POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Robotyka (ARR)

PRACA DYPLOMOWA MAGISTERSKA

Budowa mapy wysokości przy użyciu sześcionożnego robota kroczącego

Altitude map building with six-legged walking robot

AUTOR:
Dominik Urban

PROWADZĄCY PRACĘ:
dr inż. Janusz Jakubiak PWr, I-6

OCENA PRACY:

WROCŁAW 2010

Pracę tą dedykuję rodzicom.

Spis treści

1 Wstęp **3**

2 Podstawy teoretyczne **5**
2.1 Kinematyka robota 5
2.2 Proste zadanie kinematyki robota 5
2.3 Odwrotne zadanie kinematyki nogi robota. 7
2.4 Generator chodu 8
2.4.1 Trajektoria ruchu końca nogi 9
2.4.2 Trajektoria ruchu końca nogi przy zmianach kierunku ruchu 10
2.4.3 Trajektoria ruchu końca nogi przy zmiennej rotacji i nutacji korpusu 11
2.5 Generator chodu – wersja iteracyjna 12
2.6 Wyznaczanie odchylenia od pionu na podstawie pomiaru przyspieszeń . . . 14
2.7 Wyznaczanie odchylenia od pionu i orientacji 15
2.8 Filtracja Kalmana 16
2.9 Wyznaczanie prędkości oraz przemieszczenia na podstawie przyspieszeń . . 17
2.10 Triangulacja Delone 18

3 Budowa robota Mrówa **20**
3.1 Mechanika 21
3.2 Sterownik główny robota 24
3.2.1 Budowa 24
3.2.2 Komunikacja 27
3.3 Sterownik lokalny nogi robota 27
3.3.1 Budowa 29
3.3.2 Komunikacja 31
3.4 Moduł nawigacyjny INS 33
3.4.1 Akcelerometr MMA1260EG 34
3.4.2 Filtr analogowy MAX7400 35
3.4.3 Żyroskop ADIS16100 37

4 Oprogramowanie sterujące, symulator **40**
4.1 Opis aplikacji 40
4.2 Moduł komunikacji 43

5 Badania **45**
5.1 Badanie modułu INS 45
5.1.1 Wyznaczanie zera i skalowanie akcelerometrów MMA1260EG 46
5.1.2 Kompensacja temperaturowa akcelerometrów MMA1260EG oraz żyroskopów ADIS16100 47

5.1.3 Wyznaczanie odchylenia od pionu na podstawie pomiaru przyspieszeń 48
5.1.4 Wyznaczanie odchylenia od pionu i orientacji na podstawie pomiaru prędkości kątowych . . . 50
5.1.5 Wyznaczanie odchylenia od pionu i orientacji z użyciem filtracji Kalmana . . . 52
5.1.6 Wyznaczanie prędkości oraz przemieszczenia na podstawie przyspieszeń . . . 59
5.1.7 Wnioski . . . 62
5.2 Badanie nogi robota . . . 63
5.2.1 Śledzenie zadanej trajektorii . . . 63
5.2.2 Trzymanie zadanej pozycji . . . 64
5.3 Wyznaczanie pozycji i orientacji robota . . . 65
5.3.1 Wyznaczanie pozycji i orientacji robota kroczącego na podstawie odometrii . . . 65
5.3.2 Wyznaczanie pozycji i orientacji robota na podstawie pomiarów z modułu INS . . . 67
5.3.3 Wnioski . . . 68
5.4 Tworzenie mapy wysokości z użyciem robota kroczącego . . . 69
5.4.1 Wnioski . . . 71

6 Podsumowanie 72

A Szczegóły implementacji 75
A.1 Proste sposoby przyspieszenia obliczeń dla mikroprocesorów . . . 75
A.2 Implementacja filtru Kalmana . . . 76
A.2.1 Przykład implementacji w języku ANSI C . . . 77
A.3 Tablica rejestrów lokalnego sterownika nogi . . . 79
A.4 Tablica rejestrów sterownika głównego robota . . . 80

B Schematy 81

C Rozmieszczenie elementów na płytkach 89

D Opis wyprowadzeń i złączy 96

Rozdział 1

Wstęp

W ostatnich latach coraz większym zainteresowaniem cieszą się roboty kroczące. Ich zasadniczą przewagą nad kołowymi robotami mobilnymi jest zdolność poruszania się po nierównym terenie. Kołowe roboty mobilne potrafią przemieszczać się bardzo szybko i sprawnie, jednak wymagają do tego odpowiednio przygotowanego podłoża. Prowadzone w ostatnich latach badania pokazują na przykładzie robota BigDog [16], że współczesne roboty kroczące coraz lepiej radzą sobie z utrzymaniem równowagi i poruszaniem się w trudnym terenie. Prawdopodobnie w przyszłych latach dalszy rozwój technologii i układów sensorycznych pozwoli na stworzenie coraz lepszych konstrukcji.

Poruszanie się po nierównościach wymaga odpowiedniej konstrukcji mechanicznej, układów sensorycznych oraz algorytmów kroczenia. Wraz z wprowadzeniem autonomiczności pojawiają się dodatkowo problemy związane z planowaniem bezpiecznej ścieżki, wyborem punktów podporowych oraz stabilnością konstrukcji. Aby móc radzić sobie z tymi problemami, potrzebna jest wiedza o terenie, po którym porusza się robot. Najczęściej reprezentowana jest ona w postaci map wysokości. Mapy takie mogą posłużyć do planowania punktów podporowych oraz ścieżki robota.

W ciągu ostatnich lat przeprowadzono wiele badań z tej dziedziny. Robot planetarny Ambler [11] był jednym z pierwszych robotów kroczących zdolnych do rejestracji otoczenia. Mapa otoczenia budowana była na podstawie danych ze sknera laserowego. Informacje o wysokości reprezentowano w postacji dwu wymiarowej siatki, gdzie pojedyncza komórka zawiera informacje o wysokości. Inne podejście przedstawiono w pozycji [8]. Mapa otoczenia budowana była na podstawie techniki probabilistycznej odpartej o model procesu Gausowskiego. Model mapy nie zakładał równej dyskretyzacji przestrzeni w postaci siatki. Dodatkowo poruszono problem planowania punktów podporowych dla robota kroczącego LAURON III. Decyzję podejmowano na podstawie mapy zajętości i mapy wiarygodności. Robot LittleDog [15] również wykorzystuje metody gausowskie do budowania map wysokości. Opracowano metody optymalizacji, obliczeń, oraz rozwinięto proces planowania ruchu robota. Zastosowane techniki budowania map pozwalają na wyciąganie informacji na temat obszarów nie obserwowanych bezpośrednio przez skaner laserowy. Przedstawione badania na rzeczywistym robocie pokazują skuteczność zastosowanych metod. Ciekawą metodę budowania mapy wysokości przedstawiono w pozycji [9]. Estymacja powierzchni tworzona jest przy pomocy algorytmu uczenia opartego na funkcji jądra (ang. *kernel–based learning*). Metoda pozwala na jednoczesne szacowanie błędu pomiaru w postaci górnych i dolnych ograniczeń. Dodatkowa informacja o pewności pomiaru może być wykorzystana podczas planowania trajektorii robota.

Cele Pracy

Głównym celem prezentowanej pracy jest opracowanie rozwiązania zadania budowania mapy wysokości terenu przez robota kroczącego. Przedstawiona w ramach pracy metoda budowania mapy różni się od typowych rozwiązań spotykanych w literaturze, gdyż do zbierania danych pomiarowych wykorzystywane są tylko informacje o kontakcie stopy robota z podłożem. Mapa taka mogła by posłużyć jako uzupełnienie map budowanych z użyciem innych sensorów takich jak np. skanery laserowe.

W ramach pracy przeprowadzono modernizację robota skonstruowanego przez autora obejmującą: modyfikację mechaniki, wymianę elektroniki, wykonanie systemu sensorycznego do poziomowania korpusu robota oraz opracowanie algorytmów kroczenia po nierównościach. Robot wykorzystany został w badaniach nad budową mapy wysokości. Stworzone zostało również oprogramowanie sterujące robotem, które ma możliwość tworzenia trójwymiarowego modelu mapy wysokości. Mapa ta tworzona jest przy użyciu triangulacji Delone.

Struktura pracy

Niniejsza praca składa się z sześciu rozdziałów. Rozdział 2 zawiera wstęp teoretyczny. Przedstawiono w nim kinematykę prostą i odwrotną sześcionożnego robota kroczącego, następnie omówiono algorytm kroczenia robota. W dalszej części rozdziału opisano obliczenia stosowane w module nawigacji inercyjnej. Mianowicie przedstawiono sposób wyznaczania przechyłów bocznych na podstawie pomiaru wektora przyspieszenia ziemskiego, pokazano sposób przeliczenia prędkości kątowych, oraz przyspiespieszeń z lokalnego układu odniesienia robota do układu globalnego. Omówiono także filtr Kalmana. Na końcu rozdziału przedstawiono metodę budowania mapy wysokości na podstawie triangulacji Delone.

W rozdziale 3 przedstawiono opis robota Mrówa. Przedstawiona jest konstrukcja mechaniczna oraz elektronika robota. Omówiono kolejno budowę płyty głównej robota, sterownika lokalnego nogi robota oraz modułu nawigacji inercyjnej. Rozdział zawiera także opis przejętego standardu komunikacji pomiędzy nogą a sterownikiem głównym robota oraz standardu komunikacji pomiędzy robotem a komputerem PC.

Kolejny rozdział 4 zawiera opis oprogramowania napisanego specjalnie na potrzeby pracy. Opisano oprogramowanie sterujące robotem oraz symulator robota. Na końcu rozdziału przedstawiono koncepcję programową modułu komunikacji.

Rozdział 5 zawiera wyniki przeprowadzonych badań. Przedstawiono wyniki badań nad modułem nawigacji: kompensacja temperaturowa czujników, skalowanie czujników, wyznaczanie przechyłów bocznych i orientacji modułu, przemieszczenie. Przedstawiono także wyniki badania nad filtracją Kalmana. Następnie zaprezentowano wyniki badań nad określeniem pozycji robota. Badania przeprowadzono z wykorzystaniem rzeczywistej konstrukcji. Na końcu rozdziału przedstawiono wyniki badań nad tworzeniem mapy z wykorzystaniem robota kroczącego.

Na końcu pracy umieszczono dodatki. Mianowicie szczegóły implementacji A, schematy elektroniczne B, rozmieszczenie elementów na płytkach C, opis wyprowadzeń i złaczy D.

Rozdział 2

Podstawy teoretyczne

2.1 Kinematyka robota

Robotem kroczącym można sterować poprzez zmianę kątów wychylenia segmentów w nogach robota. Łącznie jest osiemnaście segmentów po trzy na każdą nogę. Aby przemieścić korpus, robot kroczący wymaga synchronicznej współpracy swoich kończyn. Pracą nóg podczas kroczenia steruje generator chodu, który wyznacza pozycję nóg w układzie kartezjańskim. Aby przenieść wartości z układu kartezjańskiego na przestrzeń zadaniową danej nogi (wychylenia kątowe), konieczna jest znajomość kinematyki odwrotnej.

W rozdziale tym przedstawiony jest sposób wyznaczenia kinematyki prostej i odwrotnej dla sześcionożnego robota kroczącego. Cały proces wyliczenia kinematyki wzorowany jest na pracy [21].

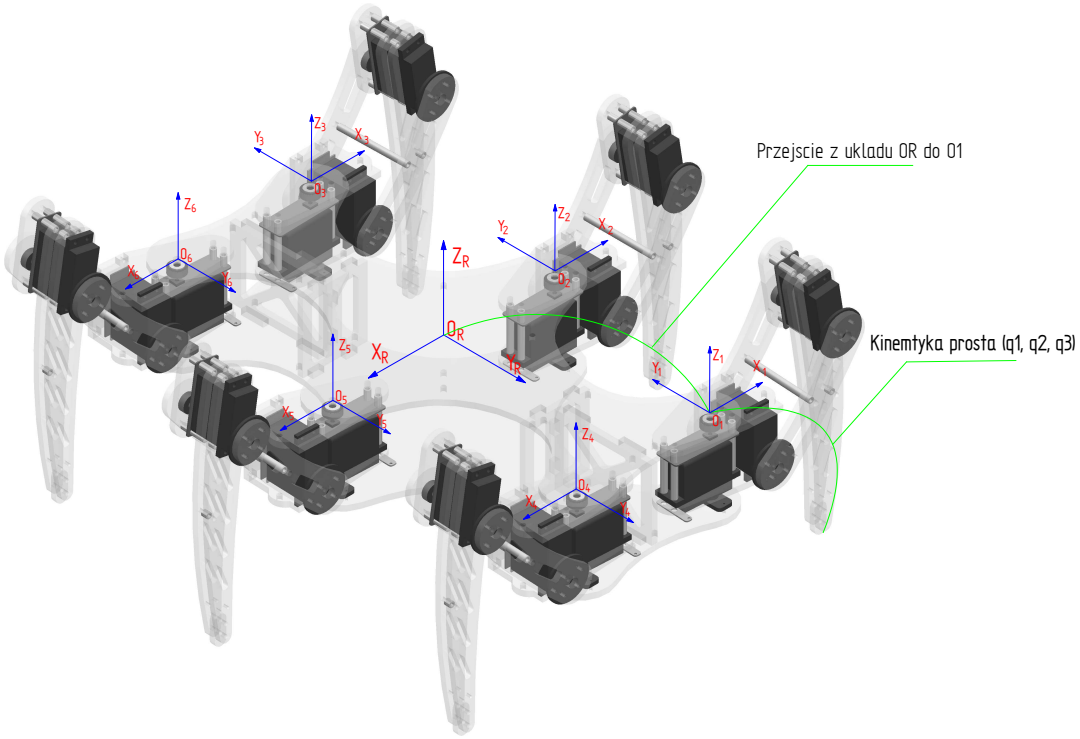
2.2 Proste zadanie kinematyki robota

Robota kroczącego można przedstawić jako połączenie sześciu manipulatorów. Pojedynczą nogę robota można przedstawić jako manipulator o trzech stopniach swobody, połączony z pozostałymi za pomocą korpusu. Podejście takie pozwala wyznaczyć rozwiązanie odwrotnego zadania kinematyki osobno dla każdej z nóg robota.

Rysunek 2.1 przedstawia rozmieszczenie lokalnych układów współrzędnych dla każdej z nóg robota. Oznaczone są one przez O_i, gdzie i – numer nogi. Środek korpusu (układ O_R) oraz układy lokalne nóg robota znajdują się na wsokości osi silnika przegbu drógiego. Na rysunku układy narysowano na wysokości górnej płyty korpusu tylko w celu poprawy czytelności rysunku. Wymiary robota przedstawione są na rysunku 2.2.

Macierz transformacji pomiędzy układem O_R, a układami O_i, ma postać:

$$T_{O_i}^{O_R} = \begin{cases} Trans\,(X, -l_1)\,Trans\,(Y, l_2)\,Rot\,(Z, \pi) & \text{dla } i = 1, 2, 3 \\ \\ Trans\,(X, l_1)\,Trans\,(Y, l_2) & \text{dla } i = 4, 5, 6 \end{cases} \tag{2.1}$$



Rysunek 2.1 Rozmieszenie lokalnych układów współrzędnych, dla każdej z nóg

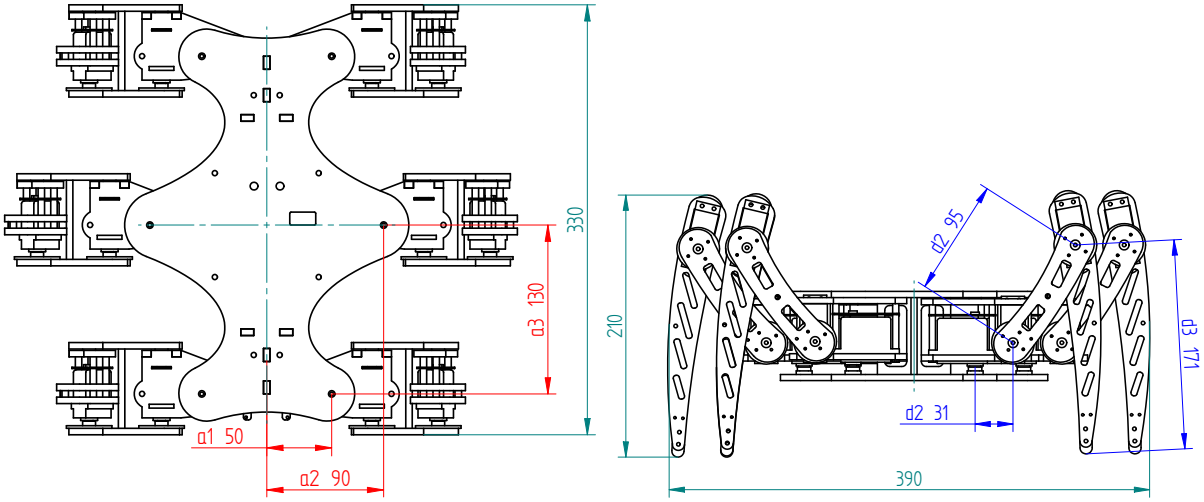
po obliczeniu uzyskuje się następujące macierze transformacji:

$$T_{O_i}^{O_R} = \begin{cases} \begin{bmatrix} -1 & 0 & 0 & -l_1 \\ 0 & -1 & 0 & l_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{dla } i = 1, 2, 3 \\ \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{dla } i = 4, 5, 6, \end{cases}$$

gdzie l_1 i l_2 oznaczają pozycję lokalnego układu współrzędnych nogi względem układu O_R. Wartości te można odczytać z rysunku 2.2. Na przykład dla nogi pierwszej ($i = 1$), $l_1 = a_1$, $l_2 = a_3$, natomiast dla $i = 4$, $l_1 = a_1$, $l_2 = a_3$.

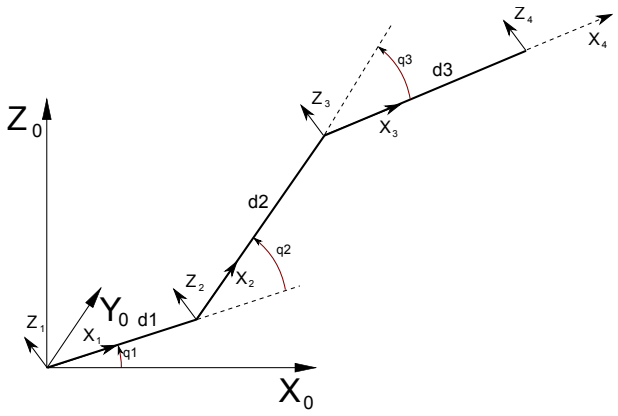
Do wyznaczenia kinematyki pojedynczej nogi (rysunek 2.3) posłużono się algorytmem Denavita–Hartenberga [18]. Przejście do kolejnych układów współrzędnych związanych z przegubami q_1, q_2, q_3 wygląda następująco:

$$\begin{aligned} {}^0_1T &= Rot\left(Z, q_1\right), \\ {}^1_2T &= Rot\left(X, -\frac{\pi}{2}\right) Trans\left(X, d_1\right) Rot\left(Z, q_2\right), \\ {}^2_3T &= Trans\left(X, d_2\right) Rot\left(Z, q_3\right), \\ {}^3_4T &= Trans\left(X, d_3\right). \end{aligned}$$



(a) widok z góry (b) widok z przodu

Rysunek 2.2 Poglądowe wymiary robota



Rysunek 2.3 Kinematyka nogi

Po obliczeniu wektor translacji ${}^0_4T$ ma postać:

$$\begin{cases} x = c_1 \left(c_{23} d_3 + c_2 d_2 + d_1\right) \\ y = s_1 \left(c_{23} d_3 + c_2 d_2 + d_1\right) . \\ z = s_{23} d_3 + s_2 d_2 \end{cases} \tag{2.2}$$

2.3 Odwrotne zadanie kinematyki nogi robota.

Do rozwiązania zadania odwrotnego kinematyki zastosowano metodę algebraiczną.

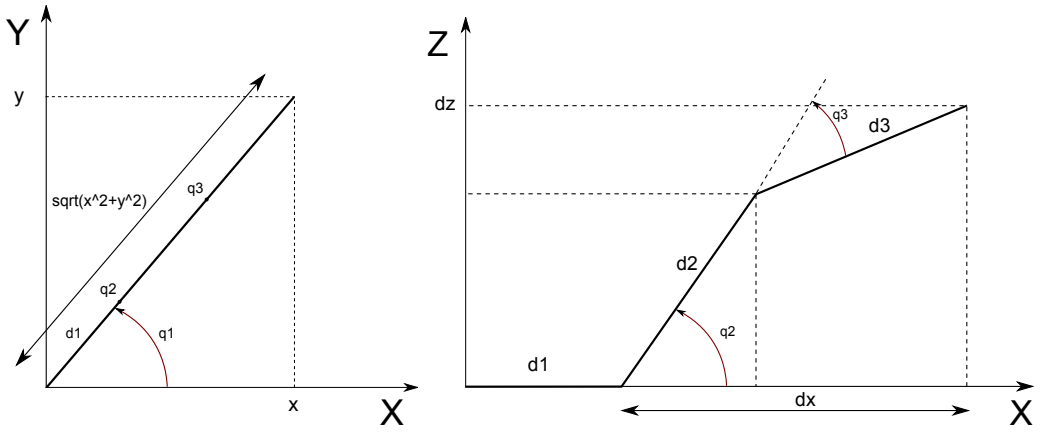
Wprowadzone są następujące ograniczenia: postura nogi typu "noga owada" (zawsze zgieta w kolanie), kąt $q_1 \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ — przekroczenie tego zakresu oznaczało by kolizję nogi z korpusem.

Dzieląc y przez x, z zależności 2.2 uzyskuje się

$$q_1 = \text{atan2}(y, x). \tag{2.3}$$

Rysunek 2.4 przedstawia rzuty nogi na płaszczyznę OXY i OXY. Z twierdzenia Pitagorasa można określić długość nogi na płaszczyźnie OXY:

$$\sqrt{x^2 + y^2}.$$



(a) rzut na płaszczyznę OXY (b) rzut na płaszczyznę OXZ

Rysunek 2.4 Kinematyka odwrotna nogi

Można więc wyznaczyć długości:

$$dx = \sqrt{x^2 + y^2} - d_1,$$
$$dz = z.$$

Następnie, patrząc tylko na segmenty d_2 i d_3, uzyskuje się

$$c_3 = \frac{dx^2 + dz^2 - d_2^2 - d_3^2}{2d_2d_3},$$
$$s_3 = \pm\sqrt{1 - c_3^2}$$

i w efekcie

$$q_3 = \text{atan2}(\pm s_3, c_3). \tag{2.4}$$

Wybór znaku funkcji s_3 jest uzależniony od postury nogi. Znak ujemny oznacza że noga jest zgięta w kolanie, a więc taka jak przyjęte założenie.

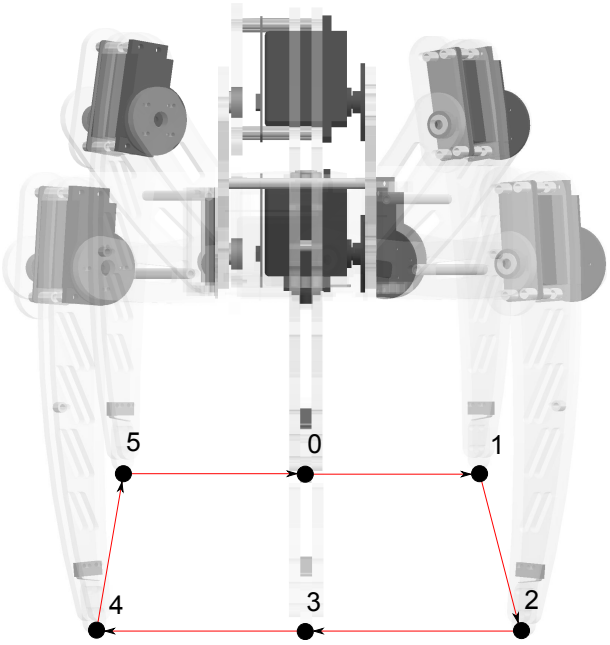
Wyprowadzenie równań do wyliczenia kąta q_2 dostępne jest w pozycji [21], poniżej przedstawiono gotowe rozwiązanie:

$$m_1 = d_2 + d_3c3,$$
$$m_2 = d_3s_3,$$
$$r = \sqrt{m_1^2 + m_2^2}$$

$$q_2 = \text{atan2}\left(\frac{dz}{r}, \frac{dx}{r}\right) - \text{atan2}(m_2, m_1). \tag{2.5}$$

2.4 Generator chodu

Sterowanie ruchem robota stanowi złożony problem. Aby robot mógł śledzić zadaną ścieżkę, konieczne jest odpowiednie sterowanie ruchem kończyn, pozwalające przeprowadzać korpus do kolejnych punktów ścieżki. Podczas generowania ruchu robota konieczne jest zdefiniowanie typu chodu robota. Typy chodu opracowywane są często na podstawie wzorców biologicznych np. ssaki, owady. O ile ssaki najczęściej korzystają z chodu dynamicznego, owady korzystają z chodu stabilnego statycznie. Pozwala on utrzymać równowagę



Rysunek 2.5 Fazy kroku

Tabela. 2.1 Etapy kroku

Etapy	0	1	2	3
noga 1,3,5	0	1	2	3
noga 2,4,6	3	4	5	0

poprzez odpowiednie przemieszczenie środka ciężkości. W maszynach sześcionożnych stosuje się najczęściej chód statyczny trójpodpodporowy. W rozdziale tym przedstawiony został algorytm chodu trójpodporowego po płaskiej powierzchni.

2.4.1 Trajektoria ruchu końca nogi

Trajektoria końca nogi zadana jest w przestrzeni kartezjańskiej i złożona jest z sześciu faz:

0 — faza środkowa niepodparta,

1 — faza wykroczna niepodparta,

2 — faza wykroczna podparta,

3 — faza środkowa podparta,

4 — faza zakroczna podparta,

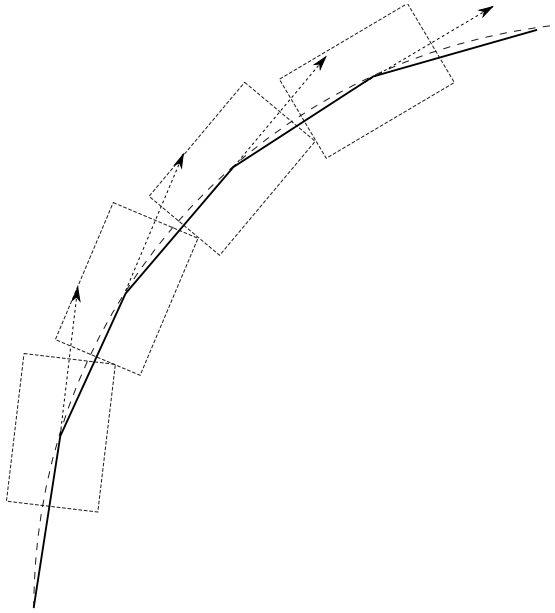
5 — faza zakroczna niepodparta.

Rysunek 2.5 przedstawia konfigurację nogi w poszczególnych fazach kroku. Część trajektorii uczestniczącej w przemieszczeniu korpusu to fazy od 2 do 4. Jako że rozpatrywany jest ruch trójpodporowy, nogi trójkami współpracują ze sobą w zadaniu przemieszczenia korpusu.

Tabela 2.1 przedstawia kolejne etapy przemieszczenia robota o jeden krok. Kompletny krok składa się z 4 etapów. Etap zerowy jest etapem wyjściowym, w którym nogi znajdują się w pozycjach środkowych. Przemieszczenie korpusu robota następuje w etapie 2–3 przez nogi nieparzyste oraz w etapie 0–1 przez nogi parzyste. Kolejno nogi parzyste i nieparzyste przemieszczają robota o połowę długości kroku. Cykliczne powtarzanie tych etapów powoduje, że robot zaczyna chodzić.

2.4.2 Trajektoria ruchu końca nogi przy zmianach kierunku ruchu

W fazie podporowej nogi stykają się z nieruchomym podłożem, dlatego też trajektorie końców nóg względem korpusu odpowiadają trajektorii ruchu korpusu względem podłoża. W trakcie ruchu do przodu końce nóg przemieszczają się w fazie podporowej wzdłuż osi OY, natomiast podczas przemieszczenia w bok — wzdłuż osi OX. Problem pojawia się w przypadku skręcania w ruchu postępowym. Trajektorie ruchu końców nóg w fazie podporowej powinny być odpowiednimi łukami, natomiast ich długość musi odpowiadać długości kroku. Rozwiązanie takie jest kłopotliwe i częściej stosuje się metodę przybliżenia łuku skrętu odcinkami (pozycja [21]). Metoda ta nie wymaga zmian kształtu trajektorii końca nogi z wyjątkiem modyfikacji współrzędnych początku i końca fazy podporowej.



Rysunek 2.6 Obrót korpusu

Koniec nogi wyrażony jest w lokalnym układzie współrzędnych danej nogi O_i ($i = 1, \ldots, 6$), a ten z kolei wyrażony jest w układzie robota O_R. Aby zrealizować przemieszczenie korpusu robota w nową pozycję, należy wyznaczyć położenie końców nóg na końcu fazy podporowej.

Niech wektor p_{i3} określa pozycję neutralną końca nogi wyrażoną w układzie O_i w fazie 3. Jest to pozycja nogi, gdy robot nie wykonuje żadnego ruchu. Dla przykładu: $p_{i3} = [100, 0, -100, 1]^T$. Niech O'_R oznacza pozycję korpusu robota na końcu fazy podporowej, wyrażoną w układzie O_R. Pojedynczy krok niech będzie opisany parametrami:

- $offset_{up}$ — wysokość, na jaką jest podnoszona noga;
- $offset_x$ — przemieszczenie robota wzdłuż osi OX w trakcie jednego kroku;

- $offset_y$ — przemieszczenie robota wzdłuż osi OY w trakcie jednego kroku;
- α_z — skręt robota wzdłuż osi OZ w trakcie jednego kroku.

Translacja układu O'_R w O_R ma postać:

$$T' = \begin{bmatrix} \cos(\alpha_z) & -\sin(\alpha_z) & 0 & -offset_x \\ \sin(\alpha_z) & \cos(\alpha_z) & 0 & offset_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.6}$$

Aby wyznaczyć pozycję nogi na końcu fazy podporowej p_{i4} wyrażoną w układzie lokalnym nogi O_i, należy wyrazić pozycję neutralną końca nogi p_{i3} w układzie wsp. korpusu robota (O_R), dokonać translacji T', a następnie otrzymaną pozycję p^R_{i4} wyrazić w układzie O_i:

$$\begin{aligned} p^R_{i3} &= O_i \cdot {}^R_{i3}, \\ p^R_{i4} &= T' \cdot {}^R_{i3}, \\ p_{i4} &= O_i^{-1} \cdot {}^R_{i4}. \end{aligned} \tag{2.7}$$

Konieczne jest jeszcze obliczenie pozycji końca nogi w pozostałych fazach kroku:

$$\begin{aligned} p^R_{i2} &= p^R_{i2} = 2 \cdot p^R_{i3} - p^R_{i4}, \\ p_{i2} &= O_i^{-1} \cdot {}^R_{i2}, \\ p_{i1} &= p_{i2} + up, \\ p_{i5} &= p_{i4} + up, \end{aligned} \tag{2.8}$$

gdzie up to wektor postaci:

$$up = \begin{bmatrix} 0 & 0 & offset_{up} & 1 \end{bmatrix}^T.$$

2.4.3 Trajektoria ruchu końca nogi przy zmiennej rotacji i nutacji korpusu

W przypadku konieczności utrzymania korpusu w poziomie podczas kroczenia po pochyłych powierzchniach konieczna jest możliwość kroczenia ze zmiennym nachyleniem korpusu.

Wprowadzone zostaną dodatkowe parametry kroku:

- $roll$ — (nutacja) obrót względem osi OX,
- $pitch$ — (rotacja) obrót względem osi OY,
- $offset_z$ — wysokość korpusu nad powierzchnią.

Niech

$$R = Trans(Z, offset_z) Rot(X, roll) Rot(Y, pitch). \tag{2.9}$$

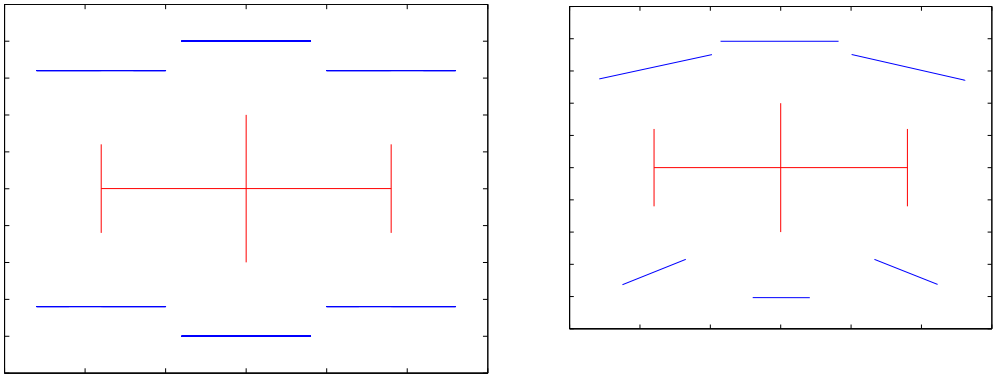
Wyznaczone w równaniach 2.7, 2.8, pozycje sotpy p^R_{i4} i p^R_{i2} należy przed transformacją do układu O_i pomnożyć przez macierz R

$$\begin{aligned} p^R_{i4} &= R \cdot p^R_{i4}, \\ p^R_{i2} &= R \cdot p^R_{i2}. \end{aligned} \tag{2.10}$$

2.5 Generator chodu – wersja iteracyjna

Przedstawiony w rozdziale 2.4 generator chodu opiera się na pewnym uproszczeniu przedstawionym w pozycji [21]. Mianowicie algorytm wymaga wyznaczenia pozycji stopy jedynie na początku i na końcu fazy podporowej. Pozwala to na znaczne uproszczenie obliczeń, wykonuje się je raz przed przystąpieniem do danego kroku. Za przemieszczenie stopy z jednej pozycji do drugiej odpowiada generator lokalny danej nogi. Przykład takiego lokalnego generatora omówiony został przy okazji omawiania lokalnego sterownika nogi w rozdziale 3.3.

Przedstawiony w poprzednim rozdziale generator niestety nie pozwala na płynne śledzenie trajektorii przez robota. Możliwość zmiany parametrów kroczenia istnieje tylko przed przystąpieniem do kolejnego kroku. Ponadto przedstawione uproszczenie zakłada, że pomiędzy kolejnymi fazami kroku stopa przemiesza się po lini prostej. Wprowadza to poślizgi podczas kroczenia po łuku (rysunek 2.7).



(a) ruch po prostej (b) ruch po łuku

Rysunek 2.7 Trajektoria stóp podczas kroczenia (generator chodu wersja 1)

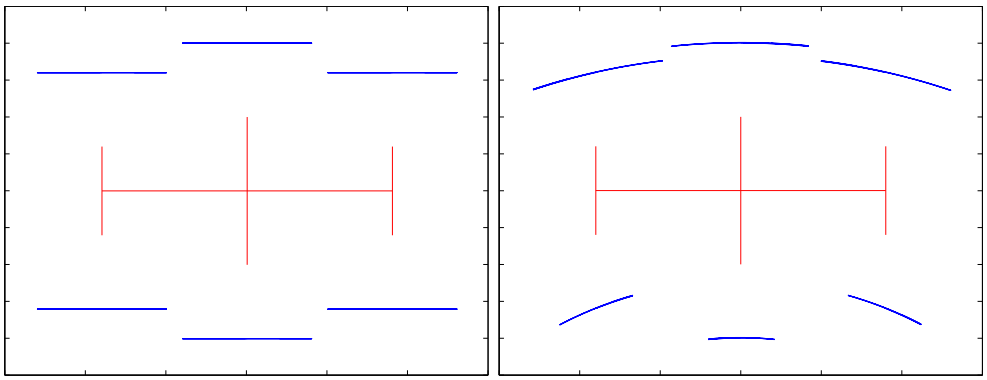
W rozdziale tym przedstawiono usprawniona wersję generatora chodu, pozbawioną wspomnianych wad.

Iteracyjny algorytm kroczenia

Podstawową zmianą w stosunku do poprzednio omówionego algorytmu, jest to, że w proponowanej modyfikacji przeliczenia dokonywane są iteracyjnie co pewien przedział czasu Δt. W praktycznej realizacji $\Delta t = 20ms$, wynika to z maksymalnej częstotliwości z jaką można zadawać pozycję dla zastosowanych serw modelarskich. Algorytm pozwala na zmianę parametrów kroczenia w dowolnym momencie, co umożliwia na śledzenie trajektorii. Ponadto algorytm umożliwia kroczenie po nierównościach oraz pozbawiony jest problemu poślizgów podczas kroczenia po łukach (rysunek 2.8).

Parametry kroczenia:

- v_y – prędkość wzdłuż osi OY,
- v_x – prędkość wzdłuż osi OX,
- v_{α_z} – prędkość kątowa wokół osi OZ.



(a) ruch po prostej (b) ruch po łuku

Rysunek 2.8 Trajektoria stóp podczas kroczenia (generator chodu wersja 2)

Algorytm wymaga, aby nogi wstępnie ustawione były w pozycję neutralną (rysunek 2.5 faza 3). Następnie podnoszone zostają nogi niepodporowe (np. nogi 1,3,5). Pozycje stóp w danej iteracji wyznaczane są na podstawie ich pozycji w iteracji poprzedniej.

Aby wyznaczyć pozycję stopy w danej iteracji stosuje się następujące przeliczenia:

$$\begin{aligned} p_{i-1}^R &= O \cdot p_{i-1}, \\ p_i^R &= T \cdot p_{i-1}^R, \\ p_i &= O^{-1} \cdot p_i^R, \end{aligned} \tag{2.11}$$

gdzie macierz translacji T ma następującą postać:

- dla nóg podporowych

$$T = \begin{bmatrix} \cos(\Delta\alpha_z) & -\sin(\Delta\alpha_z) & 0 & -\Delta x \\ \sin(\Delta\alpha_z) & \cos(\Delta\alpha_z) & 0 & \Delta y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.12}$$

- dla nóg niepodporowych

$$Tn = \begin{bmatrix} \cos(-\Delta\alpha_z) & -\sin(-\Delta\alpha_z) & 0 & \Delta x \\ \sin(-\Delta\alpha_z) & \cos(-\Delta\alpha_z) & 0 & -\Delta y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.13}$$

Δx, Δy, $\Delta\alpha_z$ są to przyrosty drogi w danej iteracji:

$$\begin{aligned} \Delta x &= \Delta t \cdot v_x, \\ \Delta y &= \Delta t \cdot v_y, \\ \Delta\alpha_z &= \Delta t \cdot v_{\alpha_z}. \end{aligned} \tag{2.14}$$

Zmiana nóg podporowych dokonywana jest, gdy któraś z nóg wykroczy poza przyjęty zakres roboczy. Podczas zmiany nóg podporowych nogi opuszczane są nezależnie do momentu kontaktu stopy z podłożem. W ten sposób w kolejnych iteracjach algorytmu uwzględniona zostanie nierówność podłoża.

Zmiany parametrów kroczenia w dowolnym momencie czasami prowadzą do zakleszczenia algorytmu. Gdy któraś z nóg znajduje się na skraju zakresu roboczego, może dojść do sytuacji, w której robot co chwilę zmienia nogi podporowe. W sytuacji takiej konieczny jest powrót do pozycji neutralnej i wznowienie ruchu z tej pozycji.

Możliwość kroczenie ze zmienną rotacją i nutacją można wprowadzić w sposób analogiczny do tego przedstawionego w rozdziale 2.4.3.

2.6 Wyznaczanie odchylenia od pionu na podstawie pomiaru przyspieszeń

Czujniki przyspieszenia dostarczają informacji o przyspieszeniu ziemskim. Odpowiednio rozmieszczone czujniki osobno w osi X, Y, Z, pozwalają na wyznaczenie wektora przyspieszenia ziemskiego (rysunek 2.9).

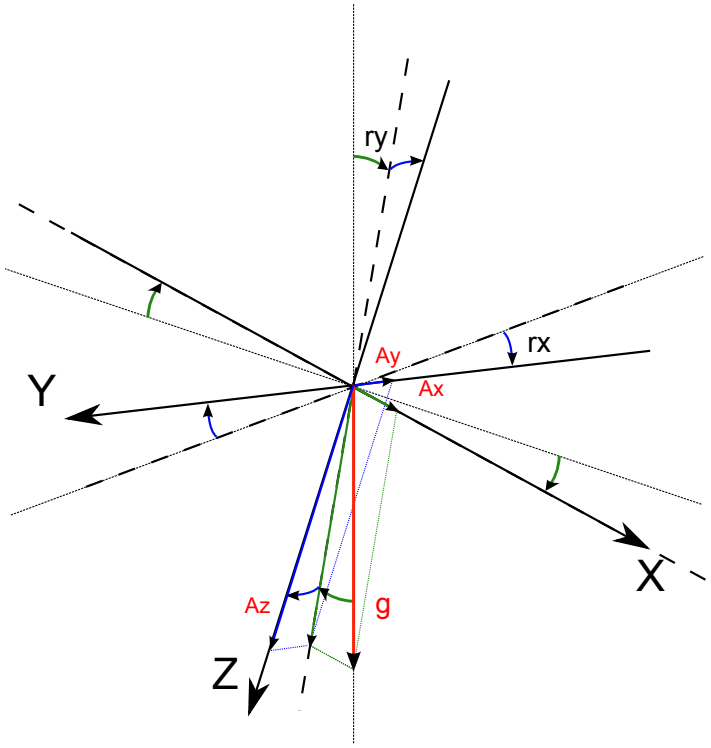
Rotację względem osi OX wyznaczyć można w jako:

$$rx = \arctan\left(\frac{Ay}{Az}\right) \tag{2.15}$$

natomiast rotację względem osi OY:

$$ry = \arctan\left(\frac{-Ax}{Ay \cdot \sin(rx) + Az \cdot \cos(rx)}\right). \tag{2.16}$$

Powyższe równania są poprawne tylko w przypadku, gdy moduł INS jest w bezruchu lub porusza się ruchem jednostajnym. w przeciwnym razie czujniki przyspieszenia, oprócz składowych przyspieszenia ziemskiego, rejestrują dodatkowo przyspieszenie wynikające ze zmian położenia modułu.



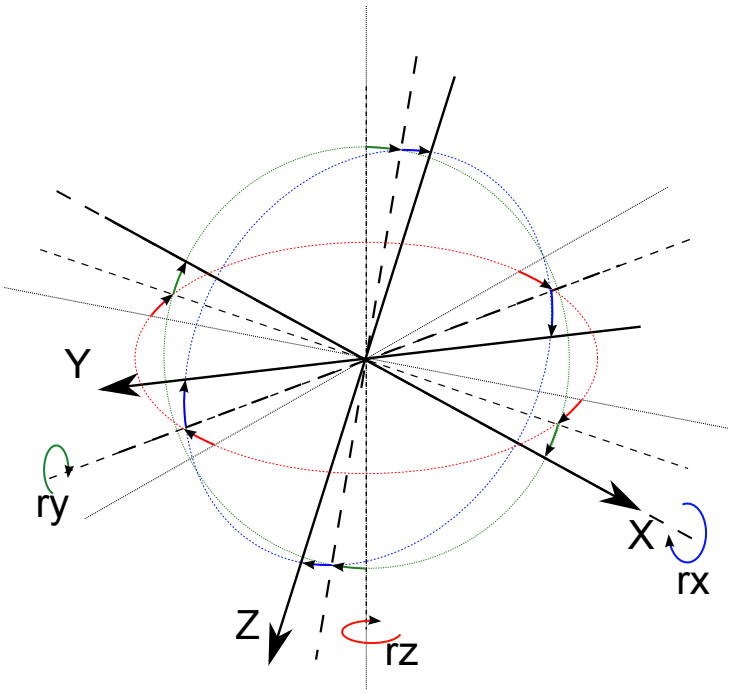
Rysunek 2.9 Rozkład wektora przyspieszenia ziemskiego

2.7 Wyznaczanie odchylenia od pionu i orientacji na podstawie pomiaru prędkości kątowych

Jednoznaczne określenie wzajemnego zorientowania układów kartezjańskich możliwe jest przez złożenie rotacji wokół poszczególnych osi. Osie, względem których dokonuje się rotacji oraz kolejność, zależą od przyjętej konwencji. Do jednej z najczęściej stosowanych należy $OZ \rightarrow OX \rightarrow OZ$. W pracy zdecydowano się zastosować konwencję $OZ \rightarrow OY \rightarrow OX$ (rysunek 2.10).

Żyroskopy pozwalają na pomiar prędkości kątowej. Poprzez scałkowanie sygnału można uzyskać wartość kąta wychylenia. Rozmieszczenie czujników w trzech płaszczyznach pozwala na rejestrację prędkości kątowej w każdej z osi. Zależność łącząca prędkości kątowe ω_x, ω_y, ω_z rejestrowane przez żyroskopy w lokalnym układzie współrzędnych z prędkościami $\dot{\theta}_x$, $\dot{\theta}_y$, $\dot{\theta}_z$ w układzie globalnym dla rotacji $OZ \rightarrow OY \rightarrow OX$, gdzie θ_x, θ_y, θ_z są to rotacje w poszczególnych osiach, przedstawiono w pozycji [12] i jest ona następująca:

$$\begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} = \begin{bmatrix} 1 & \sin\theta_x \tan\theta_y & \cos\theta_x \tan\theta_y \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x \sec\theta_y & \cos\theta_x \sec\theta_y \end{bmatrix} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.17}$$



Rysunek 2.10 Złożenie rotacji $OZ \rightarrow OY \rightarrow OX$

Zależności te poprawne są tylko w przypadku, gdy wartość kąta θ_y jest w granicach $\pm 90°$. Aby móc pominąć te ograniczenia, można posłużyć się opisem orientacji z użyciem kwaternionów, co zostało szerzej opisane w pozycji [12]. W pracy pozostano przy przedstawionym powyżej opisie, ponieważ badana konstrukcja (robot kroczący) nigdy nie będzie w sytuacji gdy kąt θ_y wykroczy poza granice $\pm 90°$.

2.8 Filtracja Kalmana

Jedną z najczęściej stosowanych metod filtracji w systemach nawigacji jest filtracja Kalmana. Jest to rodzaj filtracji dyskretnej, która estymuje stan procesu tak, aby zminimalizować błąd średnio–kwadratowy. Filtr ten korzysta z wszystkich dostępnych pomiarów, bez względu na to z jaką dokładnością i precyzją zostały one wykonane. Dokonuje fuzji sygnałów, a następnie na ich podstawie dokonuje najlepszej estymacji stanu.

Filtr Kalmana składa się z dwóch faz: predykcji, i korekcji (rysunek 2.11). Pierwsza faza to tak zwana aktualizacja czasowa, gdzie na podstawie informacji ze stanu poprzedniego wyznacza się estymowaną wartość stanu $\hat{x}$ oraz jego kowariancję. W fazie drugiej nazywanej inaczej aktualizacją pomiarową dokonuje się korekcji wartości stanu i jego kowariancji. Jest to pewien rodzaj sprzężenia zwrotnego.

Do opisu procesu oraz systemu pomiarowego stosuje się modele matematyczne

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\ z_k &= Hx_k + v_k \end{aligned} \tag{2.18}$$

Pierwsze równanie to równanie procesu, gdzie x_k to stan procesu, x_{k-1} to stan procesu z chwili poprzedniej u to sterowanie, natomiast w to szum procesu. Drugie równanie to model pomiaru, gdzie H to tzw. Wyjście filtru łączące stan procesu z filtrem, v to szum pomiaru.

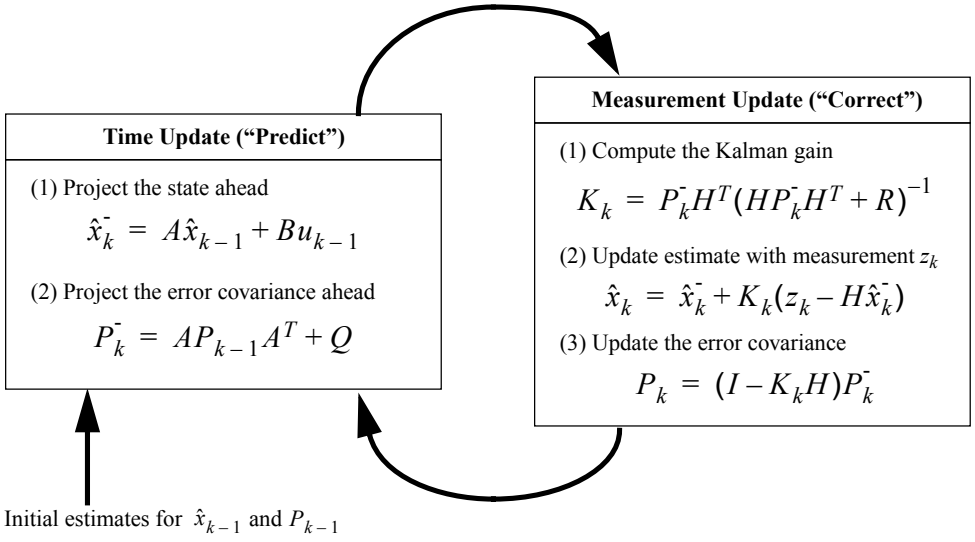
Błąd pomiaru a priori e_k^- oraz a posteriori e_k definiuje się jako

$$\begin{aligned} e_k^- &= x_k - \hat{x}_k^- \\ e_k &= x_k - \hat{x}_k \end{aligned}, \tag{2.19}$$

gdzie $\hat{x}_k^-$ to estymowany stan a priori uzyskany z procesu, $\hat{x}_k$ to estymowany stan a posteriori uwzględniający pomiar z_k, natomiast x_k to stan rzeczywisty, który nie jest znany.

Macierze kowariancji a priori P_k^- i a posteriori P_k mają postać:

$$\begin{aligned} P_k^- &= E[e_k^-, e_k^-] \\ P_k &= E[e_k, e_k]. \end{aligned} \tag{2.20}$$



Rysunek 2.11 Algorytm filtracji Kalmana [19]

Równania filtru Kalmana w fazie predykcji są następujące

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1}^- + Bu_{k-1} \\ \hat{P}_k^- &= A\hat{P}_{k-1}A^T + Q \end{aligned} \tag{2.21}$$

$\hat{x}_k^-$ oraz P_k^- to estymowanie stanu procesu i macierzy kowariancji a priori, $\hat{x}_k$ oraz P_k to szacowne wartości stanu kowariancji a posteriori z poprzedniego kroku, natomiast Q jest to macierz wariancji procesu.

W drugiej fazie wyznaczane jest wzmocnienie Kalmana

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}, \tag{2.22}$$

gdzie R jest to wariancja pomiaru.

Następnie dokonywana jest korekcja estymowanego stanu oraz korekta macierzy kowariancji

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\ P_k &= (I - K_K H)P_k^- \end{aligned} \tag{2.23}$$

gdzie z_k to pomiar, natomiast I to macierz jednostkowa.

2.9 Wyznaczanie prędkości oraz przemieszczenia na podstawie przyspieszeń

Określenie położenia oraz orientacji robota względem globalnego układu odniesienia realizowane jest jako złożenie translacji oraz rotacji:

$$T = Trans(x, y, z) \cdot Rot(\theta_z, \theta_y, \theta_x) \tag{2.24}$$

Ponieważ czujniki rejestrujące przyspieszenie i prędkość kątową robota nie są zorientowane tak samo jak układ globalny, koniecznie jest stosowne przeliczenie tych wartości. Sposób przeliczenia prędkości kątowych przedstawiono w rozdziale 2.7.

Akcelerometry umieszczone w osiach OX, OY, OZ zmieniają orientację wraz z robotem. Aby móc wykorzystać je do pomiaru przyspieszeń w układzie globalnym, konieczna jest wpierw eliminacja przyspieszenia ziemskiego, następnie przeliczenie przyspieszeń z układu lokalnego na układ globalny.

Sposób eliminacji przyspieszenia ziemskiego oraz przyspieszenia odśrodkowego, przedstawiono w pracy [12] i wygląda on następująco:

$$\begin{aligned} a_x &= A_x + V_y\omega_z - V_z\omega_y + g\sin\theta_y \\ a_y &= A_y - V_x\omega_z - V_z\omega_x - g\cos\theta_y\sin\theta_x\,. \\ a_z &= A_z + V_x\omega_y - V_y\omega_x - g\cos\theta_y\cos\theta_x \end{aligned} \tag{2.25}$$

Jak widać, przyspieszenie odśrodkowe zależy od prędkości przemieszczania się robota. Ponieważ badany w pracy robot kroczący porusza się z niewielką prędkością, postanowiono zrezygnować z korekcji przyspieszenia wynikającej z przyspieszenia odśrodkowego. Ostatecznie stosuje się jedynie eliminację przyspieszenia ziemskiego:

$$\begin{aligned} a_x &= A_x + g\sin\theta_y \\ a_y &= A_y - g\cos\theta_y\sin\theta_x\,. \\ a_z &= A_z - g\cos\theta_y\cos\theta_x \end{aligned} \tag{2.26}$$

Uzyskuje się w ten sposób przyspieszenia w układzie lokalnym. Przeliczenie wartości przyspieszeń z układu lokalnego na globalny wygląda następująco:

$$\begin{bmatrix} a_{xg} \\ a_{yg} \\ a_{zg} \end{bmatrix} = \begin{bmatrix} c_y c_z & c_z s_y s_x - c_x s_z & c_x c_z s_y + s_x s_z \\ c_y s_z & c_x c_z + s_y s_x s_z & c_x s_y s_z - c_z s_x \\ -s_y & c_y s_x & c_y c_x \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tag{2.27}$$

gdzie $s_x = \sin\theta_x$, $c_x = \cos\theta_x$ itp.

Dysponując przyspieszeniem w układzie globalnym, można wyznaczyć prędkość oraz przemieszczenie w tym układzie, stosując kolejne całkowania.

$$V = \int (a) dt \tag{2.28}$$

$$S = \int (V) dt \tag{2.29}$$

2.10 Triangulacja Delone

Jedną z metod interpolacji wysokości jest triangulacja. Dany na płaszczyźnie zbór punków łączy się odcinkami, dzieląc ich otoczkę wypukłą na trójkąty. Po dodaniu informacji o wysokości każdego z punktów otrzymuje się przybliżenie ukształtowania terenu w postaci funkcji ciągłej przedziałami liniowej. Istnieje pewna skończona liczba możliwych triangulacji jednego zbioru punktów. W celu najlepszego przybliżenia ukształtowania terenu należy unikać tworzenia wąskich dolin. Można to osiągnąć, maksymalizując wartości kątów w grafie. Uporządkowany niemalejąco ciąg wszystkich kątów tworzących triangulację nazywa się jej wektorem kątów. Triangulacja danego zbioru punktów o wektorze kątów leksykograficznie największym nazywana jest legalną i jest triangulacją Delone.

Triangulacja złożona z trójkątów $p_i p_j p_k$ i $p_i p_j p_r$ jest nielegalna, jeżeli punkt p_k leży wewnątrz okręgu opisanego na $p_i p_j p_r$. Wtedy też krawędź $\overline{p_i p_j}$ nazywana jest nielegalną. W celu uczynienia triangulacji legalną, należy zastąpić krawędź $\overline{p_i p_j}$ przez $\overline{p_r p_k}$. Na potrzeby tej pracy założono, że nie wystąpi przypadek, w którym cztery punkty położone są na jednym okręgu (wtedy nie byłoby możliwe jednoznaczne wskazanie triangulacji Delone).

Zaimplementowano przyrostowy algorytm 1 triangulacji Delone zaczerpnięty z pozycji [4]. Przyjęto zaproponowany tam sposób postępowania w przypadku, kiedy wstawiany punkt znajduje się w trójkącie o ujemnym wierzchołku. Kolejne punkty dodawane są na bieżąco, dlatego nie losuje się początkowej permutacji.

Algorithm 1 Triangulacja Delone

Require: Zbiór P zawierający n punktów na płaszczyźnie
Ensure: Triangulacja Delone $\mathcal{T}$ zbioru P

procedure TriangulacjaDelone(P)
  Niech p_{-1}, p_{-2} i p_{-3} będzie zbiorem trzech punktów takim, że P jest zawarty w trójkącie $p_{-1}p_{-2}p_{-3}$.
  Inicjuj $\mathcal{T}$ jako triangulację zawierającą pojedynczy trójkąt $p_{-1}p_{-2}p_{-3}$.
  Oblicz losową permutację p_1, p_2, ..., p_n w P.
  for $r \leftarrow 1, r$ **do** ▷ Wstaw p_r do $\mathcal{T}$:
    Znajdź trójkąt $p_ip_jp_k \in \mathcal{T}$ zawierający p_r.
    if p_r leży we wnętrzu trójkąta $p_ip_jp_k$ **then**
      Dodaj krawędzie z p_r do trzech wierzchołków $p_ip_jp_k$, dzieląc w ten sposób $p_ip_jp_k$ na trzy trójkąty.
      LegalizeEdge(p_r, $\overline{p_i, p_j}$, $\mathcal{T}$)
      LegalizeEdge(p_r, $\overline{p_j, p_k}$, $\mathcal{T}$)
      LegalizeEdge(p_r, $\overline{p_k, p_i}$, $\mathcal{T}$)
    else ▷ p_r leży na krawędzi trójkąta $p_ip_jp_k$, na przykład $\overline{p_i, p_j}$
      Dodaj krawędzie z p_r do p_k i do trzeciego wierzchołka p_l drugiego trójkąta sąsiadującego z $\overline{p_i, p_j}$, dzieląc w ten sposób dwa trójkąty sąsiadujące z $\overline{p_i, p_j}$ na cztery trójkąty.
      LegalizeEdge(p_r, $\overline{p_i, p_l}$, $\mathcal{T}$)
      LegalizeEdge(p_r, $\overline{p_l, p_j}$, $\mathcal{T}$)
      LegalizeEdge(p_r, $\overline{p_j, p_k}$, $\mathcal{T}$)
      LegalizeEdge(p_r, $\overline{p_k, p_i}$, $\mathcal{T}$)
    end if
    Usuń z $\mathcal{T}$ punkty p_{-1}, p_{-2} i p_{-3} wraz ze wszystkimi incydentnymi z nimi krawędziami
  end for
  return $\mathcal{T}$
end procedure

procedure LegalizeEdge(p_r, $\overline{p_i, p_j}$, $\mathcal{T}$) ▷ Wstawianym punktem jest p_r, a $\overline{p_i, p_j}$ jest krawędzią $\mathcal{T}$, która może wymagać zamiany („przekręcenia").
  if $\overline{p_i, p_j}$ jest nielegalna **then**
    Niech $p_ip_jp_k$ będzie trójkątem sąsiadującym z $p_rp_ip_j$ wzdłuż $\overline{p_i, p_j}$.
    Zastąp $\overline{p_i, p_j}$ przez $\overline{p_r, p_k}$ ▷ „Przekręć" $\overline{p_i, p_j}$
    LegalizeEdge(p_r, $\overline{p_i, p_k}$, $\mathcal{T}$)
    LegalizeEdge(p_r, $\overline{p_k, p_j}$, $\mathcal{T}$)
  end if
end procedure

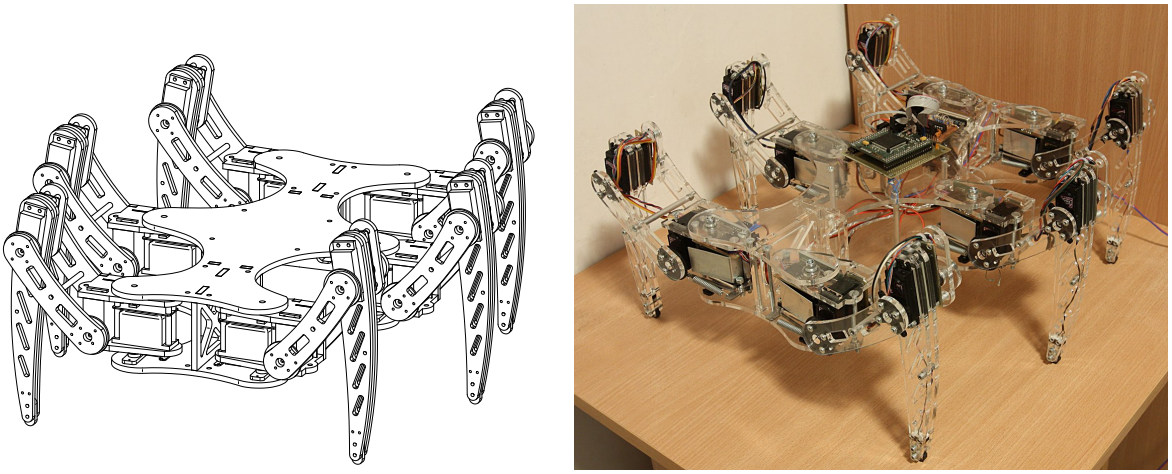
Rozdział 3

Budowa robota Mrówa

Robota Mrówa powstał w ramach działalności autora w kole naukowym "KoNaR". Prace nad konstrukcją rozpoczęto w marcu 2009 roku. Celem projektu było stworzenie sześcionożnego robota kroczącego zdolnego pokonywać przeszkody i pochyłości o nieznacznej wysokości.

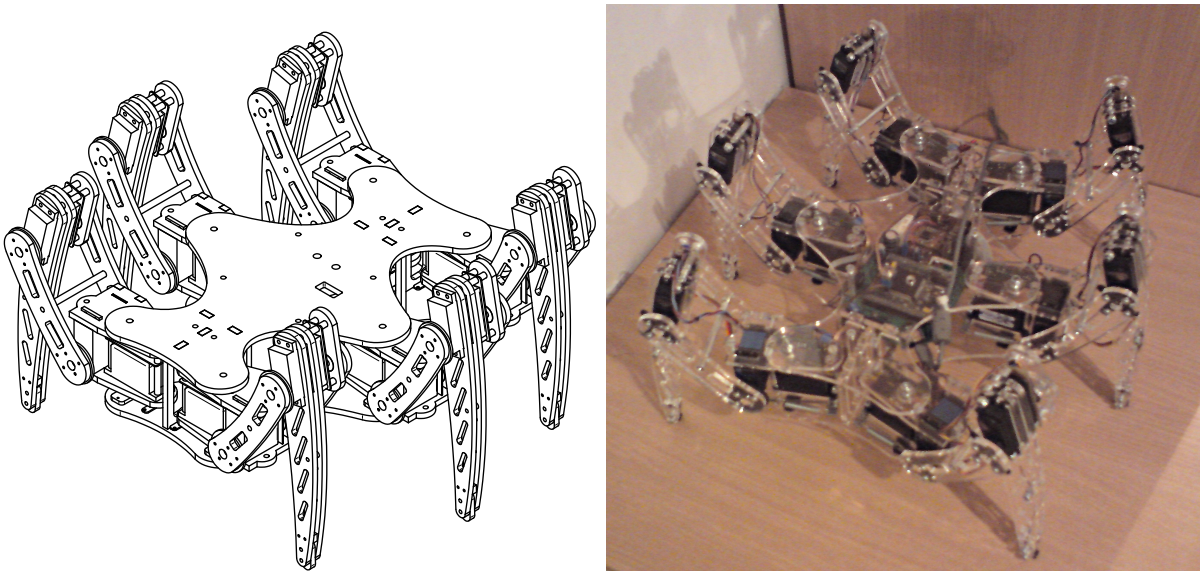
Postawione założenie wymaga znacznego prześwitu robota, a tym samym kończyn robota o znacznej długości. Początkowo robot posiadał kończyny o długościach segmentów w stosunku około 1:4:5, jednakże segment drugi został skrócony z powodu niewystarczającej mocy silników. Obecnie stosunek długości kolejnych segmentów nogi to około 1:3:5. Dodatkowo wprowadzono sprężyny wspomagające silniki. Kolejne eksperymenty pokazały, że robot ma tendencję do niebezpiecznego przechylania się, przy kroku o znacznej długości. Źle rozłożona masa wymogła konieczność zmiany wymiarów korpusu robota. Zmniejszenie korpusu oraz skupienie masy jak najbliżej centrum rozwiązało ten problem. Na rysunku 3.1 przedstawiono robota przed modernizacją korpusu. Aktualna wersja robota przedstawiona jest na rysunku 3.2.

Aby możliwe było wykorzystanie robota do budowy mapy wysokości terenu konieczna jest możliwość określenia lokalizacji robota. W tym celu opracowano moduł nawigacji inercyjnej INS. Okazał się on bardzo pomocny przy korekcji przechyłów bocznych robota podczas kroczenia po nierównościach, jednak nie do końca spełnia swoje zadanie przy określeniu lokalizacji robota. Przydało by się tu zastosować dodatkowo systemy zewnętrznej nawigacji jak np. GPS.



(a) projekt (b) realizacja

Rysunek 3.1 Robot kroczący Mrówa przed modernizacją korpusu



(a) projekt (b) realizacja

Rysunek 3.2 Robot kroczący Mrówa

Do sterowania robotem wykorzystano komputer PC. Stworzono oprogramowanie sterujące, które pozwala na trójwymiarową wizualizację robota oraz mapy wysokości. Aplikacja posiada także wbudowany symulator robota. Oprogramowanie opisano w rozdziale 4. Komunikacja z robotem możliwa jest na dwa sposoby: przewodowo przez port USB, lub bezprzewodowo przez Bluetooth[1].

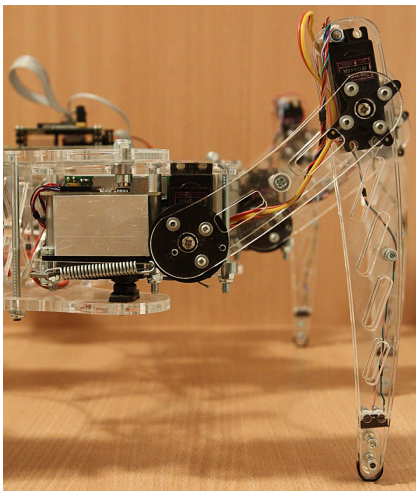
Podstawowe parametry robota:

- jednostka główna robota, mikrokontroler MC68332
- osobny sterownik dla każdej z nóg, oparty o mikrokontroler MC9S12C32
- moduł INS
- komunikacja przez USB lub bezprzewodowo przez Bluetooth
- zasilanie bateryjne Li–Pol 7.4V
- średni pobór prądu około 3A (szczytowy 10A)
- masa około 4kg
- wymiary poglądowe przy nogach w pozycji domyślnej: długość 33cm, szerokość 39cm, wysokość 21cm

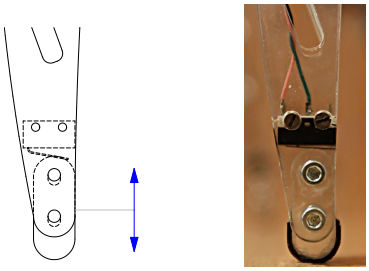
3.1 Mechanika

Mechanika robota zaprojektowana została w programie Solid Edge v19. Solid Edge jest jednym z wielu dostępnych obecnie programów wspomagających projektowanie. Program jest bardzo prosty w obsłudze, cały etap projektowania odbywa się w środowisku trójwymiarowym, dzięki czemu łatwo wyeliminować wiele błędów już na etapie projektowania.

[1] obecnie nie ma takiej możliwości, płyta sterownika jest przygotowana, jednak brak samego modułu Bluetooth



Rysunek 3.3 Sprężyna podtrzymująca ciężar robota



Rysunek 3.4 Stopa robota

Konstrukcja zaprojektowana jest z myślą o łatwym serwisowaniu robota. Wszystkie łączenia elementów wykonane są bez użycia kleju tak, aby możliwa była łatwa wymiana uszkodzonych elementów. Robot w całości wykonany został z plexi o grubości $5mm$. Wszystkie elementy cięte były z wykorzystaniem plotera laserowego. Elementy dystansowe wykonane zostały z rurek aluminiowych. Do napędu nóg robota użyte zostały standardowe serwa modelarskie. Wypróbowano serwa dwóch producentów: TowerPro MG946 oraz Alturn USA ADS–966HMG oraz AAS–750MG.

Tanie serwa firmy TowerPro zastosowano w pierwszej wersji robota. Niestety nie działały one zadowalająco, czasami wpadały one w oscylacje, podczas powolnego przemieszczania pracowały skokowo. Zdecydowano się na pozbycie się wewnętrznego sterownika i zaprojektowanie go od nowa. Zaimplementowano darmowe (open–software) rozwiązanie OpenServo. Udało się poprawić płynność przemieszczania, jednak pojawiły się problemy z dokładnością pozycjonowania. Każda próba poprawy dokładności uzuskiwania zadanej pozycji kończyła się oscylacjami. Dodatkowo pojawiły się problemy z niezawodnością. Enkodery połączone w tak zwany "daisy chain", okazały się być bardzo zawodne. Próbę wykorzystania własnego sterownika serw z powodu przedstawionych problemów zaniechano.

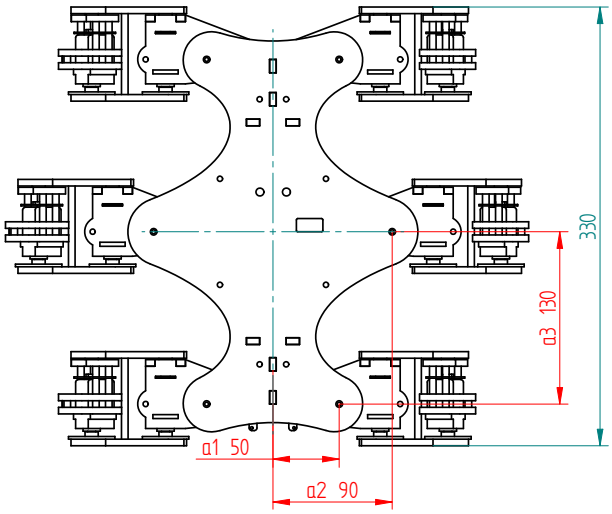
W nowej wersji robota wykorzystano serwa firmy Altrurn USA. Model ADS–966HMG to serwo cyfrowe o dużym momencie 20kg/cm, natomiast AAS–750MG to serwo analogowe o momencie 10kg/cm. W miejscach najmocniej obciążonych tj. segment środkowy nogi zastosowano silniejsze cyfrowe serwa. Serwa tej firmy podczas powolnego przemieszczania działają płynnie, brak jest oscylacji. Specjalnie dla nich zaprojektowano opisany w pracy sterownik lokalny nogi. Niestety i tutaj nie obyło się bez problemów. Producent nie wyposażył serw w wewnętrzne zabezpieczenie termiczne i prądowe. Serwo działające przez dłuższy czas pod pełnym obciążeniem przegrzewa się, i ulega silnik ulega uszkodzeniu. W ten sposób uszkodzone zostały dwa serwa ADS–966HMG. Obecnie zastąpiono je serwami TowerPro MG946. Być może dalszym punktem rozwoju robota powinno być wyposażenie serw w odpowiednie zabezpieczenia, lub też wymiana ich na serwa solidniejszego producenta.

Największym problemem jaki pojawił się podczas realizacji projektu była masa robota. Postawione wymaganie pokonywania przeszkód i nierówności, wymaga zastosowania

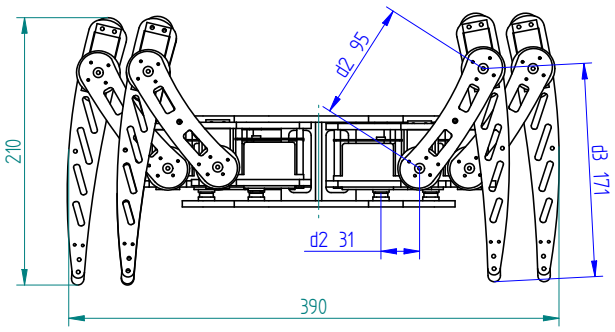
nóg o znacznej długości, co spowodowało znaczący wzrost masy robota. Zastosowane zostały sprężyny (rysunek 3.3), które pozwalają uzupełnić niedostatek mocy silników i w niektórych konfiguracjach całkowicie dźwigają ciężar robota, ograniczając tym samym pobór prądu. Sprężyna jest najmocniej naciągnięta, gdy segment drógi jest podnisiony w górę. Im wyżej, tym siła z jaką sprężyna odziaływuje jest większa. Gdy robot podpiera się na nodze będącej w takiej konfiguracji, cały ciężar dzwigany jest przez sprężynę. z dróiej strony gdy noga jest maksymalnie wyprostowana, siła oddziaływania sprężyny jest najmniejsza. Zastosowanie sprężyn ma również pewne minusy. Gdy noga nie jest nogą podporową i jest uniesiona w górę to silnik musi zmagać się z naciągnięciem sprężyn. Pobiera przy tym znacznie większy prąd, niż w przypadku nogi bez sprężyny. W ogólnym rozrachunku korzystniej jest jednak zastosować sprężyny. Gdy robot korzysta ze sprężyn średnie zużycie prądu to około 4A, natomiast gdy sprężyn brak około 6A. Oczywiście wartości te są uśrednione i w dużej mieże zależą konfiguracji.

Aby monitorować oparcie nogi robota o podłoże, zaprojektowano stopę wyposarzoną w przełącznik (rysunek 3.4).

Poglądowe wymiary robota przedstawiono na rysunku 3.5. Ze względu na złożoność konstrukcji oraz trudne do zwymiarowania na kartce krzywizny, zdecydowano się na dostarczeniu pełnej dokumentacji mechanicznej w formie elektronicznej w postaci plików projektu programu Solid Edge. Dostarczono również pliki matryc do cięcia elementów z plexi na ploterze laserowym.



(a) widok z góry



(b) widok z przodu

Rysunek 3.5 Poglądowe wymiary robota

3.2 Sterownik główny robota

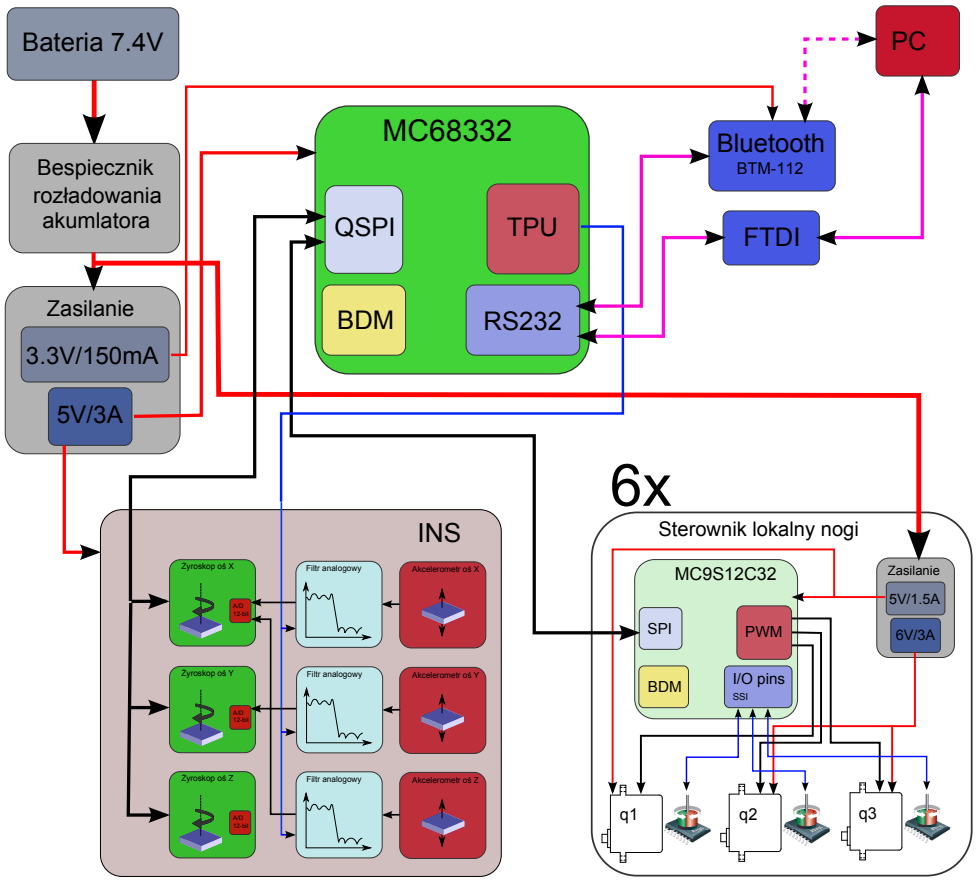
Sterownik główny robota odpowiedzialny jest przede wszystkim za wykonywanie obliczeń związanych z algorytmem kroczenia, komunikację ze sterowniami lokalnymi nóg robota, oraz za komunikację z komputerem PC. Dodatkowo w sterowniku lokalnym dokonywane są obliczenia związane z obróbką i filtracją danych z modułu INS, a następnie na ich podstawie określana jest pozycja robota. W rozdziale tym przedstawiona jest budowa sterownika oraz standard komunikacji sterownika z komputerem PC.

3.2.1 Budowa

Sterownik główny oparto o mikroprocesor MC68332 firmy *Freescale*. Głównymi zaletami jednostki jest 32 bitowa architektura procesora oraz kolejkowany interfejs synchroniczny urządzeń wewnętrznych (QSPI). Interfejs QSPI wykorzystany został do komunikacji z modułem INS oraz ze sterownikami lokalnymi nóg robota. W skład sterownika głównego robota wchodzą moduły odpowiedzialne za komunikację z komputerem PC: moduł FTDI UB232R oraz moduł Bluetooth BTM–112. Na płycie sterownika głównego umieszczono również zabezpieczenie przed nadmiernym rozładowaniem baterii.

Na rysunku 3.6 przedstawiono schemat blokowy sterownika głównego robota. Schemat ideowy, spis elementów oraz ich rozmieszczenie na płytce PCB zamieszczono w dodatkach B, C. Opis złączy i wyprowadzeń zamieszczono w dodatku D.

Kolejne elementy składowe płyty głównej opisano w dalszej części rozdziału.



Rysunek 3.6 Schemat blokowy sterownika głównego robota

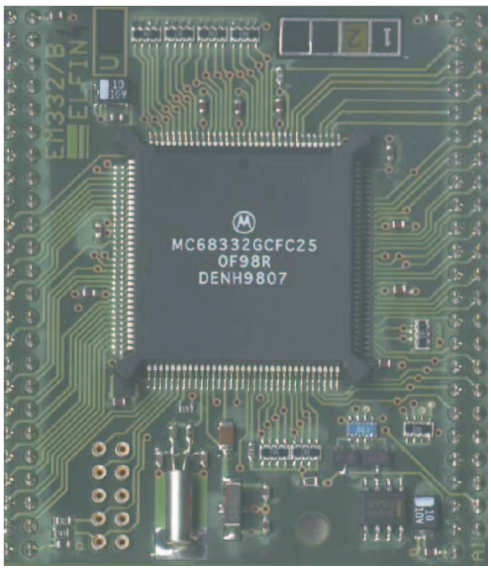
Jednostka główna MC68332

Jednostka główna to 32 bitowy mikroprocesor MC68332 firmy *Freescale*. Wyposażony jest między innymi w programowalny timer TPU (ang. Time Processor Unit), interfejs QSPI (ang. Queued Serial Peripheral Interface) oraz interfejs SCI (ang. Serial Communication Interface). Wykorzystano moduł EM332 wyposażony w taki właśnie mikroprocesor.

Główne cechy modułu z mikrokontrolerem MC68332 [20]:

- 32 bitowa architektura (CPU32),
- pamięć RAM 265kB,
- pamięć FLASH 128kB,
- moduł transmisji szeregowych QSM (ang. Queued Serial Module),
- programowalny timer TPU (ang. Time Processor Unit),
- kolejkowany interfejs synchroniczny urządzeń wewnętrznych QSPI (ang. Queued Serial Peripheral Interface),
- moduł integracji systemu SIM (ang. System Integration Module).

Dokładny opis modułu dostępny jest w pozycji [20], natomiast opis mikroprocesora w pozycji [5]. Rysunek 3.7 przedstawia zdjęcie modułu EM332.



Rysunek 3.7 Moduł EM332 z mikrokontrolerem MC68332 [20]

Moduł FTDI UB232R

UB232R jest to moduł firmy FTDI (ang. Future Technology Devices International). Jest to konwerter interfejsu RS232 na USB. Pozwala na bezpośrednie podłączenie do komputera PC przez port USB, gdzie konwerter widoczny jest jako wirtualny port szeregowy. Po stronie mikroprocesora zachowany jest standard napieć +3.3V lub +5V, co bardzo ułatwia implementację modułu. Interfejs RS232 jest znacznie łatwiejszy do oprogramowania niż USB, zarówno po stronie mikroprocesora, jak i komputera PC. Zastosowanie konwertera

pozwala na korzystanie z interfejsu RS232 na współczesnych komputerach, gdzie port szeregowy jest już rzadko spotykany.

Główne cechy modułu UB232R [7]:

- bazuje na chipie FT232RQ,
- obsługa całego standardu USB w chipie,
- osobna kolejka FIFO dla danych wchodzących i wychodzących,
- prędkość przesyłu danych od 300 baud do 3Mbaud,
- kompatybilny z standardem USB 1.1 oraz 2.0,
- zasilanie z portu USB.

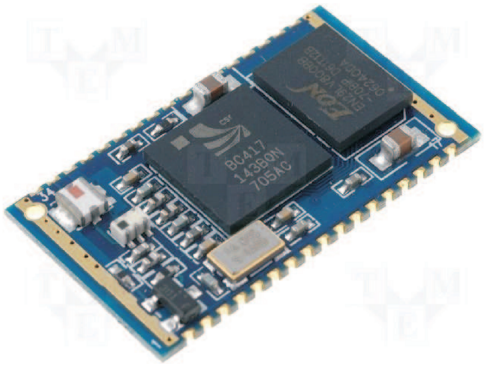
Moduł Bluetooth BTM–112

Komunikacja bezprzewodowa z robotem odbywa się poprzez Bluetooth. Wybór sposobu komunikacji z robotem pomiędzy przewodowym, a bezprzewodowym możliwy jest poprzez odpowiednie ustawienie zwor na płycie głównej. Pozycje zwor opisane są w sekcji D. Transmisja przez Bluetooth w standardzie RS232 możliwa jest poprzez zastosowanie przezroczystego dla transmisji szeregowej, modułu BTM–112. Moduł posiada kilka interfejsów przesyłu danych. W robocie wykorzystano jedynie interfejs UART. Dokładny opis pozostałych interfejsów jak i samego modułu BTM–112 dostępny jest w pozycji [17].

Główne cechy modułu BTM–112 [17]:

- Bluetooth 2.0,
- zasięg 10m,
- interfejs UART, USB, PCM,
- obsługa protokołu SPP,
- napięcie zasilania 3.3V.

Na rysunku 3.8 przedstawiono zdjęcie modułu BTM–112.



Rysunek 3.8 Moduł BTM112 [17]

3.2.2 Komunikacja

Komunikacja odbywa się przez sprzętowy interfejs komunikacji asynchronicznej SCI. Transmisja danych odbywa się z prędkością 115200 bodów, natomiast parametry transmisji są następujące: jeden bit stopu, brak parzystości.

W sterowniku robota znajdują się tablice rejestrów, 8 i 16 bitowych. Komunikacja odbywa się w trybie master/slave, gdzie komputer PC pracuje jako master, natomiast sterownik robota jako slave. Sterownik wysyła informację dopiero gdy zostanie o coś zapytany. Sterowanie robotem odbywa się poprzez zapis wartości do odpowiednich rejestrów. Odczyt informacji o stanie robota sprowadza się do odczytu danych z rejestrów.

Koncepcja taka pozwala znacznie uprościć protokół komunikacji, wystarczy obsłużyć funkcje zapisu i odczytu z rejestrów. Dodatkowo, aby przyspieszyć wymianę kluczowych informacji wprowadzona została funkcja odczytu blokowego, o stałej długości bloku danych. W funkcji tej najpierw wysyłane jest i pierwszych rejestrów 8 bitowych, a następnie j pierwszych rejestrów 16 bitowych, najbardziej znaczący bajt pierwszy. Wartości i oraz j są stałe, wynoszą odpowiednio 8, 24. Funkcje obsługiwane przez zaimplementowany protokół komunikacji przedstawiono w tabeli 3.1.

Rejestry sterownika głównego robota przedstawiono w dodatku A.4.

Tabela. 3.1 Kody funkcji obsługiwane przez sterownik główny

Kod funkcji	Opis	
0xfe	Odczyt rejestru 8 bitowego	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfe |ADR	0xfe |ADR |VALUE |0xff |0xfe
0xfd	Odczyt rejestru 16 bitowego	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfd |ADR	0xfd |ADR |VALUE_H |VALUE_L |0xfd
0xfc	Zapis do rejestru 8 bitowego	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfc |ADR |VALUE	0xfc |0xff |0xff |0xff |0xfc
0xfc	Zapis do rejestru 16 bitowego	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfb |ADR |VALUE_H |VALUE_L	0xfb |0xff |0xff |0xff |0xfb
0xfa	Odczyt blokowy	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfa |0xff	0xfa |ADR |R8_0 |... |R8_i |R16_0_H |R16_0_L |... |R16_j_H |R16_j_L |0xfa

3.3 Sterownik lokalny nogi robota

Głównym zadaniem spoczywającym na lokalnym sterowniku nogi, jest utrzymanie zadanej pozycji. Sterownik określa pozycję stopy na podstawie pomiaru z enkoderów magnetycznych. Zastosowano trzy enkodery, po jednym na każdy przegób obrotowy. Wychylenie kątowe określane jest z rozdzielczością $0,0879^\circ = 360^\circ/4096$. Zastosowano rególator proporcjonalny który utrzymuje zadaną pozycję kątową. Sygnałem sterujacym jest sygnał PWM o zmiennym wypełnieniu, poprzez który zadaje się pozycję serwa.

Sterownik lokalny każdej z nóg oparty jest o mikrokontroler MC9S12C32. Powodem dla którego sterownik zrealizowano w osobnym mikrokontrolerze jest brak wystarczającej liczby kanałów PWM w jednostce głównej robota. Mikrokontroler MC68332 posiada 16 kanałów, a potrzebnych jest 18.

Dodatkową zaletą takiego rozwiązania jest możliwość przeniesienia części obliczeń do sterownika lokalnego odciążając w ten sposób jednostkę główną robota. Sterownie nogą robota odbywa się poprzez wysyłanie współrzędnych stopy w układzie lokalnym nogi. Wszelkie przeliczenia związane z kinematyką prostą i odwrotną realizowane są w sterowniku lokalnym nogi.

Jednym z istotnych zadań jakie sterownik musi realizować to przemieszczenie stopy z punktu do punktu. Zaimplementowany został generator trajektorii przedstawiony w pozycji [21]. Generator spełnia cztery ograniczenia: dwa na prędkość początkową i końcową oraz dwa dotyczące położenia. Od generatora wymaga się, aby prędkość na początku i końcu była równa zero, natomiast maksymalną wartość przyjmował w połowie ruchu. Uzyskuje się w ten sposób gładkie, łagodne przemieszczenie nogi.

Równania opisujące współrzędną z końca nogi w zależności do czasu ruchu

$$z(t) = z_0 + 3\frac{\Delta z}{t_f^2}t^2 - 2\frac{\Delta z}{t_f^3}t^3. \tag{3.1}$$

Δz jest całkowitą zmianą współrzędnej z, z_0 jest wartością początkową w chwili $t = 0$. Współrzędna z w chwili końcowej $t = t_f$ jest równa $z_0 + \Delta z$.

Obliczając pochodną względem czasu przedstawionej funkcji otrzymujemy funkcję opisującą prędkość

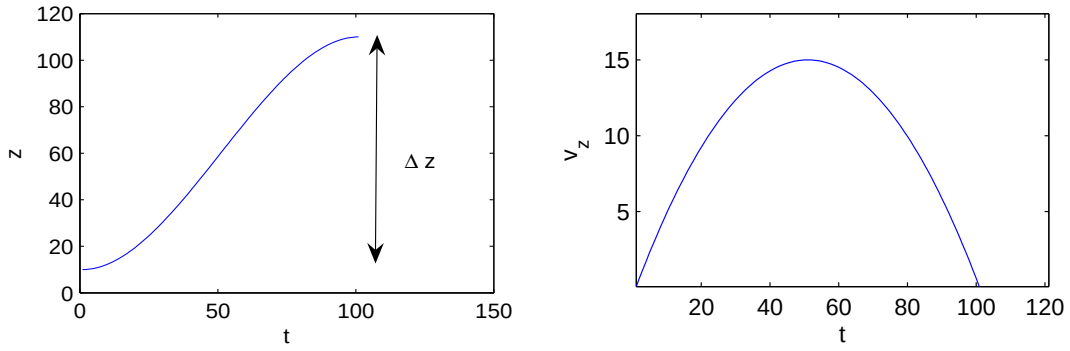
$$\dot{z}(t) = 6\frac{\Delta z}{t_f^2}t - 6\frac{\Delta z}{t_f^3}t^2. \tag{3.2}$$

Rysunek 3.9 przedstawia położenie i prędkość dla współrzędnej z wyznaczane na podstawie przedstawionych równań.

Identyczne zależności (3.1) zachodzą dla współrzędnych x, y

$$x(t) = x_0 + 3\frac{\Delta x}{t_f^2}t^2 - 2\frac{\Delta x}{t_f^3}t^3, \tag{3.3}$$

$$y(t) = y_0 + 3\frac{\Delta y}{t_f^2}t^2 - 2\frac{\Delta y}{t_f^3}t^3. \tag{3.4}$$

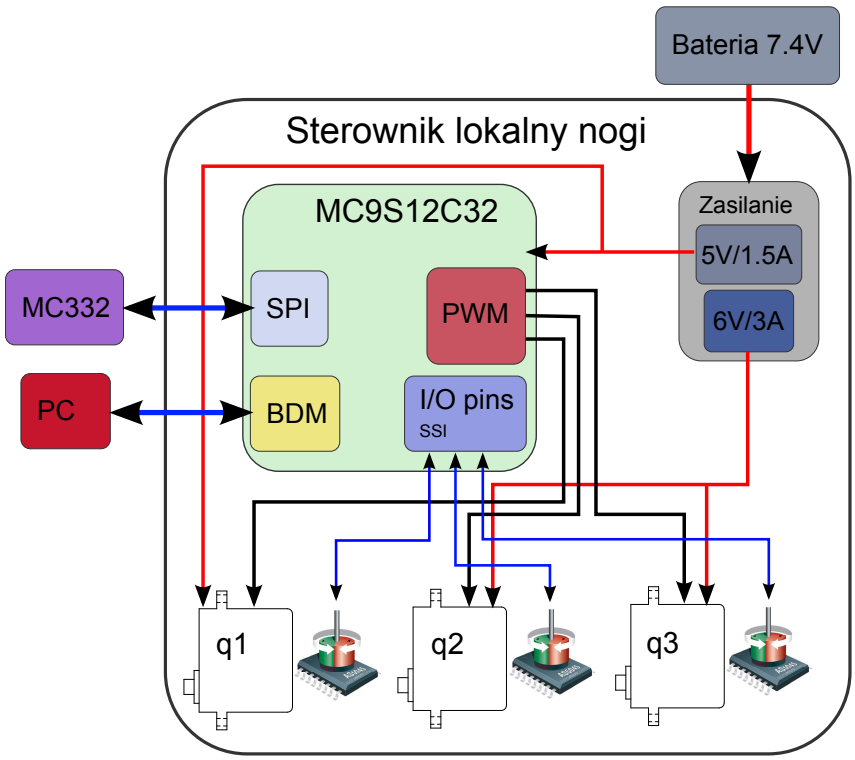


(a) położenie (b) prędkość

Rysunek 3.9 Lokalny generator trajektorii ruchu stopy ($z_0 = 10$, $\Delta z = 100$, $tf = 100$)

3.3.1 Budowa

Sterownik pojedynczej nogi składa się z mikrokontrolera MC9S12C32, trzech enkoderów kontrolujących pozycję każdego z segmentów nogi oraz układu zasilania o napięciach 5V/1.5A, 6V/3A. Sterownik wykonany jest w technologii SMD, tak aby mógł się zmieścić bezpośrednio w nodze. Schemat blokowy sterownika przedstawiono na rysunku 3.10. Schemat ideowy, spis elementów oraz ich rozmieszczenie na płytce PCB zamieszczono w dodatkach B, C. Opis złączy i wyprowadzeń zamieszczono w dodatku D.



Rysunek 3.10 Schemat blokowy sterownika lokalnego nogi

Mikrokontroler MC9S12C32

MC9S12C32 jest to mikrokontroler firmy *Freescale* z rodziny HC12. Wyposażony jest w wydajną 16-bitową jednostkę centralną (CPU16). Mikrokontroler jest bardzo wygodny podczas rozwoju oprogramowania ze względu na wbudowany debuger sprzętowy z interfejsem BDM (ang. Background Debug Module).

Główne cechy MC9S12C32:

- 16-bitowa jednostka centralna,
- 32kB wbudowanej pamięci FLASH,
- 4kB pamięci RAM,
- 8-kanałowy, 10-bitowy przetwornik A/C,
- dwa liczniki 16-bitowe,
- modulator PWM (siedem kanałów 8-bitowych lub trzy 16-bitowe),
- dwa asynchroniczne interfejsy szeregowe (SCI),

- synchroniczny interfejs szeregowy (SPI),
- interfejs zgodny z magistralą I^2C,
- częstotliwość pracy do 50Mhz (magistrala 25Mhz),
- zasilanie 3.3V–5.5V.

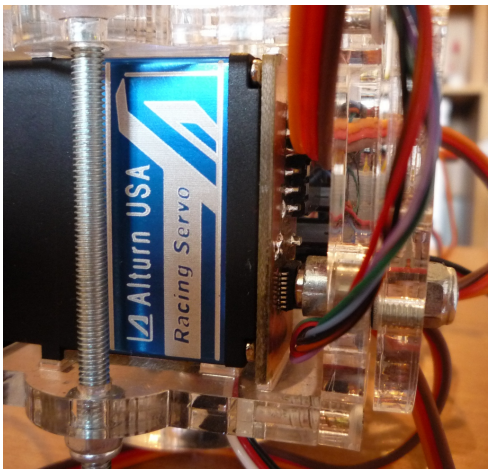
Enkoder magnetyczny AS5045

W robocie wykorzystano enkodery magnetyczne do pomiaru wychylenia kątowego segmentów nóg. Za każdym z serw umieszczona jest płytka PCB z enkoderem, natomiast magnes wciśnięty jest w oś obracającą się w łożysku (rysunek 3.11). Zastosowano enkodery firmy *Austriamicrosystems* AS5045.

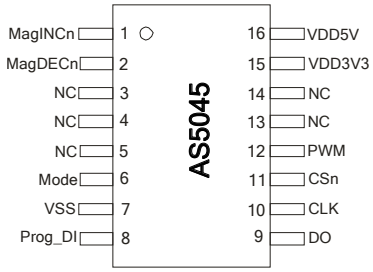
Główne cechy enkodera AS5045 [3]:

- rozdzielczość 12 – bitów na obrót,
- synchroniczny szeregowy interfejs komunikacji (SSI),
- wyjście PWM,
- napięcie zasilania 3.3V lub 5V,
- obudowa *SSOP–16*.

Do komunikacji z enkoderem wykorzystany został interfejs szeregowy SSI. W mikrokontrolerze interfejs ten zrealizowano programowo, ponieważ wbudowany sprzętowy interfejs SPI jest wykorzystany do komunikacji z jednostką centralną robota.



Rysunek 3.11 Enkoder umieszczony tuż za serwem



Rysunek 3.12 Opis wyprowadzeń enkodera AS5045 [3]

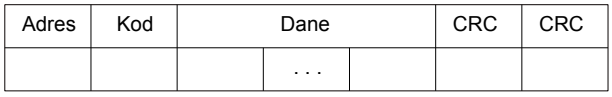
Tabela. 3.2 Opis wyprowadzeń enkodera AS5045 [3]

Pin	Nazwa	Opis
1	MagINCn	przechodzi w stan niski gdy zmniejsza się odległość pomiędzy magnesem a enkoderem
2	MagDECn	przechodzi w stan niski gdy zwiększa się odległość pomiędzy magnesem a enkoderem
3,4,5,13,14	NC	nie użyte
6	MODE	wybór trybu pracy (stan niski – tryb low, stan wysoki – tryb fast)
7	VSS	masa zasilania
8	Prog_DI	wejście danych w trybie programowania
9	DO	wyjście danych magistrali SSI
10	CLK	linie taktująca magistrali SSI
11	CSn	Chip – Select
12	PWM	wyjście sygnału PWM
15	VDD3V3	wyjście regulatora 3.3V
16	VDD5V	plus zasilania 3.3V lub 5V

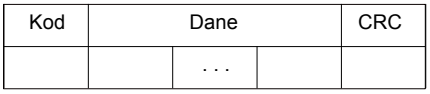
3.3.2 Komunikacja

Do komunikacji sterownika głównego z sterownikiem lokalnym nogi wykorzystano sprzętowy interfejs SPI. Wbudowany w jednostkę główną MC68332 moduł QSPI, pozwala na bardzo wygodną i efektywną komunikację. Każdy z sterowników lokalnych podpięty jest pod własny *Chip–Select*. Jednostka główna chcąc nawiązać komunikację z daną nogą wybiera dany *Chip–Select*. Moduł QSPI, umożliwia wysyłanie danych w kolejce o długości od 1 do 16 słów. Rozkaz wysyłany jest w jako pojedyncza kolejka, której długość zależy od aktualnie wysyłanego rozkazu.

Zastosowany protokół komunikacji wzorowany jest na standardzie Modbus. Porównując ramkę danych z ramką danych protokołu Modbus RTU (rysunki 3.13, 3.14), widać ich podobieństwo. W zaimplementowanym protokole funkcję bajtu adresu pełni *Chip–Select*, skrócona jest również suma CRC (zastosowano sumę 8 – bitową zamiast 16 – bitowej). Protokoły różnią się interpretowanymi kodami funkcji, Modbus posiada bardzo rozbudowaną liczbę funkcji, w zastosowanym protokole liczbę funkcji ograniczono do minimum. Pełna specyfikacja protokołu Modbus dostępna jest w pozycji [14].



Rysunek 3.13 Ramka komunikacji w protokole Modbus RTU



Rysunek 3.14 Ramka komunikacji

Kompletne zestawienie kodów funkcji odsługiwanych przez sterownik lokalny nogi robota przedstawiono w tabeli 3.3. Zaimplementowano podstawowe funkcje odczytu i zapisu

do tablicy rejestów. Dodatkowo w celu przyspieszenia komunikacji wprowadzono funkcję zadawania nowej pozycji. Sterownik nogi w odpowiedzi na funkcję zadania pozycji, zwraca aktualną pozycję stopy obliczoną z równań kinematyki.

Bajty oznaczone jako XX oznaczają dowolną wartość. W przypadku sterownika nogi w miejsce to wstawiana jest wartość rejestru statusowego.

Sterownik nogi po otrzymaniu wiadomości sprawdza jej poprawność poprzez porównanie obliczonej sumy CRC8 z tą otrzymaną w wiadomości. Gdy sumy CRC8 zgadzają się wiadomość uznana jest za poprawną, w przeciwnym przypadku jest ignorowana.

Sterownik nogi jako ostatni bajt odpowiedzi zwraca obliczoną sumę CRC8. w ten sposób sterownik główny robota ma informację zwrotną o poprawności wysłania wiadomości. Gdy sumy CRC8 zgadzają się wiadomość została wysłana poprawnie. W przypadku niepowodzenia sterownik główny powtarza wysyłanie wiadomości. Wiadomość wysyłana jest do skutku. Unikna się w ten sposób sytuacji, gdy jedna z nóg nie odpowiada (stoi w miejscu), natomiast pozostałe kontują ruch. Może to prowadzić do uszkodzenia silników.

Tablica rejestrów z opisami dostępna jest w dodatku A.3.

Bajt `TYPE` w funkcji zadawania pozycji określa rodzaj ruchu w jakim ma się przemieszczać noga:

- `SMOOTH` (wartość 0x00) ruch gładki do zadanej pozycji w zadanym czasie T. Trajektoria stopy wyznaczana jest według równań przedstawionych w rozdziale 3.3.
- `NOW` (wartość 0x01) natychmiastowe osiągnięcie zadanej pozycji. Wartość czasu T jest ignorowana. Funkcjonalność przydatna gdy rolę generatora trajektorii przejmuje sterownik główny robota.

Tabela. 3.3 Kody funkcji odsługiwane przez lokalny sterownik nogi robota

Kod funkcji	Opis	
0xfa	Odczyt rejestru statusowego nogi	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfa |XX |CRC8 |XX	0xfa |REG_VALUE |XX |CRC8
0xfb	Awaryjne zatrzymanie nogi	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfb |CRC8 |XX	0xfb |XX |CRC8
0xfc	Zapis do rejestru	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfc |ADR |REG_VALUE |CRC8 |XX	0xfc |XX |XX |XX |CRC8
0xfd	Odczyt rejestru	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfd |ADR |XX |CRC8 |XX	0xfd |XX |REG_VALUE |XX |CRC8
0xfe	Zadanie nowej pozycji	
	Ramka rozkazu:	Odpowiedz sterownika:
	0xfe |X_H |X_L |Y_H | Y_L |Z_H |Z_L |T_H | T_L |TYPE |CRC8 |XX	0xfc |X_H |X_L |Y_H | Y_L |Z_H |Z_L |XX |XX |XX |XX |CRC8

3.4 Moduł nawigacyjny INS

Na potrzeby pracy dyplomowej wykonany został moduł nawigacji inercyjnej INS (ang. Internal Nawigation System). Jest to układ trzech płytek PCB: jednej bazowej i dwóch przymocowanych prostopadle. W skład modułu wchodzą trzy żyroskopy, trzy akcelerometry oraz filtry analogowe. Moduł pozwala na pomiar prędkości kątowych oraz przyspieszeń w osiach: X,Y,Z. Układ stanowi oddzielny moduł komunikujący się z układem nadrzędnym poprzez magistralę SPI.

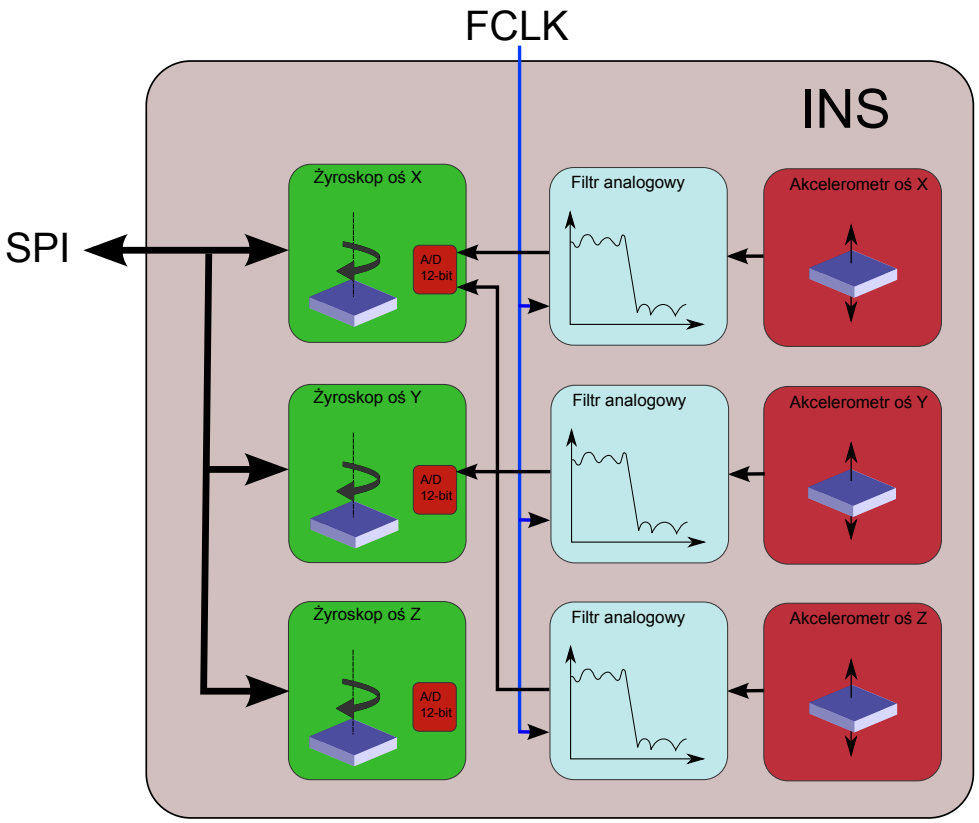
Cechy modułu:

- zasilanie: +5V,
- wejścia/wyjścia w standardzie napięć TTL,
- komunikacja przez magistralę SPI,
- trzy żyroskopy ADIS16100 osobno dla osi X,Y,Z,
- trzy jednoosiowe akcelerometry analogowe MMA1260EG, każdy z własnym filtrem analogowym MAX7400.

Tabela. 3.4 Opis wyprowadzeń modułu INS

Pin	Nazwa	Opis
1	MOSI	wejście danych magistrali SPI
2	CS_X	Chip – Select żyroskopu w osi X
3	MISO	wyjście danych magistrali SPI
4	CS_Y	Chip – Select żyroskopu w osi Y
5	SCK	linia taktująca magistrali SPI
6	VSS	masa zasilania
7	FCLK	sygnał taktujący filtry analogowe
8	VDD	zasilanie +5V
9	DIO1	nie użyte
10	CS_Z	Chip–Select żyroskopu w osi z

Schemat blokowy modułu INS przedstawiono na rysunku 3.15. Zdecydowano się na wykorzystanie akcelerometrów jednoosiowych ze względu na ich dużą czułość. Żyroskopy i akcelerometry badające oś X i Y umieszczone są na osobnych płytkach przymocowanych prostopadle do płytki bazowej. Takie rozmieszczenie elementów okazało się dużym problemem, konieczne były kilkukrotne pomiary i poprawki. Do odczytu informacji z akcelerometrów wykorzystano przetworniki A/C wbudowane w żyroskopy ADIS16100. Cały moduł wykonany został w technologi SMD.

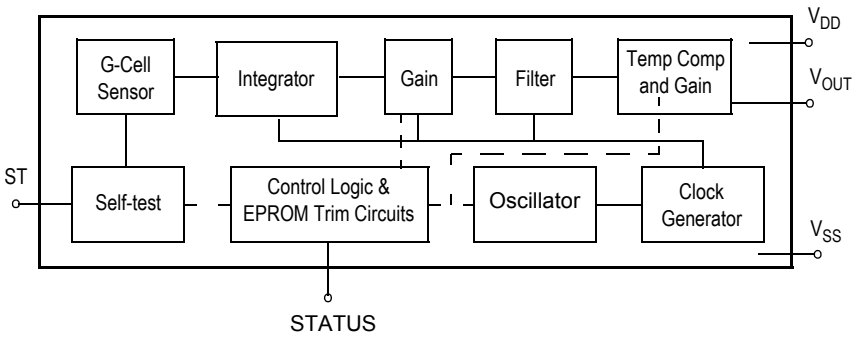


Rysunek 3.15 Diagram blokowy modułu INS

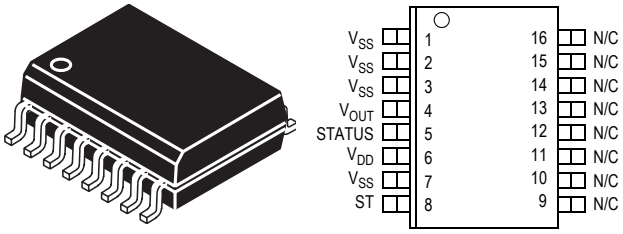
3.4.1 Akcelerometr MMA1260EG

Jednoosiowy akcelerometr firmy Freescale. Podstawowe cechy [6]:

- zakres pracy ±1.5g
- czułość 1200 [mV/g]
- zasilanie +5V
- pobór prądu 2.2mA
- wbudowany filtr Bessela 2–go rzędu (częstotliwość odcięcia 50Hz, wzmocnienie -3dB)



Rysunek 3.16 Diagram blokowy akcelerometru MMA1260EG [6]



Rysunek 3.17 Opis wyprowadzeń akcelerometru MMA1260EG [6]

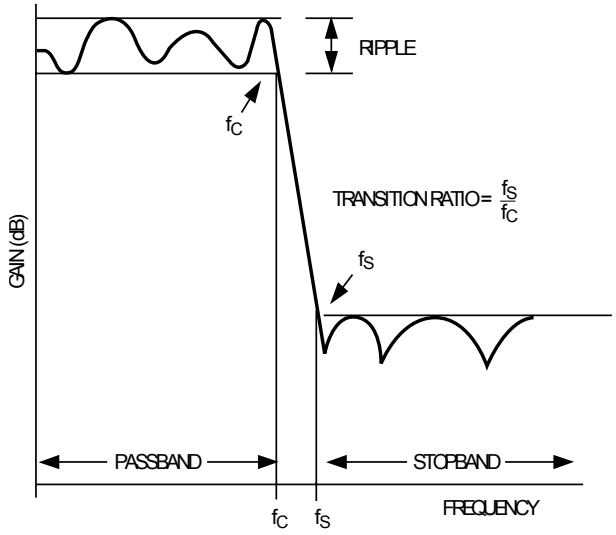
Tabela. 3.5 Opis wyprowadzen akcelerometru MMA1260EG [6]

Pin	Nazwa	Opis
1 – 3, 7	VSS	masa zasilania
4	VOUT	napięcie wyjściowe akcelerometru
5	STATUS	wyjście logiczne do wskazania awarii
6	VDD	zasilanie +5V
8	ST	wejscie logiczne do inicjalizacji wewnętrznego testu
9 – 16	N_C	nie użyte

3.4.2 Filtr analogowy MAX7400

Analogowy filtr dolnoprzepustowy (eliptyczny 8–go rzędu) firmy MAXIM zastosowano do filtrowania wysokich częstotliwości w sygnale z akcelerometrów, aby wyeliminować zjawisko aliasingu. Wbudowany w akcelerometr filtr Bessela 2–go rzędu ma zbyt małe tłumienie, co nie pozwala na wyeliminowanie niepożądanych częstotliwości. Filtr MAX7400 charakteryzuje się małym stosunkiem przejścia sygnału z pasma przenoszenia do pasma tłumienia (stroma charakterystyka amplitudowa, rysunek 3.19) oraz dużym tłumieniem powyżej częstotliwości odcięcia. Podstawowe cechy [13]:

- eliptyczny filtr dolnoprzepustowy 8–go rzędu
- tłumienie -82dB
- regulowana częstotliwość odcięcia w zakresie 1Hz – 10Khz
- zasilanie +5V
- regulacja częstotliwości odcięcia z oscylatora zewnętrznego lub z wewnętrznego poprzez odpowiedni dobór kondensatora



Rysunek 3.18 Charakterystyka amplitudowa filtru MAX7400 [13]

Filtr pozwala na ustawienie częstotliwości odcięcia f_c poprzez doprowadzenie zewnętrznego sygnału taktującego F_{CLK} o wypełnieniu 40%–60% do wejścia CLK lub przez dobór kondensatora C_{OSC} i podpięcie go między pin CLK a GND.

W przypadku wykorzystania zewnętrznego oscylatora częstotliwość odcięcia ustala się według wzoru:

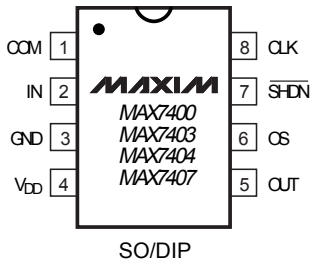
$$f_c = \frac{F_{CLK}}{100}, \tag{3.5}$$

natomiast w przypadku korzystania z oscylatora wewnętrznego częstotliwość F_{OSC} ustala się według wzoru:

$$f_{OSC}[kHz] = \frac{K * 10^3}{C_{OSC}[pF]}, \tag{3.6}$$

$$f_c = \frac{F_{OSC}}{100}, \tag{3.7}$$

gdzie K zależy od zastosowanego filtru (dla MAX7400, K = 38) [13].



Rysunek 3.19 Opis wyprowadzeń filtru MAX7400 [13]

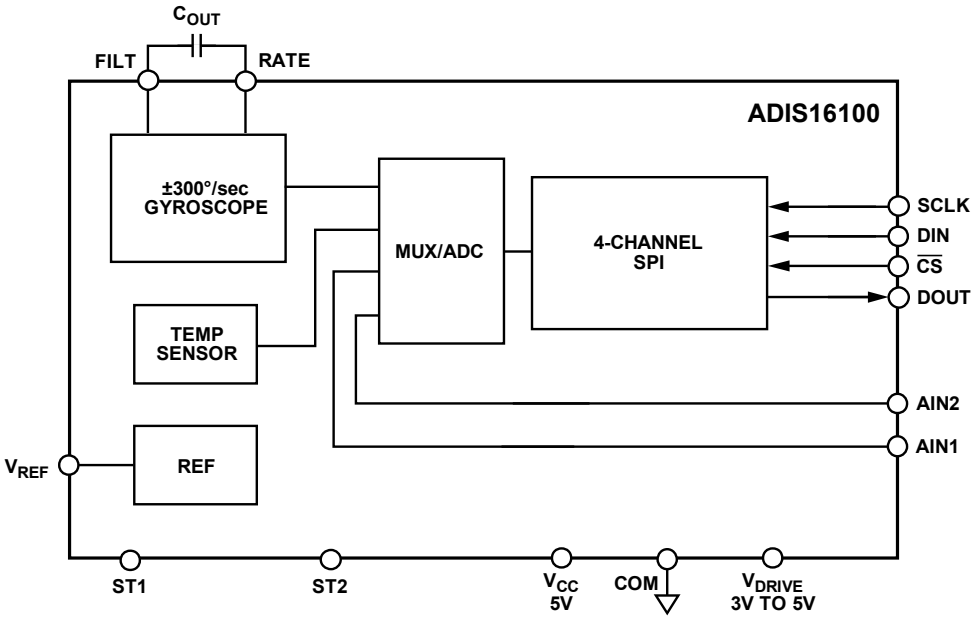
Tabela. 3.6 Opis wyprowadzeń filtru MAX7400 [13]

Pin	Nazwa	Opis
1	COM	połączony z masa przez kondensator 1μF
2	IN	wejście filtru
3	GND	masa zasilania
4	VDD	zasilanie +5V
5	OUT	wyjście filtru
6	OS	$offset$ – wejście konfiguracyjne
7	SHDN	wejście wprowadzające w stan uśpienia
8	CLK	wejście zegara taktującego

3.4.3 Żyroskop ADIS16100

Żyroskop ADIS16100 jest to 1–osiowy żyroskop firmy Analog Devices, jego podstawowe cechy to [2]:

- czułość ±300°/s
- wbudowany 12–bitowy przetwornik A/C
- pomiar prędkości kątowej w osi Z
- komunikacja przez magistralę SPI (16–bitów)
- wbudowany czujnik temperatury
- 2 dodatkowe wejścia analogowe do przetwornika A/C
- zasilanie +5V
- wbudowane źródło napięcia odniesienia +2.5V



Rysunek 3.20 Diagram blokowy żyroskopu ADIS16100 [2]

Żyroskop posiada wewnętrzny filtr dolnoprzepustowy, którego częstotliwość odcięcia ustala się poprzez dobór kondensatora i rezystora podłączonego między wyprowadzeniami RATE i FILT.

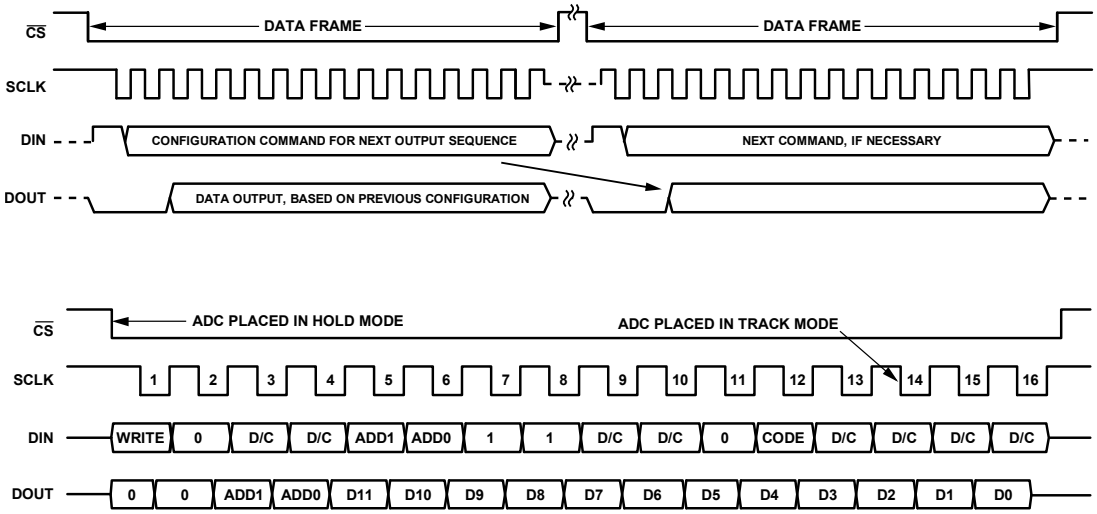
$$f_{OUT} = \frac{1}{2\pi R_{OUT}(C_{OUT} + 0.022\mu F)} \tag{3.8}$$

gdzie R_{OUT} jest to rezystancja wewnętrzna równa 180kΩ. Można dokonać zmiany tej rezystancji poprzez dołączenie rezystora R_{EXT}:

$$R_{OUT} = \frac{180k\Omega \cdot R_{EXT}}{180k\Omega + R_{EXT}} \tag{3.9}$$

Domyślnie częstotliwość odcięcia filtru ustawiona jest na 40Hz.

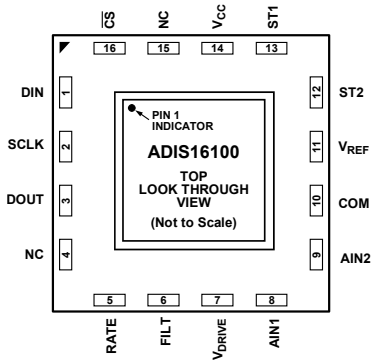
Na rysunku 3.21 przedstawiona została komunikacja z układem przez magistralę SPI w trybie 16–bitowym. W każdym cyklu komunikacji wysyłany jest rozkaz oraz odbierany wynik działania dla rozkazu z poprzedniego cyklu. Opisy poszczególnych bitów składających się na rozkaz przedstawione zostały w tabeli 3.4.3. Wbudowany w żyroskop przetwornik A/C sterowany jest przez linie CS oraz SCLK. Stan niski na linii CS wprowadza przetwornik w tryb *hold* i uruchamia próbkowanie. Po 14 takcie na linii SCLK przetwornik powraca do trybu *track*, jednak do zakończenia konwersji danych potrzebne jest 16 taktów. Po poprawnej konwersji danych wynik dostępny jest w następnym cyklu transmisji.



Rysunek 3.21 Komunikacja z żyroskopem ADIS16100 przez magistralę SPI [2]

Tabela. 3.7 Opis bitów rozkazu dla żyroskopu ADIS16100 [2]

Bit	Nazwa	Opis
15	WRITE	1 – zapis do rejestrów kontrolnych, 0 – brak zmian
14,5	0	wymagany stan niski do normalnej pracy
13–12,7 –6, 3–0	DC	bez znaczenia
11–10	ADD1,ADD0	źródło przetwornika: 00 – żyroskop, 01 – czujnik temperatury, 10 – wejście analogowe zewnętrzne 1, 11 – wejście analogowe zewnętrzne 2
9 –8	1	wymagany stan niski do normalnej pracy
4	CODE	sposób formatowania danych: 0 – uzupełnienie do 2, 1 – kod binarny bez znaku



Rysunek 3.22 Opis wyprowadzeń żyroskopu ADIS16100 [2]

Tabela. 3.8 Opis wyprowadzeń żyroskopu ADIS16100 [2]

Pin	Nazwa	Opis
1	DIN	wejście danych magistrali SPI
2	SCLK	wejście zegara magistrali SPI
3	DOUT	wyjście danych magistrali SPI
5	RATE	wyjście analogowe żyroskopu
6	FILT	wejście na zewnętrzny kondensator filtrujący
7	VDRIVE	zasilanie interfejsu cyfrowego
8	AIN1	zewnętrzne wejście przetwornika A/C
9	AIN2	zewnętrzne wejście przetwornika A/C
10	COM	masa zasilania
11	VREF	napięcie referencyjne 2,5V
12	ST2	test wewnętrzny
13	ST1	test wewnętrzny
14	VCC	zasilanie toru analogowego
15	NC	nie podłączone
16	CS	wejście wyboru urządzenia magistrali SPI

Rozdział 4

Oprogramowanie sterujące, symulator

Robot Mrówa sterowany jest za pomocą komputera PC. W tym celu stworzono specjalne oprogramowanie. W rozdziale tym przedstawione są możliwości aplikacji oraz niektóre szczegóły techniczne. W dalszej części rozdziału przedstawiona jest koncepcja zaimplementowanego modułu komunikacji.

4.1 Opis aplikacji

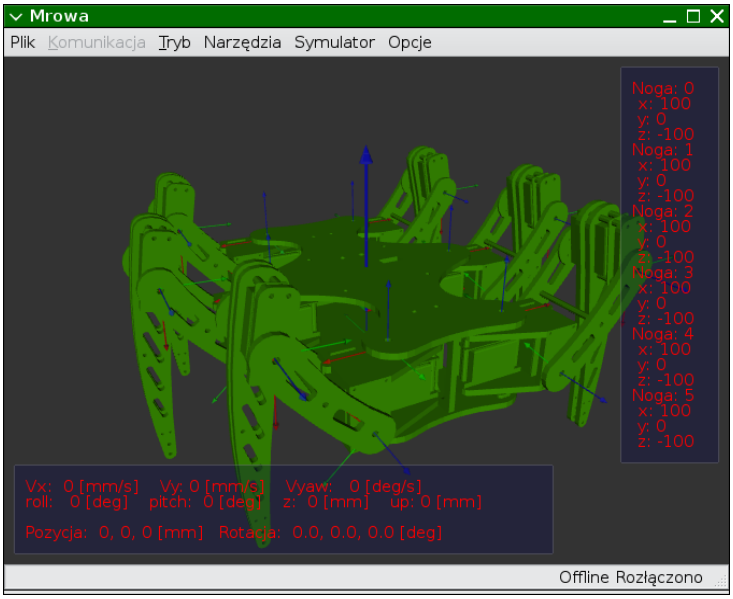
Program napisany jest w języku C++ dla systemu Linux. Wykorzystane zostały biblioteki STL, OpenGL, Boost, QT4. Bibliotekę QT4 użyto do stoworzenia okienkowego interfejsu aplikacji. Trójwymiarowa wizualizacja robota oraz mapy wysokości napisana jest z wykorzystaniem biblioteki OpenGL. Wielowątkowość do języka C++, użytą w module komunikacji z robotem wprowadzono z pomocą biblioteki Boost. Wirtualny model robota uzyskano poprzez wyeksportowanie modelu z programu Solid Edge do środowiska Blender, a stamtąd bezpośrednio do OpenGL–a.

Możliwości

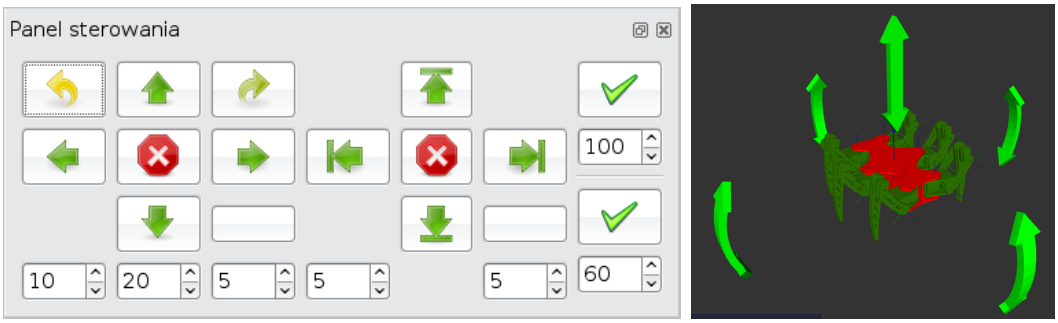
Aplikacja (rysunek 4.1) pozwala w łatwy i przejrzysty sposób kontrolować pracę robota. Prezentowany w głównym oknie aplikacji trójwymiarowy model robota odzwierciedla zachowanie rzeczywistej konstrukcji. Parametry kroczenia, aktualna pozycja korpusu, jego orientacja oraz pozycja nóg robota prezentowane są w oknie aplikacji. Aktualizacja danych następuje cyklicznie co 100ms.

Sterowanie robotem możliwe jest poprzez stworzony specjalnie do tego celu panel. Dostępny jest on w menu narzędzia. Panel (rysunek 4.2.a) pozwala na zadawanie parametrów kroczenia. Możliwe jest zadawanie prędkości liniowej w osi OX, OY oraz prędkości kątowej w osi OZ (lewa część panelu). Przechyły korpusu w osi OX, OY, wysokość podnoszenia nóg oraz wysokość korpusu można zadawać poprzez przyciski z prawej części panelu. Dokładny opis każdego z przycisków dostępny jest po najechaniu na niego kursorem. Możliwe jest awaryjne zatrzymanie robota przez przycisku stopu.

Przechyły boczne oraz wysokość korpusu można zdawać również bezpośrednio z okna wizualizacji. Dwukrotne kliknięcie w korpus robota spowoduje pojawienie się strzałek (rysunek 4.2.b), zaznaczenie której z nich, a następnie przeciągnięcie kursora pozwala na zadawanie parametrów.



Rysunek 4.1 Aplikacja sterująca robotem



(a) (b)

Rysunek 4.2 Panel sterujący

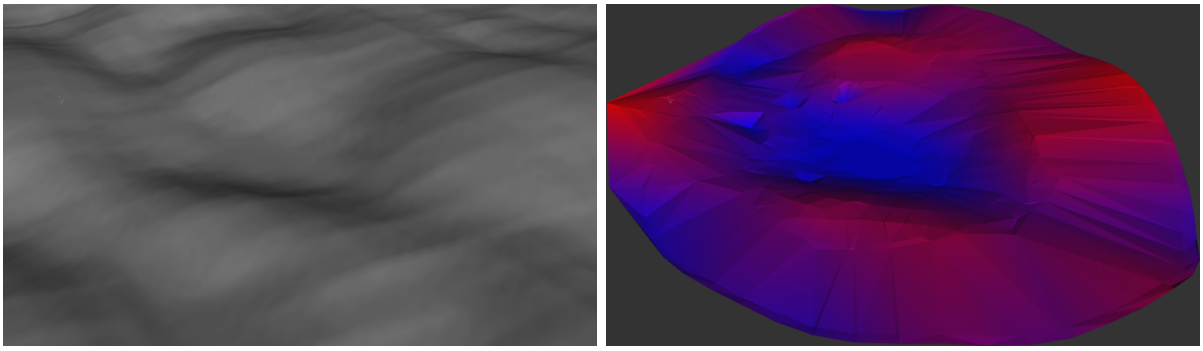
Mapa Wysokości

Program poza sterowaniem robotem ma możliwość wizualizacji trójwymiarowej mapy wysokości terenu. Mapa tworzona jest na podstawie danych uzyskanych z robota. Informacja o pozycji korpusu i pozycji każdej ze stóp robota pozwala uzyskać informację o wysokości terenu. Punkty pomiarowe uzyskane w ten sposób służą do tworzenia mapy wysokości terenu.

Mapa tworzona jest z wykorzystaniem triangulacji Delone 2.10. Punkty pomiarowe łączone są w trójkąty tak, że całość tworzy trójwymiarową siatkę będącą mapą wysokości. To które punkty należy ze sobą połączyć uzyskuje się dzieki triangulacji Delunaja. Istnieje wiele innych metod budowania mapy wysokości, część z nich przedstawiona została we wstępie pracy. Przykład mapy wysokości przedstawiono na rysunku 4.3.

Symulator

Program może pracować w dwóch trybach online i offline. W trybie online komunikuje się z robotem. W trybie offline uruchamiany jest symulator, i cała komunikacja odbywa się



(a) podłoże (b) mapa wysokości

Rysunek 4.3 Mapa wysokości

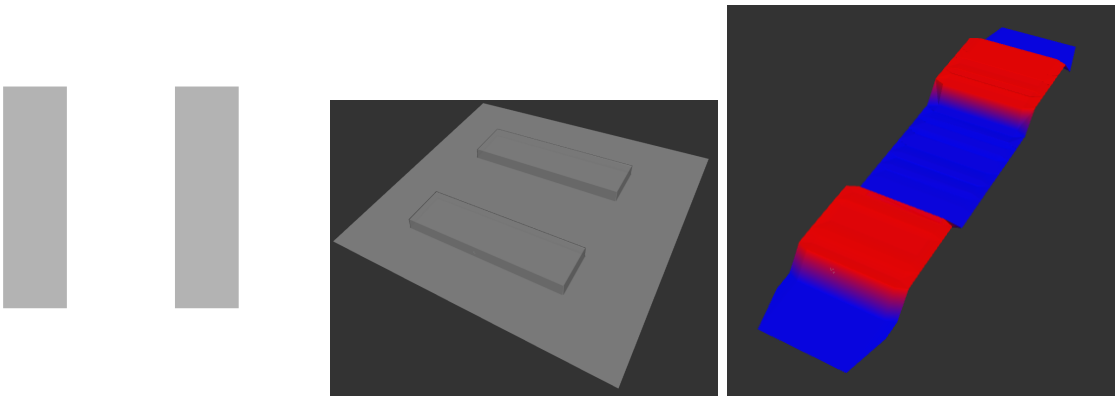
pomiędzy aplikacją a stymulatorem. W trybie tym port szeregowy nie jest wykorzystywany.

Symulator całkowicie zastępuje robota. Stworzono go, aby możliwe było rozwijanie i testowanie algorytmów kroczenia, gdy rzeczywisty robot nie był jeszcze gotowy. Wszystkie zmiany i nowe funkcjonalności robota testowane są wstępnie w stymulatorze, a dopiero potem na rzeczywistej konstrukcji. W ten sposób uniknąć można uszkodzenia konstrukcji. Większość błędów w oprogramowaniu wychwycona zostaje na poziomie testów w stymulatorze.

Symulator w większości napisano języku C, tak aby zminimalizować liczbę zmian podczas przenoszenia kodu do sterownika głównego robota. Sam symulator uruchamiany jest w osobnym wątku dzięki temu symulacja przebiega niezależnie od aplikacji sterującej.

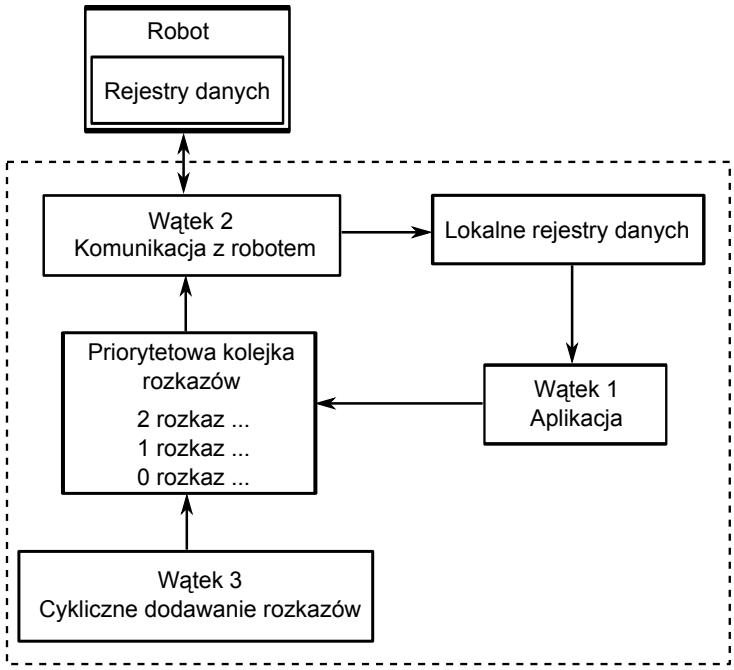
Aby móc symulować ruch robota po nierównej powierzchni, wprowadzono możliwość wczytywania wirtualnej mapy powierzchni. Podłoże wczytywane jest jako pliki graficzne PGM (ang. Portable Gray Map). Format ten służy do zapisu grafiki rastrowej. Pojedynczy piksel obrazu zapisany jest jako wartość w skali szarości. Im wartość większa tym piksel ciemniejszy. Podczas wczytywania tego obrazu do aplikacji o wysokości punkt mapy powierzchni mówi wartość piksela. Im wartość większa (piksel ciemniejszy) tym wysokość większa. Obraz taki można edytować w bardzo wielu programach graficznych. Dzięki temu w bardzo łatwy sposób można tworzyć przeróżne mapy powierzchni.

Przykład takiej mapy powierzchni przedstawiono na rysunku 4.4.



(a) obraz (b) podłoże (c) mapa wysokości

Rysunek 4.4 Mapa wysokości



Rysunek 4.5 Schemat działania modułu komunikacji

4.2 Moduł komunikacji

Wymiana informacji z urządzeniami zewnętrznymi czasami bywa dość kłopotliwa. Komunikacja z urządzeniem zewnętrznym w trybie master/slave, a więc gdy urządzenie odpowiada dopiero gdy zostanie zapytane wymaga zastosowania opóźnień czasowych aby dać czas urządzeniu na wysyłanie wiadomości. W aplikacji okienkowej wiązało by się to z spowolnieniem aplikacji. Chcąc zachować responsywność interfejsu użytkownika podczas wymiany informacji z urządzeniem zewnętrznym, konieczne jest oddzielenie warstwy komunikacji od interfejsu. W tym celu można skorzystać z dostępnego w systemie UNIX mechanizmu przerwań systemowych. Zajmuje się tym biblioteka `signal.h`. Drugą możliwością jest uruchomienie modułu komunikacji w osobnym wątku, i to właśnie to rozwiązanie zostało zastosowane.

Działanie modułu komunikacji najlepiej obrazuje diagram blokowy przedstawiony na rysunku 4.5. Za wymianę informacji z robotem odpowiada wątek 2. Wysyła on rozkazy znajdujące się w priorytetowej kolejce rozkazów. Gdy zostaną wysłane wszystkie rozkazy (kolejka jest pusta), wątek zatrzymuje się i czeka na pojawienie się nowych rozkazów. Odpowiedzi robota na wysyłane rozkazy interpretowane są w wątku 2. Jeśli wysyłany rozkaz wymaga odczytu informacji z rejestrów robota, informacje te zapisywane są to lokalnej kopi rejestrów danych. Lokalne rejestry danych dostępne są dla aplikacji w dowolnym momencie działania programu. Jakikolwiek odczyt informacji z poziomu aplikacji dokonywany jest natychmiastowo z lokalnych rejestrów danych. Dzięki temu aplikacja nie zawiesza się na czas pobierania danych. Aby w lokalnych rejestrach danych znajdowały się aktualne informacje konieczne jest cykliczne odczytywanie tych informacji z robota. Odpowiedzialny jest za to wątek 3, który cyklicznie dodaje rozkazy do kolejki. Posiada on własny zbiór rozkazów, które co określony czas dodaje do priorytetowej kolejki rozkazów. Zbiór rozkazów wysyłanych cyklicznie jest konfigurowany na początku działania programu. Umieszczane są tam rozkazy odpowiedzialne za odczyt informacji o stanie robota.

Rozkazy sterujące robotem takie jak zadanie nowej prędkości, nowej wysokości korpusu lub też awaryjne zatrzymanie robota wysyłane są z poziomu aplikacji poprzez dodanie ich

do priorytetowej kolejki rozkazów. Rozkazy takie oznaczane są wyższym priorytetem niż rozkazy odczytu informacji. Dzięki temu że kolejka rozkazów jest priorytetowa, rozkazy o największej wadze jak np. zatrzymanie robota wysyłane są jako pierwsze.

Podczas implementacji modułu komunikacji posłużono się biblioteką `Boost`, która wprowadza wielowątkowość do języka C++.

Rozdział 5

Badania

Aby móc budować mapę wyskości terenu na podstawie danych o pozycji kończyn robota konieczna jest dokładna znajomość pozycji korpusu robota oraz kątów w przegubach nóg robota. W trakcie kroczenia rzeczywstej konstrukcji po nierównym terenie pojawia się również problem przechyłów bocznych korpusu, które należy kompensować.

Przeprowadzono badania modułu INS, aby określić jego przydatność do określenia pozycji i orientacji korpusu. Wiele uwagi poświęcono na określanie przechyłów bocznych z użyciem danych modułu INS. Oczekuje się, że uzyskana w ten sposób informacja może posłużyć do korekcji postury korpusu.

Przeprowadzono również badania konstrukcji. Sprawdzono dokładność odwzorowania trajektorii przez nogę robota oraz trzymanie stałej pozycji stopy podczas wymuszeń. Podjęto również próbę określenia pozycji i orientacji robota na podstawie odometrii.

Na końcu przedstawiono wyniki badań podczas budowania mapy wysokości z wykorzystaniem robota kroczącego.

Wszystkie badania przeprowadzono przy pomocy opisanego w rodziale 3 robota kroczącego Mrówa. Do rejestracji i oprawienia danych podczas badań modułu INS posłużono się środowiskiem Matlab. Natomiast podczas badań robota oraz przy próbach budowania map wysokości wykorzystano stworzony specjalnie do tego celu program sterujący robotem (opis programu przedstawiono w rozdziale 4).

5.1 Badanie modułu INS

Badaniom poddano moduł INS opisany w rozdziale 3.4. Na czas badań płytę główną wraz z modułem wymontowano z robota. W badaniach posłużono się środowiskiem Matlab, gdzie dokonywano wszelkich obliczeń. Opierając się na badaniach przedstawionych w pracy [22], częstotliwość odcięcia filtru analogowego dla akcelerometrów ustawiono na 10Hz. Dla wszystkich odczytywanych wartości z czujników zastosowano cyfrowy filtr opóźniający

$$y_i = \alpha \cdot x_i + (1 - \alpha) y_{i-1}, \tag{5.1}$$

gdzie wyjście filtru y_i obliczane jest jako tzw. średnia krocząca z aktualnej próbki x_i i poprzednich próbek y_{i-1}. Wagę aktualnej próbki α ustawiono na $\frac{1}{8}$. Aktualizacja filtru odbywała się co 2.44ms, natomiast dane wysyłane były do Matlaba co 9.76ms, co daje częstotliwość próbkowania rzędu 100Hz.

Wpływ filtracji analogowej oraz różnych filtracji cyfrowych na uzyskiwane pomiary przedstawiono w pracy [22]. W prezentowanych badaniach modułu INS skupiono się

na próbie dokładnego określenia przechyłów bocznych robota z wykorzystaniem filtracji Kalmana. Przeprowadzono też badania nad wpływem filtracji Kalmana na określenie orientacji oraz podjęto próby określenia przemieszczenia robota na podstawie pomiaru przyspieszeń.

5.1.1 Wyznaczanie zera i skalowanie akcelerometrów MMA1260EG

Akcelerometr MMA1260EG dostarcza informacji o przyspieszeniu w granicach ±1.5g poprzez wyjście analogowe. Aby informację tę przekazać do mikrokontrolera, konieczne jest zastosowanie przetwornika analogowo cyfrowego. Informacja z przetwornika nie jest wyskalowana w żadnych jednostkach, dostarczona jest ona w najmniej znaczących bitach przetwornika (LSB). Wartość ta wynika z rozdzielczości przetwornika i z napięcia odniesienia.

Badany akcelerometr, według danych podanych przez producenta, posiada czułość 1200mV/g. Dla braku przyspieszenia wskazanie akcelerometru to połowa napięcia zasilania, a więc 2.5V. Korzystając z tych informacji oraz z informacji o rozdzielczości przetwornika i napięciu referencyjnym, można by wyznaczyć stałe skalowania. Metoda ta niestety jest nie do końca poprawna ponieważ dostarczone przez producenta informacje o czułości są typowymi wartościami. Czułość konkretnego akcelerometru może odbiegać od tej wartości według danych producenta w zakresie ±90mV/g [6].

Zdecydowanie lepszą metoda wyznaczenia czułości akcelerometrów jest metoda przedstawiona w pozycji [22]. Korzysta się w niej z faktu, że przyspieszenie ziemskie ma wartość 1g. Dokonuje się kolejno pomiaru wartości dla przyspieszenia +1g i -1g. Następnie z zależności

$$Skala = \frac{2g}{A_{1g} - A_{-1g}} \left[\frac{m}{s^2}\right] \tag{5.2}$$

można wyznaczyć stałą skalowania. Wartości A_{1g} i A_{-1g} to wskazania przetwornika odpowiednio dla przyspieszenia +1g i -1g. Pomiarów tych dokonuje się poprzez umieszczenie czujnika w poziomie przy użyciu poziomnicy, następnie robi się serię pomiarów, delikatnie odchylając czujnik w różne strony. Maksymalna odczytana wartość odpowiada przyspieszeniu 1g. Dokładnie tak samo dokonuje się pomiaru dla -1g — wystarczy obrócić czujnik. Zero przetwornika to wartość środkowa między wskazaniami dla +1g i -1g.

Czułość akcelerometru można wyznaczyć, przeliczając wskazanie przetwornika w LSB na wartość napięcia. Dla zastosowanego przetwornika 12–bitowego, przy napięciu odniesienia 5V 1LSB odpowiada wartości 1.221mV. Wyznaczona w ten sposób czułość badanych akcelerometrów jest następująca:

- czułość akcelerometru w osi OX 1243mV/g
- czułość akcelerometru w osi OY 1259mV/g
- czułość akcelerometru w osi OZ 1245mV/g.

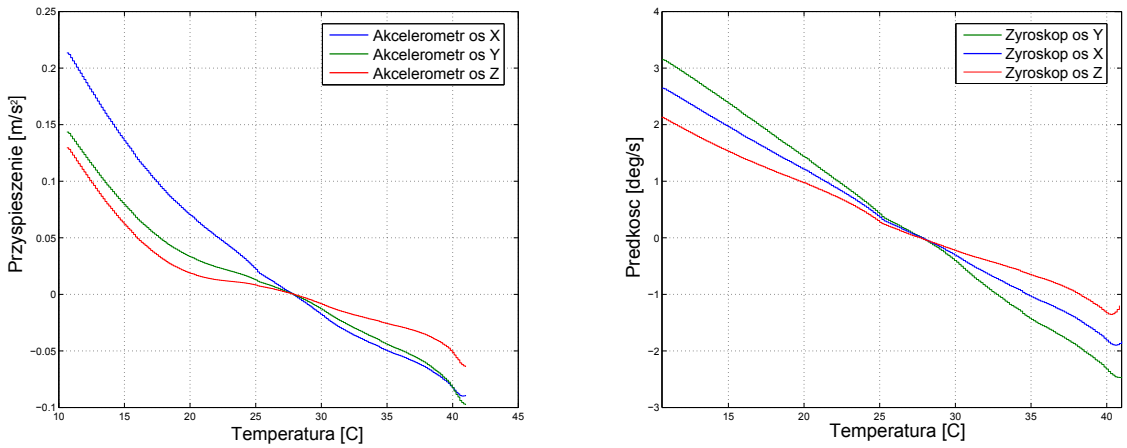
Jak widać, wartości zawierają się w podanych przez producenta granicach rozbieżności $(1200pm90mV/g)$.

5.1.2 Kompensacja temperaturowa akcelerometrów MMA1260EG oraz żyroskopów ADIS16100

Czujniki inercyjne wrażliwe są na zmianę temperatury, dlatego konieczna jest kompensacja temperaturowa. W zależności od jakości można spotkać czujniki wyposażone w wewnętrzne czujniki temperatury bądź też nawet z wbudowaną fabrycznie kompensacją temperaturową. Stosowane w pracy żyroskopy posiadają wbudowane czujniki temperatury. Kompensację temperaturową można zrobić w dwojaki sposób: poprzez wyznaczenie funkcji kompensacji temperaturowej albo poprzez wyznaczenie tablicy wartości korygujących. Wpływ temperatury na czujniki zazwyczaj nie jest liniowy, dlatego wyznaczenie funkcji nie jest łatwe. z drugiej strony, wyznaczenie tablicy wartości korygujących wymaga dużej ilości pamięci mikrokontrolera. Ponieważ zastosowany w pracy mikroprocesor posiada stosunkowo dużo pamięci, zdecydowano się zastosować drugą metodę.

W celu wyznaczenia kompensacji temperaturowej przeprowadzono eksperyment polegający na schłodzeniu modułu ISN w domowej lodówce, a następnie stopniowym powolnym podgrzaniu go do temperatury około 40 stopni przy użyciu suszarki. Cały eksperyment trwał około 20 minut. W trakcie podgrzewania rejestrowano dane z czujników. Sam moduł INS pozostawał w tym czasie w bezruchu. Odczyt temperatury z trzech żyroskopów został uśredniony dla kolejnych próbek, tak aby wyznaczyć jedną wspólną temperaturę modułu. Jak się okazało, dzięki powolnemu ogrzewaniu różnica wskazań temperatury z różnych żyroskopów była znikoma. Udało się równomiernie ogrzewać moduł. Wskazania akcelerometrów i żyroskopów zostały przeskalowane, uśrednione, a następnie przybliżone funkcją wielomianową szóstego stopnia. Na podstawie uzyskanej funkcji wyliczone zostały odrębne tablice korygujące dla każdego z czujników. Przedstawione są one na rysunku 5.1. Należy zaznaczyć, że temperatura została przeliczona na stopnie tylko na potrzeby tych rysunków. W rzeczywistości pozostaje w jednostkach LSB przetwornika, natomiast sama kompensacja odbywa się poprzez pobranie wartości korygujących z tablicy, gdzie indeks odpowiada wartości temperatury wyrażonej w LSB przetwornika. Obliczenia wykonane zostały w środowisku Matlab.

Jak widać, wpływ temperatury jest znaczący, w szczególności dla akcelerometrów gdzie dla takich samych czujników wygląd funkcji korygującej jest różny. W żyroskopach jest to niemal funkcja liniowa. Wyznaczona w ten sposób kompensacja temperaturowa nie jest bez wad. W trakcie eksperymentu dbano o to, aby zmiana temperatury była równomierna dla wszystkich czujników. Natomiast podczas normalnej pracy wcale tak nie musi być. Akcelerometry które nie posiadają własnych czujników polegają na pomiarach temperatury z żyroskopów. Ze względu na różną obudowę, różne materiały z których wykonane są akcelerometry i żyroskopy, temperatura układów może być różna.



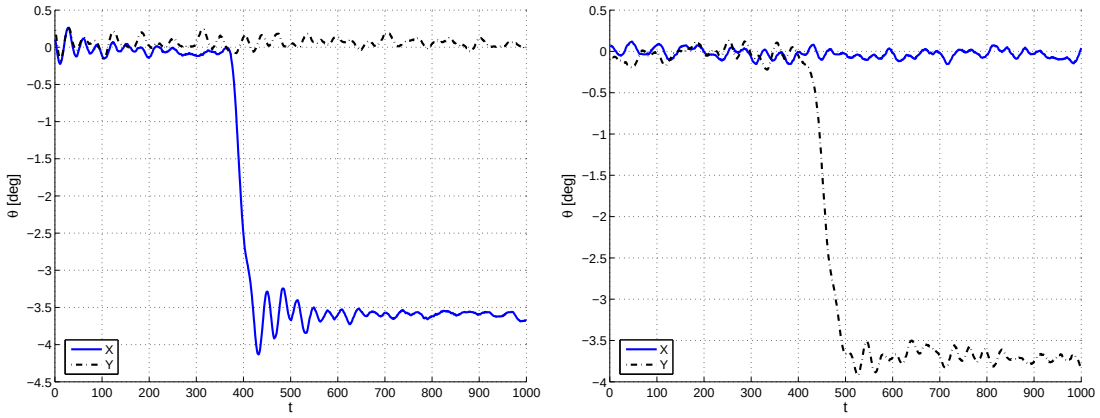
(a) akcelerometry (b) żyroskopy

Rysunek 5.1 Wpływ temperatury na wskazania sensorów

5.1.3 Wyznaczanie odchylenia od pionu na podstawie pomiaru przyspieszeń

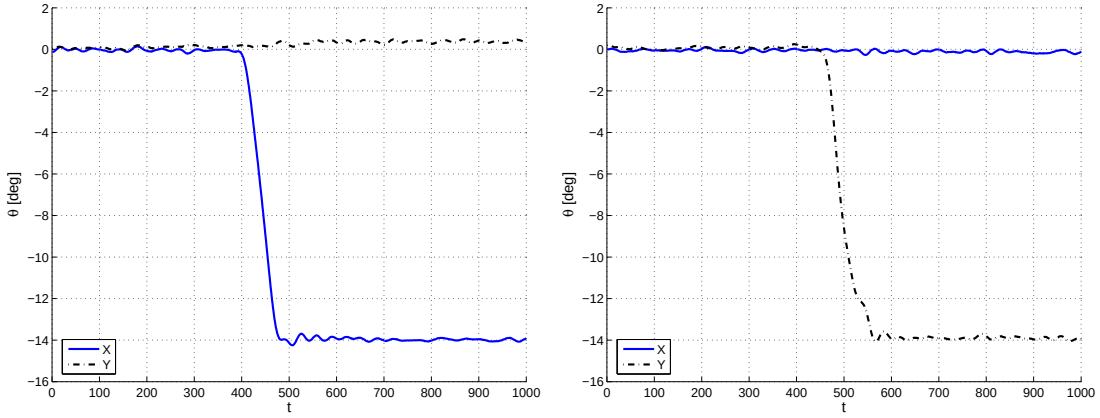
Badanie przeprowadzono dla częstotliwości odcięcia filtru analogowego ustawionej na 10Hz, oraz przy włączonym filtrze opóźniającym. Na czas badań płyta główna została wymontowana z robota. Wymuszenia odchylenia od pionu wykonywane były ręcznie przez podstawianie przedmiotów o znanej wysokości pod jedną z krawędzi płyty głównej, natomiast wartość wymuszenia wyznaczano z prostej zależności $\theta = \arcsin\left(\frac{a}{b}\right)$, gdzie a – wysokość podstawianego przedmiotu, b – odległość punktów podparcia płyty głównej. Odchylenie od pionu wyznaczano na podstawie równań przedstawionych w rozdziale 2.6.

Na rysunkach 5.2, 5.3 przedstawiono pomiar odchylenia od pionu dla wymuszeń $\theta = -3.583°$ i $\theta = -14.47°$ kolejno w osiach OX i OY. Pomiar kąta odchylenia od pionu w każdym z tych przypadków obarczony jest błędem wynikającym z szumu sygnałów z akcelerometrów. Błąd ten oscyluje w granicach $\pm 0.3°$ i, co istotne, nie narasta z czasem. Problem przy wyznaczaniu odchyleń od pionu tą metodą pojawia się, gdy moduł INS jest w ruchu. Rysunek 5.4 przedstawia pomiar odchyleń od pionu w trakcie gwałtownego przemieszczenia modułu w osiach OX, OY po poziomej powierzchni. Ze względu na fakt, że tym przypadku akcelerometry oprócz przyspieszenia ziemskiego rejestrują także to wynikające z przemieszczenia modułu, pomiar odchyleń od pionu jest nieprawidłowy. Jest to największy minus tej metody.



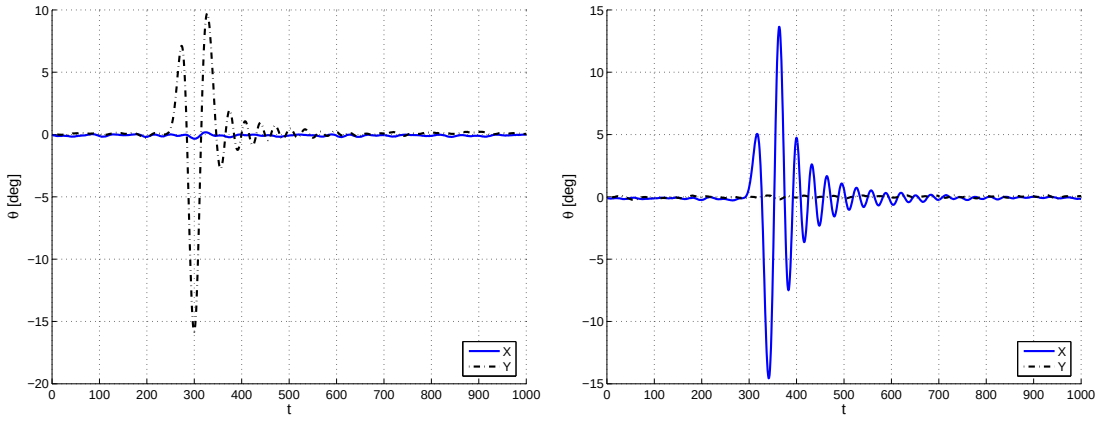
(a) wymuszenie w osi OX (b) wymuszenie w osi OY

Rysunek 5.2 Odchylenie od pionu na podstawie pomiaru przyspieszeń (wymuszenie $\theta = -3.583°$)



(a) wymuszenie w osi OX (b) wymuszenie w osi OY

Rysunek 5.3 Odchylenie od pionu na podstawie pomiaru przyspieszeń (wymuszenie $\theta = -14.47°$)



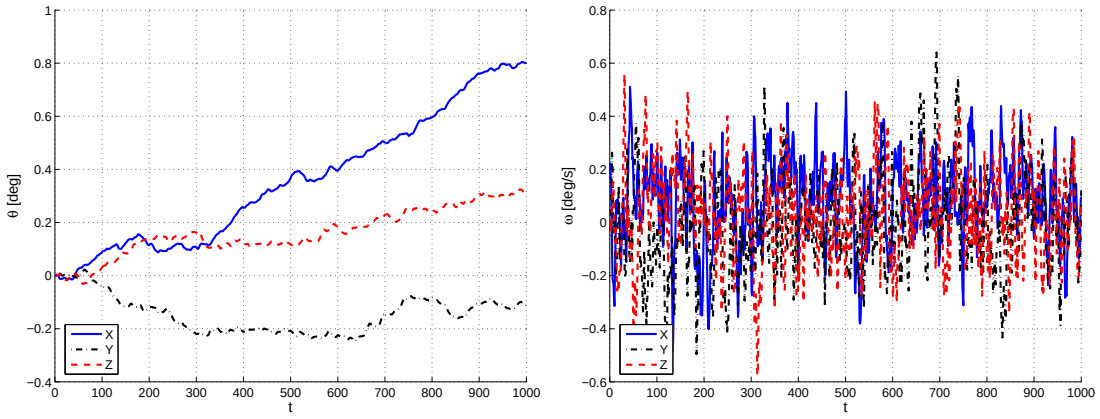
(a) wymuszenie w osi OX (b) wymuszenie w osi OY

Rysunek 5.4 Odchylenie od pionu na podstawie pomiaru przyspieszeń podczas ruchu (szarpnięcia)

5.1.4 Wyznaczanie odchylenia od pionu i orientacji na podstawie pomiaru prędkości kątowych

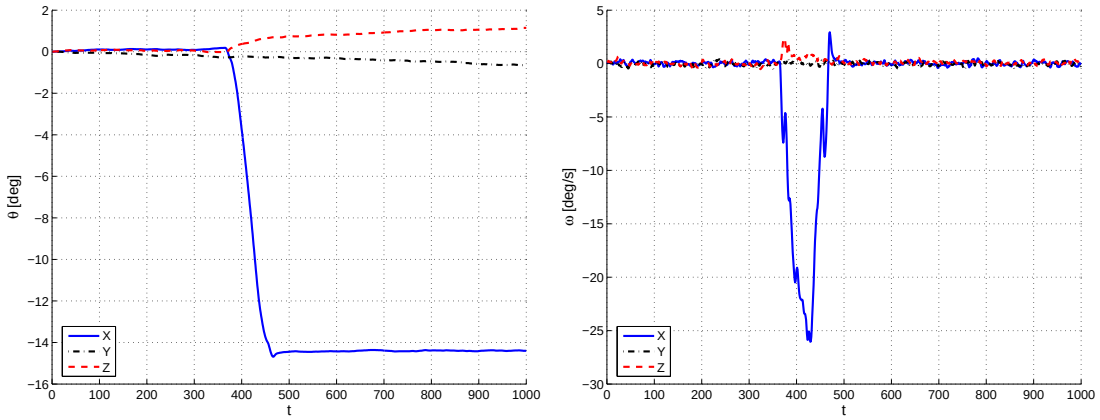
Pomiar prędkości kątowych przez żyroskopy poddany został filtracji cyfrowej. Zastosowano filtr opóźniający. Podobnie jak w poprzednim przypadku, wymuszenie odchylenia od pionu realizowane było ręcznie. Prędkość kątowa próbkowana była co 9.76ms. Sygnał poddano całkowaniu w środowisku Matlab (metoda trapezowa). Prędkości kątowe lokalne rejestrowane przez żyroskopy przeliczono na prędkości rotacji w układzie globalnym na podstawie równań przedstawionych w rozdziale 2.7.

Na rysunku 5.5 przedstawiono pomiar prędkości kątowych oraz rotacji w osiach OX, OY, OZ w sytuacji, gdy moduł INS był w spoczynku. Jak widać, wartości narastają. Pomiary obaczone są błędem systematycznym wynikającym z tzw. dryfu żyroskopu. Zjawisko to w szczególności uwidacznia się, gdy żyroskop jest w bezruchu. Sytuację, gdy moduł poddano wymuszeniom w osiach OX, OY, OZ zobrazowano na rysunkach 5.6, 5.7, 5.8. W momencie wymuszenia wartości wyznaczonych kątów odzwierciadlają rzeczywiste wychylenie, jednak w chwili ustalenia modułu w danej pozycji znowu uwidacznia się zjawisko dryfu żyroskopu. Jest to bardzo niepożądane zjawisko i niemożliwe do wyeliminowania, gdy dysponuje się tylko pomiarem z żyroskopu. Wraz z upływem czasu błąd pomiaru będzie narastać.



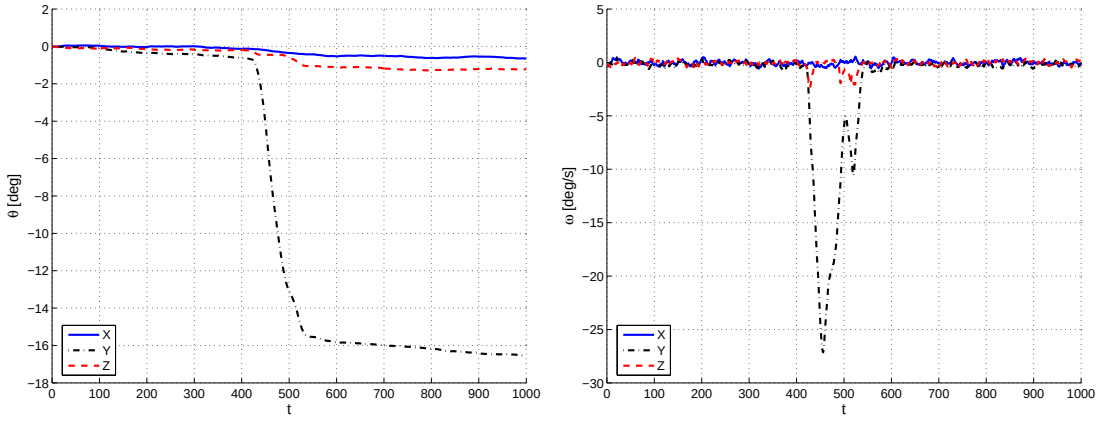
(a) Wychylenie (b) Prędkość kątowa

Rysunek 5.5 Odchylenie od pionu na podstawie pomiaru prędkości kątowych (wymuszenie $\theta = 0°$)



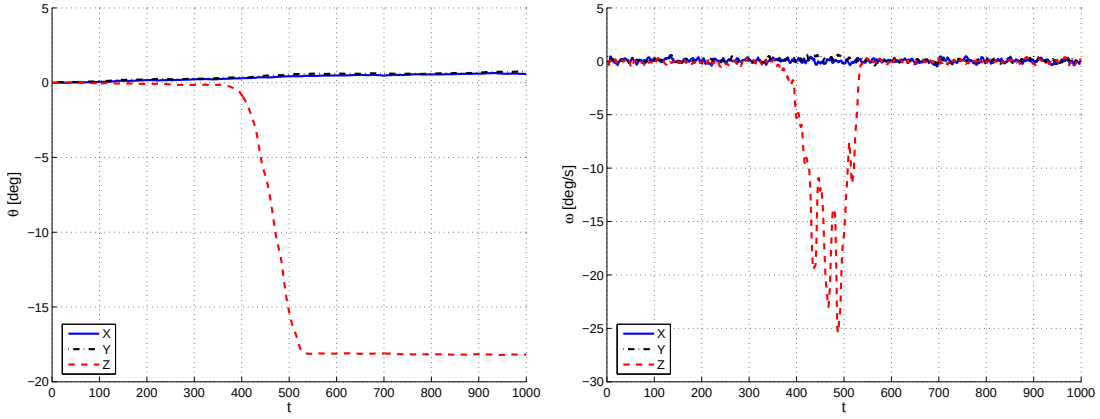
(a) Wychylenie (b) Prędkość kątowa

Rysunek 5.6 Odchylenie od pionu i orientacja na podstawie pomiaru prędkości kątowych (wymuszenie $\theta = -14.47°$ w osi OX)



(a) Wychylenie (b) Prędkość kątowa

Rysunek 5.7 Odchylenie od pionu i orientacja na podstawie pomiaru prędkości kątowych (wymuszenie $\theta = -14.47°$ w osi OY)



(a) Wychylenie (b) Prędkość kątowa

Rysunek 5.8 Odchylenie od pionu i orientacja na podstawie pomiaru prędkości kątowych (wymuszenie $\theta = -18°$ w osi OZ)

5.1.5 Wyznaczanie odchylenia od pionu i orientacji z użyciem filtracji Kalmana

W przedstawionych powyżej badaniach wykazano słabości obydwu z metod wyznaczania odchylenia od pionu i orientacji. W przypadku bazowania tylko na pomiarach z akcelerometrów wyznaczone przechyły były niepoprawne, gdy moduł był w ruchu (zmieniał prędkość). Pomiary z żyroskopów obarczone są błędem systematycznym który narasta z czasem. Błąd ten wynika z tzw. dryfu żyroskopów.

Aby dokonać fuzji obydwu sygnałów, zdecydowano się zastosować filtr Kalmana. Jest on z powodzeniem stosowany w systemach robotycznych, gdzie układy percepcji otoczenia odgrywają istotną rolę. Filtr Kalmana dokonuje optymalnej estymacji stanu na podstawie wszystkich dostępnych pomiarów. Idealnie nadaję się do fuzji różnych źródeł informacji i wyciągnięcia z nich tego, co najlepsze. Wstęp teoretyczny przedstawiono w rozdziale 2.8.

Do wyznaczenia odchylenia od pionu w osi X i Y dostępna jest informacja z żyroskopów, która wcielona zostanie w fazie predykcji jako sterowanie u (równanie 5.5), natomiast informacja uzyskana z akcelerometrów wykorzystana zostanie w fazie korekcji (równanie 5.6). Do wyznaczenia orientacji (rotacja względem osi Z) dostępna jest jedynie informacja z żyroskopu, przez co zostanie ona wprowadzona nie jako sterowanie w fazie predykcji tylko w fazie korekcji. Bardzo dobrym rozwiązaniem byłoby zastosowanie kompasu elektronicznego w połączeniu z żyroskopem, który podobnie jak informacje z akcelerometrów, jest obarczony jedynie szumem a nie błędem systematycznym. Badany moduł INS nie jest w niego wyposażony.

Badania przeprowadzono w środowisku Matlab.

Matematyczny model systemu:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{5.3}$$

oraz równanie pomiaru:

$$z_k = Hx_k + v_k. \tag{5.4}$$

Równania algorytmu filtracji Kalmana:

– faza predykcji

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1}^- + Bu \\ \hat{P}_k^- &= A\hat{P}_{k-1}A^T + Q \end{aligned} \tag{5.5}$$

– faza korekcji

$$\begin{aligned} K_k &= P_k^- H^T (HP_k^- H^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\ P_k &= (I - K_K H)P_k^- \end{aligned} \tag{5.6}$$

Wektor stanu przyjmuje postać

$$x = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \\ \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} \tag{5.7}$$

gdzie θ to rotacja w poszczególnych osiach, natomiast $\dot{\theta}$ to prędkości kątowe. Poszczególne macierze mają postać:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

$$B = \begin{bmatrix} dt & 0 & 0 \\ 0 & dt & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.9}$$

wyjście filtru

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

macierz R to szum pomiarów, wyznacza się ją doświadczalnie.

$$R = \begin{bmatrix} r_{akc_x} & 0 & 0 \\ 0 & r_{akc_y} & 0 \\ 0 & 0 & r_{zyro_z} \end{bmatrix} \tag{5.11}$$

Macierz kowariancji Q ma postać

$$Q = I \cdot q, \tag{5.12}$$

gdzie I to macierz diagonalna 6x6, natomiast współczynnik q wyznacza się doświadczalnie.

Jak już wspominano powyżej, pomiary prędkości kątowych z żyroskopów dla osi OX i OY wykorzystane są jako sterowanie u, natomiast pomiar dla osi OZ wcielony jest w fazie korekcji. Pomiar wychylenia w osiach OX OY na podstawie akcelerometrów wykorzystany został w fazie korekcji. Należy zaznaczyć, że pomiary bezpośrednio dokonane przez żyroskopy przeliczane są na prędkości w układzie globalnym na podstawie równań przedstawionych w rozdziale 2.7. Dla przypomnienia prędkości kątowe w układzie globalnym mają postać:

$$\begin{bmatrix} \phi_{xk} \\ \phi_{yk} \\ \phi_{zk} \end{bmatrix} = \begin{bmatrix} 1 & \sin\theta_{xk-1}\tan\theta_{yk-1} & \cos\theta_{xk-1}\tan\theta_{yk-1} \\ 0 & \cos\theta_{xk-1} & -\sin\theta_{xk-1} \\ 0 & \sin\theta_{xk-1}\sec\theta_{yk-1} & \cos\theta_{xk-1}\sec\theta_{yk-1} \end{bmatrix} \cdot \begin{bmatrix} \omega_{xk} \\ \omega_{yk} \\ \omega_{zk} \end{bmatrix}, \tag{5.13}$$

gdzie ω to faktyczne pomiary w poszczególnych osiach.

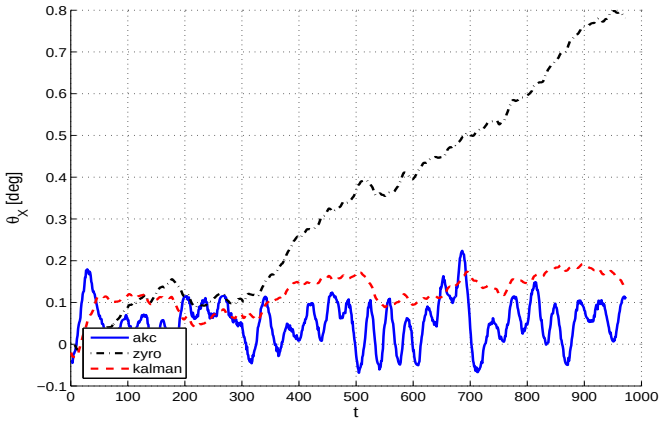
Można teraz zapisać wektor sterowania

$$u_k = \begin{bmatrix} \phi_{xk} \\ \phi_{yk} \\ 0 \end{bmatrix}, \tag{5.14}$$

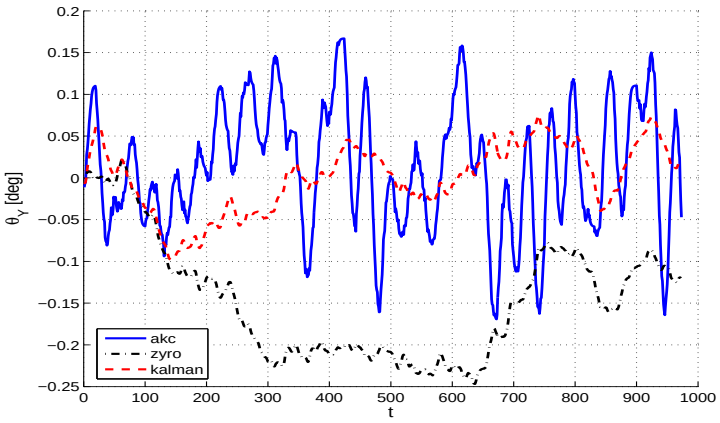
oraz wektor pomiaru

$$z_k = \begin{bmatrix} \theta_{x_{akc}k} \\ \theta_{y_{akc}k} \\ \phi_{zk} \end{bmatrix}, \tag{5.15}$$

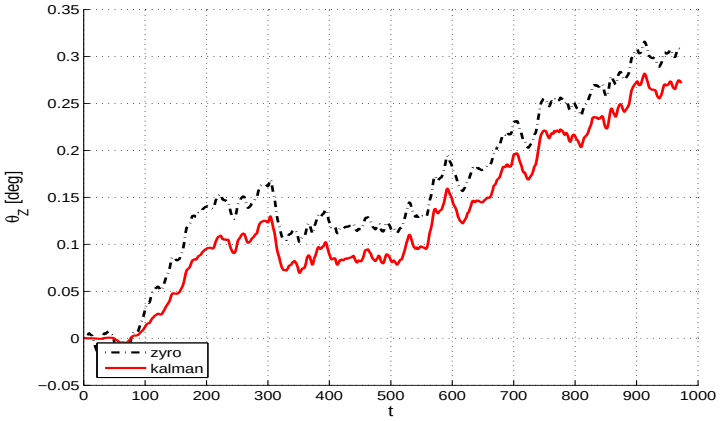
gdzie $\theta_{x_{akc}}$, $\theta_{y_{akc}}$, to odchylenia od pionu wyznaczane na podstawie pomiaru wektora przyspieszenia ziemskiego (rozdział 2.6).



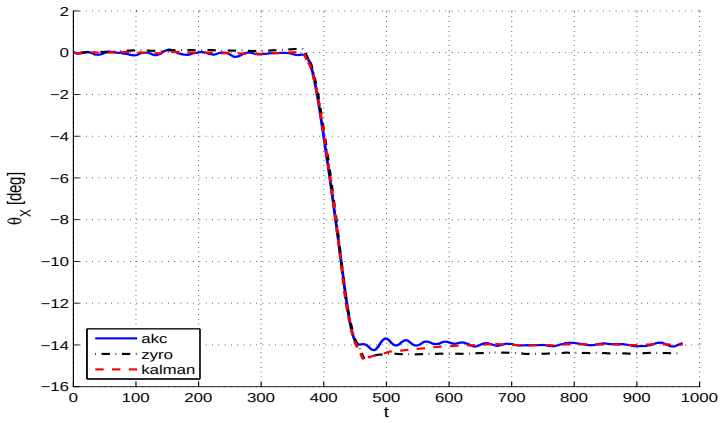
Rysunek 5.9 Rotacja w osi OX na podstawie filtracji Kalmana (wymuszenie $\theta = 0°$)



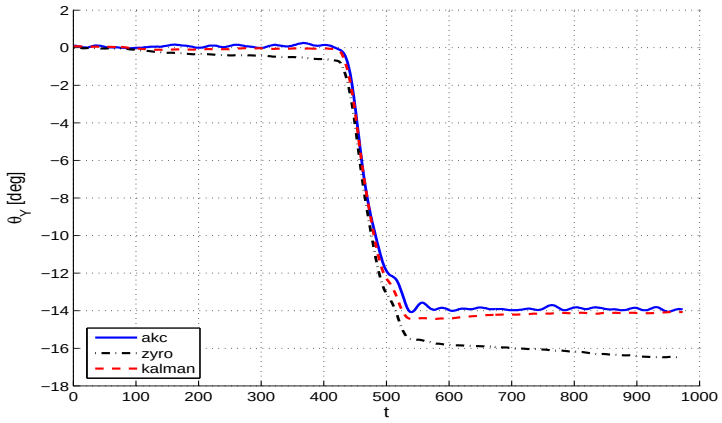
Rysunek 5.10 Rotacja w osi OY na podstawie filtracji Kalmana (wymuszenie $\theta = 0°$)



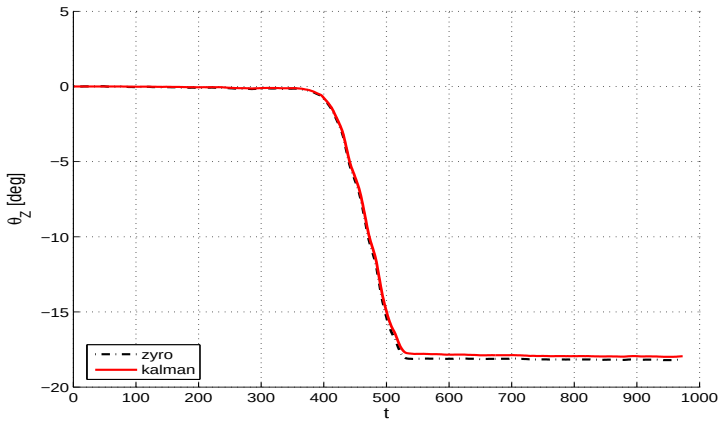
Rysunek 5.11 Rotacja w osi OZ na podstawie filtracji Kalmana (wymuszenie $\theta = 0°$)



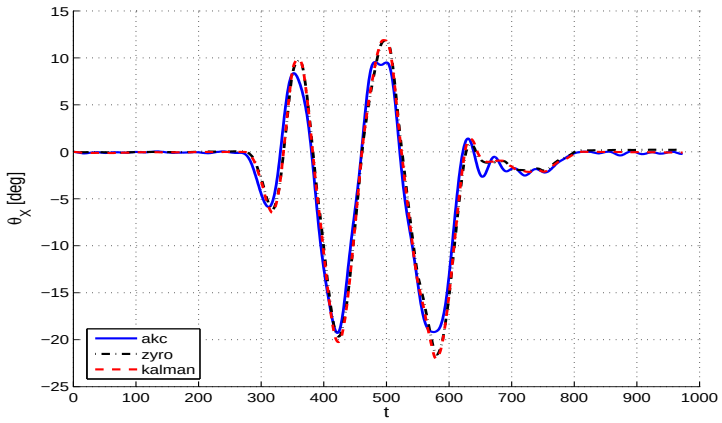
Rysunek 5.12 Rotacja w osi OX na podstawie filtracji Kalmana (wymuszenie $\theta = -14.47°$ w osi OX)



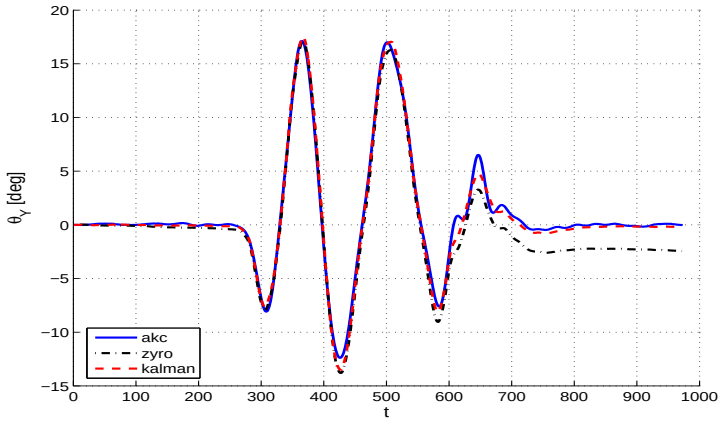
Rysunek 5.13 Rotacja w osi OY na podstawie filtracji Kalmana (wymuszenie $\theta = -14.47°$ w osi OY)



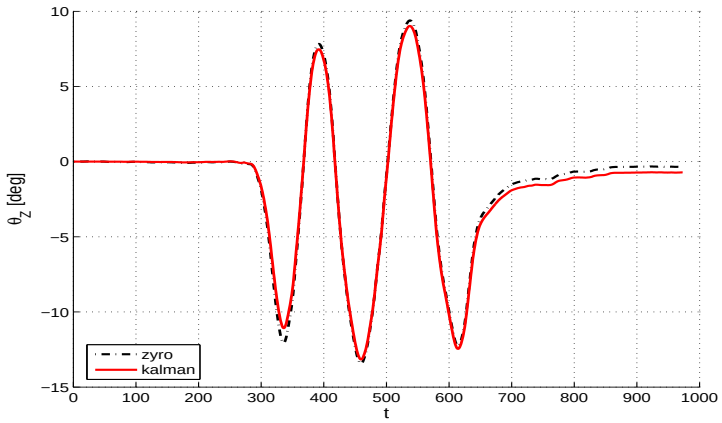
Rysunek 5.14 Rotacja w osi OZ na podstawie filtracji Kalmana (wymuszenie $\theta = -18°$ w osi OZ)



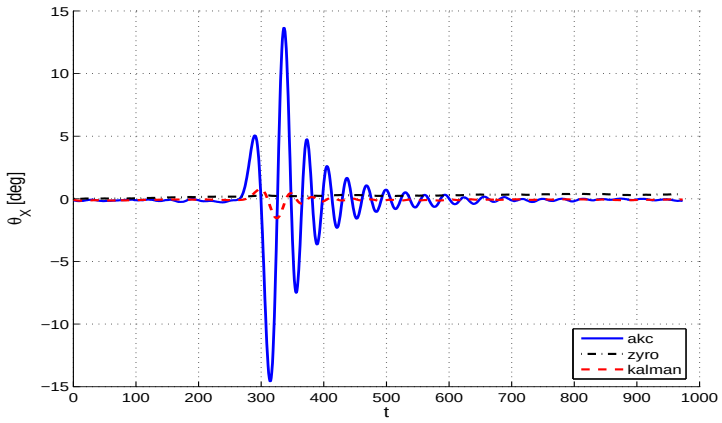
Rysunek 5.15 Rotacja w osi OX na podstawie filtracji Kalmana (wymuszenie – seria wychyleń, następnie powrót do pozycji wyjściowej)



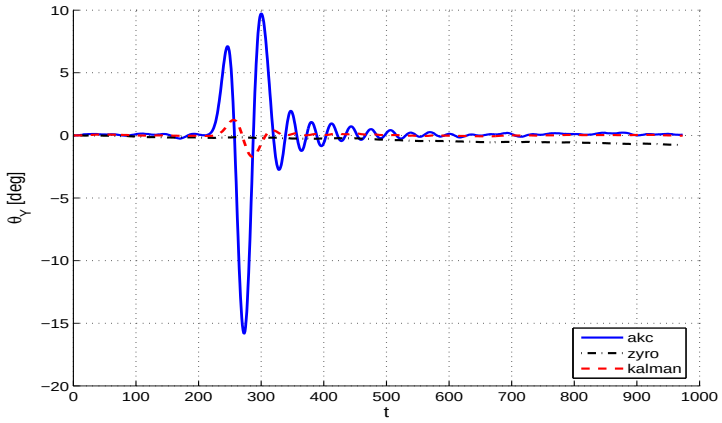
Rysunek 5.16 Rotacja w osi OY na podstawie filtracji Kalmana (wymuszenie – seria wychyleń, następnie powrót do pozycji wyjściowej)



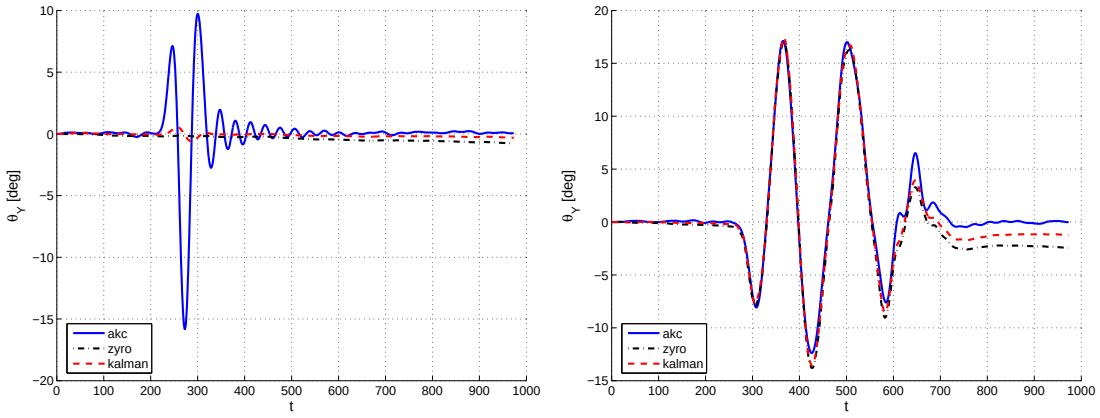
Rysunek 5.17 Rotacja w osi OZ na podstawie filtracji Kalmana (wymuszenie – seria wychyleń, następnie powrót do pozycji wyjściowej)



Rysunek 5.18 Rotacja w osi OX na podstawie filtracji Kalmana (wymuszenie – szarpnięcie w osi OY)

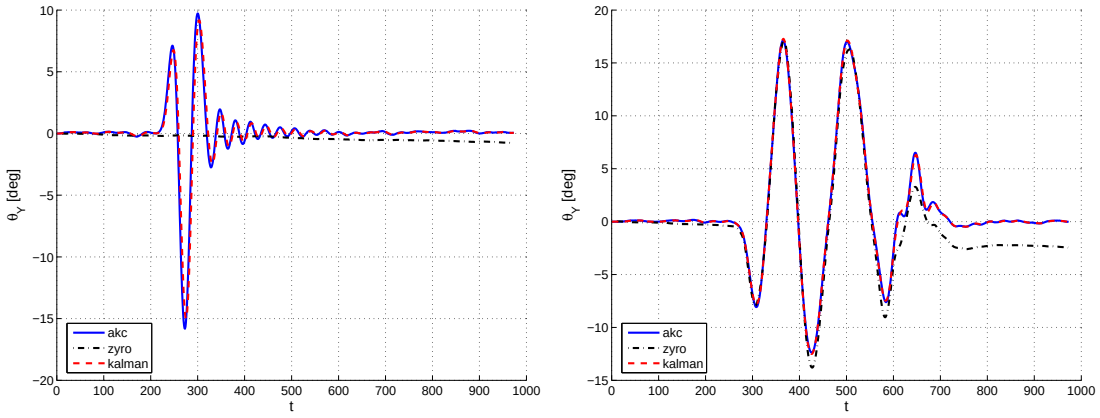


Rysunek 5.19 Rotacja w osi OY na podstawie filtracji Kalmana (wymuszenie – szarpnięcie w osi OX)



(a) (b)

Rysunek 5.20 Rotacja w osi OY na podstawie filtracji Kalmana dla parametrów $r_{akc_y} = 1000$, $q = 0.001$



(a) (b)

Rysunek 5.21 Rotacja w osi OY na podstawie filtracji Kalmana dla parametrów $r_{akc_y} = 0.01$, $q = 0.001$

Na rysunkach 5.9, 5.10, 5.11 przedstawiono sytuację, gdy moduł INS znajduje się w bezruchu (brak wymuszenia). Wyraźnie widać, że w przypadku rotacji w osiach OX, OY filtracja Kalmana skutecznie eliminuje błąd systematyczny widoczny na pomiarze z żyroskopów. Niestety w przypadku rotacji w osi OZ (orientacji), nawet filtracja Kalmana nie pozwala na skuteczne wyeliminowanie zjawiska dryfu żyroskopu. Brak dodatkowej informacji z czujników obarczonych jedynie szumem a nie błędem systematycznym. Jak już zasugerowano poprzednio, dobrym rozwiązaniem było by połączenie żyroskopu z kompasem elektronicznym. Na rysunkach 5.12, 5.13, 5.14 przedstawiono pomiar rotacji w przypadku kolejnych wymuszeń w każdej z osi. Rysunki te potwierdzają wysunięte powyżej wnioski. z kolei sytuację, gdy moduł poddano serii wymuszeń w każdej z osi (kiwanie), a następnie powrócono do pozycji wyjściowej, zobrazowano na rysunkach 5.15, 5.16, 5.17. i w tym przypadku widać, że połączenie informacji z akcelerometrów i z żyroskopów poprzez F–K sprawdza się bardzo dobrze, natomiast błędy pojawiają się w osi OZ. Ciekawy wpływ filtracji Kalmana zobrazowano na rysunkach 5.18, 5.19. Przedstawiają one pomiar wychyleń od pionu w trakcie gwałtownego przemieszczenia modułu (szarpnięcie). Wyraźnie widać, że filtr Kalmana z odpowiednio dobranymi współczynnikami R i Q pozwala na eliminację błędu pomiaru wprowadzonego przez akcelerometry rejestrujące przyspieszenie wynikające z ruchu modułu.

Należy zaznaczyć, że dla wszystkich omawianych powyżej rysunków parametry filtracji miały następujące wartości $r_{akc_x} = r_{akc_y} = r_{zyro_z} = 10$, $q = 0.001$. Sam wpływ nastaw filtru przedstawiono na rysunkach 5.20, 5.21. Wartością parametru R (szum pomiaru) można zmieniać wpływ fazy korekcji na wyjście filtru. Zwiększenie wartości R powoduje zmniejszenie znaczenia fazy korekcji, co poprawia tłumienie błędów z akcelerometrów podczas szarpnięcia (rysunek 5.20a), ale z drugiej strony w sytuacji "kiwania" wpływ korekcji jest zbyt mały i wychylenie nie powraca do rzeczywistej wartość (rysunek 5.20b). Zmniejszenie wartości R powoduje zwiększenie znaczenia fazy korekcji, co wyraźnie widać na rysunku (rysunek 5.21a). Wpływ parametru q jest podobny dlatego zrezygnowano z przedstawiania kolejnych rysunków. W wielu publikacjach zaleca się stosować q w granicach 10^{-5} co sprzyja dostrajaniu filtru [19].

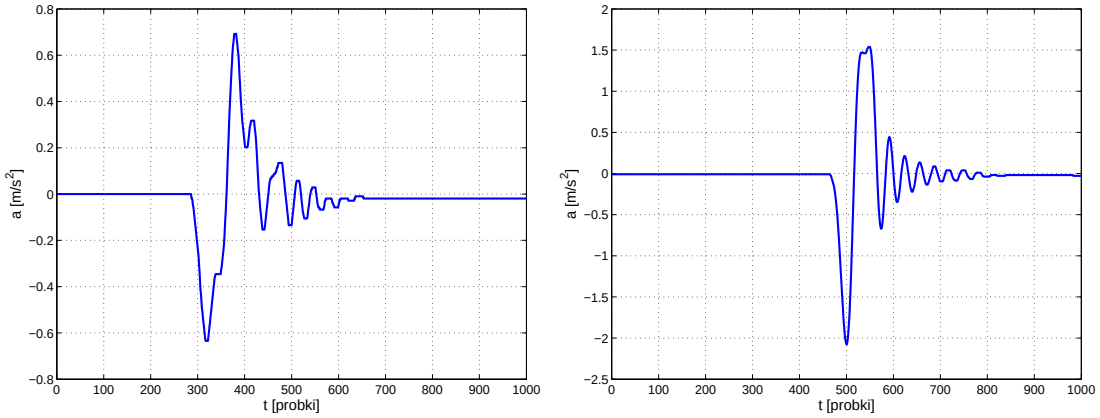
5.1.6 Wyznaczanie prędkości oraz przemieszczenia na podstawie przyspieszeń

Na czas experymentu modułu INS wraz płytą główną robota został wymontowany z robota. Badanie poległo na próbie określenie przemieszczenia i prędkość na podstawie pomiaru przyspieszenia. Wymuszenie realizowane było ręcznie poprzez przesunięcie modułu ISN po płaskiej powierzchni. Eksperyment w osi OZ polegał na podniesieniu modułu na pewną wysokość a następnie opuszczeniu w to samo miejsce.

Zastosowano filtr opóźniający oraz filtr analogowy, w którym częstotliwość odcięcia ustawiono na 10Hz.

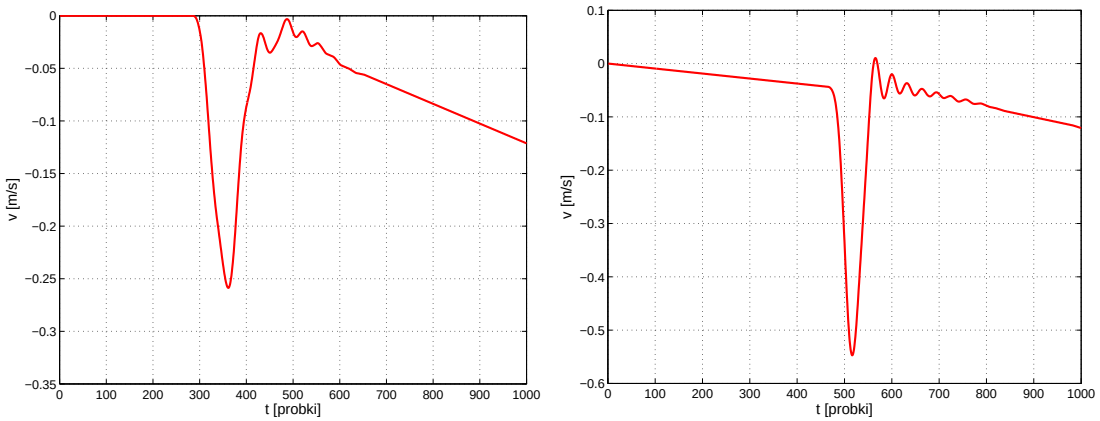
Wartości przyspieszeń rejestrowane przez akcelerometry, poddano kolejnym przeliczenią, tak aby wyeliminować przyspieszenie ziemskie oraz wyznaczyć przyspieszenie w układzie globalnym. Przeliczeń tych dokonano zgodnie z równaniami przedstawionymi w rozdziale 2.8.

Przeliczenia związane z wyznaczeniem przyspieszeń w układzie globalnym realizowane były w mikroprocesorze, natomiast w celu wyznaczanie prędkość i przemieszczenia posłużono się środowiskiem Matlab. Zastosowano trapezową metodę całkowania.



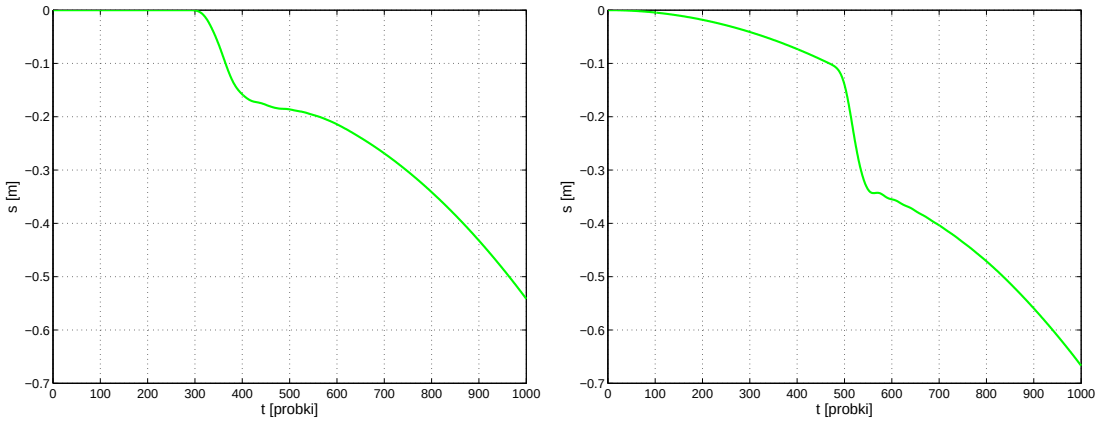
(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.22 Przyspieszenie w osi OX (wymuszenie $S = 20cm$ w osi OX)



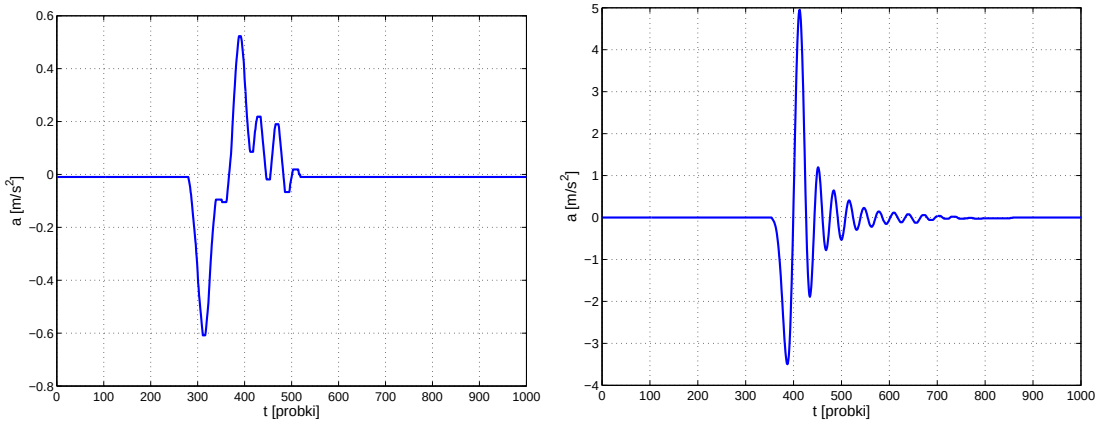
(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.23 Prędkość w osi OX (wymuszenie $S = 20cm$ w osi OX)



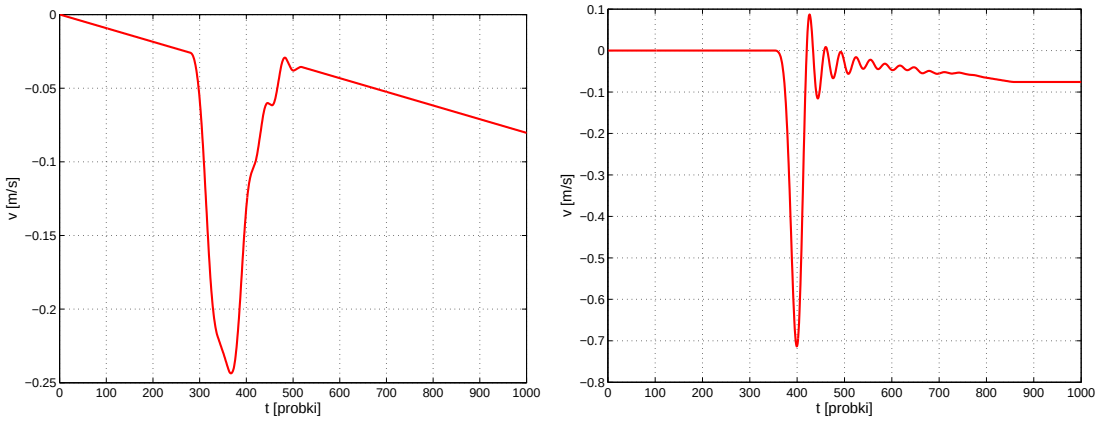
(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.24 Przemieszczenie w osi OX (wymuszenie $S = 20cm$ w osi OX)



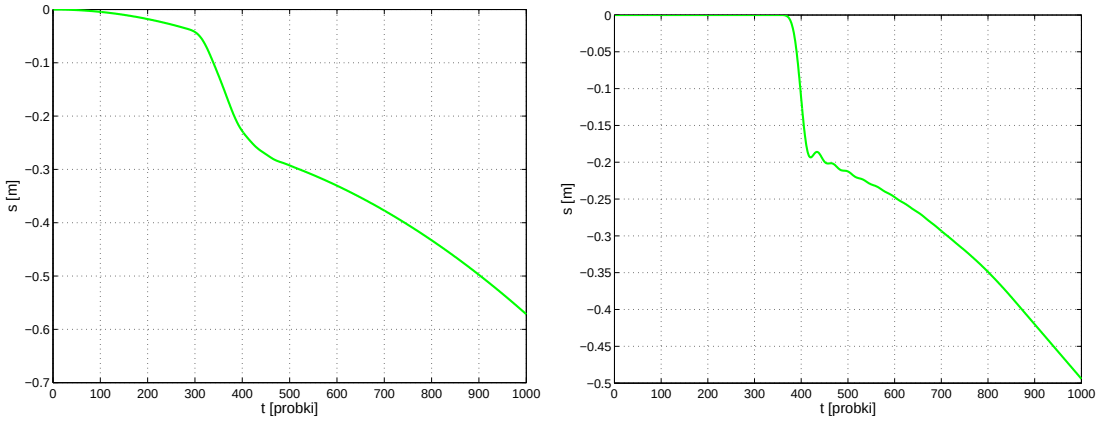
(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.25 Przyspieszenie w osi OY (wymuszenie $S = 20cm$ w osi OY)



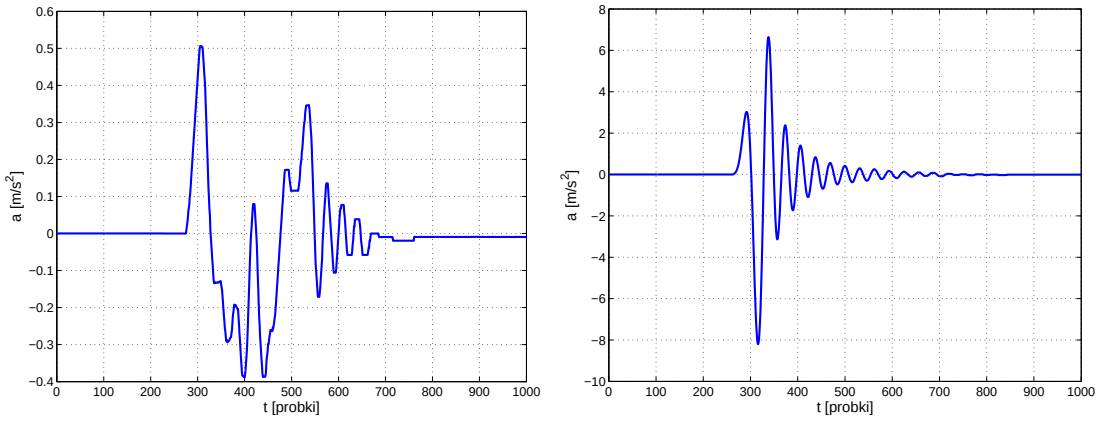
(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.26 Prędkość w osi OY (wymuszenie $S = 20cm$ w osi OY)



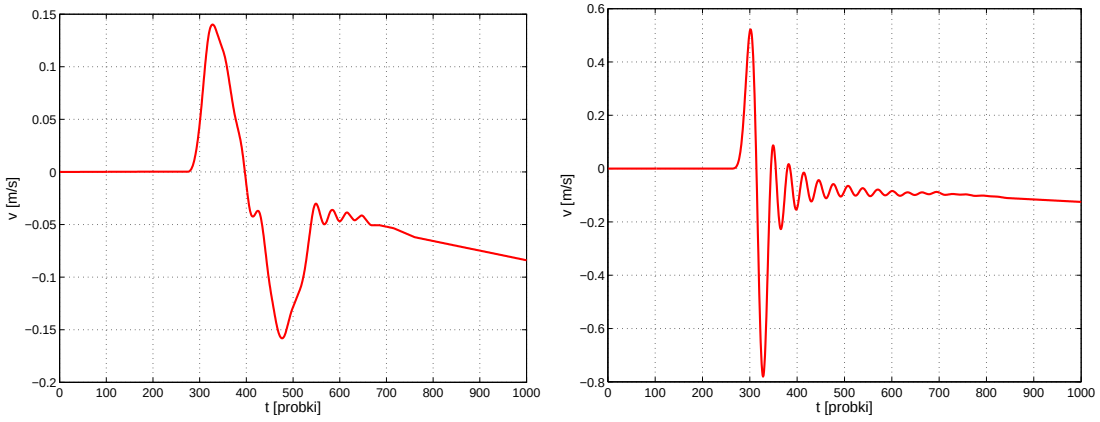
(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.27 Przemieszczenie w osi OY (wymuszenie $S = 20cm$ w osi OY)



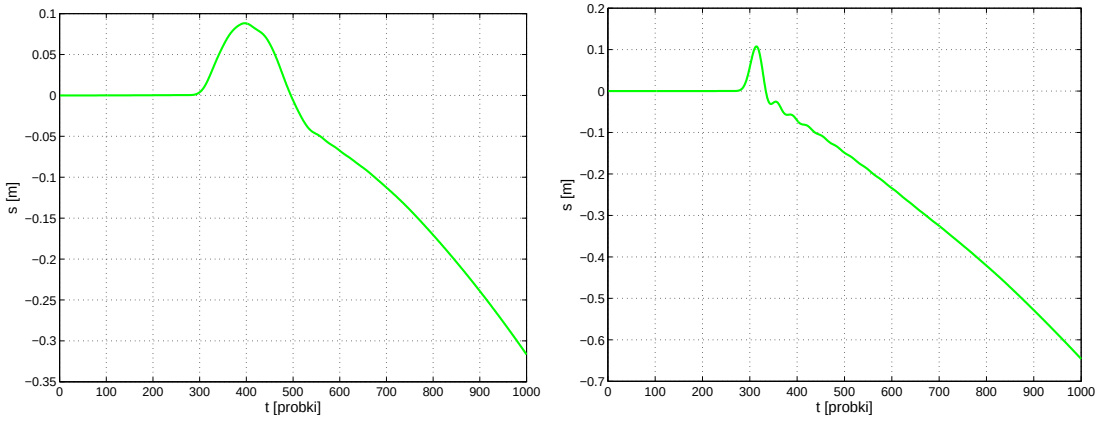
(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.28 Przyspieszenie w osi OZ (wymuszenie w osi OZ)



(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.29 Prędkość w osi OZ (wymuszenie w osi OZ)



(a) Powolne przemieszczenie (b) Gwałtowane przemieszczenie

Rysunek 5.30 Przemieszczenie w osi OZ (wymuszenie w osi OZ)

Na rysunkach 5.22, 5.25, 5.28 przedstawiono przyspieszenie w trakcie wymuszenia w kolejnych osiach. Wymuszenie w osiach OX, OY to przemieszczenie po płaskiej powierzchni o 20cm, natomiast w osi OZ to podniesienie i opuszczenie modułu spowrotem na tą samą wysokość. Dla każdej z osi przeprowadzono po dwa eksperymenty. Raz wymuszenie było łagodne (powolne), drógi raz wymuszenie było gwałtowne (szarpnięcie). Prędkość przedstawiono na rysunkach 5.23, 5.26, 5.29, natomiast przemieszczenie na rysunkach 5.24, 5.27, 5.30. Największym problemem jest to, że prędkość po wymuszeniu nie powraca do zera, co skutkuje nieustannym narastaniem drogi. Wystarczy mała niedokładność przy kalibracji czujnika, w stanie spoczynku wskazanie odbiega nieznacznie od zera, a prędkość i droga nieustannie narasta.

Jak widać, pomimo zastosowania bardzo dobrych analogowych filtrów oraz cyfrowego filtru opóźniającego, określanie położenia bazując tylko na pomiarach z akcelerometrów nie ma większego sensu. Już przy pojedynczym wymuszeniu w danej osi wyznaczone przemieszczenie jest niepoprawne, nie wspominajac o sytuacji, gdy robot będzie się poruszać i pojawią się dodatkowe zakłócenia związane z drganiami konstrukcji.

5.1.7 Wnioski

Połączenie informacji z akcelerometrów i żyroskopów poprzez filtr Kalmana pozwala uzyskać dokładną informację o rotacji w osi X i Y. Informacja ta może być z powodzeniem wykorzystana do korekcji postury korpusu robota podczas kroczenia po nierównym terenie.

Określenie orientacji z wykorzystaniem tylko informacji z żyroskopu nie jest możliwe. Po dłuższym czasie daje znać o sobie zjawisko dryfu oraz błędy kompensacji i kalibracji czujnika. Dobrym rozwiązaniem było by wprowadzenie dodatkowo informacji z kompasu elektronicznego i zastosowanie filtru Kalmana.

Pomiar przemieszczenia na podstawie informacji o przyspieszeniu z akcelerometrów jest praktycznie niemożliwy. Mimo zastosowanie bardzo dobrej filtracji analogowej dają o sobie znać błędy kompensacji i kalibracji czujnika. Już po pojedynczym wymuszeniu pomiary znacznie różnią się od wartości rzeczywistych.

5.2 Badanie nogi robota

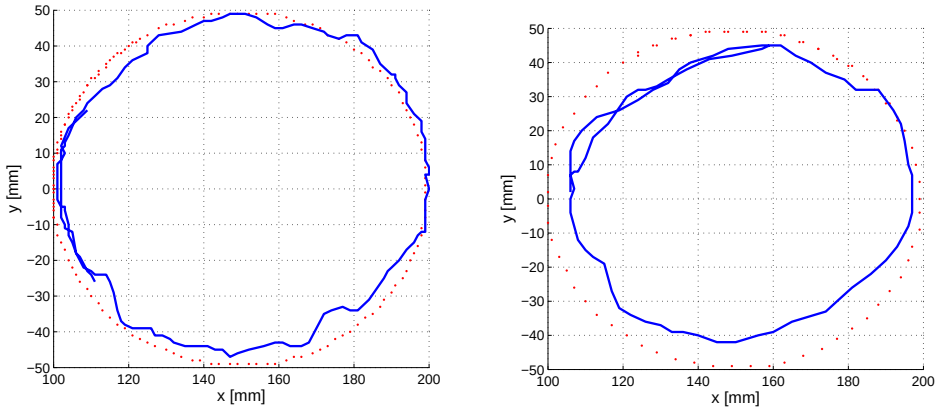
Podczas budowania mapy wysokości terenu wykorzystywana jest informacja o pozycji stóp robota. Dzięki kinematyce prostej robota oraz znajomości kątów w przegubach nogi, pozycji i orientacji korpusu robota można określić pozycję stopy, a tym samym wysokość terenu. Na dokładność takiego pomiaru, dokładność śledzenia trajektorii przez robota oraz określanie pozycji robota ma wpływ jakość sterowania pojedynczą nogą robota. Zadanie to spoczywa na lokalnym sterowniku nogi robota opisanym w rozdziale 3.3.

W rozdziale tym przedstawiono wyniki badań, jakim poddano lokalny sterownik nogi robota. Przeprowadzono badania śledzenia trajektorii oraz trzymania zadanej pozycji.

5.2.1 Śledzenie zadanej trajektorii

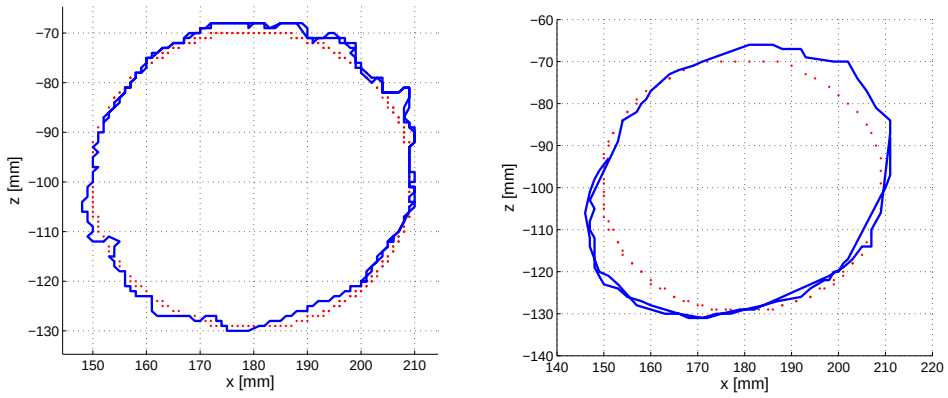
Przy pomocy sterownika głównego robota wykonano badanie śledzenia trajektorii przez nogę robota. Generator trajektorii zaimplementowano w sterowniku głównym, skąd wysyłano informacje o pozycji stopy do lokalnego sterownika nogi. Jednocześnie odczytywano informacje z enkoderów o kątach w przegubach nogi, a te z kolei przeliczano na pozycję stopy. Informacje o zadanej i rzeczywistej pozycji stopy wysyłano cyklicznie do komputera. Informacje rejestrowano w środowisku Matlab.

Zadana trajektoria to okrąg kolejno w płaszczyźnie OXY, OXZ, OYZ dla różnych prędkości. Na rysunkach 5.31.a, 5.32.a i 5.33.a przedstawiono pozycję stopy podczas śledzenia trajektorii przy małej prędkości, z kolei na rysunkach 5.31.b, 5.32.b, 5.33.b przy dużej. Widać, że wpływ prędkości na jakość śledzenia trajektorii jest bardzo znaczący. Nawet dla niewielkiej prędkości dokładność odwzorowania trajektorii jest niezadowalająca. Wynika to przede wszystkim z jakości zastosowanych serwomechanizmów, dużych luzów na przekładni oraz niewielkiej maksymalnej prędkości serwa. Nie bez znaczenia jest też sposób sterowania. Serwomechanizm modelarski posiada wbudowany fabrycznie regulator, który ma za zadanie utrzymać zadaną pozycję. Regulator ten jest bardzo prosty i jedynym parametrem, jakim można sterować, jest pozycja. Nie ma możliwości zadania prędkości. Zaimplementowany w sterowniku lokalnym nogi regulator ze sprzężeniem od enkoderów pozwala jedynie na proporcjonalne kontrolowanie serwa.



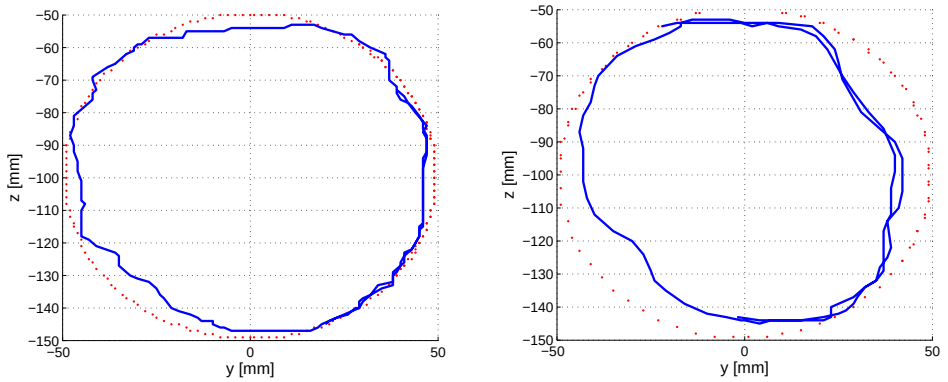
(a) wolno (b) szybko

Rysunek 5.31 Śledzenie trajektorii dla różnych prędkości. Zadana trajektoria to okrąg o promieniu 50mm w płaszczyźnie OXY



(a) wolno (b) szybko

Rysunek 5.32 Śledzenie trajektorii dla różnych prędkości. Zadana trajektoria to okrąg o promieniu 30mm w płaszczyźnie OXZ



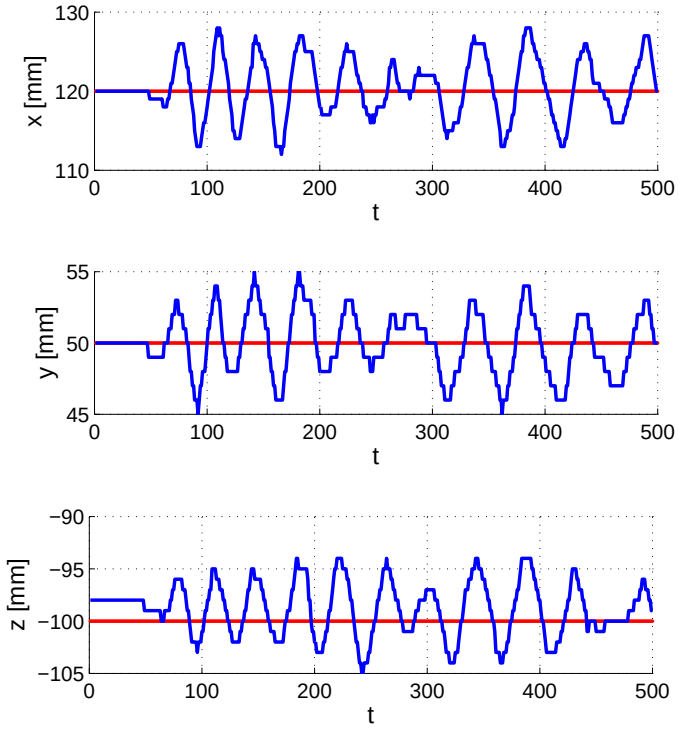
(a) wolno (b) szybko

Rysunek 5.33 Śledzenie trajektorii dla różnych prędkości. Zadana trajektoria to okrąg o promieniu 50mm w płaszczyźnie OYZ

5.2.2 Trzymanie zadanej pozycji

W badaniu tym, podobnie jak poprzednio, posłużono się sterownikiem głównym robota oraz środowiskiem Matlab. Eksperyment polegał na zbadaniu trzymania zadanej pozycji przez nogę robota podczas wymuszeń. Wymuszenia realizowane były ręcznie poprzez wypychanie stopy z zadanej pozycji z niewielką siłą, aż do momentu napotkania oporu ze strony serwa. Badanie to miało na celu zgrubne określenie luzów w serwie (przekładni).

Na rysunku 5.34 przedstawiono pozycję stopy podczas wymuszeń. Zadano następującą pozycję: $x = 120$, $y = 50$, $z = -100$. Widać, że błąd trzymania zadanej pozycji oscyluje w granicach ±5mm. Co istotne, po ustąpieniu wymuszeń błąd nie powraca do zera. Błędy te wynikają z luzów w przekładni serwa oraz z samego sterownika serwa. Serwo modelarskie posiada stopień nieczułości rzędu 0,2–0,5 stopnia [1], co przekłada się na dokładność pozycjonowania stopy. Wartości tych błędów są różne w zależności od konfiguracji nogi. Na rysunku przedstawiono wynik eksperymentu dla typowej konfiguracji nogi.



Rysunek 5.34 Trzymanie zadanej pozycji podczas wymuszeń

5.3 Wyznaczanie pozycji i orientacji robota

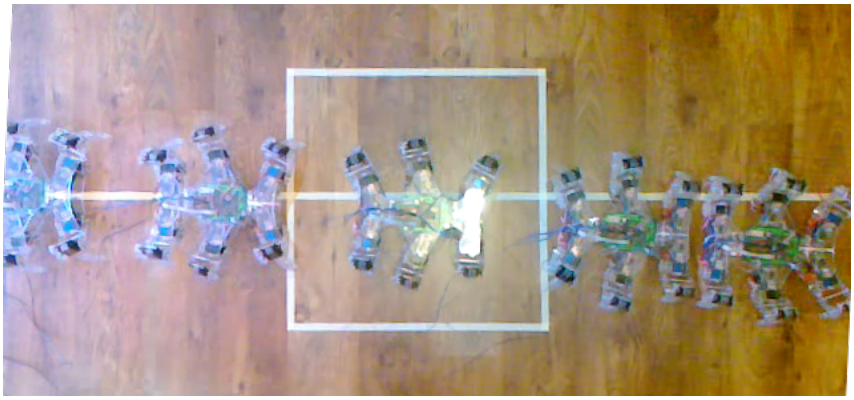
W rozdziale tym przedstawiono wyniki badań nad określeniem pozycji i orientacji robota. Podjęto próbę wyznaczenia położenia robota za pomocą odometrii z użyciem czujników inercyjnych. W badaniach posłużono się robotem kroczącym Mrówa.

5.3.1 Wyznaczanie pozycji i orientacji robota kroczącego na podstawie odometrii

Przeprowadzono eksperyment polegający na pomiarze przemieszczenia robota z wykorzystaniem odometrii. Odometria to sposób na wyznaczenie pozycji robota opierający się na założeniu braku poślizgów między nogą a podłożem. Przemieszczenie robota wyliczane jest wprost z algorytmu kroczenia.

Ruch po prostej

W pierwszym eksperymencie robot miał za zadanie pokonać odcinek prostej o długości 2 m, po czym dokonano pomiaru pozycji robota za pomocą miarki. Eksperyment powtórzono kilkakrotnie dla różnych prędkości i różnych wysokości korpusu. Przebieg jednego z eksperymentów przedstawiono na rysunku 5.35. Wyniki badań zamieszczono w tabeli 5.3.1.



Rysunek 5.35 Przemieszczenie robota wzdłuż prostej ($V_y = 40mm/s$, wysokość korpusu 90mm)

Tabela. 5.1 Pozycja robota po przejściu prostej długości 2m

Prędkość [mm/s]	Wysokość korpusu [mm]	wsp. x [cm]	wsp. y [cm]
$V_y = 40$	90	13	196
$V_y = 80$	90	10	187
$V_y = 40$	130	20	192
$V_y = 80$	130	15	185

Obrót w miejscu

Podczas drugiego badania sprawdzono odwzorowanie trajektorii przez robota podaczas obrotu. Robota miał za zadanie zmienić orientacje o 90 stopni. Badanie powtórzono kilkakrotnie dla obrotu w lewo i w prawo dla różnych prędkości obrotu i różnych wysokości korpusu. Wyniki przedstawiono w tabeli 5.3.1. Jak widać jest znacząca różnica pomiędzy obrotem w lewo a obrotem w prawo. Nie bez znaczenia jest też prędkość obrotu. Zdecydowano się wprowadzić współczynniki korekcji. Wyznaczono je poprzez uśrednienie pomiarów, a następnie podzielenie uzyskanej wartości przez 90 stopni. Wartości przyjętych współczynników korekcji to 0.8 dla obrotu w lewo i 1.2 dla obrotu w prawo. W każdej iteracji algorytmu kroczenia przyrost kątowy mnożony jest przez odpowieni współczynnik.

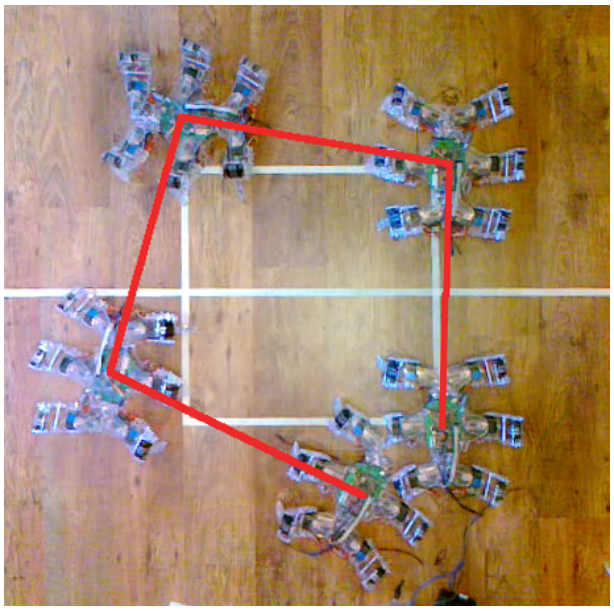
Rozwiązanie to oczywiście nie rozwiązało wszystkich problemów z określeniem orientacji. Pozostaje problem różnej przyczepności w zależności od podłoża. Badań takich nie przeprowadzono ale intuicja wskazuje że rodzaj podłoża będzie mieć duży wpływ na poślizgi robota.

Tabela. 5.2 Orientacja robota po obrocie o 90 stopni

Prędkość [mm/s]	Wysokość korpusu [mm]	kierunek	orientacja [stopnie]
$\alpha_z = 5$	90	lewo	74
$\alpha_z = 10$	90	lewo	66
$\alpha_z = 5$	130	lewo	78
$\alpha_z = 10$	130	lewo	70
$\alpha_z = 5$	90	prawo	-112
$\alpha_z = 10$	90	prawo	-99
$\alpha_z = 5$	130	prawo	-113
$\alpha_z = 10$	130	prawo	-108

Ruch po kwadracie

Ostateczną przeprowadzoną próbą pokazującą skuteczność określenia pozycji i orientacji z użyciem odometrii było przejście robota wzdłuż kwadratu. Robot miał do pokonania ścieżkę w kształcie kwadratu o bokach 70 cm. Wynik jednego z badań przedstawiono na rysunku 5.36. Robot ukończył zadanie w pozycji: $x = -16$ cm, $y = -21$ cm, $r_z = -30°$ (licząc od punktu startowego). Podobną próbę przeprowadzono dla ruchu w kierunku przeciwnym. Uzyskana pozycja to: $x = 20$ cm, $y = 10$ cm, $r_z = -20°$.



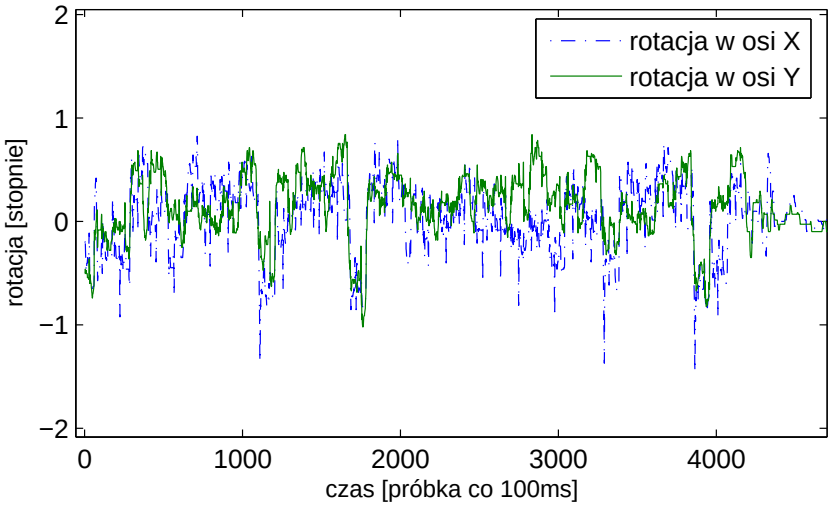
Rysunek 5.36 Przemieszczenie robota wzdłuż kwadratu o boku 70cm

5.3.2 Wyznaczanie pozycji i orientacji robota na podstawie pomiarów z modułu INS

W trakcie wykonywania eksperymentów opisanych w poprzednim rozdziale prowadzono także rejestrację danych z modułu INS. Informacje o przechyłach: bocznym i wzdłużnym robota są na bieżąco wykorzystywane do korekcji postury korpusu. Po każdym cyklu zmiany nóg podporowych następuje faza korekcji, w której na podstawie informacji z modułu INS niwelowane jest przechylenie korpusu. Na rysunku 5.37 przedstawiono odczyt przechyłów korpusu w trakcie pokonywania prostej.

Z prób określenia przemieszczenia robota na podstawie pomiaru przyspieszeń przez akcelerometry zrezygnowano. Proste eksperymenty wykonane w trakcie badań modułu INS (rozdział 5.1.5) pokazały wady metody całkowicie eliminujące ją z praktycznego zastosowania.

Niestety nie udało się przeprowadzić badań nad określeniem orientacji robota (rotacja w osi Z) na podstawie pomiaru prędkości kątowej. Czujnik uległ uszkodzeniu. W trakcie badań modułu INS dotyczących określenia orientacji (rozdział 5.1.4) wykazano problemy metody wnikające z tzw. dryfu żyroskopu. Wnioskując na podstawie wyników tych badań, można stwierdzić, że w trakcie kroczenia błąd orientacji będzie systematycznie narastał.



Rysunek 5.37 Przechyły robota w trakcie ruchu po prostej

5.3.3 Wnioski

Dokładność określenia pozycji robota na podstawie odometrii jest silnie zależna od przyczepności nóg robota. Prędkość przemieszczenia, wysokość korpusu oraz rodzaj podłoża mają duży wpływ na końcowy wynik. Problem jest w szczególności widoczny podczas obrotu korpusu. Być może zastosowanie stóp o innej konstrukcji, które zapewniłyby lepszą przyczepność dałoby lepsze rezultaty.

Próba wyznaczenia pozycji i orientacji robota na podstawie informacji z czujników inercyjnych zakończyła się niepowodzeniem. Informacje z czujników inercyjnych mogą posłużyć jedynie jako uzupełnienie informacji z innych czujników, których wynik nie jest obarczony błędem systematycznym.

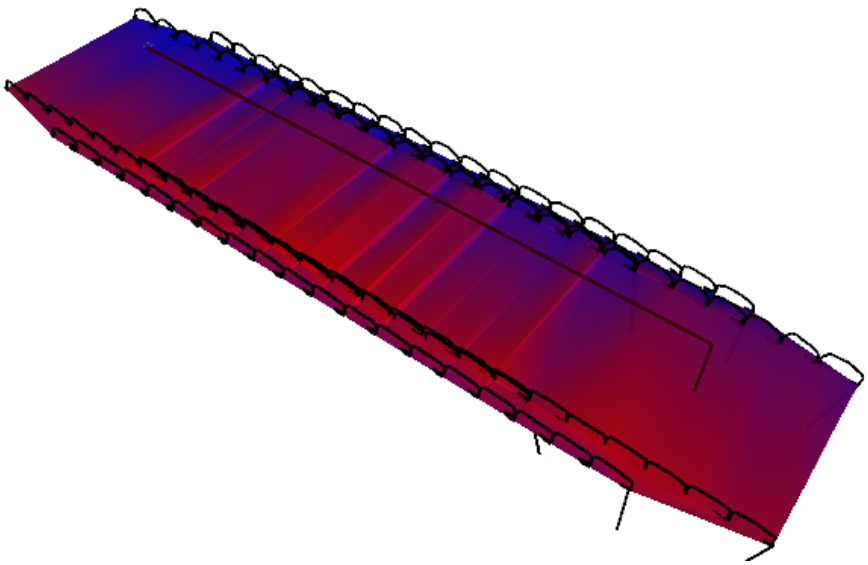
Reasumując, zastosowane metody nawigacji nie dają dobrych rezultatów. o ile zastosowanie odometrii na krótkim dystansie ma sens, to wraz z przebytą drogą błędy narastają. Konieczne wydaje się rozwinięcie układu nawigacji robota o zewnętrzne systemy, które co jakiś czas dokonywałyby korekty. Jednym z rozwiązań, które mogłoby się sprawdzić, jest przedstawiony w pozycji [10] system nawigacji, który poprzez system optyczny potrafi określić pozycję i orientację specjalnie przygotowanego markera. Została nawet przygotowana biblioteka `ARToolKit`, która dokonuje wszystkich przeliczeń i w prosty sposób pozwala zastosować metodę. Aby zastosować ten system do nawigacji robota można umieścić kamerę w znanym miejscu, natomiast marker przymocować do robota. W celu poprawy dokładności i zwiększenia obszaru roboczego można zwiększyć liczbę kamer. System ten na pewno ma duże możliwości i wart jest zbadania.

5.4 Tworzenie mapy wysokości z użyciem robota kroczącego

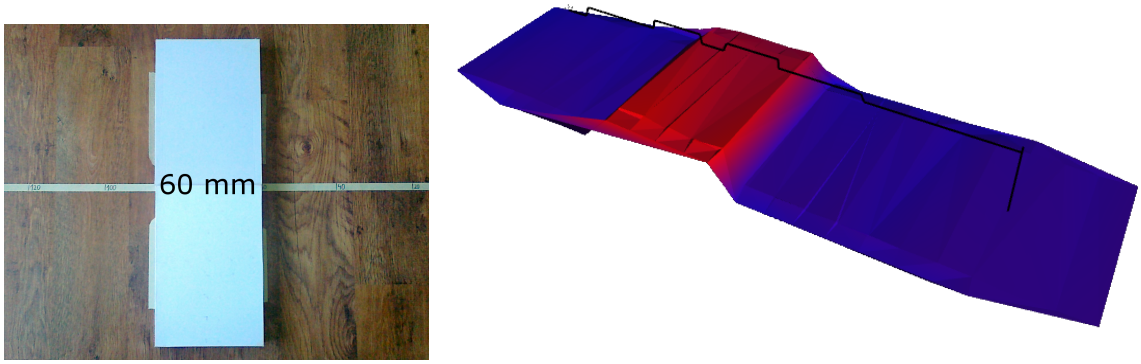
Przeprowadzone zostały badania nad użyciem robota kroczącego do budowania mapy wysokości terenu. Badania przeprowadzono na rzeczywistej konstrukcji. Do tworzenia mapy na podstawie punktów pomiarowych wykorzystano triangulację Delone. Do gromadzenia informacji i prezentacji wyników posłużył program sterujący robotem 4.

W trakcie badań robot poruszał się czteropodporowym algorytmem kroczenia. Algorytm ten jest bardzo zbliżony do tego omówionego w rozdziale 2.5. Różnica polega na tym, że w trakcie fazy przemieszczenia korpusu robot podparty jest na wszystkich nogach. Następnie następuje faza przemieszczenia nóg w nową pozycję. Nogi przenoszone są parami, tak że korpus jest zawsze podparty przez co najmniej cztery nogi. Algorytm ten pozwala zachować większą stabilność konstrukcji podczas ruchu po nierównościach.

Przeprowadzono trzy eksperymenty. W pierwszym robot poruszał się wzdłuż prostej po płaskiej powierzchni. Na rysunku 5.38 przedstawiano mapę terenu. Kolory mapy są skalowane względem najwyższego i najniższego punku pomiarowego. Wysokość odpowiadająca danym kolorom zawarta jest w opisie rysunku. Na rysunku dodatkowo zaprezentowano trajektorię korpusu i nóg robota. w kolejnych dwóch eksperymentach robot poruszał się wzdłuż prostej po specjalnie przygotowanych trasach z przeszkodami. Widok badanego podłoża przedstawiono na zdjęciach 5.39.a, 5.41.a, natomiast mapy wysokości na rysunkach 5.39.b, 5.41.b. Z kolei na rysunkach 5.40, 5.42 przedstawiono wysokość uzyskaną z nóg środkowych w trakcie próby nr 2 i nr 3.

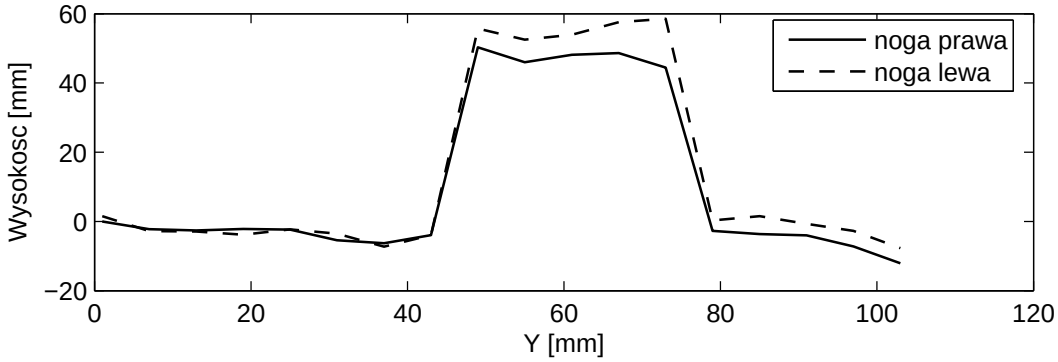


Rysunek 5.38 Mapa wysokości podczas ruchu po płaskiej powierzchni (niebieski -8mm, czerwony 8mm)

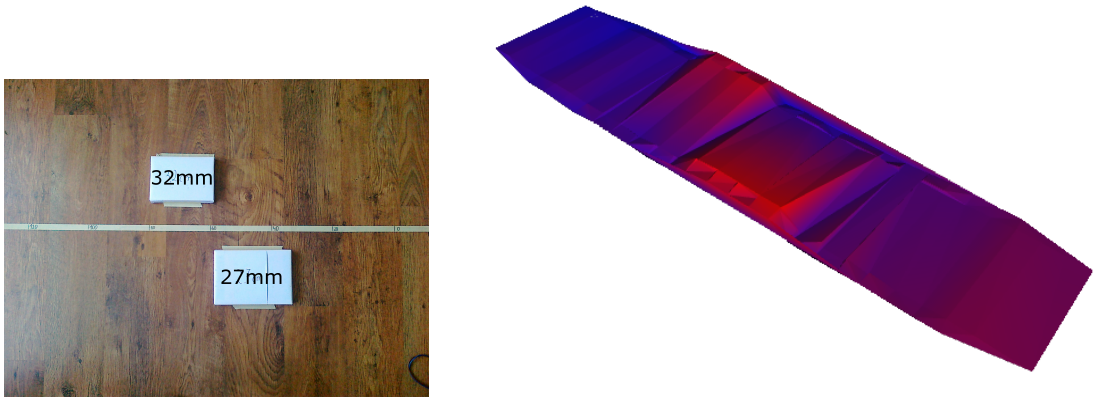


(a) teren rzeczywisty (b) mapa wysokości

Rysunek 5.39 Mapa wysokości próba nr 2 (niebieski -10mm, czerwony 63mm)

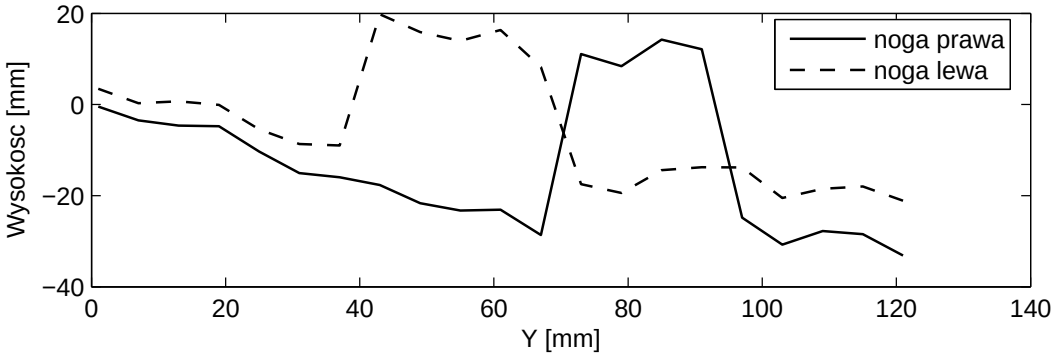


Rysunek 5.40 Wysokość nóg środkowych w trakcie próby nr 2



(a) teren rzeczywisty (b) mapa wysokości

Rysunek 5.41 Mapa wysokości próba nr 3 (niebieski -33mm, czerwony 26mm)



Rysunek 5.42 Wysokość nóg środkowych w trakcie próby nr 3

5.4.1 Wnioski

Największym problemem podczas budowania mapy jest określenie wysokości korpusu robota. Robot podczas pokonywania przeszkód musi zmieniać wysokość korpusu tak, aby utrzymać go na zadanej wysokości. W trakcie kroczenia wysokość korpusu wyliczana jest jako średnia z wysokości na jaką wyciągnięte są nogi robota. W momencie gdy któraś z nóg napotka na przeszkodę konieczna jest korekta wysokości korpusu. Wysokość korpusu robota w globalnym układzie odniesienia wyznaczana jest jako wysokość zadana plus suma korekt. W teorii taka metoda się sprawdza, jednak w trakcie rzeczywistych prób pojawiają się błędy. Eksperymenty wykazały, że robot w trakcie kroczenia ma tendencję do zniżania korpusu. Nawet robot kroczący po płaskiej powierzchni wymaga stosowania korekt wysokości korpusu. Wyznaczana w ten sposób wysokość korpusu w układzie globalnym robota obarczona jest błędem systematycznym. Problem wyraźnie widoczny jest podczas próby nr 3. Robot po pokonaniu przeszkód znajdował się na tej samej wysokości z jakiej rozpoczynał próbę. Natomiast na mapie wysokości widać że jest różnica w wysokości pomiędzy punktem startu a końcem. Sytuację tą dokładniej obrazuje wysokość nóg robota w trakcie próby przedstawiona na rysunku 5.42. Podobny problem zaobserwowano w trakcie próby nr 2 (rysunek 5.40).

Błąd ten można wyeliminować poprzez poprawę metody określania wysokości korpusu w układzie globalnym. Można by tutaj z powodzeniem zastosować system nawigacji, oparty o system wizyjny. Wspominano o nim w rozdziale 5.3 w trakcie omawiania prób określenia pozycji robota.

Reasumując dokładność odwzorowania terenu zależy przede wszystkim od dokładności określenia pozycji robota. Dodatkowo mała liczba próbek pomiarowych nie pozwala na uzyskanie szczegółowej mapy terenu.

Rozdział 6

Podsumowanie

Praca miała na celu przybliżenie odbiorcy tematu poruszania się robotów kroczących po trudnym terenie oraz poruszenie problemu budowania mapy wysokości terenu. Przedstawiono dość nietypową metodę budowania mapy wysokości, w której dane pomiarowe zbierane są przez samego robota. Uzyska w ten sposób mapa nie nadaje się do planowania ścieżki robota, jednak może być wykorzystana jako uzupełnienie map tworzonych innymi metodami.

W ramach pracy zbudowano robota kroczącego zdolnego poruszać się w trudnym terenie. Wykonany został moduł nawigacji inercyjnej INS, którego podstawowym zadaniem jest określenie przechyłów robota. Informacje uzyskane z modułu INS zostały z powodzeniem wykorzystane do poziomowania korpusu robota w trakcie kroczenia. Zastały także przeprowadzone badania nad określeniem przemieszczenia z użyciem akcelerometrów. Badania wykazały, że nawet pomimo zastosowania dobrych filtrów analogowych czujnik te nie nadają się do pomiaru przemieszczenia. Błędy kompensacji i kalibracji czujnika wprowadzają systematyczny błąd pomiaru, którego nie udało się wyeliminować. Przeprowadzono eksperymenty nad określeniem pozycji robota przy użyciu odometrii. Wyniki pokazały, że dokładność pomiaru jest silnie zależna od przyczepności nóg robota. Błędy uwidaczniają się w trakcie ruchu robota po łuku. Niestety nie udało się przeprowadzić badań nad określeniem orientacji robota na podstawie pomiaru prędkości kątowej. Czujnik uległ uszkodzeniu. Wstępne eksperymenty podczas badań modułu INS wykazały, że zastosowany żyroskop nadaje się do tego pomiaru. Próby budowania map wysokości przy pomocy robota kroczącego pokazały, że dokładność tworzonych map zależy od dokładności określenia pozycji robota. Niesyty zastosowane metody nawigacji dają kiepskie rezultaty. Dużym problemem jest też niewielka ilość punktów pomiarowych. Przedstawiona metoda nie pozwala na uzyskanie szczegółowej mapy terenu.

W celu poprawy dokładności map powinno się poprawić system nawigacji robota. Dobrym pomysłem wydaje się zastosowanie systemu nawigacji optycznej, gdzie kamera obserwuje specjalny znacznik umieszczony na korpusie robota. Dostępna jest biblioteka `ARToolKit` [10], która potrafi określić pozycje takiego znacznika z duża dokładnością. Wykonany w ramach pracy robot kroczący Mrówa może posłużyć do dalszych badań nad problemem poruszania się w trudnym terenie. Zastosowanie skanera laserowego pozwoliło by na budowanie map na podstawie których można by planować trajektorię robota. Przedstawiona praca jest jedynie wstępem do tematu poruszania się robotów kroczących po trudnym terenie, daje ona podwaliny pod przyszłe badania.

Bibliografia

[1] AlturnUSA.LLC. Alturn USA ADS-966HTG, 2009.

[2] Analog Devices. Adis16100 ±300°/sec Yaw Rate Gyroscope with SPI, 2006-2009.

[3] Austriamicrosystems. AS5045 12 Bit Programmable Magnetic Rotary Encoder Datasheet, rev. 1.6.

[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. *Geometria obliczeniowa Algorytmy i zastosowania.* 2007.

[5] Freescale Semiconductor. MC68332 User's Manual, 2004.

[6] Freescale Semiconductor. MMA1260EG Low G Micromachined Accelerometer, 2007.

[7] FTDI. UB232R USB Mini-B FT232R Evaluation Module Datasheet, 2010.

[8] B. Gaßmann, L. Frommberger, R. Dillmann, K. Berns. Real-Time 3D Map Building for Local Navigation of a Walking Robot in Unstructured Terrain, 2003.

[9] R. Hadsell, J. A. Bagnell, D. Huber, M. Hebert. Accurate Rough Terrain Estimation with Space-Carving Kernels.

[10] H. Kato, M. Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System, 1999.

[11] E. Krotkov, R. Hoffman. Terrain Mapping for a Walking Planetary Rover. *IEEE Transactions on Robotics and Automation*, 10:728–739, 1994.

[12] V. Kumar. Integration of Inertial Navigation System and Global Positioning System Using Kalman Filtering, 2004.

[13] MAXIM. MAX7400 8th-order, Lowpass, Elliptic, Switched-Capacitor Filters, 1999.

[14] Modbus Organization. Modbus over Serial Line Specification & Implementation guide, 2002.

[15] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, W. Burgard. Learning Predictive Terrain Models for Legged Robot Locomotion.

[16] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, the BigDog Team. Bigdog, the Rough-Terrain Quaduped Robot.

[17] Rayson Technology Co., Ltd. BTM-112 Bluetooth Class 2 CSR BC4-ext module Datasheet, 2006.

[18] K. Tchoń, A. Mazur, I. Dulęba, R. Hossa, R. Muszyński. *Manipulatory i roboty mobilne*. 2000.

[19] G. Welch, G. Bishop. An Introduction to the Kalman Filter. Raport instytutowy, Chapel Hill, NC, USA, 1995.

[20] M. Wnuk. Moduł z mikrokontrolerem mc68332. Raport instytutowy, 2007.

[21] T. Zielińska. *Maszyny Kroczące*. 2003.

[22] Łukasz Tułacz. Nawigacja robota mobilnego z użyciem czujników inercyjnych, 2009.

Dodatek A

Szczegóły implementacji

A.1 Proste sposoby przyspieszenia obliczeń dla mikroprocesorów

Zastosowany w pracy mikroprocesor MC68332 nie jest wyposażony w jedostkę zmienno–przecinkową. Obliczenia na danych typu `float` są bardzo czasochłonne. W celu usprawnienia szybkości obliczeń często zmiast obliczeń zmienno–przecinkowych stosuje się operacje stało–przecinkowe. Pozwala to w istotnym stopniu pryspieszyć obliczenia. Niestety niesie to za soba kilka istotnych konsekwencji: obliczenia wykonywane są z odgórnie ustaloną precyzją, przez co istnieje ryzyko przekroczenia zakresu wartości przechowywanych w zmiennej; sam zapis operacji jest dość niewygodny i nieczytelny.

W celu implementacji operacji stało–przecinkowych w języku C wygodnie jest skorzystać z makr preprocesora. Przykład takiej implementacji przedstawiono poniżej.

```
typedef int32_t fixint;

#define FXBASE 12

#define FX2INT(a)       ((int32_t)((a==0?a:a>0?a+1:a-1) >> FXBASE))
#define INT2FX(a)       ((fixint)(((int32_t)a) << FXBASE))
#define FL2FX(a)        ((fixint)(a * (1 << FXBASE)))
#define FX2FL(a)        ((float)((float)a / (float)(1 << FXBASE)))
#define FXADD(a,b)      ((fixint)(a + b))
#define FXSUB(a,b)      ((fixint)(a - b))
#define FXMUL(a,b)      ((fixint)((a * b) >> FXBASE))
#define FXDIV(a,b)      ((fixint)((a << FXBASE) / b))

#define FXABS(a)        ((a >= 0)? a: -a)
```

Dla przejrzystości kodu zdefiniowano nowy typ danych `fixint`, reprezentujący wartość stało przecinkową. o precyzji obliczeń decyduje stała preprocesora `FXBASE`. Wartość przyjętej precyzji obliczen należy dostosować do zakresu danych w aplikacji. Dla przykładau przy precyzji 16 bitów pomnożenie dwóch jedynek spowoduje przekroczenie zakresu dla zmiennej `int32_t`. Powoduje to dość często trudne do wychwycenia błędy obliczeń.

Kolejnym wąskim gardłem dla obliczeń realizownych w prostych mikroprocesorach jest liczenie funkcji trygonometrycznych. Dostarczone przez producenta standardowe biblioteki języka C zaiwerają implementację funkcji trygonometrycznych. Implementacje te daja bardzo dokładne wyniki, jednak są bardzo czasochłonne. Często stosowanym usprawnieniem obliczeń jest tablicowanie funkcji. Mikroprocesor na początku dokonuje obliczeń z wykorzystaniem funkcji bibliotecznych zapisując wyniki w tablicy. Obliczenie funkcji

podczas działania progrmu sprowadza się do odczytania wartości z konkretnego miejsca w tablicy. Wykorzystuje się tutaj fakt okresowaści funkcji trygonomtrycznych np. dla funkcji `sin` w tablicy wystrczy zachować wartości funkcji w przedziale x od 0 od $\frac{\pi}{2}$.

W pracy posłóżono się gotową implementacją tablicowanych funkcji trygonometrycznych autorstwa doktora Marka Wnuka i doktora Marka Kabały. Biblioteka dodatkowo posiada implementację funkcji pierwiastka kwadratowego. Precyzję obliczeń dla biblioteki można zmieniać, w pracy przyjęto 12 bitów.

A.2 Implementacja filtru Kalmana

Przedstawiony w rozdziale 5.1.5 filtr Kalmana dla wyznaczania rotacji wymaga operacji na macierzach rozmiaru 6x6. o ile dla Matlaba operacje takie nie są niczym wymagającym, w przypadku stosowanego mikroprocesora MC68332 już tak nie jest. Wykonywanie takich obliczeń w czasie rzeczywistym pomiędzy kolejnymi próbkami jest praktycznie niemożliwe.

Skorzystano z faktu, że przedstawiony filtr 5.1.5 można rozdzielić na trzy osobne, w których rozmiar macierzy to 2x2, a także z faktu, że w macierzach A i H występuje wiele zer. Pozwala to na eliminację obliczania zbędnych wyrazów.

Filtr F-K dla rotacji w osi OX, OY można zapisać w następujący sposób:

$$x = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}, \quad A = \begin{bmatrix}1 & 0 \\ 0 & 0\end{bmatrix}, \quad B = \begin{bmatrix}dt \\ 1\end{bmatrix}, \quad H = \begin{bmatrix}1 & 0\end{bmatrix}. \tag{A.1}$$

Rozwiązując to analitycznie, można uprościć poszczególne fazy obliczeń (tabela A.1).

Tabela. A.1 Fazy obliczeń dla F-K (rotacja w osi OX, OY)

Faza predykcji	
$x_k^- = A\hat{x}_{k-1}^- + Bu$	$\theta = \theta + u \cdot dt$ $\dot{\theta} = u$
$\hat{P}_k^- = A\hat{P}_{k-1}A^T + Q$	$p_{11} = p_{11} + q$
Faza korekcji	
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$	$k_1 = p_{11} \cdot \frac{1}{p_{11}+r}$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$	$\theta = \theta + k_1(z - \theta)$
$P_k = (I - K_K H)P_k^-$	$p_{11} = (1 - k_1)p_{11}$

Filtr F-K dla rotacji w osi OZ:

$$x = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}, \quad A = \begin{bmatrix}1 & dt \\ 0 & 1\end{bmatrix}, \quad B = \begin{bmatrix}0 \\ 0\end{bmatrix}, \quad H = \begin{bmatrix}0 & 1\end{bmatrix}. \tag{A.2}$$

Podobnie w tym przypadku, rozwiązując analitycznie równania w kolejnych fazach filtru, można przyspieszyć proces obliczeń (tabela A.2), jednak nie jest to już tak efektywne jak w przypadku poprzednim.

Tabela. A.2 Fazy obliczeń dla F-K (rotacja w osi OZ)

Faza predykcji	
$x_k^- = A\hat{x}_{k-1}^- + Bu$	$\theta = \theta + \dot{\theta}dt$ $\dot{\theta} = \dot{\theta}$
$\hat{P}_k^- = A\hat{P}_{k-1}A^T + Q$	$p_{11} = p_{11} + p_{21}dt + (p_{12} + p_{22})dt + q$ $p_{12} = p_{12} + p_{22}dt$ $p_{21} = p_{21} + p_{22}dt$ $p_{22} = p_{22} + q$
Faza korekcji	
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$	$k_1 = p_{12} \cdot \frac{1}{p_{22}+r}$ $k_2 = p_{22} \cdot \frac{1}{p_{22}+r}$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$	$\theta = \theta + k_1(z - \dot{\theta})$ $\dot{\theta} = \dot{\theta} + k_2(z - \dot{\theta})$
$P_k = (I - K_K H)P_k^-$	$p_{11} = p_{11} - k_1 p_{21}$ $p_{12} = p_{12} - (1 - k_2)p_{22}$ $p_{21} = (1 - k_2)p_{21}$ $p_{22} = (1 - k_2)p_{22}$

W obu przypadkach A.1, A.2 sterowanie u oraz pomiar r wyznaczane są według wzorów przedstawionych w rozdziale 5.1.5.

A.2.1 Przykład implementacji w języku ANSI C

Zastosowane struktury danych:

```
typedef signed long int int32_t;
typedef int32_t fixint;

typedef struct {
        fixint ax,ay,az,wx,wy,wz;
} SensorsState;

typedef struct {
        fixint p[2][2], k[2], x[2], q, r;
} KalmanStateR2;

typedef struct{
        fixint rot[3], pos[3];
        fixint sx,sy,sz,cx,cy,cz;
} State;
```

Typ `fixint` używany jest do operacji stało–przecinkowych opisanych w dodatku A.1. Struktura `SensorsState` przechowuje aktualny stan sensorów. To na niej dokonywane jest skalowanie, kompensacja oraz filtracja cyfrowa (filtr opóźniający). Struktura `KalmanStateR2` przechowuje aktualne wartości parametrów F-K, natomiast `State` aktualny stan robota (rotację i pozycję). Dodatkowo w strukturze tej umieszczono pola przechowujące wartości funkcji `sin` i `cos` dla kątów rotacji. Zabieg ten ma na celu przyspieszenie obliczeń. Obliczanie funkcji `sin` i `cos`, pomimo ich stablicowania, jest czasochłonne, a są one wykorzystywane przy wielu przeliczeniach. Dzięki umieszczeniu ich w tej strukturze można liczyć je tylko raz dla danej iteracji.

Funkcja inicjalizująca strukturę `KalmanStateR2`:

```
void  KalmanR2_initialize(KalmanStateR2 *ks, float q, float r)
{
        ks->p[0][0] = ks->p[0][1] = ks->p[1][0] = ks->p[1][1] = 0;
        ks->k[0] = ks->k[1] = 0;
        ks->x[0] = ks->x[1] = 0;
        ks->r = FL2FX(r);
        ks->q = FL2FX(q);
}
```

Funkcja uaktualniająca F-K:

```
void KalmanR2_update(KalmanStateR2 *ks, fixint u, fixint z)
{
        static const fixint dt = FL2FX(INS_SAMPLE_TIME);
        /* PREDYKCJA */

        /* x = Ax + Bu */

        ks->x[0] = FXADD( ks->x[0], FXMUL(u, dt ) );
        ks->x[1] = u;

        /* P = APA^T + Q */

        ks->p[0][0] = FXADD( ks->p[0][0], ks->q );

        /* KOREKCJA */

        /* K = PH^T(HPH^T+R)^-1 */

        ks->k[0] = FXDIV( ks->p[0][0], FXADD( ks->p[0][0], ks->r ) );

        /* x = x+ K(z-Hx) */

        ks->x[0] = FXADD( ks->x[0], FXMUL( ks->k[0], FXSUB( z, ks->x[0] ) ) );

        /* P = (I-KH)P */

        ks->p[0][0] = FXMUL( FXSUB( INT2FX(1), ks->k[0] ), ks->p[0][0] );
}
```

Przedstawiono jedynie implementację F-K dla rotacji OX, OY. Wszystkie operacje dodawania, mnożenia itp. wykonywane są z użyciem makr preprocesora dla operacji stało–przecinkowych (omówiono je w rozdziale A.1). Funkcja ta uruchamiana jest w przerwaniu podczas aktualizacji stanu robota. Dzięki zastosowaniu operacji stało–przecinkowych oraz tablicowaniu funkcji `sin` i `cos` udało się zredukować czas aktualizacji stanu robota do około 2ms. Na aktualizację stanu robota składa się komplet operacji związanych z określeniem rotacji oraz pozycji robota. Czas, co jaki wywoływane jest przerwanie uaktualniające stan robota, ustawiono na 4.88ms, z czego na aktualizację stanu potrzeba około 2ms, natomiast reszta przeznaczona jest na operacje związane z kroczeniem robota i komunikację.

Przykład funkcji uruchamianej w przerwaniu, uaktualniającym stan robota:

```
void INS_update()
{
        static SensorsState ss;
        static State s;
        INS_read_sensors(&ss);
        INS_dealy_filter_update(&ss);
        INS_scale(&ss);
        INS_temp_comp(&ss);
```

```
        INS_calibration_offset(&ss);
        INS_calculate_sincos(&s);
        INS_rotation_update(&s, &ss);
        INS_position_update(&s, &ss);
        INS_update_registers(&s);
}
```

A.3 Tablica rejestrów lokalnego sterownika nogi

Tabela. A.3 Tablicza rejestrów lokalnego sterownika nogi

Nazwa	Adres	Typ	Opis
REG_STATUS	0x00	R	Rejestr Statusowy. Znaczenie bitów: bit 0 – przyjmuje wartość 1, gdy noga ma kontakt z podłożem, bit 1 – przyjmuje wartość 1, gdy zadana pozycja jest poza zasięgiem, bit 2 – przyjmuje wartość 1, gdy osiągnięto zadaną pozycję.
REG_TICTOCK	0x01	X	Rejestr licznika, używany w wewnętrznej implementacji.
REG_X_H	0x02	R	Aktualna pozycja stopy wsp. X (LSB).
REG_X_L	0x03	R	Aktualna pozycja stopy wsp. X (MSB).
REG_Y_H	0x04	R	Aktualna pozycja stopy wsp. Y (LSB).
REG_Y_L	0x05	R	Aktualna pozycja stopy wsp. Y (MSB).
REG_Z_H	0x06	R	Aktualna pozycja stopy wsp. Z (LSB).
REG_Z_L	0x07	R	Aktualna pozycja stopy wsp. Z (MSB).
REG_PGAIN	0x08	R/W	Wzmocnienie rególatora proporcjonalnego. Wartość domyślna 9.
REG_DEADBAND	0x09	R/W	Stopień nieczułości rególatora. Wartość domyślna 10.
REG_POS0_L	0x0a	R	Aktualna wartość kąta wychylenia segmentu $q1$ (LSB). Wartość z dokładnością enkodera (12–bitów na obrót).
REG_POS0_H	0x0b	R	Aktualna wartość kąta wychylenia segmentu $q1$ (MSB).
REG_POS1_L	0x0c	R	Aktualna wartość kąta wychylenia segmentu $q2$ (LSB).
REG_POS1_H	0x0d	R	Aktualna wartość kąta wychylenia segmentu $q2$ (MSB).
REG_POS2_L	0x0e	R	Aktualna wartość kąta wychylenia segmentu $q3$ (LSB).
REG_POS2_H	0x0f	R	Aktualna wartość kąta wychylenia segmentu $q3$ (MSB).
REG_SEEK_POS0_L	0x10	X	Zadana wartość kąta wychylenia segmentu $q1$ (LSB). Rejestr używany przez generator trajektroii.
REG_SEEK_POS0_H	0x11	X	Zadana wartość kąta wychylenia segmentu $q1$ (MSB). Rejestr używany przez generator trajektroii.
REG_SEEK_POS1_L	0x12	X	Zadana wartość kąta wychylenia segmentu $q2$ (LSB). Rejestr używany przez generator trajektroii.
REG_SEEK_POS1_H	0x13	X	Zadana wartość kąta wychylenia segmentu $q2$ (MSB). Rejestr używany przez generator trajektroii.
REG_SEEK_POS2_L	0x14	X	Zadana wartość kąta wychylenia segmentu $q3$ (LSB). Rejestr używany przez generator trajektroii.
REG_SEEK_POS2_H	0x15	X	Zadana wartość kąta wychylenia segmentu $q3$ (MSB). Rejestr używany przez generator trajektroii.
REG_ENCODERS_TEST	0x1f	R/W	Test enkoderów, gdy wartość = 1 serwa ustwiane są w położeniu domyślnym. Można wykorzystać do kalibracji magnesów.
REG_START	0x20	R/W	Start sterownika. Wartość domyślna 0x00. Sterownik po włączeniu zaslania czeka na wpisanie do rejestru wartości 0x01. Wpisanie wartości 0x00 po prawidłowym starcie sterownika spowoduje wyłączenie silników (PWM = 0).

A.4 Tablica rejestrów sterownika głównego robota

Tabela. A.4 Tablicza rejestrów 8–bitowych sterownika głównego robota

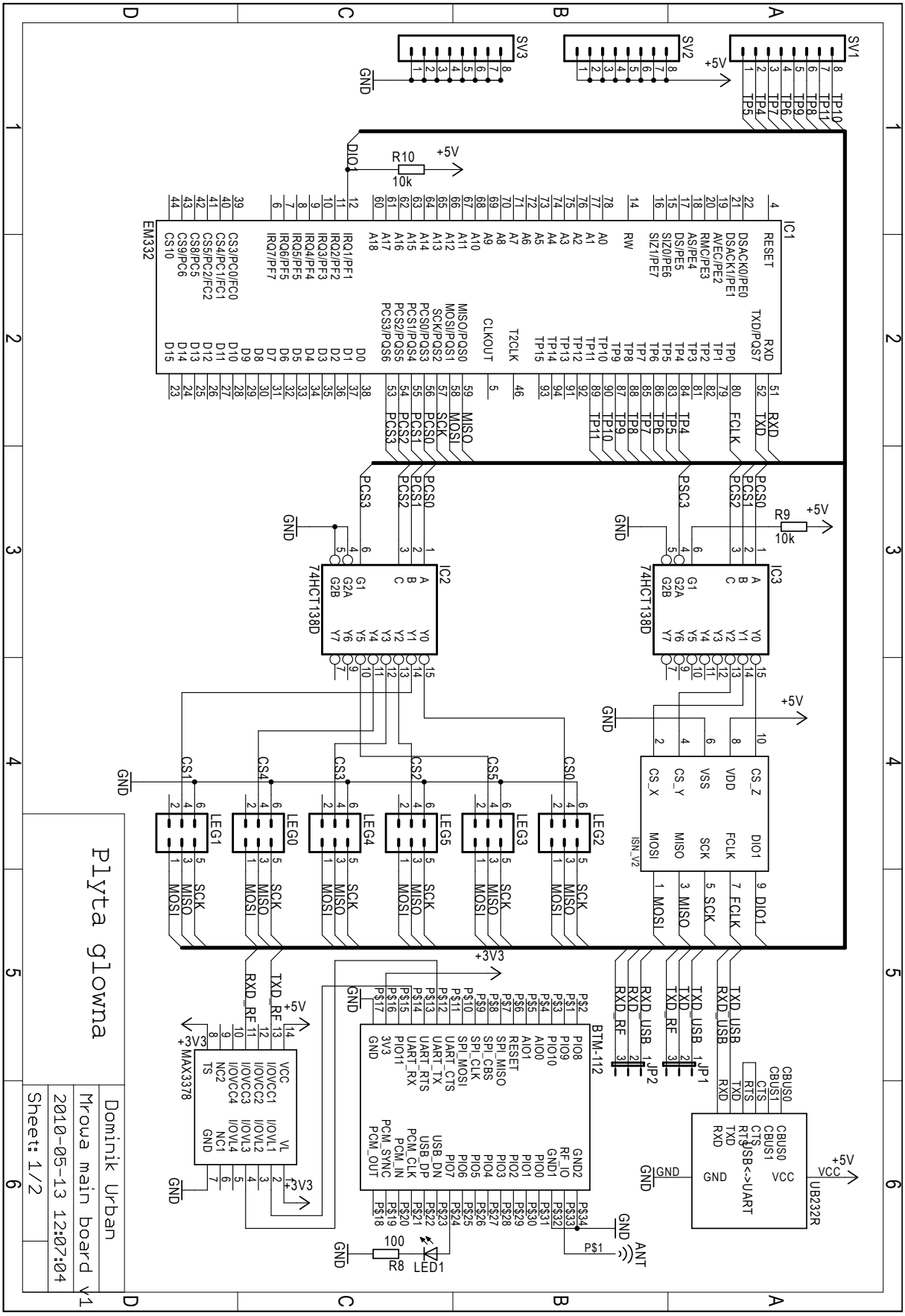
Nazwa	Adres	Typ	Opis
REG_V_X	0x00	R/W	Prędkość kroczenia w osi X [mm/s].
REG_V_Y	0x01	R/W	Prędkość kroczenia w osi Y [mm/s].
REG_V_YAW	0x02	R/W	Prędkość oborotu względem osi Z [stopnie/s].
REG_ROLL	0x03	R/W	Kąt przechyłu w osi X [stopnie].
REG_PITCH	0x04	R/W	Kąt przechyłu w osi Y [stopnie].
REG_OFFSET_Z	0x05	R/W	Wysokość korpusu [mm].
REG_OFFSET_UP	0x06	R/W	Wysokość podnoszenia nóg [mm].
REG_GROUNDED_LEGS	0x07	R	Nogi mające kontak z podłożem gdy bit = 1, bit 0 noga nr. 0 ... bit 5 noga nr. 5.
REG_LEG0_STATUS	0x08	R	Rejestr statusowy sterownika lokalnego nogi 0.
REG_LEG1_STATUS	0x09	R	Rejestr statusowy sterownika lokalnego nogi 1.
REG_LEG2_STATUS	0x0a	R	Rejestr statusowy sterownika lokalnego nogi 2.
REG_LEG3_STATUS	0x0b	R	Rejestr statusowy sterownika lokalnego nogi 3.
REG_LEG4_STATUS	0x0c	R	Rejestr statusowy sterownika lokalnego nogi 4.
REG_LEG5_STATUS	0x0d	R	Rejestr statusowy sterownika lokalnego nogi 5.
REG_EMERGENCY_STOP	0x10	W	Awaryjne zatrzymanie robota gdy wartość równa 1.

Tabela. A.5 Tablicza rejestrów 16–bitowych sterownika głównego robota

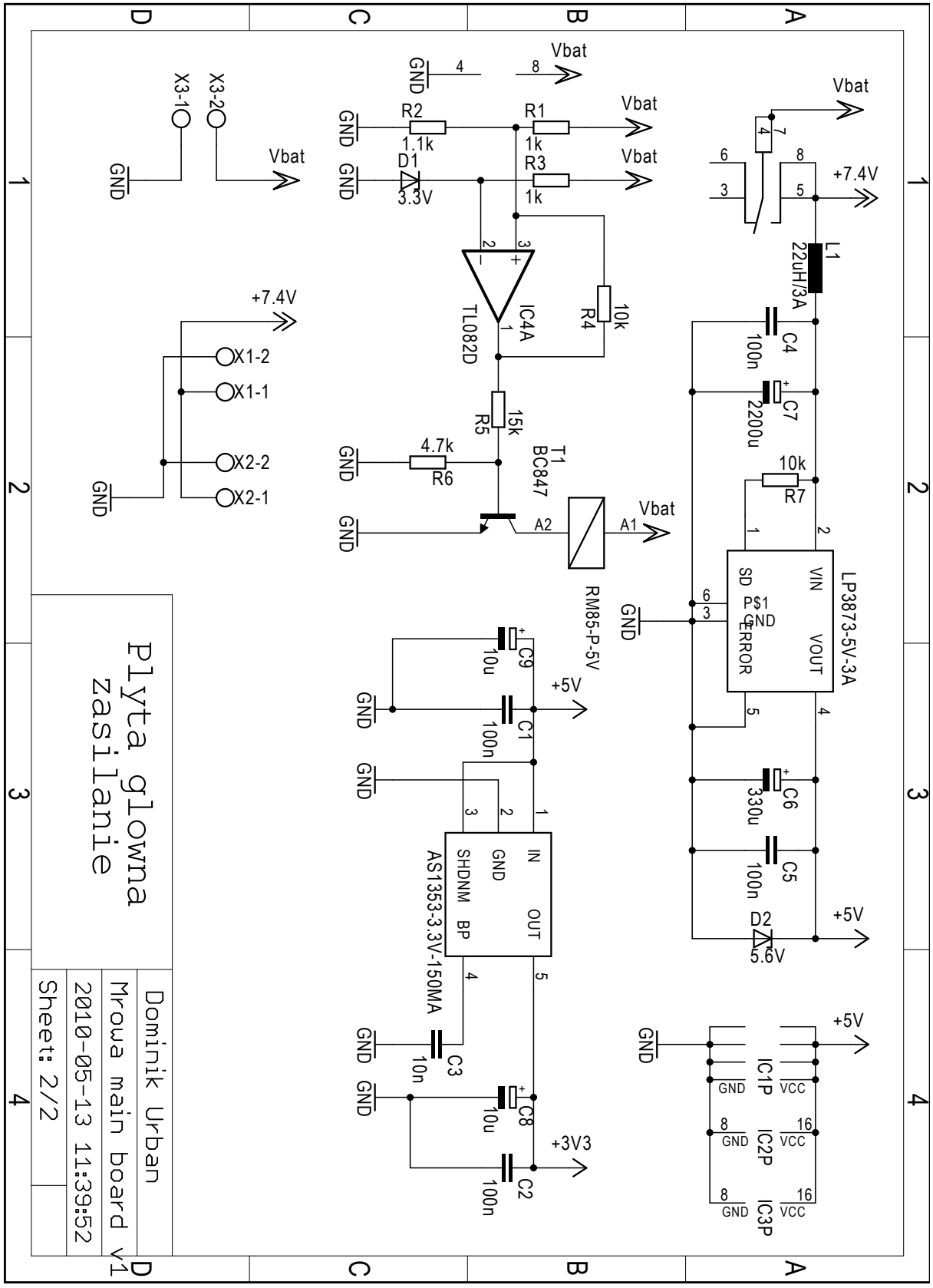
Nazwa	Adres	Typ	Opis
REG_POSITION_X0	0x00	R	Współrzędna X nogi 0 [mm].
REG_POSITION_X1	0x01	R	Współrzędna X nogi 1 [mm].
REG_POSITION_X2	0x02	R	Współrzędna X nogi 2 [mm].
REG_POSITION_X3	0x03	R	Współrzędna X nogi 3 [mm].
REG_POSITION_X4	0x04	R	Współrzędna X nogi 4 [mm].
REG_POSITION_X5	0x05	R	Współrzędna X nogi 5 [mm].
REG_POSITION_Y0	0x06	R	Współrzędna Y nogi 0 [mm].
REG_POSITION_Y1	0x07	R	Współrzędna Y nogi 1 [mm].
REG_POSITION_Y2	0x08	R	Współrzędna Y nogi 2 [mm].
REG_POSITION_Y3	0x09	R	Współrzędna Y nogi 3 [mm].
REG_POSITION_Y4	0x0a	R	Współrzędna Y nogi 4 [mm].
REG_POSITION_Y5	0x0b	R	Współrzędna Y nogi 5 [mm].
REG_POSITION_Z0	0x0c	R	Współrzędna Z nogi 0 [mm].
REG_POSITION_Z1	0x0d	R	Współrzędna Z nogi 1 [mm].
REG_POSITION_Z2	0x0e	R	Współrzędna Z nogi 2 [mm].
REG_POSITION_Z3	0x0f	R	Współrzędna Z nogi 3 [mm].
REG_POSITION_Z4	0x10	R	Współrzędna Z nogi 4 [mm].
REG_POSITION_Z5	0x11	R	Współrzędna Z nogi 5 [mm].
REG_X_MEASURE	0x12	R	Pozycja robota wsp. X [mm].
REG_Y_MEASURE	0x13	R	Pozycja robota wsp. Y [mm].
REG_Z_MEASURE	0x14	R	Pozycja robota wsp. Z [mm].
REG_YAW_MEASURE	0x15	R	Orientacja robota [rad] (dokładność 12 bitów).
REG_ROLL_MEASURE	0x16	R	Rotacja w osi X [rad] (dokładność 12 bitów).
REG_PITCH_MEASURE	0x17	R	Potacja w osi Y [rad] (dokładność 12 bitów).

Dodatek B

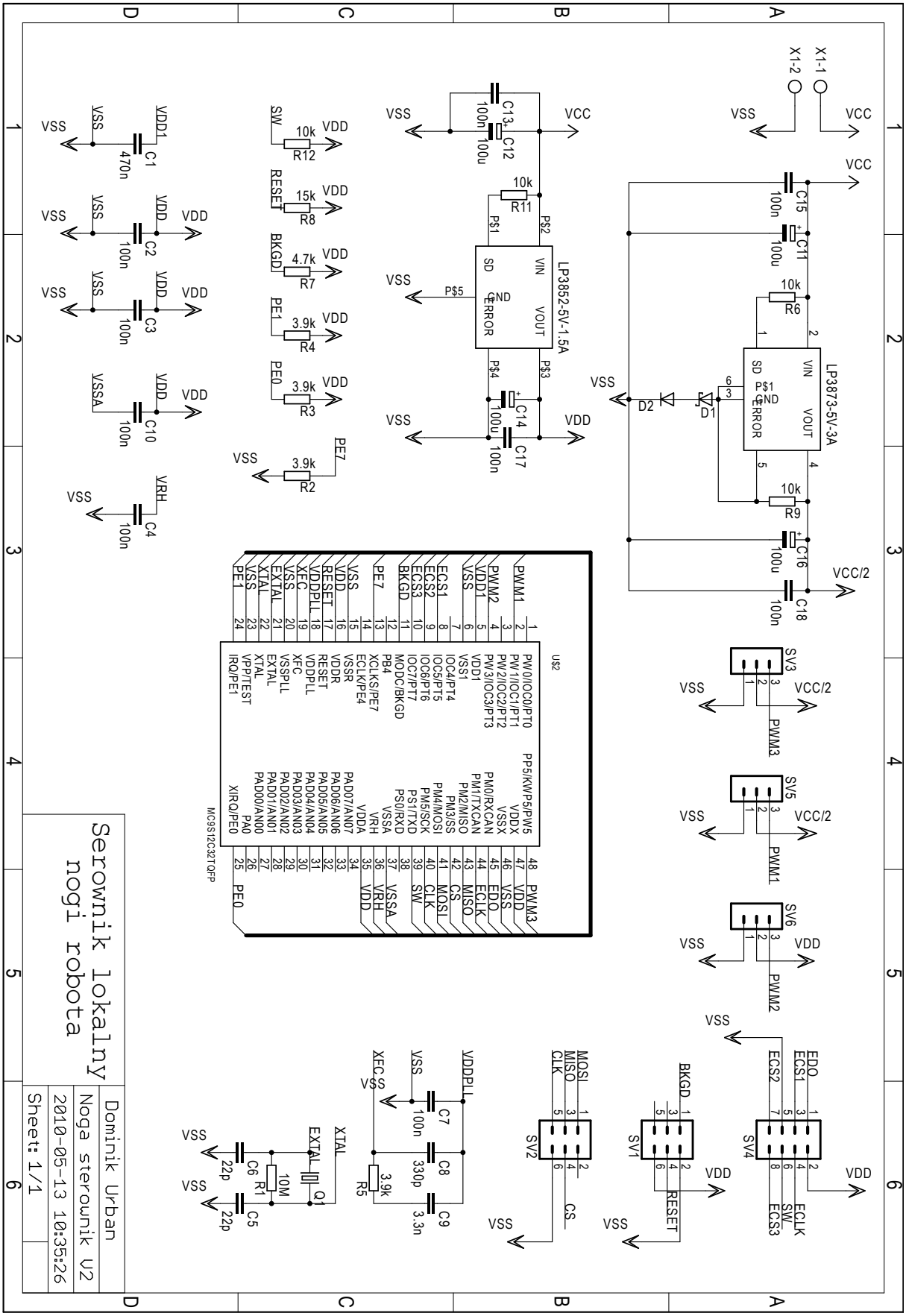
Schematy



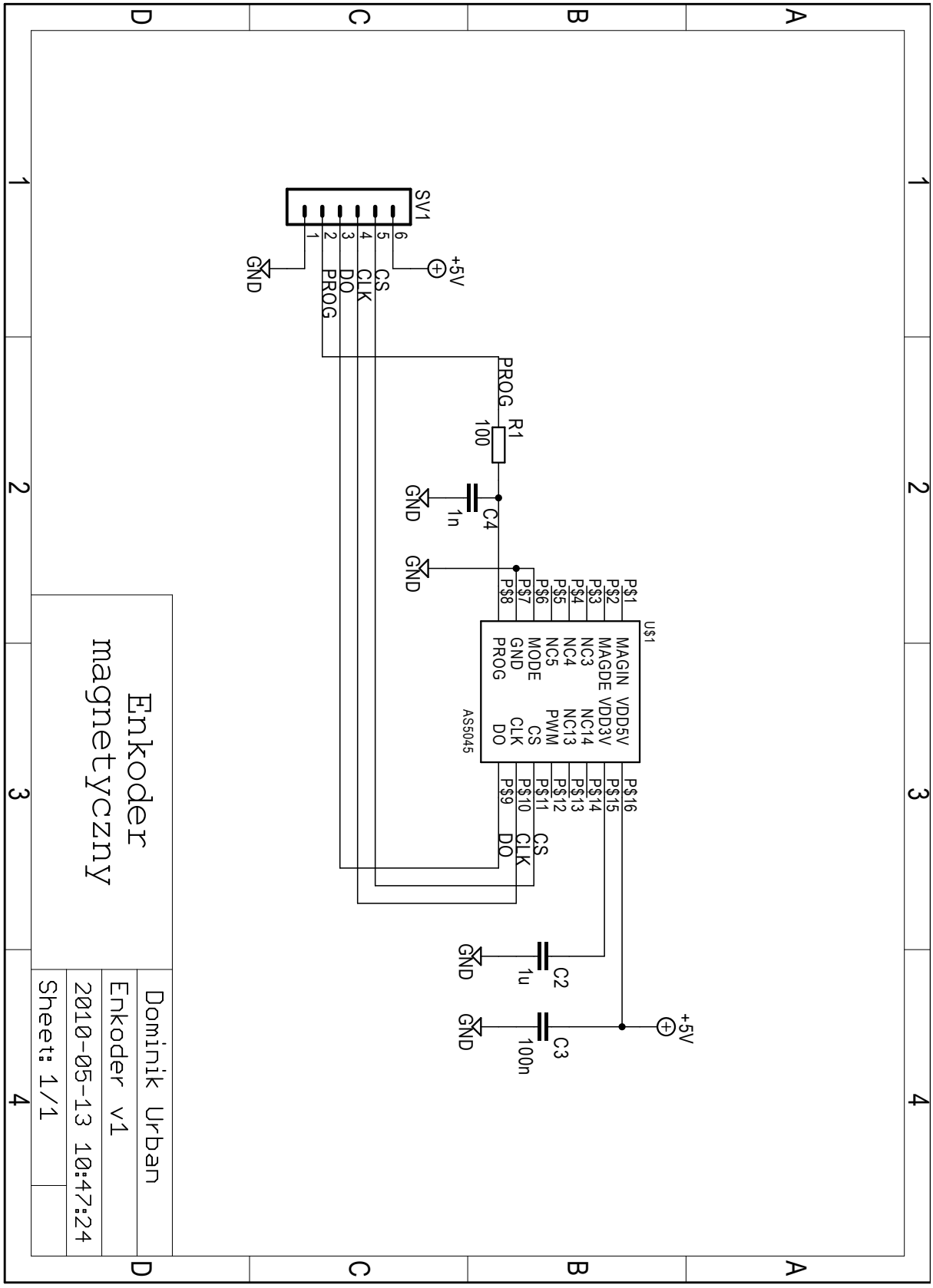
Rysunek B.1 Schemat ideowy płyty głównej robota



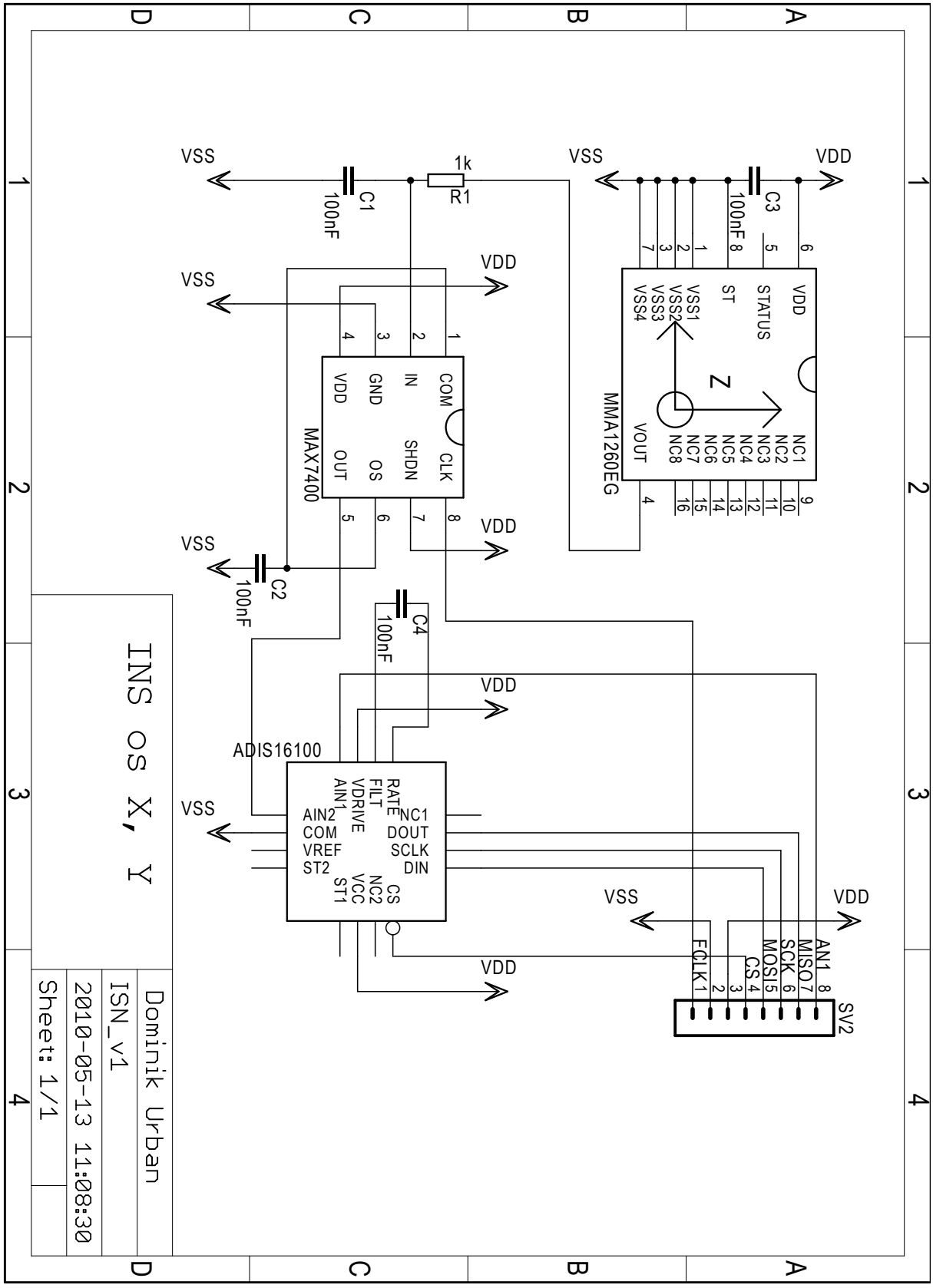
Rysunek B.2 Schemat ideowy płyty głównej robota (zasilanie)



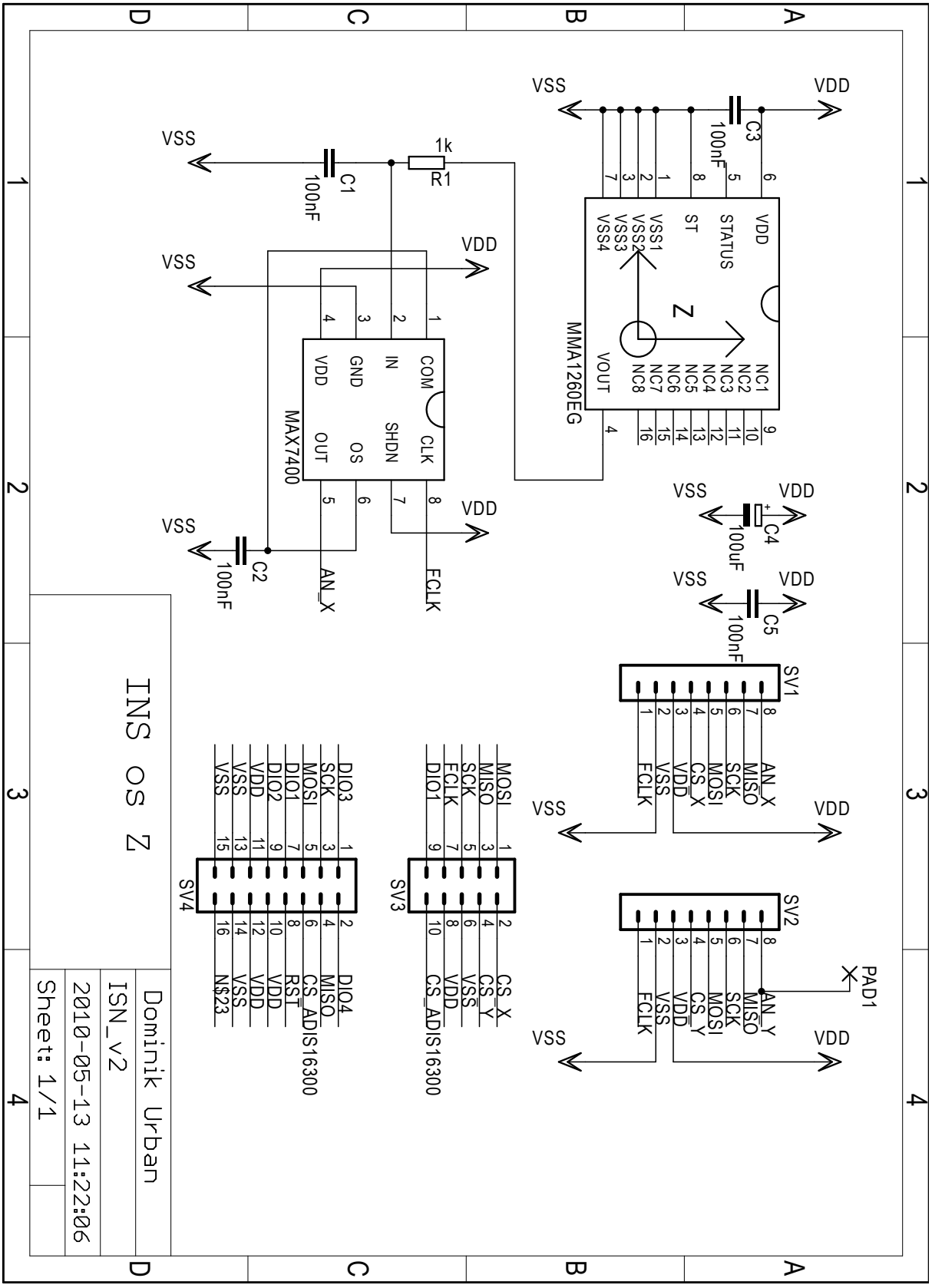
Rysunek B.3 Schemat ideowy sterownika lokalnego nogoi robota



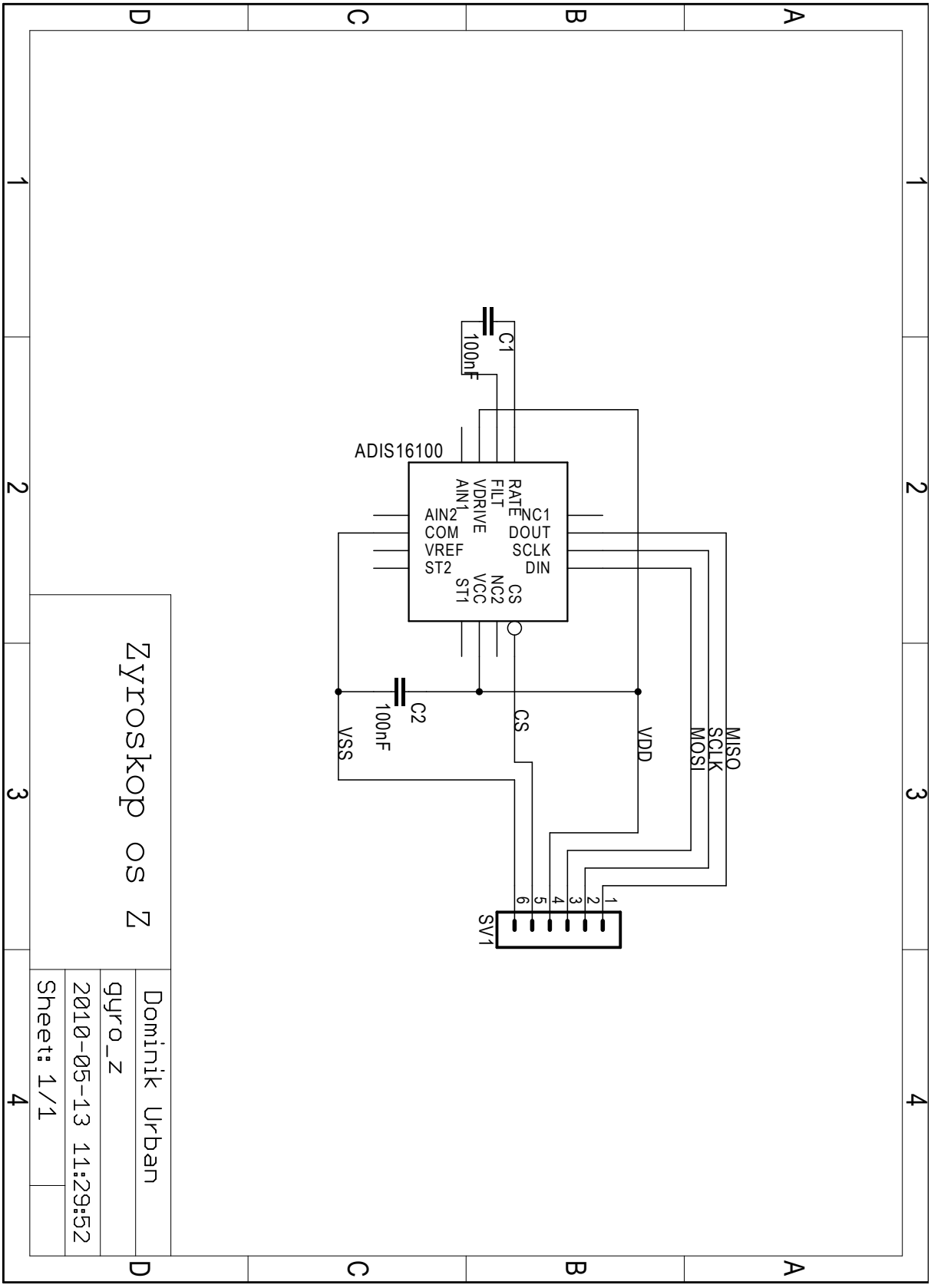
Rysunek B.4 Schemat ideowy enkodera magnetycznego



Rysunek B.5 Schemat ideowy modułu INS, płytki poionowe w osi x, y



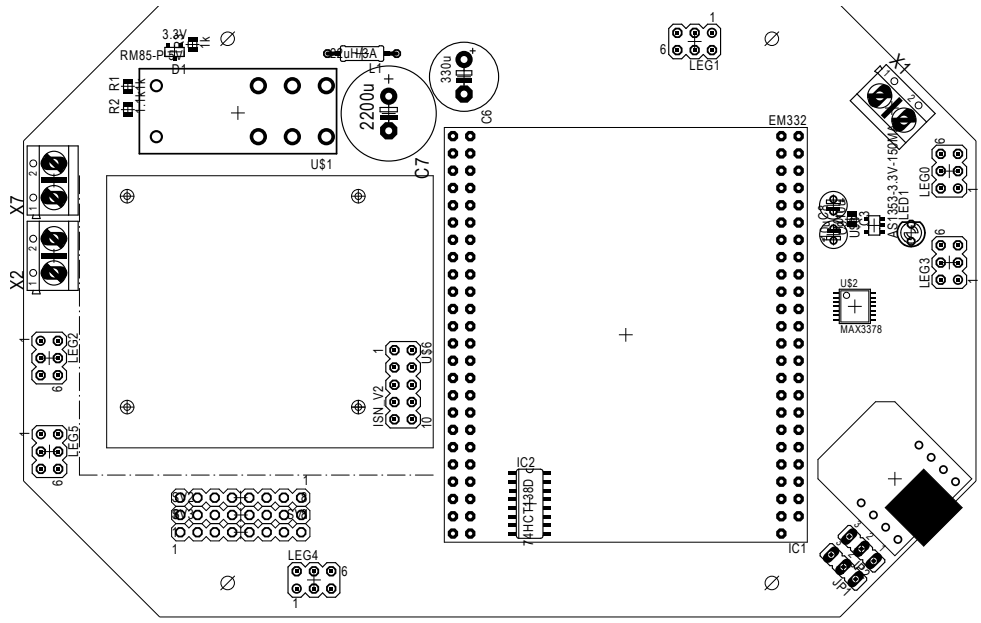
Rysunek B.6 Schemat ideowy modułu INS



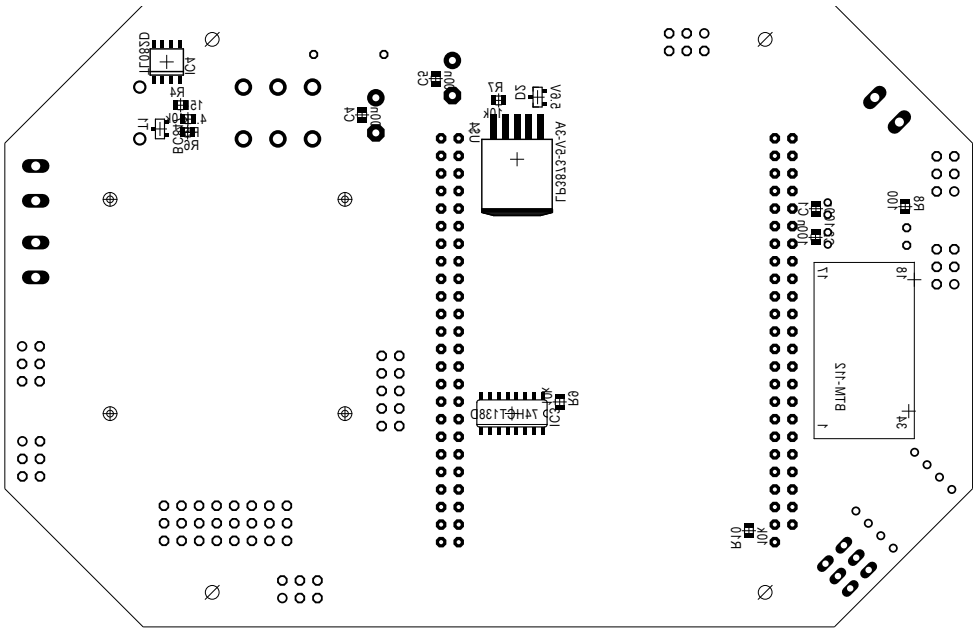
Rysunek B.7 Schemat ideowy żyroskopu w osi Z

Dodatek C

Rozmieszczenie elementów na płytkach

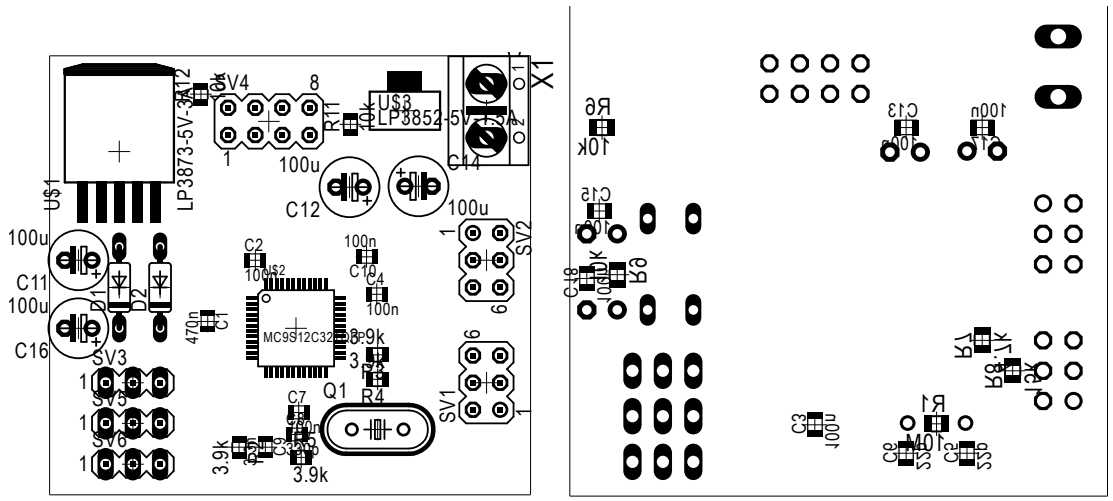


(a) widok z góry



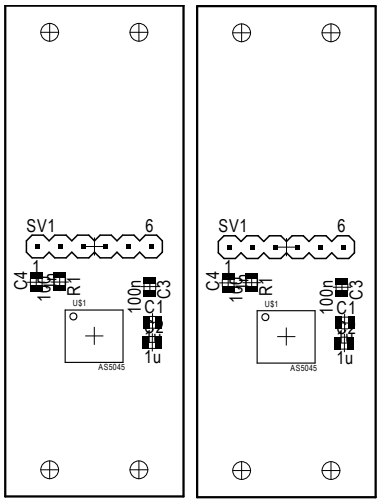
(b) widok z dołu

Rysunek C.1 Rozmieszczenie elementów na płycie głównej robota



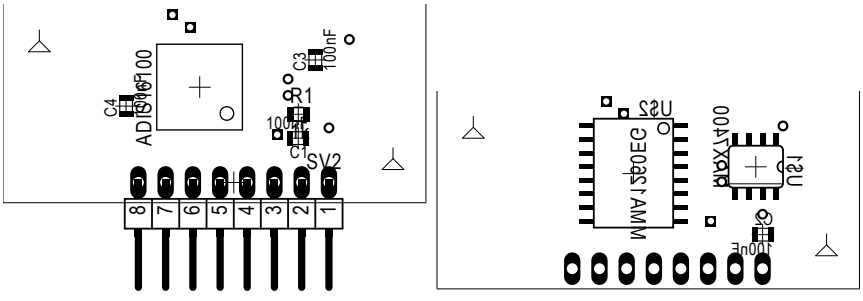
(a) widok z góry (b) widok z dołu

Rysunek C.2 Rozmieszczenie elementów na płycie sterownika lokalnego nogi robota



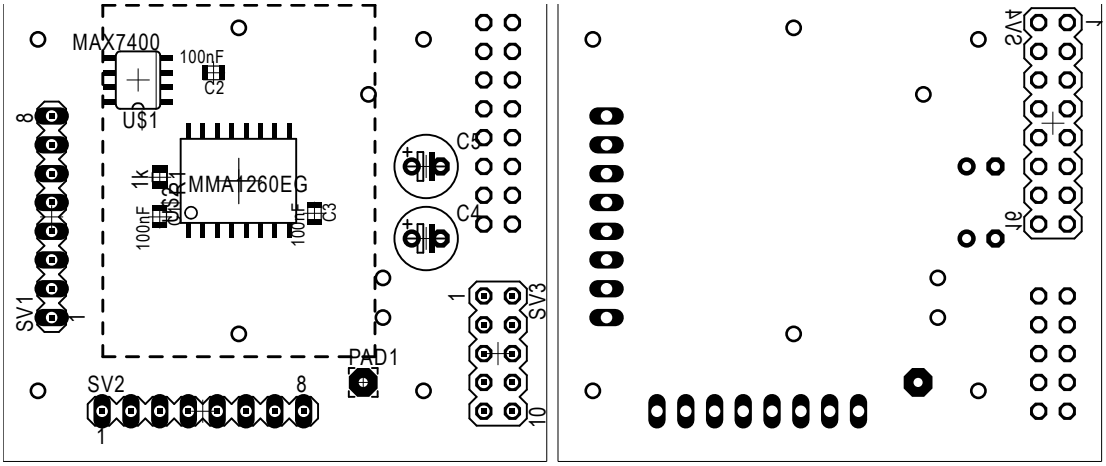
(a) płytka wersja 1 (b) płytka wersja 2

Rysunek C.3 Rozmieszczenie elementów na płycie enkodera magnetycznego



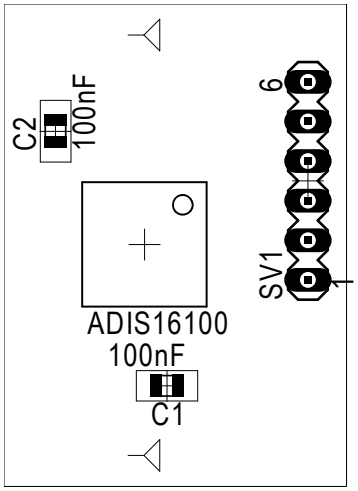
(a) widok z góry (b) widok z dołu

Rysunek C.4 Rozmieszczenie elementów na płycie modułu INS (os x, y)



(a) widok z góry (b) widok z dołu

Rysunek C.5 Rozmieszczenie elementów na płycie modułu INS



Rysunek C.6 Rozmieszczenie elementów na płycie żyroskopu w osi z

Tabela. C.1 Spis elementów płyty sterownika głównego

Element	Wartość	Opis	Obudowa
C1	100n	C-EUC0805	C0805
C2	100n	C-EUC0805	C0805
C3	10n	C-EUC0805	C0805
C4	100n	C-EUC0805	C0805
C5	100n	C-EUC0805	C0805
C6	330u	CPOL-EUE5-10.5	E5-10,5
C7	2200u	CPOL-EUE5-13	E5-13
C8	10u	CPOL-EUE1.8-4	E1,8-4
C9	10u	CPOL-EUE1.8-4	E1,8-4
D1	3.3V	ZENER-DIODESOT23	SOT23
D2	5.6V	ZENER-DIODESOT23	SOT23
IC1	EM332	EM332	EM332
IC2	74HCT138D	74HCT138D	SO16
IC3	74HCT138D	74HCT138D	SO16
IC4	TL082D	TL082D	SO08
JP1		JP2E	JP2
JP2		JP2E	JP2
L1	22uH/3A	L-EU0207/10	0207/10
LED1		LED3MM	LED3MM
LEG0		MA03-2	MA03-2
LEG1		MA03-2	MA03-2
LEG2		MA03-2	MA03-2
LEG3		MA03-2	MA03-2
LEG4		MA03-2	MA03-2
LEG5		MA03-2	MA03-2
R1	1k	R-EU_R0805	R0805
R2	1.1k	R-EU_R0805	R0805
R3	1k	R-EU_R0805	R0805
R4	10k	R-EU_R0805	R0805
R5	15k	R-EU_R0805	R0805
R6	4.7k	R-EU_R0805	R0805
R7	10k	R-EU_R0805	R0805
R8	100	R-EU_R0805	R0805
R9	10k	R-EU_R0805	R0805
R10	10k	R-EU_R0805	R0805
SV1		MA08-1S	MA08-1S
SV2		MA08-1S	MA08-1S
SV3		MA08-1S	MA08-1S
T1	BC847	BC847	SOT23
U$1	RM85-P-5V	RM85-P-5V	RM85
U$2	MAX3378	MAX3378	TSSOP14
U$3	AS1353-3.3V-150MA	AS1353-3.3V-150MA	SOT23-5L
U$4	LP3873-5V-3A	LP3873-5V-3A	TO263-5
U$5	BTM-112	Bluetooth	BTM-112
U$6	ISN_V2	ISN_V2	MA05-2
U$7	UB232R	UB232R	FTDI_BOARD
U$8	ANTENA	ANTENA	ANTENA
X1		AK500/2	AK500/2
X2		AK500/2	AK500/2
X3		AK500/2	AK500/2

Tabela. C.2 Spis elementów płyty sterownik lokalnego nogi

Element	Wartość	Opis	Obudowa
C1	470n	C-EUC0805K	C0805K
C2	100n	C-EUC0805K	C0805K
C3	100n	C-EUC0805K	C0805K
C4	100n	C-EUC0805K	C0805K
C5	22p	C-EUC0805K	C0805K
C6	22p	C-EUC0805K	C0805K
C7	100n	C-EUC0805K	C0805K
C8	330p	C-EUC0805K	C0805K
C9	3.3n	C-EUC0805K	C0805K
C10	100n	C-EUC0805K	C0805K
C11	100u	CPOL-EUE2.5-6	E2,5-6
C12	100u	CPOL-EUE2.5-6	E2,5-6
C13	100n	C-EUC0805K	C0805K
C14	100u	CPOL-EUE2.5-6	E2,5-6
C15	100n	C-EUC0805K	C0805K
C16	100u	CPOL-EUE2.5-6	E2,5-6
C17	100n	C-EUC0805K	C0805K
C18	100n	C-EUC0805K	C0805K
D1		SCHOTTKY-DIODEDO35-7	DO35-7
D2		1N4148DO35-7	DO35-7
Q1		CRYTALHC49U-V	HC49U-V
R1	10M	R-EU_R0805	R0805
R2	3.9k	R-EU_R0805	R0805
R3	3.9k	R-EU_R0805	R0805
R4	3.9k	R-EU_R0805	R0805
R5	3.9k	R-EU_R0805	R0805
R6	10k	R-EU_R0805	R0805
R7	4.7k	R-EU_R0805	R0805
R8	15k	R-EU_R0805	R0805
R9	10k	R-EU_R0805	R0805
R11	10k	R-EU_R0805	R0805
R12	10k	R-EU_R0805	R0805
SV1		MA03-2	MA03-2
SV2		MA03-2	MA03-2
SV3		MA03-1	MA03-1
SV4		MA04-2	MA04-2
SV5		MA03-1	MA03-1
SV6		MA03-1	MA03-1
U$1	LP3873-5V-3A	LP3873-5V-3A	TO263-5
U$2	MC9S12C32TQFP	MC9S12C32TQFP	TQFP48
U$3	LP3852-5V-1.5A	LP3852-5V-1.5A	SOT223
X1		AK500/2	AK500/2

Tabela. C.3 Spis elementów płyty INS os Z

Element	Wartość	Opis	Obudowa
C1	100nF	C-EUC0805K	C0805K
C2	100nF	C-EUC0805K	C0805K
C3	100nF	C-EUC0805K	C0805K
C4	100uF	CPOL-EUE2.5-6	E2,5-6
C5	100nF	C-EUC0805K	C0805K
PAD1		2,54/0,8	2,54/0,8
R1	1k	R-EU_R0805	R0805
SV1		MA08-1	MA08-1
SV2		MA08-1	MA08-1
SV3		MA05-2	MA05-2
SV4		MA08-2	MA08-2
U$1	MAX7400	MAX7400	SO08
U$2	MMA1260EG	MMA1260EG	SO16-1

Tabela. C.4 Spis elementów płyty INS os X, Y

Element	Wartość	Opis	Obudowa
C1	100nF	C-EUC0805K	C0805K
C2	100nF	C-EUC0805K	C0805K
C3	100nF	C-EUC0805K	C0805K
C4	100nF	C-EUC0805K	C0805K
R1	1k	R-EU_R0805	R0805
SV2		MA08-1W	MA08-1W
U$1	MAX7400	MAX7400	SO08
U$2	MMA1260EG	MMA1260EG	SO16-1
U$3	ADIS16100	ADIS16100	GYROPCB

Tabela. C.5 Spis elementów płyty żyroskopu w osi Z

Element	Wartość	Opis	Obudowa
C1	100nF	C-EUC0805	C0805
C2	100nF	C-EUC0805	C0805
SV1		MA06-1	MA06-1
U$1	ADIS16100	ADIS16100	GYROPCB

Tabela. C.6 Spis elementów płyty enkodera magnetycznego

Element	Wartość	Opis	Obudowa
C2	1u	C-EUC0805	C0805
C3	100n	C-EUC0805	C0805
C4	1n	C-EUC0805	C0805
R1	100	R-EU_R0805	R0805
SV1		MA06-1	MA06-1
U$1	AS5045	AS5045	SSOP16

Dodatek D

Opis wyprowadzeń i złączy

Tabela. D.1 Opis złączy płyty głównej

Złącze (płyta gł.)	Złącze (połączenie)	Opis
INS_V2	INS os z (SV3)	złącze modułu INS
1	1	wyjście danych MOSI (SPI)
2	2	Chip – Select żyroskopu w osi X
3	3	wejście danych MISO (SPI)
4	4	Chip – Select żyroskopu w osi Y
5	5	sygnał taktujący SCK (SPI)
6	6	masa zasilania
7	7	sygnał taktujący filtry analogowe
8	8	plus zasilania (+5V)
9	9	sygnał przerwania (nie użyte)
10	10	Chip – Select żyroskopu w osi Z
LEG_0 ... LEG_5	Sterownik lokalny 1... 6(SV2)	złącze danych nogi robota
1	1	wyjście danych MOSI (SPI)
2	2	—
3	3	wejście danych MISO (SPI)
4	4	Chip – Select
5	5	sygnał taktujący SCK (SPI)
6	6	masa zasilania
SV1	—	wyjścia kanałów TPU
1 ... 8	—	wyjście kanału TPU ogólnego przeznaczenia
SV2	—	wyjścia plusa zalilania (+5V)
1 ... 8	—	wyjście zasilania +5V ogólnego przeznaczenia
SV2	—	wyjścia masy zalilania
1 ... 8	—	wyjście masy ogólnego przeznaczenia
JP1	—	Zwora wyboru rodzaju komunikacji z PC (TXD)
pozycja 2 – 1(JP2)[1]	—	komunikacja przez USB, aktywne FTDI
pozycja 2 – 3	—	komunikacja przez Bluetooth
JP2	—	Zwora wyboru rodzaju komunikacji z PC (RXD)
pozycja 2 – 1(JP1)[2]	—	komunikacja przez USB, aktywne FTDI
pozycja 2 – 3	—	komunikacja przez Bluetooth
X3	bateria	złącze zasilania (bateria Li–Pol 7.4V)
1	masa	masa zasilania
2	plus	plus zasilania
X1	Sterownik lokalny 1... 3(X1)	złącze zasilania dla nóg robota
1	1	masa zasilania
2	2	plus zasilania
X2	Sterownik lokalny 4... 6(X1)	złącze zasilania dla nóg robota
1	1	masa zasilania
2	2	plus zasilania

Tabela. D.2 Opis złączy sterownika lokalnego nogi robota

Złącze (płyta)	Złącze (połączenie)	Opis
X1	Płyta główna (X1, X2)	złącze zasilania
1	1	masa zasilania
2	2	plus zasilania (7.4V)
SV1	—	złacze programatora
1	—	BKGD
2	—	masa zasilania
3	—	—
4	—	RESET
5	—	—
6	—	plus zasilania (+5V)
SV2	Płyta główna (LEG_0...LEG_5)	złącze danych
1	1	wyjście danych MOSI (SPI)
2	2	—
3	3	wejście danych MISO (SPI)
4	4	Chip – Select
5	5	sygnał taktujący SCK (SPI)
6	6	masa zasilania
SV3	serwo	złacze serwa segmentu q_2
1	—	masa zasilania
2	—	plus zasilania (6V)
3	—	sygnał sterujący PWM
SV5	serwo	złacze serwa segmentu q_3
1	—	masa zasilania
2	—	plus zasilania (6V)
3	—	sygnał sterujący PWM
SV6	serwo	złacze serwa segmentu q_1
1	—	masa zasilania
2	—	plus zasilania (5V)
3	—	sygnał sterujący PWM
SV4	Enkoder (SV1)	złącze danych
1	3	wejscie danych EDO
2	6	plus zasilania (5V)
3	5	Chip – Select enkodera segmentu q_1
4	4	sygnał taktujący ECKL
5	1	masa zasilania
6	—	czujnik podparcia stopy
3	5	Chip – Select enkodera segmentu q_2
3	5	Chip – Select enkodera segmentu q_3

Tabela. D.3 Opis złączy modułu INS (płytka INS os Z)

Złącze (płyta)	Złącze (połączenie)	Opis
SV3	płyta główna (INS_V2)	złącze danych
1	1	wyjście danych MOSI (SPI)
2	2	Chip – Select żyroskopu w osi X
3	3	wejście danych MISO (SPI)
4	4	Chip – Select żyroskopu w osi Y
5	5	sygnał taktujący SCK (SPI)
6	6	masa zasilania
7	7	sygnał taktujący filtry analogowe
8	8	plus zasilania (+5V)
9	9	sygnał przerwania (nie użyte)
10	10	Chip – Select żyroskopu w osi Z
SV1	płytka INS os X, Y	złącze żyroskopu i akcelerometru osi X
1	1	sygnał taktujący filtry analogowe
2	2	masa zasilania
3	3	plus zasilania (+5V)
4	4	Chip – Select żyroskopu w osi X
5	5	wyjście danych MOSI (SPI)
6	6	sygnał taktujący SCK (SPI)
7	7	wejście danych MISO (SPI)
8	8	wejście przetwornika A/C żyroskopu w osi X
SV2	płytka INS os X, Y	złącze żyroskopu i akcelerometru osi Y
1	1	sygnał taktujący filtry analogowe
2	2	masa zasilania
3	3	plus zasilania (+5V)
4	4	Chip – Select żyroskopu w osi Y
5	5	wyjście danych MOSI (SPI)
6	6	sygnał taktujący SCK (SPI)
7	7	wejście danych MISO (SPI)
8	8	wejście przetwornika A/C żyroskopu w osi Y
SV4	płytka żyroskou w osi z	złącze żyroskopu w osi Z
1,2,7...11, 14 ...16	—	nie użyte, złacze zaprojektowano pierwotnie do innego czujnika (ADIS16100)
3	2	sygnał taktujący SCK (SPI)
4	1	wejście danych MISO (SPI)
5	3	wyjście danych MOSI (SPI)
6	5	Chip – Select żyroskopu w osi Z
12	4	plus zasilania (+5V)
13	6	masa zasilania