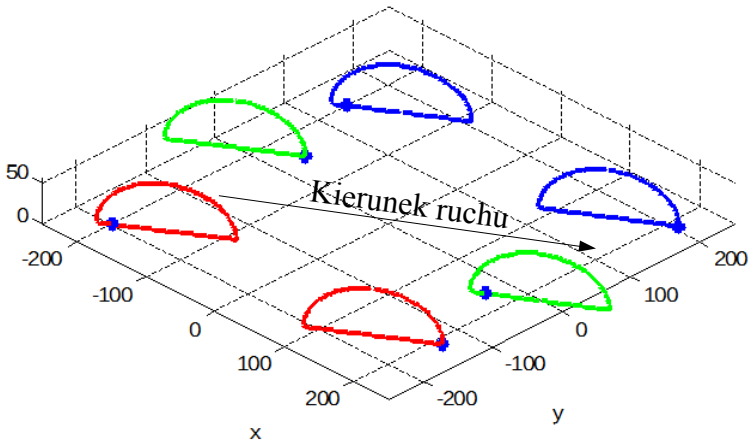
4.2. Realizacja ruchu prostoliniowego w dowolnym kierunku:

Jak pokazuje praca [5], do osiągnięcia zadowalających parametrów chodu w przypadku platformy sześcionożnej o kinematyce nóg zastosowanej w naszym przypadku, wystarczająca jest realizacja chodu prostoliniowego w kilku dyskretnych kierunkach. W pracy tej przeprowadzono symulację w której zaimplementowano możliwość chodu w dwunastu dyskretnych kierunkach rozmieszczonych co 30°, co okazało się być wystarczające do realizacji dość złożonych trajektorii.

Zaproponowany w poprzednim paragrafie układ równań opisuje ruch w kierunku osi y. Aby uzyskać równania opisujące ruch po linii prostej w dowolnym kierunku równoległym do płaszczyzny OXY wystarczy dokonać obrotu zaproponowanych trajektorii wokół neutralnych położeń odpowiednich stóp (21).

$$\begin{cases} x_i = x_{0_i} + \left(w \cdot \frac{2}{\pi} \cdot asin(\sin(p)) \right) \cdot \sin(\alpha) \\ y_i = y_{0_i} + \left(w \cdot \frac{2}{\pi} \cdot asin(\sin(p)) \right) \cdot \cos(\alpha) \\ z_i = z_{0_i} \end{cases} \quad p \in \left[k\pi - \pi, k\pi - \frac{\pi}{2} \right) \cup \left[k\pi + \frac{\pi}{2}, k\pi + \pi \right),$$

$$\begin{cases} x_i = x_{0_i} + (w \cdot \sin(p)) \cdot \sin(\alpha) \\ y_i = y_{0_i} + (w \cdot \sin(p)) \cdot \cos(\alpha) \\ z_i = z_{0_i} + h \cdot \cos(p) \end{cases} \quad p \in \left[k\pi - \frac{\pi}{2}, k\pi + \frac{\pi}{2} \right), \qquad (21)$$



Rys. 29: Przebieg trajektorii stóp robota. Ruch prostoliniowy w dowolnym kierunku.

W rezultacie, dla kierunku α = 0 rad otrzymujemy taką samą zależność, jak w poprzednim przypadku. Przykładowa trajektoria dla kierunku α = 1 rad przedstawiona została na rysunku 29.

4.3. Realizacja ruchu obrotowego

Aby zrealizować ruch obrotowy, wprowadźmy jednoczesny obrót wszystkich trajektorii wyprowadzonych dla chodu prostoliniowego wokół środka układu współrzędnych robota. Wprowadźmy oznaczenia:

$$\begin{aligned} &x_{0_i}+\left(w\cdot\sin(p)\right)\cdot\sin(\alpha)=f_{0\mathrm{xi}}\\ &y_{0_i}+\left(w\cdot\sin(p)\right)\cdot\cos(\alpha)=f_{0\mathrm{yi}}\\ &x_{0_i}+\left(w\cdot\frac{2}{\pi}\cdot asin(\sin(p))\right)\cdot\sin(\alpha)=f_{1\mathrm{xi}}\\ &y_{0_i}+\left(w\cdot\frac{2}{\pi}\cdot asin(\sin(p))\right)\cdot\cos(\alpha)=f_{1\mathrm{yi}} \end{aligned} \tag{22}$$

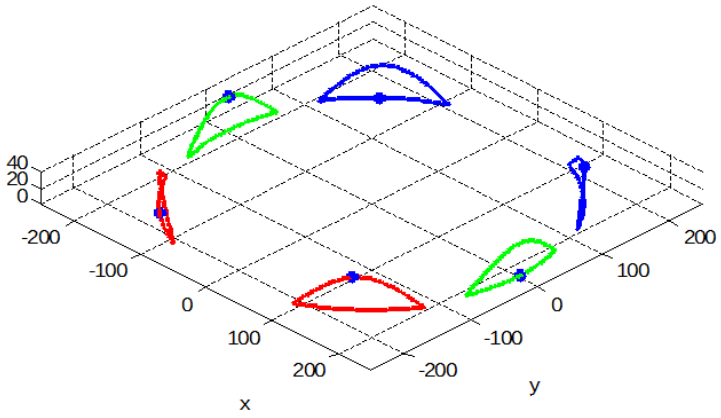
ruch w osi Z pozostawiony został bez zmian:

(23)

$$\begin{cases} x_i=f_{1\mathrm{xi}}\cdot\cos(\arcsin(\sin(p))\cdot\omega)+f_{1\mathrm{yi}}\cdot\sin(\arcsin(\sin(p))\cdot\omega)\\ y_i=f_{1\mathrm{yi}}\cdot\cos(\arcsin(\sin(p))\cdot\omega)-f_{1\mathrm{xi}}\cdot\sin(\arcsin(\sin(p))\cdot\omega) & p\in\left[k\pi-\pi,k\pi-\frac{\pi}{2}\right)\cup\left[k\pi+\frac{\pi}{2},k\pi+\pi\right),\\ z_i=z_{0_i} \end{cases}$$

$$\begin{cases} x_i=f_{0\mathrm{xi}}\cdot\cos(\omega\cdot p)+f_{0\mathrm{yi}}\cdot\sin(\omega\cdot p)\\ y_i=f_{0\mathrm{yi}}\cdot\cos(\omega\cdot p)-f_{0\mathrm{xi}}\cdot\sin(\omega\cdot p) & p\in\left[k\pi-\frac{\pi}{2},k\pi+\frac{\pi}{2}\right),\\ z_i=z_{0_i}+h\cdot\cos(p) \end{cases}$$

przy czym omega jest prędkością obrotową. Ostateczna prędkość obrotu zależy od omegi i prędkości działania całego mechanizm, czyli od ω i v. Dla $\omega = 0$, $\sin(\omega p)$ jest równy zero, natomiast $\cos(\arcsin(\sin(p)\omega))$ wynosi 1, czyli otrzymujemy wyprowadzoną wcześniej zależność.

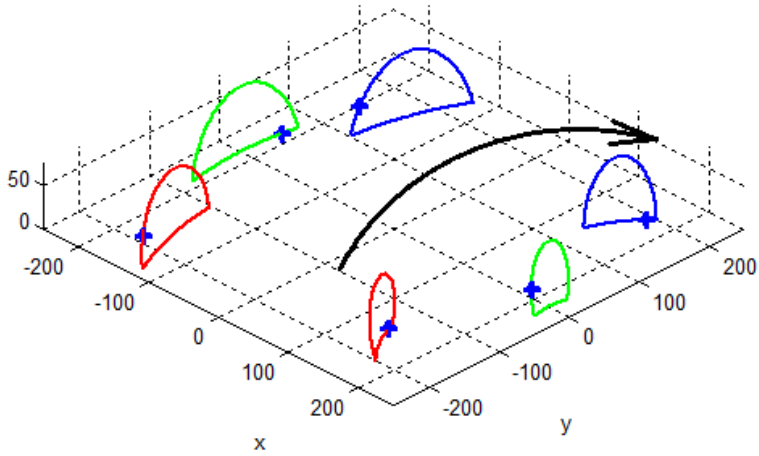


Rys. 30: Przebieg trajektorii stóp robota dla ruchu obrotowego.

dla $\omega = 0.2$ oraz długości kroku $w = 0$ otrzymujemy rezultat zaprezentowany na rys 30.
Dla $w = 0$, w fazie podporowej stopy robota przemieszczają się po łukach których środki pokrywają się ze środkiem układu współrzędnych robota, a promień jest równy odległości neutralnego położenia danej stopy od środka układu współrzędnych robota.

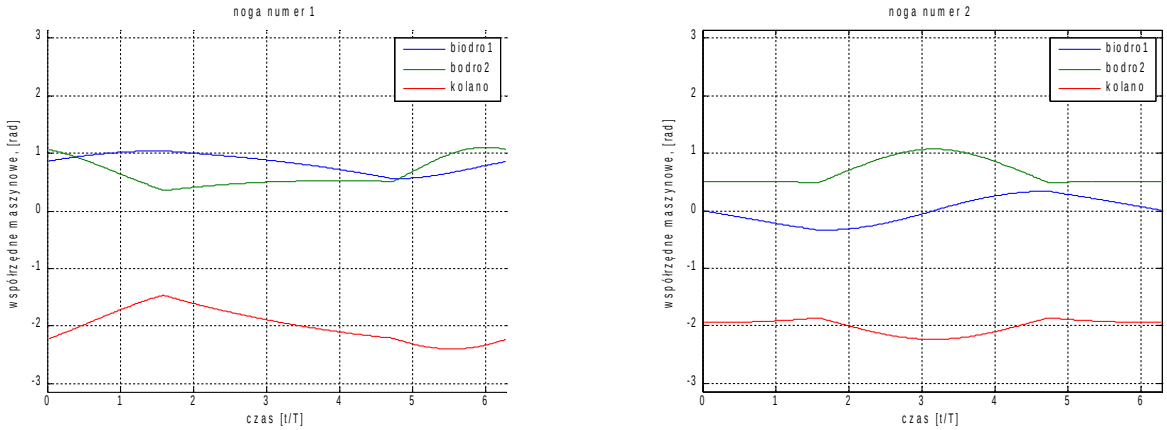
4.4. Realizacja ruchu w dowolnym kierunku po dowolnej krzywej z jednoczesnym obrotem robota.

Ponieważ ruch na płaszczyźnie można zapisać jako sumę przesunięć i obrotów, zastosowanie odpowiednio sparametryzowanych, wyprowadzonych wyżej równań pozwoli na realizację dowolnej trasy robota, oraz ustawienie go w dowolnej orientacji w dowolnym punkcie realizowanej trajektorii (rys. 31).

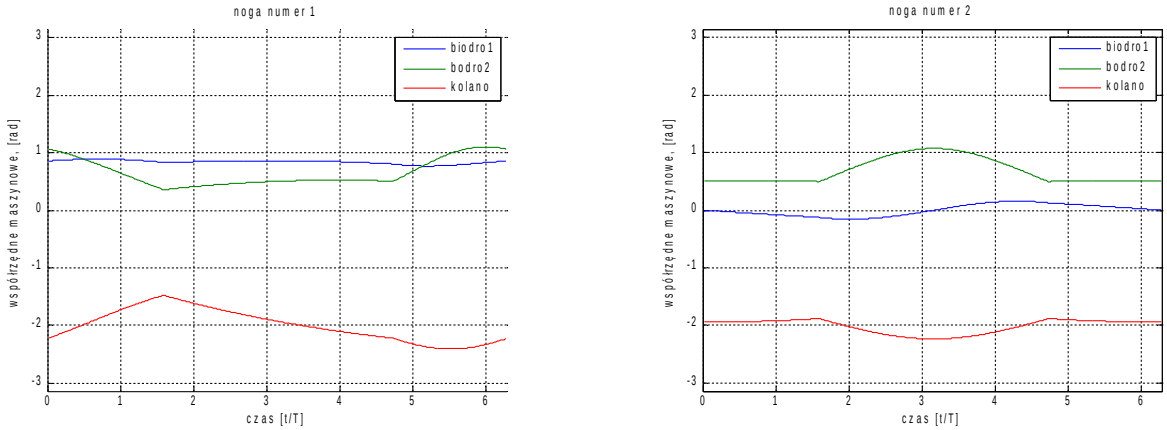


Rys. 31: przebieg trajektorii stóp robota w przypadku złożenia ruchu prostoliniowego i obrotowego

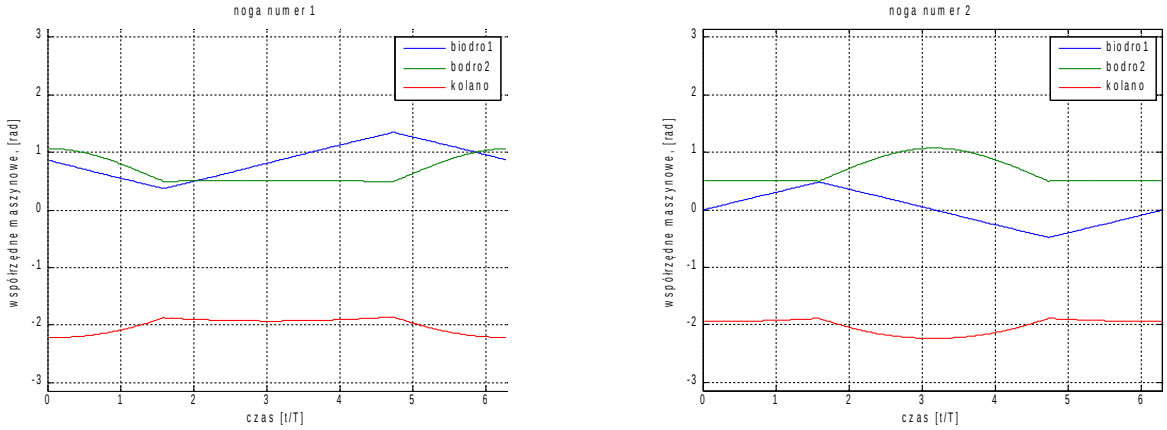
Korzystając z wyprowadzonych w drugim rozdziale zależności można obliczyć przebiegi współrzędnych maszynowych dla poszczególnych połączeń poszczególnych nóg. Na rysunkach 32-34 przedstawiono kilka przebiegów jednego okresu chodu dla różnych parametrów. Ze względu na symetrię przedstawiono tylko wykresy dla nogi przedniej i środkowej. Na przedstawionych wykresach przedział $\left[\frac{\pi}{4}, \frac{5\pi}{4}\right)$ odpowiada fazie podporowej nogi pierwszej (noga przednia) i fazie przenoszenia nogi drugiej (noga środkowa).



Rys. 32: Wykresy współrzędnych maszynowych nogi pierwszej i drugiej dla chodu prostoliniowego w kierunku 0 rad i długości kroku 100 mm.



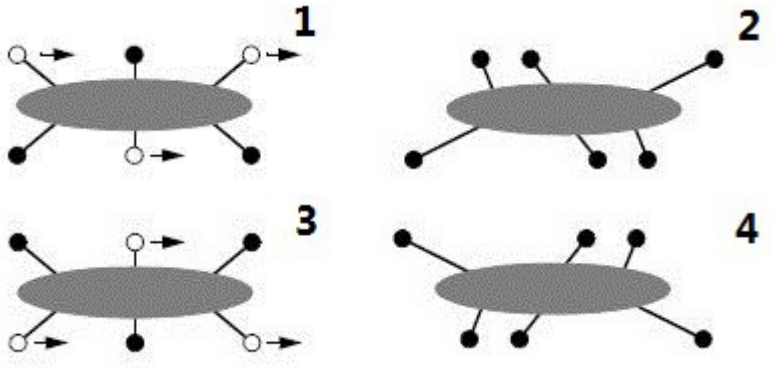
Rys. 33: Wykresy współrzędnych maszynowych nogi pierwszej i drugiej dla ruchu obrotowego omega = 0.05 i długości kroku w kierunku liniowym 100 mm.



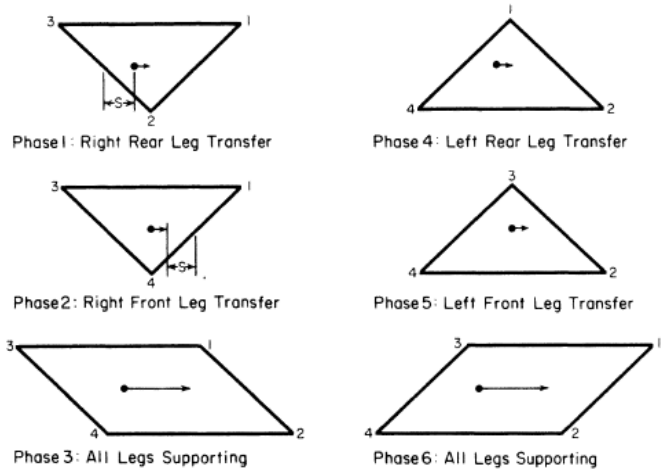
Rys. 34: Wykresy współrzędnych maszynowych nogi pierwszej i drugiej dla ruchu obrotowego omega = 0.2 i długości kroku w kierunku liniowym 0 mm.

4.5. Schemat kroków

Kolejnym aspektem generowania chodu jest sposób przenoszenia nóg. Jak pokazują badania, w przypadku zwierząt sześcionożych, i ośmiomożnych, najczęściej spotykane są chody polegające na falach kroków po każdej ze stron ciała zwierzęcia (najpierw przenoszona jest ostatnia noga, następnie poprzednia, i poprzednia). W przypadku owadów współczynnik obciążenia β (definiowany jako stosunek czasu trwania fazy podporowej do okresu chodu) wynosi 0.5 dla każdej pary nóg. Falowe przenoszenie stóp jest powszechne u owadów dla każdej prędkości poruszania się, oraz u czworonogów w przypadku małych prędkości.[3] Dla bardzo małych prędkości współczynnik obciążenia stóp znacznie przewyższa 0.5 (przenoszona jest tylko jedna noga, trzy są w fazie podporowej, w konsekwencji faza podporowa trwa zdecydowanie dłużej), chód taki został przedstawiony na rysunku 36.



Rys. 35: Chód sześcionożny, stabilny statycznie,[6]
czarne kropki – faza podporowa,
białe kropki – faza przenoszenia.



Rys. 36: Chód czteronożny, stabilny statycznie.[3]

4.6. Implementacja w robocie

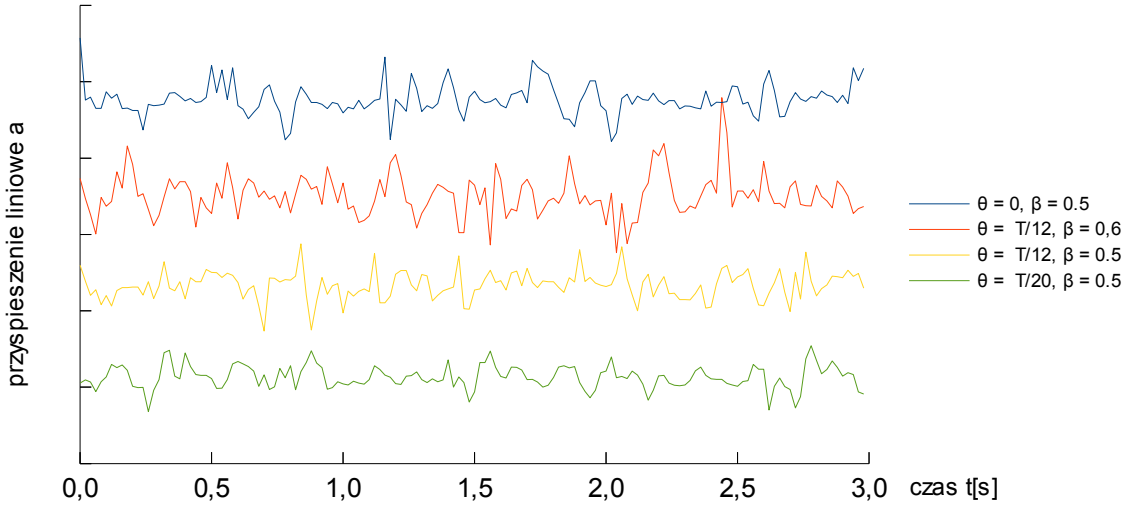
Początkowo w prototypie zaimplementowaliśmy chód trójpodporowy, o kształcie trajektorii stóp zaprezentowanej w tym rozdziale, ze współczynnikiem obciążenia stóp równym 0.5, oraz przesunięciem fazowym dla kolejnych par nóg poczynając od ostatniej wynoszącym 0, T, 2T. Biorąc pod uwagę, że przesunięcie o 2T jest równoważne z przesunięciem o pełną wartość okresu, oraz że nogi należące do jednej pary zawsze pracują w przesunięciu równym połowie okresu (gdy jedna jest w połowie fazy podporowej, przeciwna jest w połowie fazy przenoszenia), przy tak określonych warunkach, nogi są trójkami zgodne w fazie (noga pierwsza i ostatnia z danej strony i środkowa z przeciwnej).

Uzyskany efekt przy tak określonym algorytmie był zadowalający w czasie trwania fazy podporowej danej trójki nóg, jednak w momencie przejścia pomiędzy fazami obserwowane były znaczne drgania korpusu spowodowane zarówno uderzeniem o podłoże stóp wchodzących w fazę podporową jak i gwałtowną utratą podparcia przez nogi wchodzące w fazę przenoszenia. Możliwym rozwiązaniem zaistniałego problemu jest wygładzenie trajektorii w miejscach zmiany fazy. Pozwoliłoby to zmniejszyć prędkość, z jaką stopy uderzają w podłoże. Likwidację drgań wynikających ze zmiany zestawu nóg podporowych można także uzyskać poprzez nałożenie faz podporowych na siebie w taki sposób, aby pomiędzy dwoma wyróżnionymi fazami trójpodporowymi występowała faza, w której robot podpierany jest przez wszystkie nogi (zwiększenie współczynnika obciążenia). Kolejnym czynnikiem powodującym spotęgowanie wspomnianych wyżej efektów jest fakt, że nogi działają trójkami zgodnie w fazie –kontakt z podłożem trójki nóg pojawia się w dokładnie tym samym momencie – poprawę powinno zatem przynieść wprowadzenie dodatkowego przesunięcia fazowego.

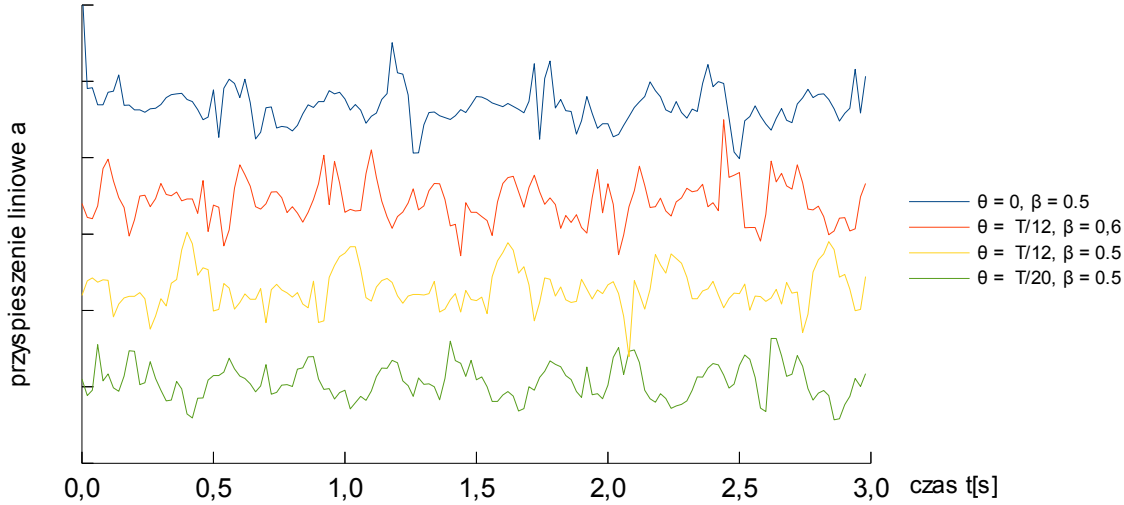
W celu poprawy stabilności platformy do algorytmu wprowadzone zostały dwa dodatkowe parametry:

- a, zdefiniowany jako stosunek czasu trwania fazy podporowej do fazy przenoszenia nóg,
- θ, zdefiniowany jako dodatkowe przesunięcie fazowe poszczególnych par nóg.

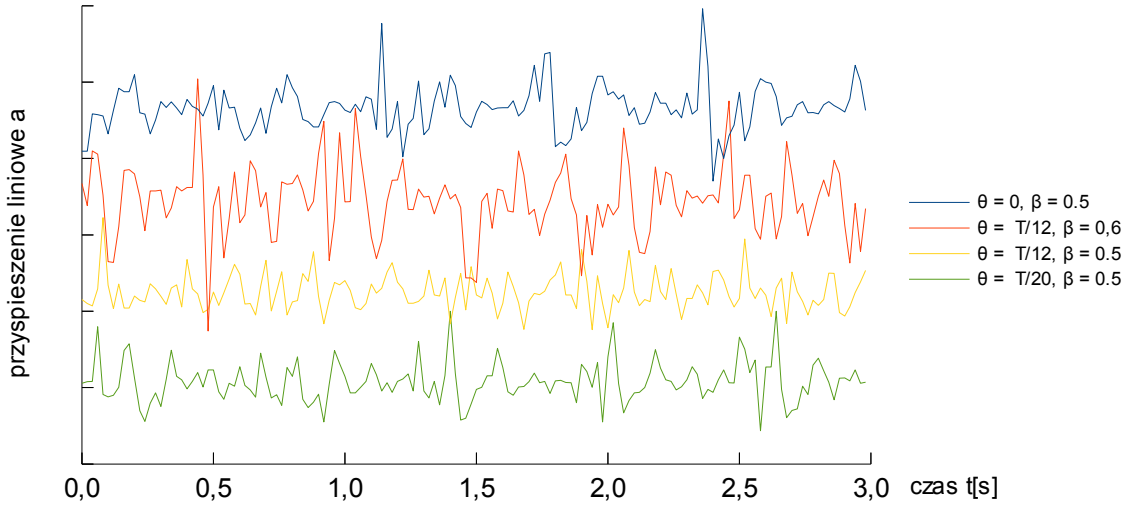
W celu pomiaru przyspieszeń liniowych oraz kątowych w trakcie chodu, na korpusie zamontowany został trzyosiowy żyroskop, oraz trzyosiowy akcelerometr. Zbadano wartości przyspieszeń liniowych oraz prędkości kątowych w trzech osiach, dla kilku różnych parametrów chodu. We wszystkich przypadkach prędkość chodu była jednakowa i stała, robot poruszał się ruchem prostoliniowym. Wyniki pomiarów zaprezentowano na rysunkach 37-42 oraz w tabeli 2. Oś X jest równoległa do kierunku chodu, oś Z jest osią pionową.



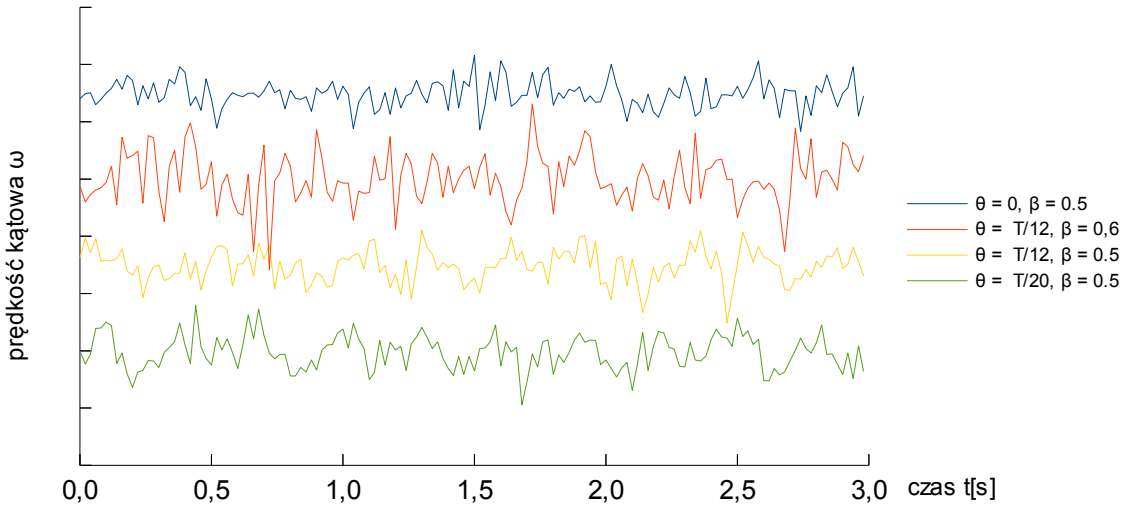
Rys. 37: Wykresy przyspieszeń liniowych w osi X.



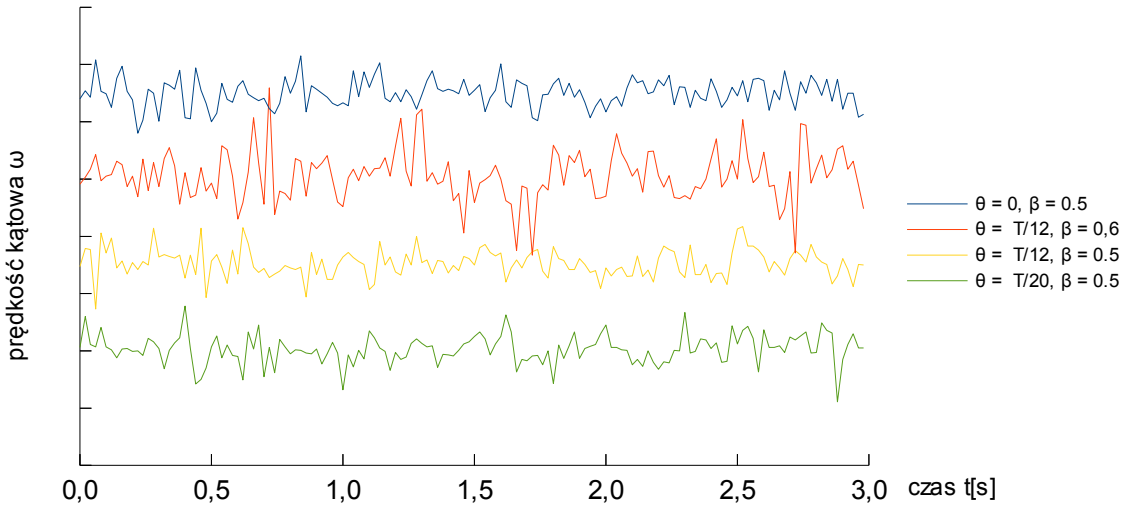
Rys. 38: Wykresy przyspieszeń liniowych w osi Y.



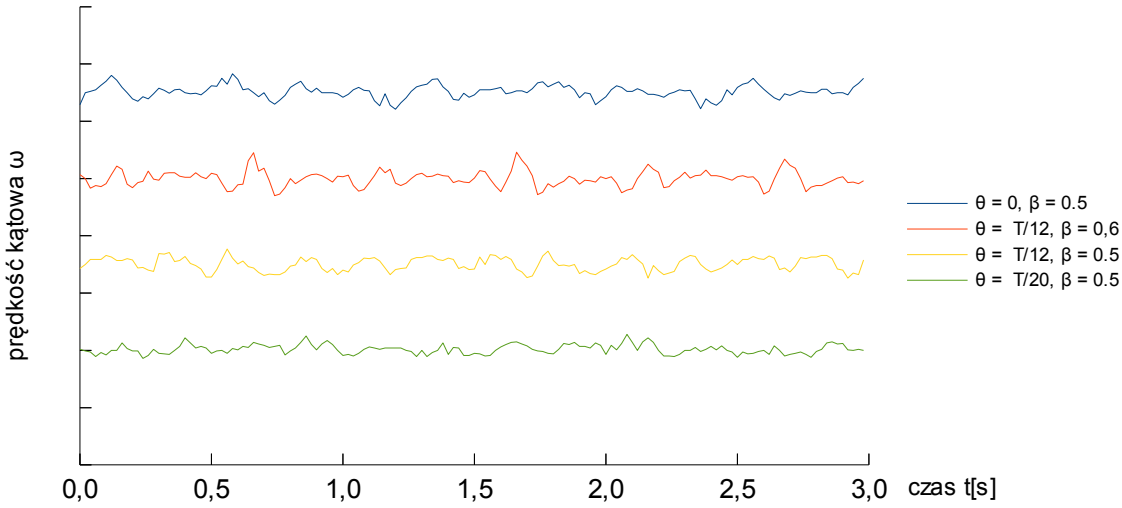
Rys. 39: Wykresy przyspieszeń liniowych w osi Z



Rys. 40: Wykresy prędkości kątowych w osi X



Rys. 41: Wykresy prędkości kątowych w osi Y



Rys. 42: Wykresy prędkości kątowych w osi Z

Tabela 2: Przyspieszenia liniowe i kątowe dla różnych parametrów ruchu

	θ = 0, β = 0.5	θ = T/12, β = 0,6	θ = T/12, β = 0.5	θ = T/20, β = 0.5
a_{XMAX} [g]	0,4240	0,5352	0,4458	0,2839
a_{YMAX} [g]	0,2769	0,6382	0,3002	0,2250
a_{ZMAX} [g]	0,6508	0,8537	0,4874	0,4607
ω_{XMAX} [rad/s]	0,1940	0,4251	0,2219	0,2554
ω_{YMAX} [rad/s]	0,1881	0,4398	0,2755	0,2537
ω_{ZMAX} [rad/s]	0,0866	0,1287	0,0738	0,0724
σ_{aX} [g]	0,1217	0,1293	0,1347	0,1075
σ_{aY} [g]	0,1040	0,1353	0,0915	0,0722
σ_{aZ} [g]	0,1372	0,2417	0,1181	0,1275
$\sigma_{\omega X}$ [rad/s]	0,0699	0,1205	0,0672	0,0671
$\sigma_{\omega Y}$ [rad/s]	0,0658	0,1275	0,0777	0,0846
$\sigma_{\omega Z}$ [rad/s]	0,0318	0,0387	0,0333	0,0240

Przez „MAX” oznaczono maksymalną różnicę pomiędzy wartością chwilową a średnią, przez symbol „σ” odchylenie standardowe odpowiednich wielkości. Na zielono zakreślono najlepsze uzyskane rezultaty.

Zmiana parametru *a* nie przyniosła spodziewanych efektów – powodem prawdopodobnie jest fakt, iż w zaimplementowanym algorytmie zwiększenie współczynnika obciążenia wiąże się z większą prędkością nóg w fazie przenoszenia, a co za tym idzie większą prędkością w punkcie zmiany fazy, przez co nogi silniej uderzają o podłoże. Zdecydowanie najlepsze efekty osiągnięte zostały dla przesunięcia w fazie o dodatkowe 0.05 okresu przy współczynniku β = 0.5, przesunięcie o 0.083 przyniosły podobne rezultaty jak brak przesunięcia fazowego. Efekt wprowadzenia przesunięcia można stwierdzić wizualnie – chód stał się dużo płynniejszy, brak zatrzymań w kierunku X oraz gwałtownych przyspieszeń w kierunku osi Z w momencie zmiany trójki nóg podporowych.

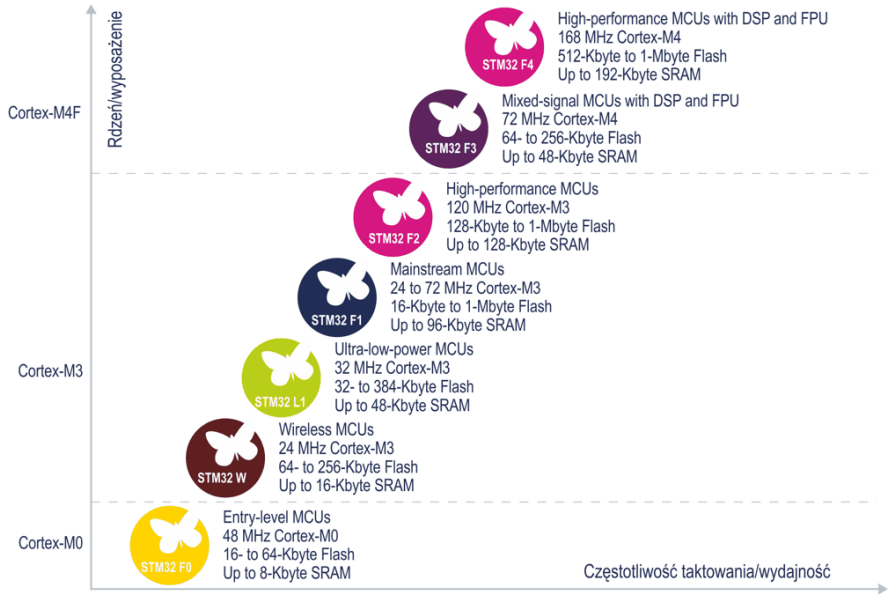
Rozdział 5

Oprogramowanie mikrokontrolera

5.1. Opis urządzenia

Jedno z ogniw łańcucha systemu sterowania platformy stanowi mikrokontroler. Jego zadaniem jest niskopoziomowe obliczanie pożądanych położeń stóp robota, realizacja odwrotnego zagadnienie kinematyki oraz generowanie sekwencji sterujących serwami. W mikrokontrolerze zaimplementowany został algorytm ruchu oraz zaimplementowana zostanie w przyszłości obsługa planowanej sensoryki nóg oraz korpusu.

Proponowanym przez nas urządzeniem jest mikrokontroler STM32 firmy STMicroelectronics. Konkretnym modelem zastosowanym w prototypie jest STM32F103VB.



Rys. 43: Wykres wydajności mikrokontrolerów STM32

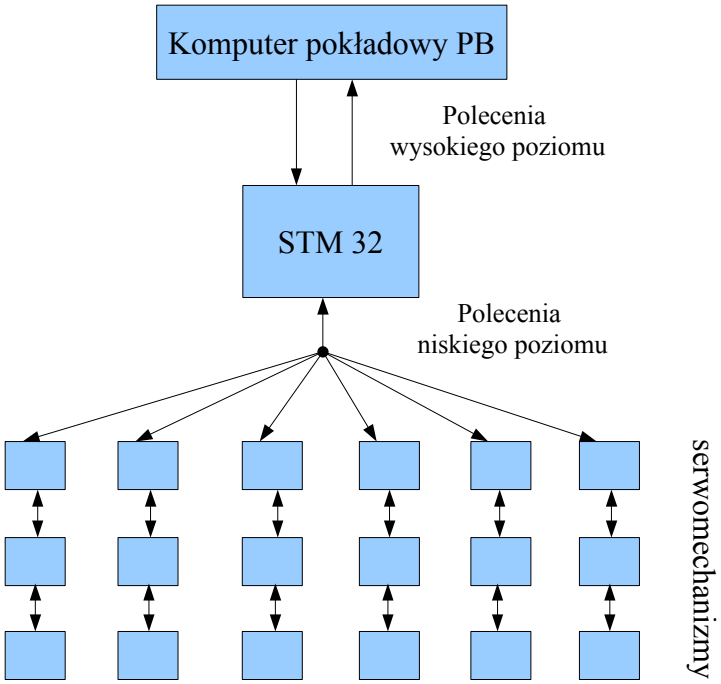
STM32 wyposażony został w procesor Cortex-M3. Rodzina rdzeni Cortex składa się z trzech grup procesorów :

- Cortex-Ax, przeznaczonych do pracy z systemami operacyjnymi (Linux, Symbian),
- Cortex-Rx, przeznaczonych do realizacji systemów czasu rzeczywistego,
- Cortex-Mx, zoptymalizowanych pod kątem ceny i wydajności do zastosowań konsumenckich i przemysłowych.

Cortex-M3 jest rdzeniem 32 bitowym, taktowanym częstotliwością do 72MHz.

5.2. System sterowania nogami

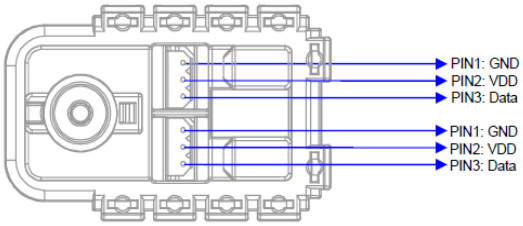
Schemat działania systemu sterowania nogami:



Rys. 44: Schemat układu sterowania robota.

Komunikacja mikrokontroler – serwomechanizmy odbywa się szeregowo z wykorzystaniem interfejsu USART w trybie half-duplex. Interfejs USART jest jednym z wielu interfejsów komunikacyjnych zaimplementowanych w STM32, przystosowanym do współpracy z układem bezpośredniego dostępu do pamięci DMA.[4] Tryb half-dupleks jest trybem, w którym transmisja w obu kierunkach odbywa się przy użyciu jednej linii komunikacyjnej. Jest to jedyny tryb komunikacji realizowany w zastosowanych przez nas serwach. Komunikacja komputer – mikrokontroler została opisana w innym rozdziale.

Schemat wyprowadzeń serwa AX-12 przedstawiono na rysunku 45.



Rys. 45: Schemat wyprowadzeń serwomechanizmów AX-12[2]

Serwa AX-12 zostały wyposażone w dwa gniazda po trzy piny. Odpowiadające sobie piny są wewnętrznie zwarte, co umożliwia szeregowe łączenie serw. Jedno gniazdo w takim przypadku pracuje jako wejście, drugie jako wyjście. Gniazda wyposażone są w trzy piny, odpowiednio: masa, napięcie zasilania, łącze danych. Producent umieścił pin zasilający na środku, co zabezpiecza przed uszkodzeniem serwomechanizmów nawet w przypadku nieprawidłowego podłączenia kabla (napięcie i tak jest zawsze na tym samym pinie).

5.3. Komunikacja

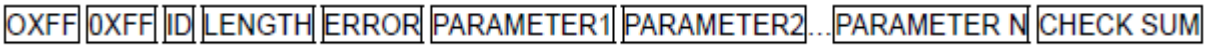
Komunikacja odbywa się przy napięciach rzędu 5V. Możliwe jest nadawanie i odbiór w kilku określonych prędkościach, z górną granicą 1Mbps. W naszym przypadku nie udało się uzyskać komunikacji z tą prędkością, zastosowaliśmy 0.5Mbps. Interfejs komunikacji narzucony przez producenta definiuje kształt ramki:

OXFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	...	PARAMETER N	CHECK SUM

Przy czym:

0xFF, 0xFF	są dwoma bajtami otwierającymi każdą ramkę.
ID	jest identyfikatorem serwa,
LENGTH	jest długością ramki, przy czym jej wartość jest równa liczbie parametrów (N) + 2
INSTRUCTION	jest identyfikatorem instrukcji
PARAMETER1...N	jest parametrem instrukcji
CHECK SUM	jest sumą kontrolną, liczoną w poniższy sposób: Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N). W przypadku, gdy obliczona wartość przekracza 0xFF to sumą kontrolną staje się młodszy bajt obliczonej wartości.

Po odebraniu ramki danych serwo odpowiada ramką powrotną:



Jest ona analogiczna do ramki wysyłanej w kierunku serwa.

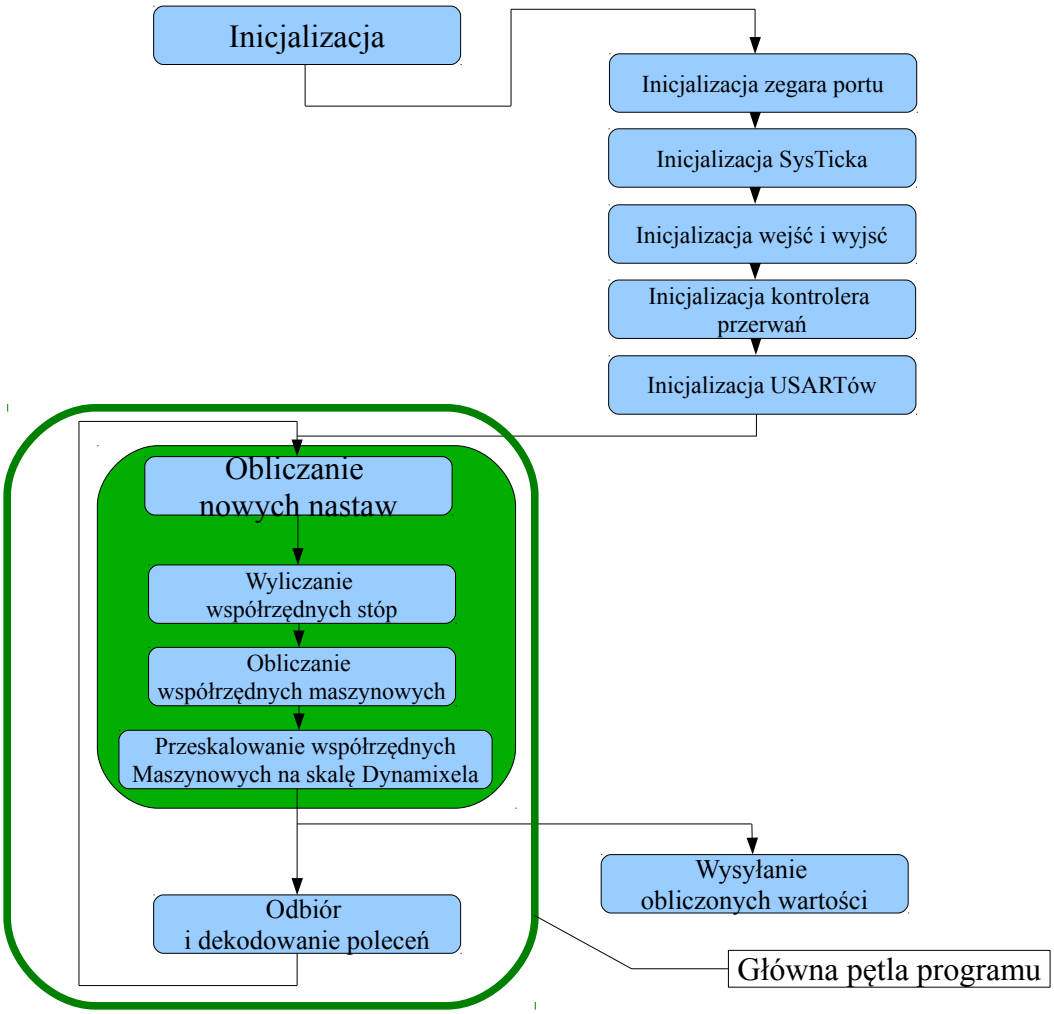
5.4. Sterowanie serwami

Nastawianie wartości zadanych położenia i prędkości poszczególnych serwomechanizmów jest możliwe, między innymi, poprzez bezpośrednie wpisywanie odpowiednich wartości do odpowiednich miejsc w pamięci tych urządzeń. Serwa AX-12 są zdolne do komunikacji w trybie rozgłoszeniowym, co w znacznym stopniu zwiększa prędkość komunikacji w przypadku dużej ilości tych urządzeń. Główną zaletą jest brak odpowiedzi na otrzymane dane wysyłanej przez serwa, dzięki czemu nie ma konieczności przełączania się kontrolera w tryb nasłuchu po wysłaniu nastaw do każdego serwa, oczekiwania na odpowiedź i ponownego przełączania się w tryb nadawania. Nie bez znaczenie pozostaje też fakt zmniejszenia ilości transmitowanych danych – początek ramki i suma kontrolna, oraz instrukcja wysyłane są tylko raz. Ramka danych w tym trybie komunikacji została opisana dokładnie w nocie katalogowej urządzania, nie będzie więc tu bliżej omawiana.

Do komunikacji został wykorzystany interfejs USART zaimplementowany w mikrokontrolerze. W trybie half-duplex wykorzystywany jest tylko pin nadawczy TX. Pozostałe piny portu zostają niewykorzystane. W prototypie wszystkie serwa połączone są z jednym portem mikrokontrolera. Docelowo planowany jest podział zadania komunikacji pomiędzy kilka portów. (STM32 dysponuje pięcioma takimi portami). W zaimplementowanym systemie, USART jest zdolny do współpracy z układem DMA (Direct Memory Access). Dzięki temu dane wysyłane są współbieżnie do wykonywanych obliczeń, co przez wzgląd na stosunkowo dużą ilość danych i stosunkowo wolną prędkość transmisji, oznacza istotne przyspieszenie działania systemu.

5.5. Program sterujący

5.5.1. Struktura programu



Rys. 46: Struktura programu kontroli kończyn

Program odczytuje na wejściu jednego z USARTów dane z komputera pokładowego dane, których dokładniejszy opis zawarty jest w rozdziale 9. Na ich podstawie wyliczane są pożądane ustawienia nóg robota w kolejnych chwilach czasu.

Pozycja stóp zamieniana jest na współrzędne kątowe poszczególnych par kinematycznych, a te z kolei przeliczane są na odpowiednią liczbę wysyłaną następnie do aktuatorów. Do realizacji zadania obliczania trajektorii stopy zaimplementowany został algorytm przedstawiony w poprzednich rozdziałach. Obliczane są, w zależności od fazy ruchu, kolejne dyskretne położenia stopy, dla każdej nogi oddzielnie. Każda noga ma przypisaną własną zmienną (y[i]) przechowującą informację o fazie ruchu. Zastosowanie oddzielnych zmiennych pozwala na wprowadzenie indywidualnego przesunięcia fazowego dla każdej nogi. Każda iteracja powoduje inkrementację zmiennej y[i], a inkrement zależy

od żądanej prędkości działania mechanizmu. Funkcje trygonometryczne, w celu usprawnienia działania programu, zostały stabelaryzowane.

```
int x = 0;
int z = 0;
int i;
float zakres = PI/2;
float promien = wysokosc*0.2;
float incr = (speed*time)/300.0;

for(i = 0; i< 6; i++)
{
	if(y[i] > PI) y[i] -=2*PI;
	if(y[i] < -PI)y[i] +=2*PI;
}

for( i = 0; i<6; i++)
{

	if(Cos(y[i]) > 0)
	{
		tmp1 = (x+wspssr[i][0]+Sin(direction) * odleglosc*Sin(y[i]));
		tmp2 = (wspssr[i][1] + Cos(direction) * odleglosc*Sin(y[i]));

		wspstopy[i][0] = Cos(y[i] *obrot)*tmp1 + Sin(y[i]*obrot)*tmp2 ;
		wspstopy[i][1] = -Sin(y[i] *obrot)*tmp1 + Cos(y[i]*obrot)*tmp2;
		wspstopy[i][2] = z+wysokosc*Cos(y[i])*Cos(y[i]);
		y[i] = y[i] + incr*nakladka;

	}
	else
	{
		tmp1 = (x+wspssr[i][0]+Sin(direction) * odleglosc*AsinSin(y[i])*2/PI);
		tmp2 = (wspssr[i][1]+Cos(direction) * odleglosc*AsinSin(y[i])*2/PI);

		wspstopy[i][0] =  Cos(AsinSin(y[i])*obrot)*tmp1 + Sin(AsinSin(y[i])*obrot)*tmp2;
		wspstopy[i][1] = -Sin(AsinSin(y[i])*obrot)*tmp1 + Cos(AsinSin(y[i])*obrot)*tmp2;
		wspstopy[i][2] = z;
		y[i]=y[i] + incr;
	}
}
```

Listing 1: Kod odpowiadający za obliczenia trajektorii stóp.

Na obecnym etapie prac robot działa bez sprzężenia od jakichkolwiek czujników położenia nóg. Planowane jest w przyszłości uwzględnienie w algorytmie poruszania oraz generowania chodu odczytu z czujników, w tym odczyt parametrów pracy serw. W szczególności serwa mogą zwracać moment obciążający, na podstawie którego jesteśmy w stanie obliczyć siłę nacisku stopy, oraz wysokość na której nastąpił kontakt z podłożem (nagły wzrost momentu, lub nawet zmiana jego znaku).

5.5.2 Działanie programu

Zaimplementowany program wykonuje się około 200 razy na sekundę, co daje odświeżanie pozycji serw co 5ms. Przy takiej częstotliwości oraz stosunkowo dużej bezwładności aktuatorów nie ma potrzeby obliczania pożądanej prędkości serw, wystarczy ustawić ją jednorazowo na wartość maksymalną. Położenie próbkowane jest na tyle szybko, że nie obserwujemy skokowych zmian położenia, a ruch robota jest płynny.

Rozdział 6

System sterowania

6.1. Wprowadzenie

Sterowanie jest podstawowym zagadnieniem przy budowie robotów dowolnego typu. Sterowanie można określić jako odpowiednie kształtowanie sygnałów wejściowych tak, aby uzyskać pożądaną postać sygnałów wyjściowych. Rozdział ten przedstawia sposób, w jaki rozkazy wytworzone przez algorytm sterowania na podstawie sygnałów wejściowych przemierzają drogę od ośrodka decyzyjnego do efektorów. Opisuje on zarówno elektroniczne urządzenia biorące udział w komunikacji jak i protokoły, które definiują tę komunikację. Algorytmy sterowania zaimplementowane w robocie opisane zostały w rozdziale "Programy sterujące".

Projektowanie systemu sterowania robotem wiąże się z koniecznością postawienia precyzyjnych wymagań, które powinien on spełniać. W przypadku każdego typu maszyny wywierającej fizycznie wpływ na swoje otoczenie podstawowym wymaganiem jest jak największy stopień niezawodności komunikacji lub odpowiednie zabezpieczenie maszyny przed uszkodzeniem jej lub otoczenia. Sześcionożny robot kroczący, będący przedmiotem pracy, przy obecnych rozmiarach, nie stanowi zagrożenia dla środowiska. Duża zawodność komunikacji operatora z robotem zmniejsza jednak jakość obsługi urządzenia. W celu uzyskania pełnej mobilności robota, komunikacja z operatorem musi odbywać się bezprzewodowo.

Kolejnym ważnym parametrem, wpływającym bezpośrednio na jakość obsługi robota jest prędkość komunikacji pomiędzy odpowiednimi urządzeniami. Komunikacja ośrodka decyzyjnego z efektorami robota powinna być możliwie jak najszybsza. Opracowywany sześcionożny robot kroczący składa się z dużej liczby serwomechanizmów realizujących skomplikowany algorytm ruchu. Duża szybkość komunikacji daje możliwość ustawiania wartości zadanych silników z dużą częstotliwością. Skutkuje to szybką reakcją robota

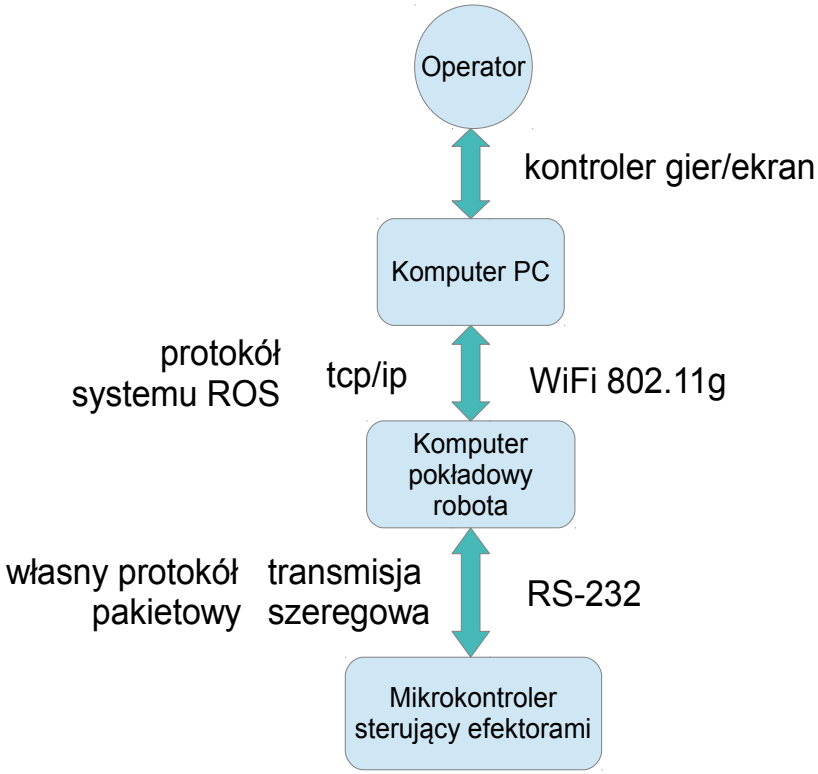
na zmiany środowiska oraz znacznie zwiększa płynność jego ruchów, co z kolei pozwala na lepszą stabilizację platformy.

Ostatnim wymaganiem dotyczącym systemu sterowania jest skalowalność. Urządzenia składające się na system powinny być dobrane tak, aby umożliwiać rozbudowę o kolejne sensory. Protokoły komunikacji pomiędzy nimi powinny umożliwiać przesył danych z tych czujników.

6.2. System sterowania zastosowany w robocie

6.2.1. Opis systemu

Na potrzeby robota będącego przedmiotem pracy zaprojektowano i zaimplementowano system sterowania spełniający wyżej przedstawione wymagania. Ogólny schemat tego systemu przedstawia rysunek 47.



Rys. 47.Schemat zastosowanego systemu sterowania robotem

Na schemacie, z prawej strony strzałek pomiędzy urządzeniami umieszczono nazwy warstw fizycznych realizujących komunikację. Po lewej stronie przedstawiono nazwy protokołów warstw wyższych.

Operator steruje robotem za pomocą dowolnego komputera klasy PC wyposażonego w adapter Wi-Fi oraz specjalny program. Robot nosi ze sobą dwa urządzenia – własny komputer pokładowy, podejmujący decyzje wysokiego poziomu oraz mikrokontroler zajmujący się podejmowaniem decyzji niskopoziomowych.

Komputer pokładowy jest głównym ośrodkiem decyzyjnym robota. W jego obrębie, na podstawie przetwarzania obrazu z kamery umieszczonej na robocie lub danych o stanie kontrolera gier odebranych od komputera PC tworzone są rozkazy ruchu wynikające z zaimplementowanego algorytmu sterowania. Urządzenie wysyła rozkazy do mikrokontrolera sterującego serwomechanizmami oraz odbiera od niego dane z sensorów. Do komputera PC wysyłane są informacje potrzebne do wizualizacji stanu robota.

Do zadań mikrokontrolera należy wykonywanie rozkazów ruchu wydawanych przez komputer pokładowy. W tym celu steruje on wszystkimi serwomechanizmami robota zgodnie z zaimplementowanym w nim algorytmem ruchu przedstawionym we wcześniejszych rozdziałach pracy. Do mikrokontrolera podłączone są również sensory takie jak akcelerometr, żyroskop lub sensory nacisku odnóż robota. Kolejnym zadaniem jest odczytywanie danych z tych sensorów oraz przesyłanie części z nich do komputera pokładowego. Na poziomie mikrokontrolera zaimplementowane ma być także sprzężenie zwrotne od sensorów korygujące stawianie członów robota.

Zadaniem komputera PC jest przełączanie trybu działania robota - autonomicznego lub zdalnej kontroli. W trybie zdalnej kontroli odbierany jest obraz z kamery i wysyłany jest stan kontrolera gier podłączonego do urządzenia. Komputer PC odbiera również dane od komputera pokładowego robota oraz przedstawia je operatorowi przy pomocy ekranu.

6.2.2. Zastosowane urządzenia

W robocie będącym przedmiotem pracy rolę komputera pokładowego spełnia system wbudowany PandaBoard ES. Jest on oparty na układzie OMAP i posiada złącza RS-232, Ethernet, oraz interfejsy WiFi i Bluetooth. Dokładny opis tego urządzenia znajduje się w dalszym rozdziale. Funkcję mikrokontrolera sterującego serwomechanizmami spełnia płytka ewaluacyjna zawierającą mikrokontroler o architekturze ARM.

6.2.3. Komunikacja pomiędzy PC a komputerem pokładowym

Do realizacji komunikacji pomiędzy komputerem PC a systemem PandaBoard wykorzystano standard 802.11, potocznie nazywany Wi-Fi. Jest to obecnie jeden z najpopularniejszych standardów wykorzystywanych do budowy bezprzewodowych sieci

komputerowych[3]. Umożliwia on komunikację z teoretyczną prędkością 54 Mb/s [7]. Zaletą jest jego duża powszechność. Obecnie prawie wszystkie produkowane komputery przenośne typu notebook są wyposażone w adapter realizujący ten standard.

Teoretycznie zasięg sieci Wi-Fi wynosi od 90 metrów w zamkniętych pomieszczeniach do 150 na zewnątrz. W praktyce występuje jednak wiele zakłóceń, na które sieć Wi-Fi jest podatna. Częstotliwość, która jest wykorzystywana przez wybrany standard 802.11g wynosi 2,4 GHz. W tym samym zakresie pracują również urządzenia takie jak Bluetooth, kuchenki mikrofalowe, telefony bezprzewodowe, radary meteorologiczne, radiowa telewizja przemysłowa oraz wiele innych. W efekcie dochodzić może do chwilowych zaników połączenia pomiędzy robotem a operatorem.

Aby zmaksymalizować jakość połączenia należy użyć nadajnika wytwarzającego silny sygnał. Z powodu rozmiarów takich nadajników, obowiązek tworzenia punktu dostępu powinien być przeniesiony na stronę operatora. Takie rozwiązanie minimalizuje zapotrzebowanie na energię robota oraz rozmiar noszonych przez niego urządzeń. Drugie rozwiązanie prezentuje bardziej zwarte podejście. Po uruchomieniu robot tworzyłby punkt dostępu o znanych parametrach, do którego podłączaliby się operatorzy. Likwiduje to potrzebę dodatkowego urządzenia, którego obsługa należałaby do obowiązków operatora. Podczas rozwoju robota opisanego w tej pracy wykorzystywane jest rozwiązanie pierwsze.

Bilans przedstawionych powyżej wad i zalet wykazuje, że wykorzystanie standardu Wi-Fi jest odpowiednie na czas badań i rozwoju robota. Końcowe wersje robota powinny być wyposażone w bardziej wyspecjalizowane rozwiązanie, które jest lepiej przystosowane pod względem wymienionych wyżej parametrów.

Protokołem wykorzystywanym do skomunikowania komputera PC z komputerem pokładowym robota jest TCP/IP. Komunikacja pomiędzy wyższymi warstwami sieci realizowana jest poprzez system ROS i opisana została w kolejnych rozdziałach.

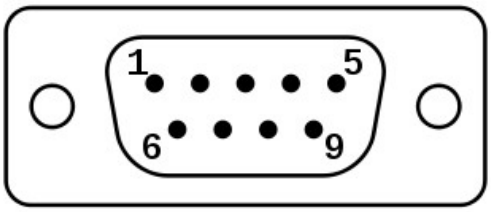
6.2.4. Komunikacja pomiędzy komputerem pokładowym a mikrokontrolerem

Komunikacja pomiędzy systemem PandaBoard a mikrokontrolerem sterującym serwomechanizmami zrealizowana została przy pomocy standardu RS-232. Określa on nazwy styków złącza oraz przypisane im sygnały a także specyfikację elektryczną obwodów wewnętrznych[4]. Definiuje też normy wtyczek i kabli portów szeregowych. Standard ten, opracowany w 1962 roku, wciąż jest bardzo powszechny w zastosowaniach technicznych.

3 http://pl.wikipedia.org/wiki/Wi-Fi
4 http://pl.wikipedia.org/wiki/RS-232

Tabela 3. Opis złącza standardu RS-232

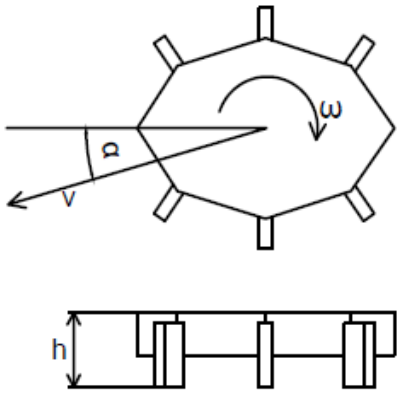
Numer pinu	Oznaczenie	Nazwa
1	DCD	sygnał wykrycia nośnej
2	RxD	odbiór danych
3	TxD	transmisja danych
4	DTR	gotowość terminala
5	GND	masa
6	DSR	gotowość "modemu"
7	RTS	żądanie wysłania
8	CTS	gotowość wysłania
9	RI	wskaźnik dzwonka



Rys. 48.Widok gniazda PC typu DE-9

Protokół komunikacji warstwy aplikacji

Aby sterować ruchem robota, komputer pokładowy wysyła do mikrokontrolera następujące parametry: kąt wektora prędkości, wartość wektora prędkości, wartość prędkości obrotowej, wysokość platformy nad ziemią oraz kąt pochylenia kamery. Każda z tych wartości przechowywana jest jako osobny bajt. W języku C reprezentowane są przez typ *unsigned char*. Zajmuje on dokładnie jeden bajt i umożliwia zapis liczby z zakresu od 0 do 255.



Rysunek 49.Parametry służące do sterowania robotem

Ramkę rozpoczyna bajt 0xFF. Kąt wektora predkości α przyjmuje wartości od 0 do 200 z punktem centralnym 100. Wartości mniejsze od 100 oznaczają kierunek “w lewo”, wartości większe zaś - “w prawo”. Podobnie zakodowana jest wartość predkości obrotowej ω.

Kąt pochylenia kamery jest zdyskretyzowany co jeden stopień i jego zakres wynosi od 0 do 90 stopni. Ramka zakończona jest bajtem 0xFE.

Ramkę można rozbudować o kolejne informacje. Ilość informacji w ramce jest ustalona *explicite* w programach robota. W przyszłości, powinna być ona uzupełniona o długość, a na jej końcu należałoby dodać sumę kontrolną, która umożliwi sprawdzanie poprawności i wyeliminowanie błędnych rozkazów. Schemat ramki został przedstawiony na poniższym rysunku (rys. 50).

<table>
<tr><td colspan="7">0B 7B</td></tr>
<tr><td rowspan="2">0xFF</td><td>kąt</td><td>prędkość</td><td>obrót</td><td>wysokość</td><td>kąt kamery</td><td rowspan="2">0xFE</td></tr>
<tr><td>0 - 200</td><td>0 - 200</td><td>0 - 200</td><td>0 - 200</td><td>0 - 90</td></tr>
</table>

Rys. 50. Ramka danych wykorzystywana w komunikacji

6.3. Podsumowanie

Opisana w punkcie 6.2 koncepcja systemu sterowania spełnia postawione na początku rozdziału wymagania w satysfakcjonującym stopniu. System ten jest na tyle uniwersalny, że przy niewielkich zmianach protokołów komunikacji może być zastosowany w innych typach robotów mobilnych.

Rozdział 7

Konfiguracja komputera pokładowego

7.1. Wprowadzenie

Każdy robot, który ma posiadać funkcje autonomiczne bazujące na wizji maszynowej, wymaga elektroniki o odpowiedniej mocy obliczeniowej. Roboty mobilne często wykorzystują urządzenia takie jak kamera, porty wejścia/wyjścia, mikrofon czy służący do przetwarzania obrazów układ graficzny. Urządzenia te muszą być obsługiwane przez procesor. Często wymagane jest też zrównoleglenie wielu procesów. Przykładem może być proces monitorowania wystąpienia zagrożeń robota, takich jak uderzenie w przeszkodę, który musi być przeprowadzany niezależnie od innych funkcji robota. Do wypełniania tego typu zadań potrzebny jest system operacyjny, który potrafi sprawnie zarządzać wieloma urządzeniami oraz wykonywaniem wielu zadań jednocześnie (multitasking).

Prostym rozwiązaniem jest zastosowanie komputera przenośnego w roli komputera pokładowego robota. Takie podejście pozwala na łatwy rozwój oprogramowania i konfigurację sprzętu – nie istnieje wtedy potrzeba kompilacji skrośnej programów z architektury komputera, na którym tworzone jest oprogramowanie, na architekturę mikrokontrolera. Istnieje również możliwość bezpośredniego wglądu do komputera pokładowego poprzez monitor komputera przenośnego. Jednakże z powodu nadmiernej liczby funkcji i rozmiarów zwykłych komputerów użytkowych nie jest to zazwyczaj rozwiązanie optymalne. Przy budowie robotów o mniejszych rozmiarach, takich jak robot będący przedmiotem niniejszej pracy, wymagane jest zastosowanie sprzętu bardziej wyspecjalizowanego, o mniejszych rozmiarach.

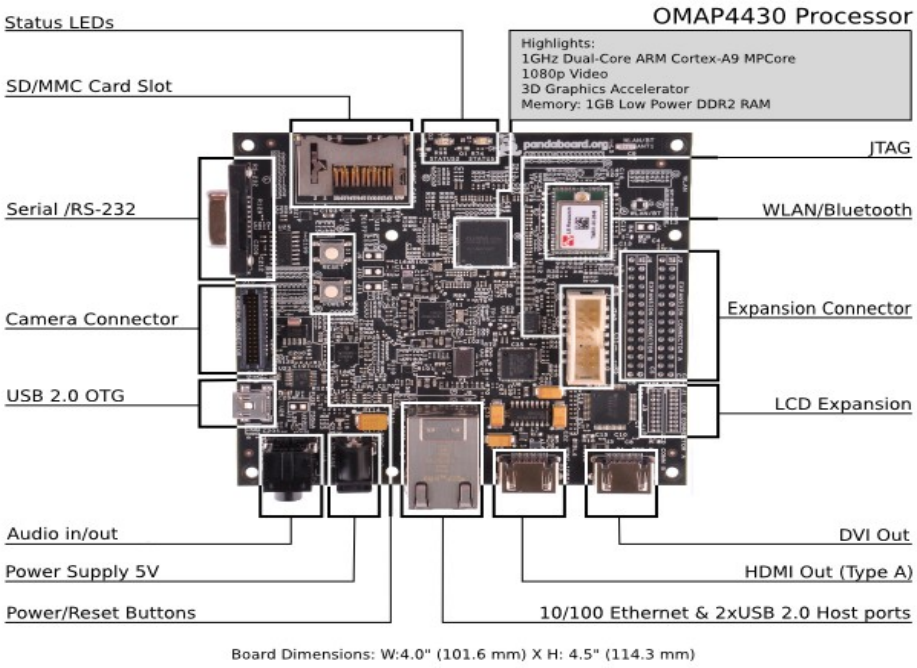
Zapotrzebowanie to, mające źródło w dziedzinach starszych niż robotyka mobilna, spowodowało powstanie klasy sprzętu elektronicznego nazwanego wbudowanymi systemami sterowania (Embedded Systems). Są to systemy komputerowe specjalnego przeznaczenia, oparte na mikrokontrolerach, które stają się częścią obsługiwanego przez nie sprzętu[5].

5 http://pl.wikipedia.org/wiki/System_wbudowany

Systemy tego typu stosowane są we wszystkich dziedzinach – urządzeniach AGD, telefonach komórkowych itp. Spowodowane jest to obecnym trendem uczynienia wszystkich urządzeń bardziej inteligentnymi.

7.2. Urządzenie zastosowane w pracy

Do spełniania funkcji komputera pokładowego robota budowanego w ramach tej pracy wykorzystano system wbudowany PandaBoard. Jest on oparty na popularnie stosowanym w telefonach komórkowych układzie OMAP (Open Multimedia Application Platform). Płytka PandaBoard wyposażona jest w wiele różnych peryferiów, między innymi port RS-232, moduł Wi-Fi oraz Bluetooth, złącze video HDMI oraz 2 porty USB. Ogólne zestawienie parametrów systemu przedstawia dostarczony przez producenta rysunek, który został zamieszczony poniżej.



Rys. 51.Specyfikacja układu PandaBoard

7.3. Wybór systemu operacyjnego

Do pracy na przedstawionym powyżej urządzeniu wybrany został system operacyjny z rodziny GNU/Linux. Powodem podjęcia takiej decyzji jest między innymi duża przejrzystość mechanizmów działania systemu oraz dostępność kodu źródłowego. Pozwala to na modyfikacje systemu do potrzeb zagadnienia. W wyniku tego system operacyjny w mniejszym stopniu zajmuje zasoby sprzętu, co skutkuje również mniejszym zapotrzebowaniem na energię, istotnym przy zasilaniu akumulatorami.

W celu minimalizacji zużycia energii oraz zwiększenia efektywności działania systemu istnieje możliwość własnej kompilacji jądra systemu oraz programów i bibliotek potrzebnych do działania robota. Rozwiązanie takie jest jednak opłacalne jedynie w końcowej fazie rozwoju robota, w której znane są wszystkie potrzebne programy. Istnieje wówczas możliwość przygotowania obrazu całego systemu, który może być łatwo wgrany na komputer pokładowy. Zazwyczaj nie dochodzi wtedy do dużych zmian oprogramowania.

W przypadku robota będącego przedmiotem pracy opłacalne jest wykorzystanie gotowej dystrybucji systemu Linux, która ponadto zawiera repozytoria z popularnym oprogramowaniem przygotowanym na architekturę komputera. Kompilowane ze źródeł muszą być wtedy jedynie programy, które nie są zawarte w repozytoriach. Przyspiesza to w znacznym stopniu dynamiczny rozwój oprogramowania robota.

Ze względu na duże zasoby repozytoriów, wsparcie społeczności oraz kompatybilność z oprogramowaniem, które ma być wykorzystywane w projekcie w pracy wykorzystano dystrybucję Ubuntu Linux.

7.4. Instalacja systemu operacyjnego

Instalacja systemu operacyjnego na systemach wbudowanych może przebiegać w różny sposób, w zależności od parametrów systemu. W przypadku systemów posiadających jedynie złącze RS-232 oraz port Ethernet stosowana jest technologia PXE (Preboot Execution Environment). Jest to tryb pracy urządzenia, który pozwala na uruchomienie systemu operacyjnego, mimo że nie jest on na nim zainstalowany[6]. W trybie tym komputer łączy się z serwerem obsługującym protokoły DHCP oraz TFTP, z którego pobiera system operacyjny. W dalszej kolejności, za pomocą uruchomionego przez PXE systemu operacyjnego, instaluje się go na pamięci zewnętrznej komputera.

W przypadku systemów wbudowanych posiadających wyjście video i porty USB istnieje możliwość instalacji systemu operacyjnego z specjalnie przygotowanej pamięci USB, podpinając do urządzenia klawiaturę oraz monitor.

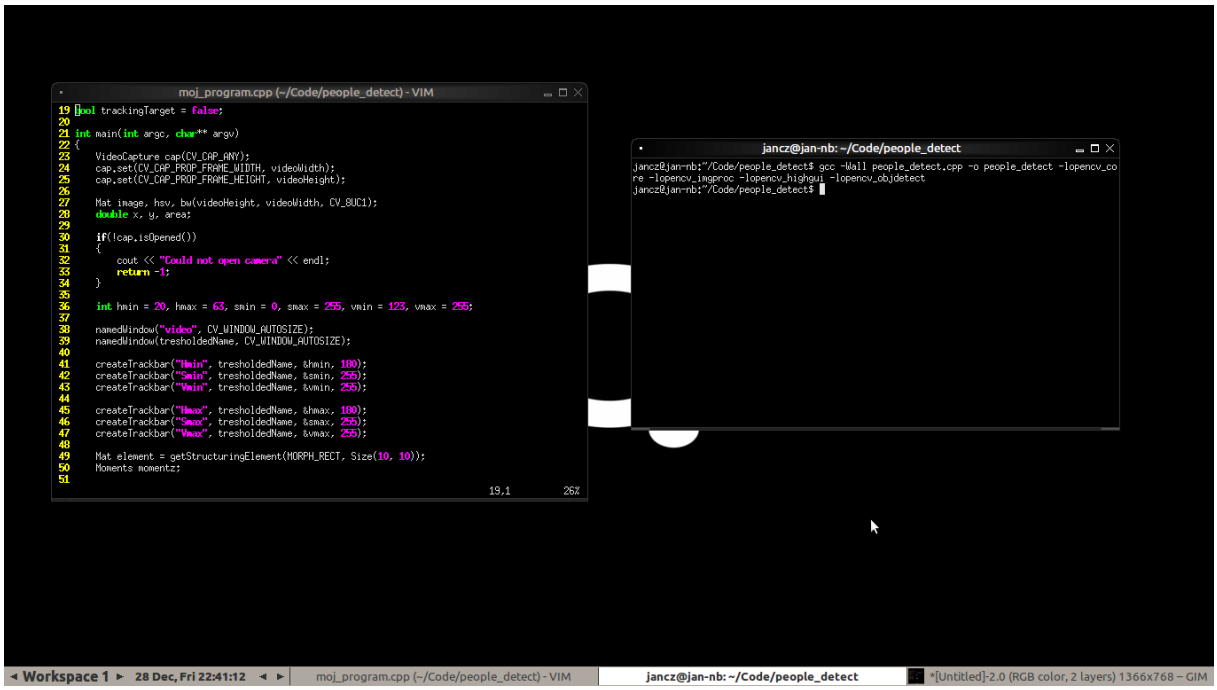
6 http://pl.wikipedia.org/wiki/Preboot_Execution_Environment

7.5. Konfiguracja systemu operacyjnego

7.5.1. Konfiguracja środowiska pracy

W celu maksymalnego uproszczenia i przyspieszenia rozwoju robota, tworzenie oprogramowania odbywało się bezpośrednio na komputerze pokładowym. Spowalnia to kompilację programów ze względu na jego mniejszą moc obliczeniową. W przypadku systemu klasy PandaBoard, wyposażonego w dwurdzeniowy procesor ARM Cortex-A9 oraz 1 GB pamięci DDR2 RAM, spowolnienie to jest na akceptowalnym poziomie.

Aby umożliwić komfortową i sprawną pracę zainstalowane zostało środowisko graficzne FluxBox. Nie obciąża ono znacznie procesora, umożliwiając jednocześnie korzystanie z przeglądarki internetowej oraz innych programów graficznych. Do pisania kodu źródłowego użyto bardzo powszechnego wśród systemów Unix edytora tekstu ViM, a do kompilacji - zestawu kompilatorów GNU – GCC.



Rys. 52.Widok środowiska graficznego użytego do rozwoju oprogramowania robota

7.5.2. Sterowniki do układu OMAP oraz procesora graficznego

Aby uzyskać pełny dostęp do funkcji układu OMAP, takich jak sprzętowe kodowanie/dekodowanie wideo należy zainstalować odpowiednie sterowniki. Są one dostępne w repozytoriach Ubuntu w paczce o nazwie “ubuntu-omap4-extras”.

Sprzętową akceleracje grafiki uzyskać można poprzez instalację sterowników graficznych do karty SGX. W nowszych wersjach systemu Ubuntu (>12.04) instalacja odbywa się automatycznie i jest uruchamiana poprzez potwierdzenie monitu menedżera sterowników sprzętowych.

7.5.4. Przesyłanie obrazu z kamery

Do komputera pokładowego robota podłączona została kamera internetowa USB QuickCam® Pro for Notebooks firmy Logitech. Posiada ona wbudowany system autofocus i umożliwia odbiór obrazu w maksymalnej rozdzielczości 1024x768 przy 15 klatkach na sekundę. Dzięki zastosowaniu kamery obraz z robota może być transmitowany na żywo do komputera operatora. Ponadto możliwe jest przetwarzanie klatek uzyskanych z kamery i na podstawie wyników samodzielne poruszanie się robota.

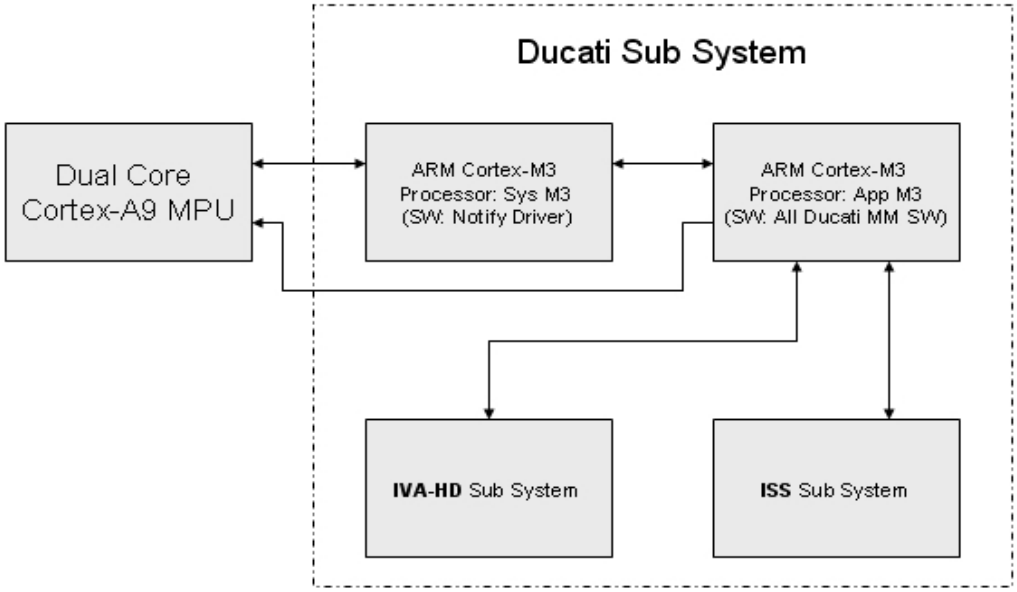
Rysunek 53.Zastosowana kamera

Większość współczesnych kamer internetowych jest urządzeniami spełniającymi standard UVC (Universal Video Class). Wszystkie urządzenia klasy UVC są obsługiwane przez systemy GNU/Linux za pomocą sterownika Linux UVC driver. Sterownik ten jest zawarty w jądrze systemu Linux od wersji 2.6.26.

Zastosowana w robocie kamera internetowa przekazuje do komputera niezakodowany strumień obrazu o formacie pikseli YUV. Przed wysłaniem obrazu do komputera operatora strumien wideo musi zostac zakodowany. Domyślnie, w programach umożliwiających strumieniowanie wideo, tego typu operacje są wykonywane przez procesor komputera. Programowe kodowanie obrazu wideo (software encoding) jest procesem bardzo obciążającym procesor. Z tego powodu, w przypadku płytki PandaBoard, zastosowanej w omawianym robocie jest to sytuacja niepożądana. Podczas testów tego rozwiązania, obciążenie procesora utrzymywało się na stałym poziomie 100%.

Optymalnym rozwiązaniem jest wykorzystanie specjalnego układu elektronicznego zawartego na płytce PandaBoard, służącego do sprzętowego kodowania/dekodowania obrazu wideo (hardware encoding/decoding). Umożliwia to opisana w poprzednich punktach instalacja sterowników do układu OMAP, którego częścią jest sprzętowy koder/dekoder o nazwie Ducati[7].

7 http://omappedia.org/wiki/Ducati_For_Dummies



Rys. 54. System Ducati

Na system Ducati składają się dwa mikroprocesory ARM Cortex-M3. Jeden z nich komunikuje się z głównym procesorem płytki PandaBoard, na której działa system operacyjny wyższego poziomu (HLOS – High Level Operating System). Pierwszy mikroprocesor otrzymuje polecenia z systemu operacyjnego i przekazuje je do drugiego. Drugi mikroprocesor wykorzystuje dwa podsystemy – IVA-HD (Image Video Accelerator – High Definition) oraz ISS (Imaging Sub System) do wykonania tych poleceń (na przykład kodowania strumienia wideo).

Obsługa sprzętowego kodowania Ducati jest możliwa między innymi przez wykorzystanie odpowiedniej wtyczki do popularnego framework'u multimedialnego GStreamer. Jest on używany do tworzenia aplikacji, które umożliwiają edycję/odtwarzanie wideo[8]. Framework GStreamer składa się z wtyczek, które reprezentują różne kodery/dekodery wideo lub funkcjonalności. Wtyczki te mogą być połączone w łącza (pipelines), które definiują przepływ danych pomiędzy nimi. Do uruchamiania tak przygotowanych łącz służy program *gst-launch*. Poniżej umieszczono przykład gotowej komendy służącej do kodowania sprzętowego obrazu z kamery za pomocą Ducati oraz wysyłania go do komputera operatora przez protokół RTP.

```
gst-launch v4l2src device=/dev/video0 ! 'video/x-raw-yuv,width=160,height=120' ! ffmpegcolorspace !
ducatih264enc ! rtph264pay ! udpsink host=jan-nb port=1234
```

8 http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/chapter-gstreamer.html

Poszczególne wtyczki w łączu oddzielone są znakami wykrzyknika. Wtyczki przyjmują również określone argumenty (przykładem jest wtyczka udpsink, która przyjmuje za parametr nazwę lub adres ip hosta docelowego oraz port docelowy).

Po uruchomieniu na systemie PandaBoard powyższego programu można zauważyć znaczny spadek użycia procesora. Jednakże wciąż pozostaje on na nieprawidłowo wysokim poziomie (około 80% przy rozdzielczości obrazu 800x600). Powodem tego jest potrzeba konwersji formatu kolorów obrazu ze źródłowego (YUV) do formatu, który przyjmuje enkoder Ducati (NV12). Konwersja przestrzeni kolorów dokonywana jest za pomocą wtyczki "ffmpegcolorspace". Operacja ta wykonywana jest na głównym procesorze PandaBoard. Koszt przetwarzania rośnie wraz z wielkością obrazu. Aby zmniejszyć więc obciążenie procesora istnieje możliwość znalezienia kompromisu pomiędzy jakością obrazu (przez jego rozmiar) a kosztem konwersji.

Rozdział 8

Platforma ROS

8.1. Platformy programistyczne

Podczas rozwoju projektów z dziedziny robotyki mobilnej zachodzi potrzeba rozwiązania pojawiających się problemów. Chcąc wyposażyć robota w skaner mierzący odległość należy napisać program, który komunikując się z sensorem pobierze od niego dane. Ze względu na ich powszechność powstały już opracowania tego typu zagadnień. Aby uniknąć zbytecznej pracy, należy maksymalnie wykorzystać gotowe rozwiązania. Umożliwiają to platformy programistyczne, służące do tworzenia oprogramowania robotów.

Istnieje kilka różnych projektów tego typu. Jednym z nich jest Microsoft Robotic Developer Studio (MRDS). Jest to komercyjny produkt firmy Microsoft, działający jedynie na systemie operacyjnym Windows. Fakt ten wyklucza użycie go w robocie budowanym w ramach tej pracy. Platforma ta wspiera platformę .NET (języki programowania takie jak C#, C++, VisualBasic). Jej dużą zaletą jest rozbudowane środowisko symulacji.

Kolejne istotne rozwiązanie to Mobile Robot Programming Toolkit (MRPT). Jest on kolekcją bibliotek C++ oraz zbiorem narzędzi, który wspiera wiele systemów operacyjnych (GNU/Linux, MacOS, Windows) oraz architektur (amd64, x86, mips). Zawiera ona części kodu opublikowane na licencjach, które mogą spowodować kolizje z licencjonowaniem projektu wykorzystującego tę platformę[9]. Istnieje jednak możliwość przygotowania wersji MRPT pozbawionej problematycznych części. Dodatkowo na stronie projektu udostępnione jest repozytorium zbiorów danych testowych związanych z robotyką.

Ostatnią opisywaną platformą programistyczną jest system ROS (Robot Operating System). Został on wykorzystany do stworzenia oprogramowania dla robota opracowywanego w ramach niniejszej pracy. System ten został stworzony przez Laboratorium Sztucznej Inteligencji Uniwersytetu Stanford jako część budowanego przez nie robota. Obecnie jest on rozwijany przez firmę Willow Garage, znaną przez bibliotekę

9 http://www.mrpt.org/License

do przetwarzania obrazów OpenCV. Na stronie projektu dystrybucja Ubuntu została oznaczona jako oficjalnie przez niego wspierana. System ten jest oprogramowaniem o otwartym kodzie źródłowym, dostępnym do wykorzystania w pracach naukowych oraz projektach komercyjnych. Dużą zaletą systemu jest liczba dostępnych gotowych rozwiązań różnych zagadnień z dziedziny robotyki mobilnej.

8.2. Działanie systemu ROS

Punkt ten został stworzony bazując na informacjach zawartych na stronie projektu ROS (www.ros.org).

Podstawowymi założeniami systemu ROS są:

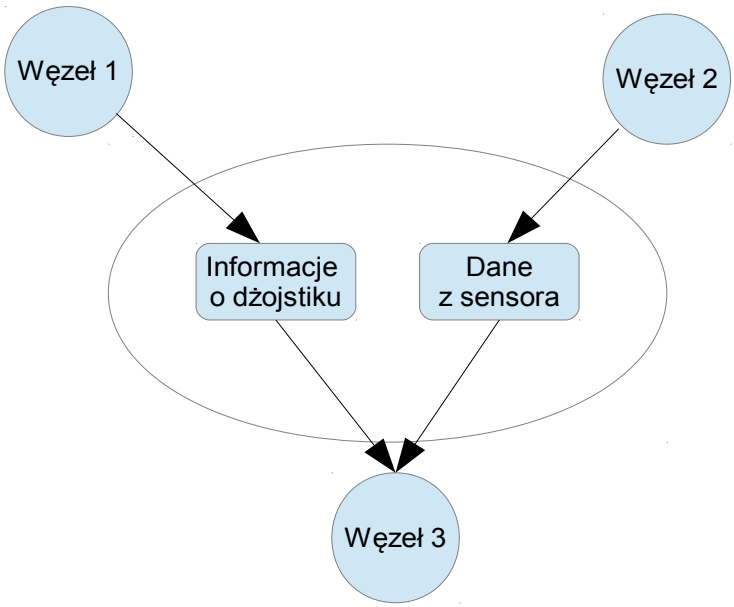
- lekkość – system ma w jak najmniejszym stopniu obciążać sprzęt oraz być łatwym w integracji z innymi framework'ami,
- niezależność od języka programowania – możliwość pisania kodu przy użyciu dowolnego języka programowania (w tej chwili Python, C++ oraz Lisp),
- łatwość testowania – ROS zawiera aplikacje ułatwiające testowanie oraz wizualizację,
- skalowalność – system ma spełniać swoje funkcje niezależnie od poziomu skomplikowania.

Działanie systemu ROS można rozpatrywać na trzech poziomach: poziomie systemu plików, grafu przetwarzania oraz społeczności.

System plików ROS składa się z **Paczek (Packages)** – najmniejszych części reprezentujących pewną częściową funkcjonalność. Paczki opisywane są za pomocą **Manifestów (Manifests).** Są to pliki XML zawierające między innymi informacje o zależnościach od innych paczek, nazwie, licencji oraz wersji paczki. Paczki zorganizowane są w **Stosy**, które reprezentują kompletny zbiór funkcjonalności – na przykład stos służący do nawigacji robota. Stosy również są opisane przez ich własne manifesty. Istnieją również dwa specjalne typy plików: definicji typów **Wiadomości (Messages)** oraz **Usług (Services)**. Pozwalają one na wymianę pomiędzy urządzeniami w sieci ROS samodzielnie zdefiniowanych typów wiadomości oraz usług.

Komunikacja w sieci ROS opiera się na modelu *Publisher-Subscriber.* W modelu tym jedna strona publikuje pewien zestaw tematów, które może subskrybować dowolna ilość odbiorców. Podczas procesu subskrypcji pomiędzy odbiorcą a stroną publikującą zestawiane jest połączenie, któym wysyłane są dane określone przez subskrybowany temat.

Graf przetwarzania systemu ROS reprezentuje przepływ danych ze źródeł do ich odbiorców. Sieć ROS może być przedstawiona w postaci grafu. Węzłami (Nodes) tego grafu są procesy, które przetwarzają dane. Węzły komunikują się ze sobą za pomocą opisanego wcześniej systemu publikacji tematów, usług które udostępniają oraz parametrów obsługiwanych przez serwer parametrów. Przykładem takiej sieci jest sieć złożona z trzech węzłów: pierwszego, który zajmuje się publikowaniem informacji o podłączonym do komputera operatora dżojstiku, drugiego, zajmującego się odczytywaniem i publikacją danych z sensora oraz trzeciego – subskrybującego publikowanych przez poprzednie węzły danych, sterującego na ich podstawie robotem.

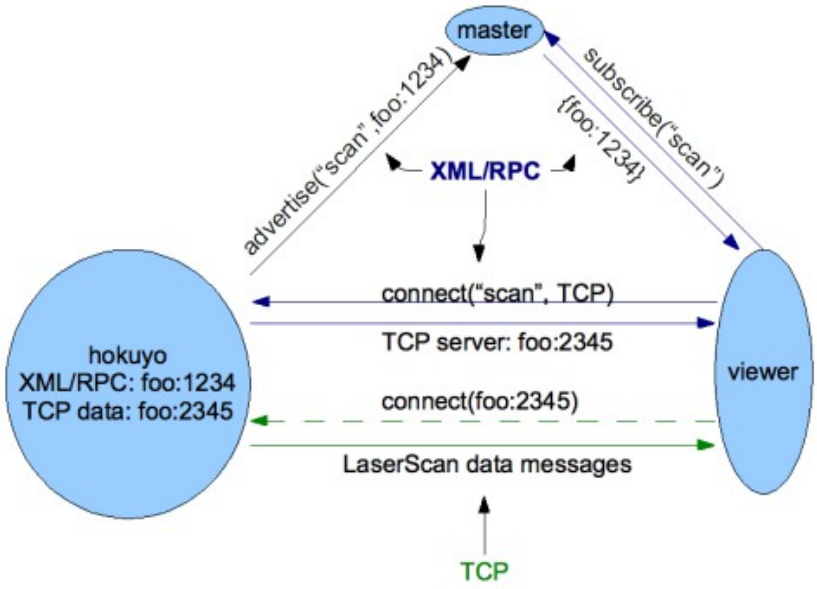


Rys. 55. Przykładowa wymiana informacji w sieci ROS

Kolejnym przykładem, który przedstawia dokładnie proces subskrypcji oraz publikacji jest sytuacja w której dostępny jest skaner firmy Hokuyo, z którego należy pobrać i wyświetlić dane. Aby to uczynić należy uruchomić odpowiedni węzeł, który połączy się ze skanerem i opublikuje wiadomości typu “LaserScan” (typ ten jest wbudowany w instalację ROS) pod tematem “scan”.

Do wyświetlania danych ze skanera musi zostać uruchomiony węzeł, który zasubskrybuje temat “scan”. Węzeł nadawczy i odbiorczy znajdują się poprzez wykorzystanie węzła głównego (master) o znanej lokalizacji. Przechowuje on listę wszystkich opublikowanych tematów. Każdy węzeł, który chce opublikować temat, musi poinformować o tym węzeł główny. Komunikaty publikacji/subskrypcji przeprowadzane są przy użyciu

protokołu XML/RPC. Subskrybujący węzeł otrzymuje od węzła głównego adres, za pomocą którego łączy się z węzłem publikującym interesujący go temat.



Rys. 56. Przykład przebiegu procesu subskrypcji publikacji tematu.
Źródło: http://www.ros.org/wiki/ROS/Technical%20Overview

8.3. Instalacja platformy ROS na komputerze pokładowym

System ROS dostępny jest w repozytoriach dystrybucji Ubuntu w postaci gotowej paczki, co znacznie ułatwia instalację systemu na komputerze operatora (laptopie). Jednak w przypadku architektury komputera pokładowego użytego w robocie będącym przedmiotem pracy (ARM) konieczna jest kompilacja systemu z kodu źródłowego. W tym celu należy zainstalować lub skompilować wszystkie biblioteki konieczne do przeprowadzenia tego procesu, które również mogą nie być dostępne w repozytoriach. Po wykonaniu instalacji należy pobrać oraz skompilować potrzebne dodatkowe paczki oraz stosy ROS.

8.4. Podsumowanie

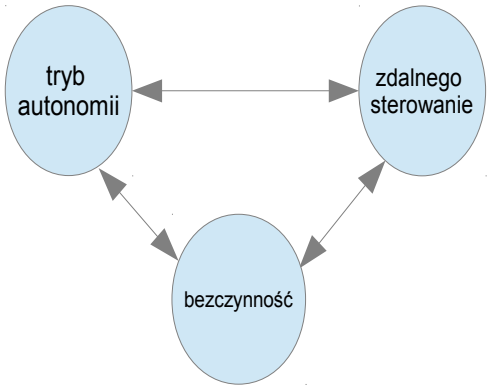
Mechanizmy zastosowane w systemie ROS w znacznym stopniu przyspieszają proces tworzenia oprogramowania przeznaczonego do obsługi robotów mobilnych. System ten zapewnia łatwą w obsłudze i pewną komunikację pomiędzy urządzeniami połączonymi siecią różnego rodzaju. Pozwala to na skupieniu uwagi użytkownika na rozwiązywaniu problemów z dziedziny robotyki mobilnej oraz zminimalizowaniu czasu na opracowywanie znanych i gotowych już rozwiązań.

Rozdział 9

Programy sterujące

9.1. Wprowadzenie

Projektując algorytmy sterowania, z ogólnej koncepcji zachowania robota wydzielić można stany, w których on przebywa. W pracy założono, że robot ma posiadać dwa tryby. Pierwszym trybem pracy ma być sterowanie zdalne z komputera operatora za pomocą kontrolera gier. Robot ma też mieć możliwość autonomicznego poruszania się na podstawie przetwarzania na jego komputerze pokładowym obrazu z kamery. Założenia te implikują istnienie co najmniej dwóch stanów. Dodatkowo potrzebny jest jeszcze trzeci, pośredni, stan reprezentujący bezczynność maszyny. Przejścia pomiędzy stanami wymuszane są przez użytkownika, z poziomu komputera operatorskiego. Z każdego stanu istnieje możliwość przejścia do pozostałych. Z tego powodu graf maszyny stanów jest grafem pełnym o trzech wierzchołkach. Przedstawiony jest on na rysunku 57.



Rys. 57. Graf maszyny stanów robota

9.2. Koncepcja programu

Do zarządzania zachowaniem robota napisany został program implementujący opisaną wcześniej maszynę stanów. Program ten stworzony jest jako węzeł systemu ROS, co umożliwia mu komunikację z innymi węzłami w sieci. Napisany jest w języku C++, przy wykorzystaniu wersji biblioteki ROS do tego języka.

Przełączanie pomiędzy stanami odbywa się przy pomocy serwera parametrów systemu ROS. Jest on współdzielonym słownikiem, który jest dostępny w sieci ROS[10]. Węzły ROS mogą przechowywać oraz odczytywać parametry w trakcie wykonywania. Serwer parametrów jest zaimplementowany przy pomocy XMLRPC i jest częścią głównego węzła systemu ROS.

Po uruchomieniu program robota rejestruje w serwerze parametrów parametr o nazwie robot_state. Użytkownik może ustawić go na dowolny ciąg znaków za pomocą komendy *rosparam* z opcją *set*. Program publikuje również temat o nazwie *error,* który jest zmienną typu liczbowego reprezentującą kod błędu robota (kod 0 oznacza poprawne działanie). Przy ustawieniu niepoprawnej wartości parametru robot_state publikowany jest odpowiedni kod błędu i robot przechodzi w stan bezczynności. Na chwilę obecną poprawne stany to *idle, auto* oraz *joystick*.

Program składa się z pętli głównej, wykonującej się co co najmniej 20 milisekund. Spowodowane jest to zamieszczonym w niej poleceniem czekania, które zapobiega nadmiernemu wykorzystywaniu procesora. Przy każdej iteracji pętli sprawdzana jest wartość parametru robot_state. Pobrana wartość porównywana jest z dowolnymi stanami. Fragment kodu odpowiedzialny za opisaną wyżej funkcjonalność zamieszczony został poniżej.

```
rosnh.getParam("state", s);
if(s == "autonomous") {
       cout << "entering autonomous state" <<
endl;
       autonomous_ctrl(fd, rosnh);
}
else if(s == "joystick") {
       cout << "entering joystick state" <<
endl;
     joystick_ctrl(fd, rosnh);
}
else if(s == "idle") {
       cout << "entering idle state" << endl;
       idle_ctrl(fd, rosnh);
}
else {
       cout << "wrong state parameter! going
back to idle state" << endl;
       rosnh.setParam("state", "idle");
}
```

Listing 2: Kod odpowiedzialny za obsługę stanów robota.

10 http://www.ros.org/wiki/Parameter%20Server

W poszczególnych przypadkach wartości *robot_state* wywoływana jest odpowiednia funkcja sterująca robotem. Funkcje te umieszczone zostały w oddzielnych plikach z kodem źródłowym i zdefiniowane w plikach nagłówkowych *autonomous_ctrl.h* i *joystick_ctrl.h.* Z powodu prostoty funkcji bezczynności została ona umieszczona w tym samym pliku co instrukcja *switch.* Każda z tych funkcji sprawdza co pewien czas wartość parametru *robot_state* i jeżeli jest on różny od stanu przez nią reprezentowanego, zakańcza ona wykonywanie swojego kodu.

9.3. Problem współdzielenia kamery

W systemie Linuks urządzenia wejścia/wyjścia reprezentowane są przez pliki w katalogu /dev/. Procesy mogą czytać lub zapisywać do tych urządzeń poprzez otwieranie tych plików. Jednakże jednoczesny dostęp przez kilka procesów do jednego pliku urządzenia jest niemożliwy. Z tego powodu jednoczesne przesyłanie obrazu z kamery do komputera operatora za pomocą jednego procesu oraz jego przetwarzanie za pomocą drugiego nie jest możliwe. Naturalnym rozwiązaniem jest napisanie jednego programu korzystającego z kamery. Program ten pobierałby obraz z urządzenia, wysyłał do komputera operatora, a następnie wykorzystywał do sterowania robotem. Rozwiązanie to jest jednak bardzo skomplikowane, ponieważ w tym celu należałoby wykorzystywać framework GStreamer oraz interfejs programistyczny sterownika V4L (Video 4 Linux). Optymalnym pod względem czasu rozwiązaniem jest wykorzystanie gotowego programu oferowanego przez framework GStreamer o nazwie *gst-play.* Pozwala on na przesyłanie obrazu z kamery za pomocą protokołu RTP oraz kodowanie obrazu na poziomie sprzętu.

Aby umożliwić poprawną pracę programu w trybie sterowania zdalnego (w którym obraz z kamery przesyłany jest do operatora) zachodzi potrzeba programowego tworzenia nowego procesu, który zajmuje się przesyłaniem obrazu. Przy przechodzeniu z trybu zdalnego do innego trybu proces potomny musi zostać zakończony w celu zwolnienia pliku urządzenia. Tworzenie procesów potomnych możliwe jest przy użyciu funkcji *fork()* zamieszczonej w systemowym pliku nagłówkowym *unistd.h*[11]. Funkcja ta nie przyjmuje żadnych parametrów i zwraca identyfikator utworzonego procesu potomnego. Proces potomny jest prawie dokładną kopią procesu macierzystego. Zmiany wykonywanego przez proces kodu dokonuje się przy pomocy rodziny funkcji *exec()*. Proces potomny likwidowany jest poprzez wysłanie do niego sygnału SIGTERM powodującego łagodne zakończenie programu.

11 Linux Man Pages – dokumentacja funkcji fork

9.4. Sterowanie zdalne

Tryb sterowania zdalnego umożliwia kontrolę robota przy pomocy podłączonego do komputera operatora kontrolera gier. Wykorzystany w ramach pracy kontroler oraz oznaczenie numerów osi i przycisków przedstawiają rysunki 58 i 59.

Rys. 58. Opis przycisków użytego kontrolera gier – widok z boku

Rys. 59. Opis przycisków użytego kontrolera gier – widok z przodu

Stany osi analogowych reprezentowane są przez liczby zmiennoprzecinkowe z zakresu (-1,1). Stan przycisków oznaczany jest jako 0 przy braku wciśnięcia oraz 1 przy wciśnięciu. Informacje o stanach osi i przycisków muszą być odczytywane oraz wysyłane przez komputer operatora do komputera pokładowego. Jest to realizowane przez wykorzystanie gotowego programu – udostępnionego przez społeczność węzła systemu ROS o nazwie *joy*. Odczytuje on stan kontrolera i publikuje go w systemie ROS w postaci wiadomości typu *Joy.* Kod realizujący zdalną kontrolę subskrybuje te wiadomości, odbierając je w specjalnie przygotowanej funkcji typu *Callback,* wywoływaną przez system ROS w momencie nadejścia nowej wiadomości. Funkcja te zapisuje informacje o stanie osi i przycisków w globalnych tablicach o nazwach *axes* oraz *buttons*. Jej kod zamieszczony został w listingu 3.

```
void joystickCallback(const sensor_msgs::Joy::ConstPtr&
msg)
{
    for(int i = 0; i < NUM_OF_AXES; ++i)
        axes[i] = msg->axes[i];

    for(int i = 0; i < NUM_OF_BUTTONS; ++i)
        buttons[i] = msg->buttons[i];
}
```

Listing 3: Kod odbierający stan kontrolera gier.

Główna pętla, wykonująca się co co najmniej 20 milisekund sprawdza tablice stanów kontrolera i na ich podstawie generuje aktualne wartości parametrów robota. Wysyłaniem tych wiadomości do mikrokontrolera, zgodnie z opisanym w dziale „System sterowania" protokołem, zajmuje się funkcja serial_send. Za argumenty przyjmuje ona wartości parametrów robota do wysłania oraz otwarty deskryptor pliku odpowiadający portowi szeregowemu podłączonego z mikrokontrolerem. Zwraca też ilość wysłanych z powodzeniem bajtów. Kod funkcji serial_send przedstawiony został w listingu 4.

```
int serial_send(int fd, int angle, int vel, int
omega, int height, int cameraPos)
{
    buffer[0] = 0xFF;
    buffer[6] = 0xFE;
    buffer[1] = (unsigned char) angle;
    buffer[2] = (unsigned char) vel;
    buffer[3] = (unsigned char) omega;
    buffer[4] = (unsigned char) height;
    buffer[5] = (unsigned char) cameraPos;

    return write(fd, buffer, 7);
}
```

Listing 4: Kod funkcji służącej do wysyłania parametrów do robota.

Wartość „alpha" jest kątem wektora przemieszczenia robota, przyjmującym wartości od 0 do 200 z punktem neutralnym w 100. Obliczana jest na podstawie kąta wychylenia lewego drążka względem maksymalnego jego wychylenia do przodu. Kąt wychylenia drążka otrzymywany jest za pomocą funkcji *atan2,* której wynikiem jest wartość z zakresu $[-\pi, \pi]$. Zakres tego kąta jest przekształcany do [0, 200]. Wysokość platformy robota nad ziemią, czyli wartość „height", jest na początku programu ustawiana na 100. Następnie, przy wciśnięciu przycisku 5, jest ona zwiększana, a przy wciśnięciu przycisku 4 – zmniejszana. Długość wektora przemieszczenia robota, będąca jego prędkością liniową, wyznaczana jest za pomocą twierdzenia Pitagorasa na podstawie wychylenia osi 0 i 1

oraz skracana do wartości 200 przy jej przekroczeniu. Pozycja kamery zmieniana jest poprzez wychylenie osi 2 lub 5. Stopień wychylenia tych osi mnożony jest poprzez stałą wartość maksymalnej zmiany. Wynik tej operacji jest dodawany lub odejmowany od aktualnego wychylenia kamery. Prędkość obrotowa robota *omega* wyliczana jest przy użyciu stopnia wychylenia osi numer 3. Kod wyliczający wartości *alpha, omega, vel, height, cameraPos* do wysłania zamieszczony został w listingu 5.

```
if(buttons[5] && height < 199)
      height += 2;

 if(buttons[4] && height > 1)
      height -= 2;

// obliczanie wartosci
alpha = (!axes[0] && !axes[1] ? 100 : (int) (100 + atan2(axes[0], axes[1])/M_PI * 100));
omega = 100 + 100 * axes[3];

vel = sqrt(pow(axes[0],2) + pow(axes[1],2)) * 200;
if(vel > 200)
      vel = 200;

cameraPos += axes[5] * CAMERA_INCREMENT;
cameraPos -= axes[2] * CAMERA_INCREMENT;

if(cameraPos < 0)
      cameraPos = 0;
if(cameraPos > 90)
     cameraPos = 90;
```

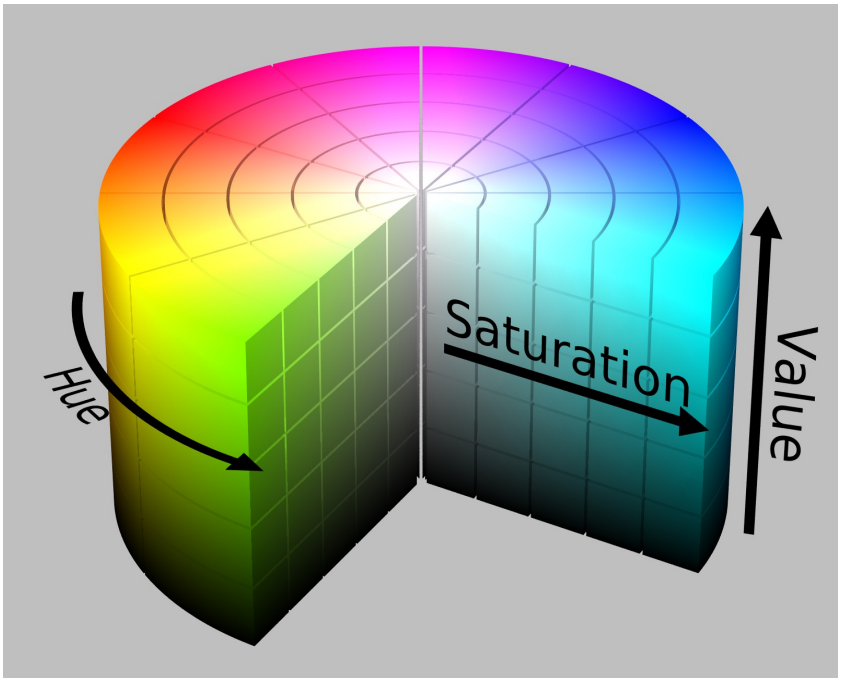
Listing 5: Kod odpowiedzialny za obliczanie parametrów robota.

9.5. Sterowanie autonomiczne

Tryb sterowania autonomicznego pozwala na samodzielne poruszanie się robota na podstawie obrazu z noszonej przez niego kamery. Napisany w ramach pracy program powoduje śledzenie przez kamerę robota markera o ustalonym kolorze oraz podążanie za nim. W tym celu obraz z kamery poddawany jest serii przekształceń, mających na celu rozpoznanie obiektu oraz wyznaczenie jego pozycji. Program napisany został przy wykorzystaniu biblioteki OpenCV.

Pierwszym etapem zastosowanego algorytmu przetwarzania obrazu jest jego progowanie na podstawie koloru. Dla każdego piksela sprawdzane są współrzędne jego koloru. Jeśli mieszczą się one w zadanych granicach, w wynikowym obrazie binarnym wstawiany jest w tym miejscu piksel o kolorze białym. Wykrywanie koloru markera przy różnych warunkach oświetlenia jest trudnym zadaniem. W przestrzeni barw RGB, zmiana oświetlenia powoduje zmianę współrzędnych koloru. Aby maksymalnie uniezależnić wykrywanie danego koloru od warunków oświetlenia zastosowana została przestrzeń barw HSV. Dana barwa jest w niej reprezentowana przez trzy parametry – odcień (Hue), nasycenie

(Saturation) oraz jasność (Value). Ilustrację współrzędnych przestrzeni HSV prezentuje rysunek 60.



Rys. 60. Cylinder przestrzeni barw HSV
Źródło: http://en.wikipedia.org/wiki/HSL_and_HSV

Zastosowanie tej przestrzeni barw pozwala na ustalenie wąskiego zakresu parametru odcienia, który opisuje kolor markera. Pozostałe dwa parametry mogą przyjmować dowolne wartości. Są to jedyne parametry ulegające zmianie wraz ze zmianą oświetlenia.

W celu kalibracji operacji progowania napisany został program umożliwiający dobór zakresów parametrów H, S oraz V. Dobór właściwych dla nich wartości jest kluczowym czynnikiem decydującym o jakości wykrywania markera. Kod z listingu 6 pobiera obraz z kamery, konwertuje jego przestrzeń barw do HSV oraz dokonuje progowania.

```
cap >> image;
cvtColor(image, hsv, CV_BGR2HSV);
inRange(hsv, Scalar(hmin, smin, vmin),
Scalar(hmax, smax, vmax), bw);
```

Listing 6: Kod odpowiedzialny za pobieranie obrazu z kamery, konwersję kolorów oraz progowanie.

Kolejnym etapem jest kilkukrotne wykonanie operacji erozji oraz dylatacji obrazu binarnego. Pozwala to na likwidację szumów w postaci małych grup pikseli, które silnie wpływają na zastosowaną metodę wykrywania pozycji markera. Efekty zastosowanych przekształceń zaprezentowane zostały na rysunkach 61 oraz 62.

Rys. 61. Obraz przed operacjami morfologicznymi

Rys. 62. Obraz po operacjach morfologicznych

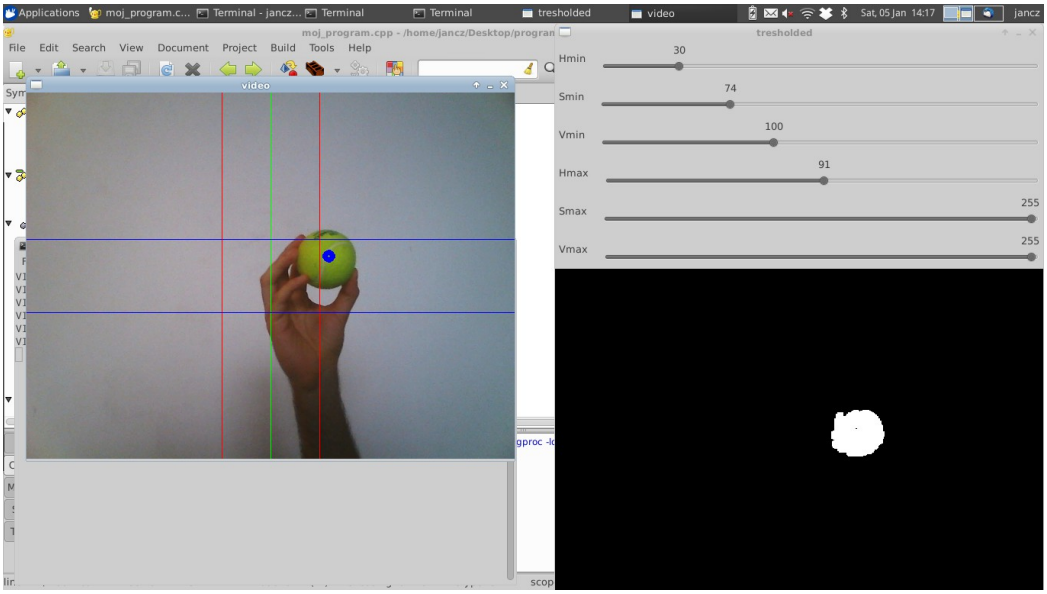
W celu wyznaczenia pozycji markera na ekranie liczone są momenty obrazu przy wykorzystaniu funkcji *moments* z biblioteki OpenCV[1]. Zwraca ona strukturę zawierającą momenty przestrzenne, centralne oraz centralne znormalizowane [2]. Współrzędne środka masy obrazu wyliczane są za pomocą zależności:

$$\bar{x}=\frac{m_{10}}{m_{00}} \qquad \bar{y}=\frac{m_{01}}{m_{00}}$$

gdzie: m_{00} - moment przestrzenny zerowego rzędu, reprezentujący pole,

$m_{10,}m_{01}$ - momenty przestrzenne pierwszego rzędu.

Wyniki detekcji pozycji markera prezentuje rysunek 63.



Rys. 63. Rozpoznawanie markera

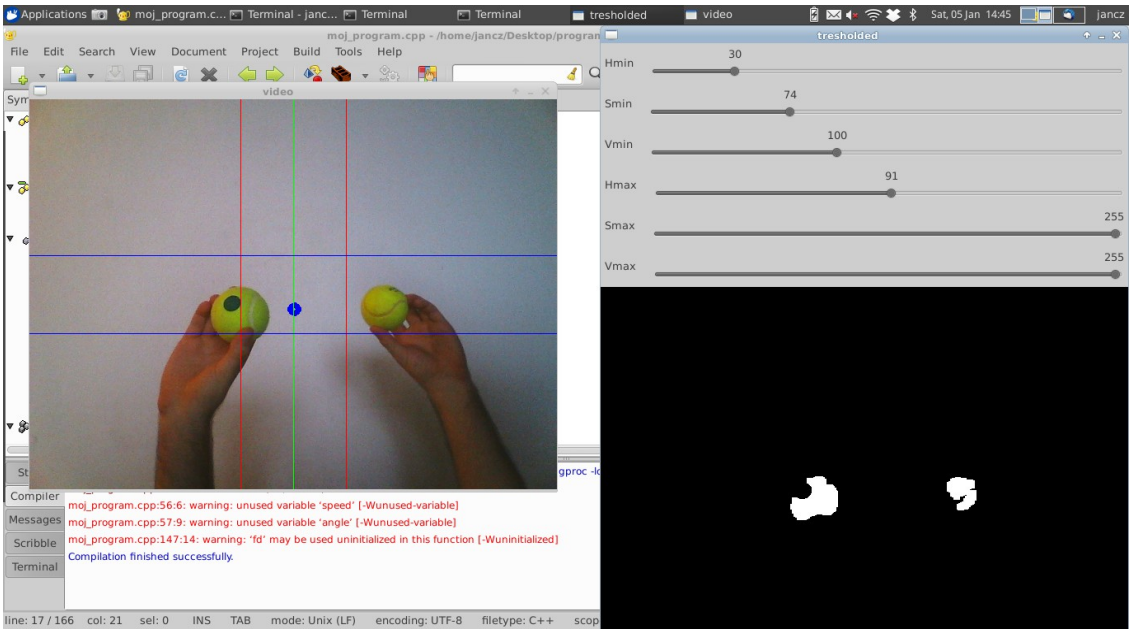
Poziome i pionowe linie na rysunku oznaczają granice, po przekroczeniu których dokonywana jest korekcja ustawienia kamery. Algorytm sterowania pozycją kamery jest bardzo prosty – w przypadku przekroczenia granicy górnej lub dolnej jej kąt pochylenia zwiększa się/zmniejsza o stałą wartość. Centrowanie w poziomie otrzymuje się przez skręcanie całego robota. Algorytm sterowania jest w tym przypadku taki sam jak przy centrowaniu w pionie. Dodane zostały również zabezpieczenia zapobiegające wykrywania pozycji szumów w postaci małych grup pikseli – badany w tym celu jest moment

1 http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=moments#moments

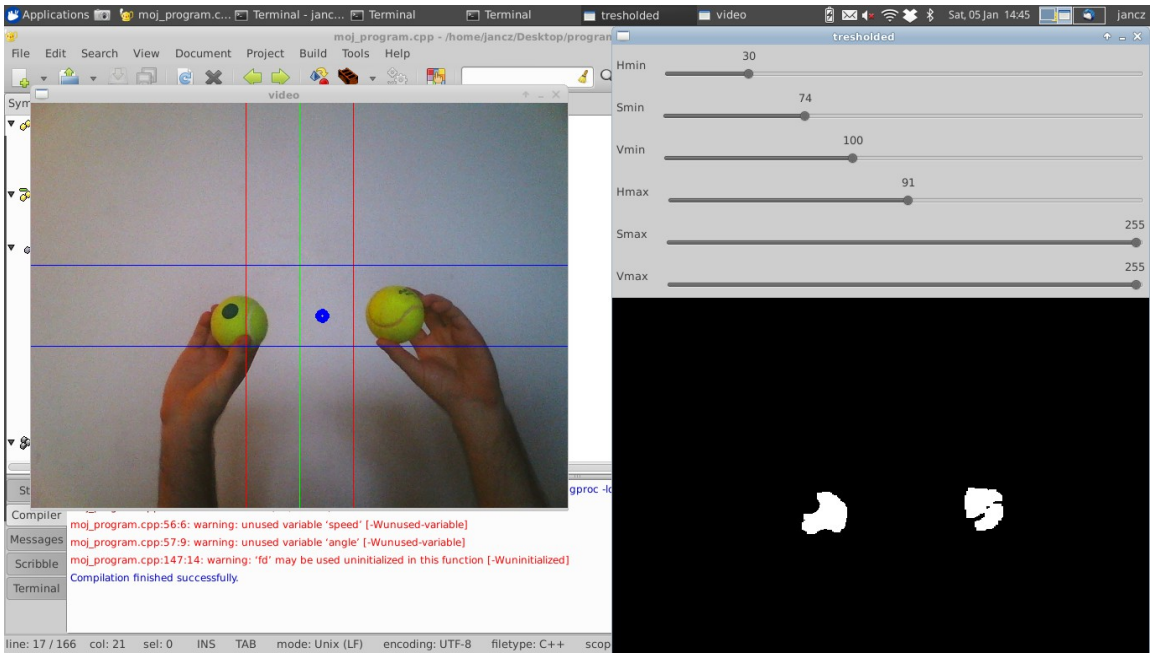
zerowego rzędu obrazu. Jeśli jest on mniejszy od pewnej granicznej wartości, robot nie śledzi celu i pozostaje w spoczynku.

Zbliżanie się oraz oddalanie robota inicjowane jest na podstawie badania kąta wychylenia kamery. Jeżeli kąt ten jest większy od 60° lub mniejszy od 40° do mikrokontrolera wysyłane są parametry powodujące poruszanie się robota odpowiednio w tył i w przód. Jednocześnie, podczas ruchu robota kamera jest centrowana na obiekcie. Taki algorytm powoduje ustawianie się robota w pewnej odległości od obiektu. Jeśli jednak obiekt znajduje się na ziemi, robot podejdzie do niego.

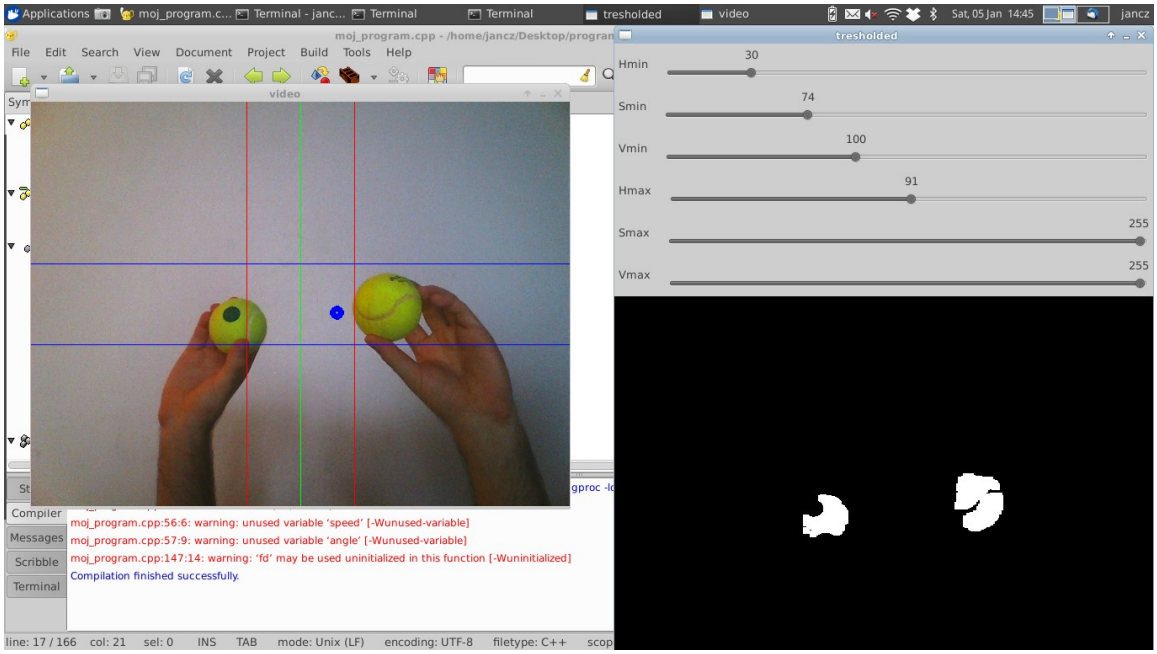
Wykryte podczas progowania elementy niebędące markerem mają silny wpływ na jakość wykrywania jego pozycji. Momenty są liczone dla całego obrazu. Z tego powodu jego środek ciężkości może wypadać poza markerem. Pojawianie się obiektów innych niż marker przesuwa środek ciężkości proporcjonalnie do pola tych obiektów. Wpływ błędów progowania przy pomocy koloru na jakość wykrywania pozycji markera przedstawiają rysunki 64, 65 oraz 66.



Rys. 64. Wpływ zakłóceń na wykrywanie pozycji markera 1



Rys. 65. Wpływ zakłóceń na wykrywanie pozycji markera 2



Rys. 66. Wpływ zakłóceń na wykrywanie pozycji markera 3

Markerem w powyższych rysunkach jest piłka z zieloną kropką. Druga piłka symbolizuje zakłócenie. Wraz ze wzrastającym polem zakłócenia, środek ciężkości obrazu (czyli wykryta pozycja markera) przesuwa się w jego stronę.

9.6. Podsumowanie

Przedstawiony w tym rozdziale program pozwala na podążanie robota za wybranym markerem. Po dokonaniu kalibracji parametrów progowania robot centruje swoją kamerę na markerze oraz podąża za nim.

Aby poprawić otrzymywane wyniki, należy zastosować marker o jak najbardziej wyraźnym kolorze. Zastosowanie w środowisku laboratoryjnym równomiernego oświetlenia również ma pozytywny wpływ na wykrywanie markera.

Algorytm sterowania jest bardzo prosty. Dużą poprawę spowodowałoby zastosowanie bardziej skomplikowanego algorytmu, uwzględniającego wstrząsy podczas przemieszczania się robota oraz wartość odchylenia kamery od pozycji zadanej. Istnieje również możliwość zastosowania sterowania rozmytego do kontrolowania ruchu robota w przód oraz w tył.

Zastosowany program do rozpoznawania obiektu nie obciąża w znacznym stopniu procesora. Bardziej skomplikowana jego odmiana potrzebowałaby większych zasobów systemowych, których być może zastosowana elektronika nie jest w stanie dostarczyć. Aby maksymalnie odciążyć procesor układu, należałoby wykorzystać wbudowaną kartę graficzną systemu PandaBoard. Biblioteka OpenCV posiada funkcje wykorzystujące framework OpenCL (Open Computing Library) pozwalającą wykorzystać do obliczeń procesor graficzny. Płytka PandaBoard nie posiada jednak sterowników do niego i wykorzystanie na niej OpenCL nie jest możliwe. Inna możliwością jest wykorzystanie biblioteki OpenGL (Open Graphics Library), służącej do generowania grafiki. Istnieją metody wykorzystania języka GLSL do przeprowadzania operacji na macierzach. Operacje takie jak konwersja przestrzeni kolorów, progowanie, dylatacja oraz erozja mogłyby być zaimplementowane przy pomocy tych metod.

Rozdział 10

Podsumowanie

Efektem przedstawionej pracy jest prototypowa konstrukcja robota realizująca stabilny statycznie chód sześcionożny, o napędzie elektrycznym, kontrolowana przy pomocy trzypoziomowego systemu komputerowego, w skład którego wchodzi komputer operatora, komputer pokładowy PandaBoard oraz mikrokontroler. Opracowany robot jest zdolny do prostych autonomicznych zachowań, z możliwością przejęcia kontroli przez operatora w dowolnym momencie. Komunikacja z robotem jest bezprzewodowa i wykorzystuje technologię WiFi, dzięki czemu rolę panelu operatorskiego może pełnić dowolny komputer wyposażony w kartę sieciową oraz odpowiednią aplikację, a obszar pracy robota ogranicza zasięg sieci w której się on znajduje. Zastosowanie takiego rozwiązania pozwala na kontrolę z dowolnego miejsca w zasięgu sieci.

Algorytm ruchu został opracowany w stopniu umożliwiającym efektywne poruszanie się po płaskich powierzchniach, z uwzględnieniem dowolnego kierunku ruchu i jednoczesnego obrotu. Pozwala on na bierne pokonywanie niewielkich przeszkód, jednak poruszanie się w nieregularnym terenie wymaga opracowania bardziej zaawansowanego algorytmu oraz wprowadzenia sprzężenia zwrotnego od kończyn. Robot, zgodnie z przewidywaniami, jest w stanie przenosić niewielkie ładunki, których masa jest ograniczona przez maksymalny moment obciążający serwomechanizmy, jaki są one w stanie pokonać. Jak wynika z przeprowadzonej analizy, nogi nie są obciążone równomiernie. Wymiana dwóch najbardziej obciążonych aktuatorów odpowiadających za napędzanie stawów biodrowych środkowych nóg na wydajniejszy model powinno w znacznym stopniu poprawić możliwości platformy.

Prace związane z oprogramowaniem mikrokontrolera pokazały, że przy potrzebnej do realizacji omawianego zadania ilości obliczeń, efektywność implementacji algorytmu nie pozostaje bez znaczenia. Kod programu został pod tym kątem przeanalizowany, wszystkie powtarzające się obliczenia zostały wyeliminowane, pamięć rezerwowana na odpowiednie

zmienne ograniczona do minimum a niektóre zależności stabelaryzowane. Osiągnięta efektywność jest zadowalająca w chwili obecnej, jednak w przyszłości będzie musiała zostać poprawiona jeżeli konieczne okaże się przetwarzanie danych z sensorów. Ze względu na złożoność obliczeniową, dużą zmianę powinna przynieść konwersja wszystkich zmiennych typu float na zmienne stałoprzecinkowe.

Opracowany system sterowania, pomimo swoich zalet związanych z podziałem zadań pomiędzy autonomiczne warstwy, okazał się podatny na awarię. Wystąpienie błędu w którymkolwiek z elementów powoduje zawieszenie całego systemu. W szczególności jakość sterowania zależy od jakości połączenia, które w przypadku technologi WiFi okazuje się być zawodne. W trakcie testów obserwowaliśmy kilkusekundowe przerwy w połączeniu, które skutkowały całkowitym brakiem kontroli nad robotem oraz przerwaniem transmisji wideo. Aby możliwe było praktyczne zastosowanie opracowanego prototypu, konieczne jest udoskonalenie systemu łączności oraz opracowanie korpusu zdolnego pomieścić całą elektronikę i zasilanie robota.

Bibliografia

[1] Bis M.: *Linux w systemach embedded*. BTC, Legionowo 2011.

[2] Bradski G., Kaehler A.: *Learning OpenCV: Computer Vision in C++ with the OpenCV Library.* O'Reilly Media, Inc., Sebastopol, CA, 2008.

[3] McGhee R. B.: *Adaptive Locomotion of a Multilegged Robot over Rough Terrain*, IEEE Transactions on systems, man, and cybernetics, Vol SMC-9, No 4 (1979), 176-182.

[4] Paprocki K.: *Mikrokontrolery STM32 w praktyce.* Wydawnictwo BTC, Legionowo 2011.

[5] Seljanko F.: *Towards Omnidirectional Locomotion Strategy for Hexapod Walking Robot*. International Symposium on Safety, Security and Rescue Robotics, Kyoto, Japonia 2011.

[6] Siegwart R., Nourbakhsh I., Scaramuzza D.: *Introduction to Autonomous Mobile Robots*. The MIT Press, Londyn 2011.

[7] Tanenbaum A. S.: *Computer Networks*. Prentice Hall PTR, New Jersey 2003.

[8] Tomaszewski K.: *Maszyny przemysłowe. Projektownie układów mechanicznych*. Wydawnictwo Naukowo-Techniczne, Warszawa 1993.

[9] Zielińska T.: *Maszyny kroczące. Podstawy, projektowanie, sterowanie i wzorce biologiczne*. Wydawnictwo Naukowe PWN, Warszawa 2003.