POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

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

# PRACA DYPLOMOWA MAGISTERSKA

Robot balansujący - model i sterowanie.

Balancing robot - modeling and control.

AUTOR:
Jan Kędzierski

PROWADZĄCY PRACĘ:
dr inż. Alicja Mazur, I-6

OCENA PRACY:

WROCŁAW 2008

*Pracę tą dedykuje rodzicom oraz przyszłej żonie.*

*Składam serdeczne podziękowania Pani* ***dr inż. Alicji Mazur*** *za pomoc, wyrozumiałość oraz czas poświęcony tej pracy. Chciałem również podziękować* ***dr inż. Markowi Wnukowi*** *za pomoc w uruchomieniu jednostki MPC555 oraz wszelkie uwagi techniczne. Dziękuje także* ***Kołu Naukowemu Robotyków „KoNaR”*** *za wsparcie finansowe oraz zaplecze techniczne udostępnione w trakcie realizacji projektu.*

# Spis treści

# Rozdział 1

# Wstęp

Od kilku lat znacząco wzrosło zainteresowanie robotami balansującymi. Odwrócone wahadło już od dawna służy w uczelnianych laboratoriach jako przykład nieliniowego, niestabilnego systemu dynamicznego. Dzięki implementacji odpowiednich systemów sterowania możliwe jest badanie jego własności. Robot balansujący jest w uproszczeniu niczym innym jak odwróconym, mobilnych wahadłem. W konstrukcji nośnej mocuje się dwa napędy wraz z kołami. To dzięki nim możliwe jest utrzymywanie równowagi oraz przemieszczanie się robota. Idea takiego rozwiązania pozwala lepiej zrozumieć skomplikowane, kroczące układy lokomocji robotów humanoidalnych. Roboty balansujące przyczyniły się także do rozwoju różnego rodzaju platform transportowych. Najpopularniejszą tego typu konstrukcją jest "Segway" - Human Transporter [22]. Doczekał się już swojej drugiej, znacznie bardziej dopracowanej wersji. Rozważa się także zastosowanie balansującego podwozia w wózkach inwalidzkich. Dwukołowe podwozie łatwo pokonuje różne nierówności. Ponadto zarówno sterowanie jak i planowanie ruchu nie wymaga stosowania skomplikowanych algorytmów, gdyż w przypadku tej klasy robotów mobilnych możliwa jest zmiana orientacji w miejscu.

W większości popularnych algorytmów sterowania zakłada się, że wszystkie przeguby są napędzane. W przypadku odwróconego wahadła, odchylenie spowodowane jest działaniem siły grawitacji. Przeciwdziałać temu zjawisku można jedynie poprzez odpowiednie wysterowanie kół. Te same napędy tworzą także układ lokomocji robota. Dostępnych jest wiele prac opisujących systemy sterowania robota utrzymującego równowagę [8], [23]. Sposób postępowania w większości przypadków jest podobny. Matematyczny opis platformy traktuje się jak liniowy obiekt, a następnie implementuje się liniowe sterowniki.

W pracy skupiono się głównie na części symulacyjnej. Wyliczono model dynamiki robota balansującego zarówno we współrzędnych uogólnionych jak i pomocniczych. Wyznaczono przybliżenie liniowe w punkcie równowagi, a także podjęto próbę wyznaczenia sterownika liniowego w oparciu o dwie metody LQR oraz PP. Przeprowadzono szereg symulacji w celu weryfikacji poprawnego działania sterownika. Następnie zaprojektowano i przebadano nieliniowy sterownik, działający w oparciu o model dynamiki opisujący zachowanie robota. Wyniki symulacji zaprezentowano na wykresach.

W praktycznej części pracy podjęto próbę budowy opisywanej konstrukcji. Z uwagi na to, że tematem pracy jest głównie model matematyczny robota oraz jego sterowanie, zdecydowano o szczegółowym opisaniu tylko niektórych elementów składowych robota. Pominięto te elementy, które nie są konieczne do powtórzenia praktycznego eksperymentu.

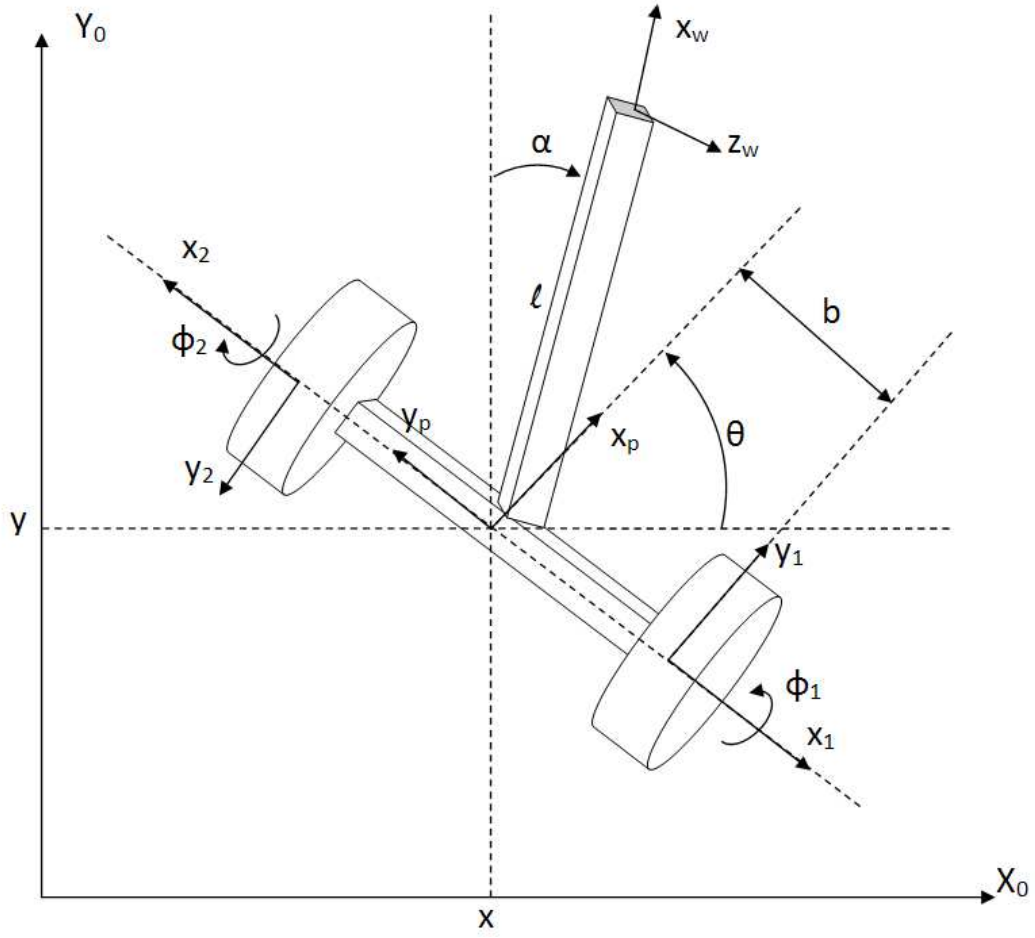
Niestety w trakcie budowy robota okazało się, że niektóre parametry fizyczne znacznie się różniły od tych przyjętych w symulacjach. Użyte niskiej jakości napędy przyczyniły się do tego, że autor nie zdążył ponownie przeprowadzić symulacji. Aby jednak przetestować

poprawność działania wszystkich modułów robota, zaimplementowano w sterowniku algorytm rekursywnej filtracji Kalmana oraz prosty podwójny sterownik PID. Wyniki eksperymentu przedstawiono na wykresach, na podstawie danych z robota wysłanych do komputera poprzez interfejs RS232.

# Rozdział 2

# Model robota balansującego

Robot balansujący przedstawiony schematycznie na rysunku 2.1 może być rozważany jako odwrócone, mobilne wahadło. Opisywana konstrukcja różni się od klasycznego odwróconego wahadła tym, że nie posiada wyróżnionej platformy jezdnej [1]. Konstrukcja nośna jest jednocześnie wahadłem, do której mocuje się dwa napędy wraz z kołami. To dzięki nim możliwe jest utrzymywanie równowagi oraz przemieszczanie się robota. Przedstawiona konstrukcja jest nazywana robotem balansującym, gdyż celem sterowania jest utrzymywanie robota w pozycji pionowej, ponieważ jego środek ciężkości znajduje się znacznie wyżej niż podwozie. W dalszej części pracy rozważono jedynie dynamikę wahadła, kół oraz napędów robota.




Rysunek 2.1 Odwrócone wahadło na kołach - robot balansujący

## 2.1 Model dynamiki robota balansującego we współrzędnych uogólnionych.

### 2.1.1 Energia kinetyczna kół.

Opisywany robot balansujący można potraktować jako manipulator mobilny z niedosterowanym ramieniem manipulacyjnym [11]. W konstrukcji tej można wyróżnić dwa układy lokalne. Jeden z nich jest stowarzyszony z osią łączącą koła $X_pY_pZ_p$, drugi zaś z odwróconym wahadłem $X_wY_wZ_w$. Układ lokalny stowarzyszony z podsystemem mobilnym $X_pY_pZ_p$ jest umiejscowiony na osi łączącej koła, w połowie odległości między nimi. Transformacja opisująca położenie i orientację robota względem globalnego układu odniesienia $X_0Y_0Z_0$ jest wyrażona wzorem

$$T_0^p(x,y,\theta) = Trans(X,x)Trans(Y,y)Rot(Z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & x \\ \sin\theta & \cos\theta & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Opisywana konstrukcja nie posiada wyróżnionego nadwozia. Napędy wraz z kołami są dołączone bezpośrednio do wahadła, które spełnia rolę nadwozia, dlatego w tym przypadku nie oblicza się energii kinetycznej nadwozia. Wyznaczamy jedynie energię kinetyczną kół. Do wyliczenia energii poszczególnych kół konieczne jest wyznaczenie położenia układów lokalnych $X_iY_iZ_i$ stowarzyszonych z obracającymi się kołami. Transformacja dla pierwszego koła ma postać

$$T_0^{k1} = T_0^p Rot(Z,-\frac{\pi}{2})Trans(X,b)Rot(X,-\phi_1). \tag{2.1}$$

Z pracy [3] wiadomo, że dla transformacji (2.1) energia kinetyczna pierwszego koła będzie równa

$$K_1 = \frac{1}{2}I_{xx1}\dot{\phi}_1^2 + \frac{1}{2}I_{zz1}\dot{\theta}^2 + \frac{1}{2}M_{k1}(\dot{x}^2+\dot{y}^2+b^2\dot{\theta}^2) + M_{k1}b\dot{\theta}(\dot{y}\sin\theta + \dot{x}\cos\theta).$$

Podobnie dla drugiego koła transformacja przekształcająca układ lokalny $X_2Y_2Z_2$ do układu podstawowego $X_0Y_0Z_0$ jest następująca

$$T_0^{k2} = T_0^p Rot(Z,\frac{\pi}{2})Trans(X,b)Rot(X,\phi_2),$$

a energia kinetyczna tego koła wynosi

$$K_2 = \frac{1}{2}I_{xx2}\dot{\phi}_2^2 + \frac{1}{2}I_{zz2}\dot{\theta}^2 + \frac{1}{2}M_{k2}(\dot{x}^2+\dot{y}^2+b^2\dot{\theta}^2) - M_{k2}b\dot{\theta}(\dot{y}\sin\theta + \dot{x}\cos\theta),$$

gdzie poszczególne symbole oznaczają:

- $I_{xxi} = \frac{1}{2}M_{ki}R_i^2$ - moment bezwładności $i$-tego koła względem osi $X_i$,
- $I_{zzi} = \frac{1}{12}M_{ki}w_i^2 + \frac{1}{4}M_{ki}R_i^2$ - moment bezwładności $i$-tego koła względem osi $Z_i$,
- $M_{ki}$ - masa $i$-tego koła,

- $R_i$ - promień $i$-tego koła,
- $w_i$ - grubość $i$-tego koła,
- $b$ - odległość $i$-tego koła od początku układu lokalnego robota $X_pY_pZ_p$.

W dalszych rozważaniach przyjmujemy, że oba koła są jednakowe, a więc mają takie same wszystkie parametry. Po zsumowaniu energii kinetycznej obydwu kół

$$K_k = K_1 + K_2$$

otrzymujemy

$$K_k = \frac{1}{2} I_{xx}(\dot{\phi}_1^2 + \dot{\phi}_2^2) + I_p\dot{\theta}^2 + (\dot{x}^2 + \dot{y}^2)M_k,$$

gdzie

$$I_p = I_{zz} + M_k b^2.$$

Energię kinetyczną kół można wyrazić w postaci formy kwadratowej jako

$$K_k = \frac{1}{2}\dot{q}_k^T Q_k(q_k)\dot{q}_k, \quad q_k = (x, y, \theta, \phi_1, \phi_2)^T$$

z diagonalną macierzą bezwładności

$$Q_k = \begin{bmatrix} 2M_k & 0 & 0 & 0 & 0 \\ 0 & 2M_k & 0 & 0 & 0 \\ 0 & 0 & 2I_p & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 \\ 0 & 0 & 0 & 0 & I_{xx} \end{bmatrix}.$$

### 2.1.2 Energia kinetyczna odwróconego wahadła.

Aby obliczyć energię kinetyczną wahadła, należy wyznaczyć transformację kinematyczną końca wahadła $X_wY_wZ_w$ względem układu podstawowego [4]. Wahadło jest zaczepione w środku lokalnego układu robota $X_pY_pZ_p$. Wówczas transformację układu $X_wY_wZ_w$ względem układu podstawowego można wyrazić jako

$$T_0^w = T_0^p Rot(X, 90^o) Rot(Z, 90^o - \alpha) Trans(X, l)$$

gdzie $l$ to długość wahadła, a $\alpha$ to odchylenie wahadła od pionu, jak oznaczono na rysunku 2.1. Transformacja ta jest równa

$$T_0^w = \begin{bmatrix} \cos\theta\sin\alpha & -\cos\theta\cos\alpha & \sin\theta & x + l\cos\theta\sin\alpha \\ \sin\theta\sin\alpha & -\sin\theta\cos\alpha & -\cos\theta & y + l\sin\theta\sin\alpha \\ \cos\alpha & \sin\alpha & 0 & l\cos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Do wyznaczenia energii kinetycznej potrzebna jest także macierz pseudoinercji wahadła (ogniwa L) względem układu lokalnego. Macierz ta jest zdefiniowana w następujący sposób

$$J_w = \begin{bmatrix} \int_L x^2 \cdot dm & \int_L xy \cdot dm & \int_L xz \cdot dm & \int_L x \cdot dm \\ \int_L xy \cdot dm & \int_L y^2 \cdot dm & \int_L zy \cdot dm & \int_L y \cdot dm \\ \int_L xz \cdot dm & \int_L zy \cdot dm & \int_L z^2 \cdot dm & \int_L z \cdot dm \\ \int_L x \cdot dm & \int_L y \cdot dm & \int_L z \cdot dm & \int_L dm \end{bmatrix}.$$

Przyjęto założenie, że wahadło ma postać pręta jednorodnego, którego długość jest ponad 10 razy większa niż średnica przekroju. Wówczas macierz pseudoinercji wynosi

$$J_w = \begin{bmatrix} \frac{M_w l^2}{3} & 0 & 0 & -\frac{M_w l}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{M_w l}{2} & 0 & 0 & M_w \end{bmatrix}$$

Energię kinetyczną wahadła można wyliczyć z następującego wzoru

$$K_w = \frac{1}{2} tr \left[ \dot{T}_0^w J_w (\dot{T}_0^w)^T \right].$$

Energia ta jest równa

$$\begin{aligned} K_w = -\frac{1}{6} M l^2 (-3\dot{x}^2 + 3\dot{\theta}\dot{x} l \sin\theta \sin\alpha - 3\dot{\alpha}\dot{x} l \cos\theta \cos\alpha - \dot{\theta}^2 l^2 + \\ + \dot{\theta}^2 l^2 \cos\alpha^2 - 3\dot{y}^2 - 3\dot{\theta}\dot{y} l \cos\theta \sin\alpha - 3\dot{\alpha}\dot{y} l \sin\theta \cos\alpha - \dot{\alpha}^2). \end{aligned} \tag{2.2}$$

Energię wahadła (2.2) można również przedstawić w postaci formy kwadratowej

$$K_w = \frac{1}{2} \dot{q}^T Q_w(q) \dot{q}, \quad q = (x, y, \theta, \phi_1, \phi_2, \alpha)^T$$

z diagonalną macierzą bezwładności

$$Q_w(q) = \begin{bmatrix} Q_{11} & 0 & Q_{13} & 0 & 0 & Q_{16} \\ 0 & Q_{22} & Q_{23} & 0 & 0 & Q_{26} \\ Q_{13} & Q_{23} & Q_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ Q_{16} & Q_{26} & 0 & 0 & 0 & Q_{66} \end{bmatrix},$$

przy czym poszczególne elementy tej macierzy są równe

$$\begin{aligned} Q_{11} &= M_w, \\ Q_{22} &= M_w, \\ Q_{33} &= -\frac{1}{3} M_w l^2 (\cos^2\alpha - 1, ) \\ Q_{66} &= \frac{1}{3} M_w l^2, \\ Q_{13} &= -\frac{1}{2} M_w l \sin\theta \sin\alpha, \\ Q_{23} &= \frac{1}{2} M_w l \cos\theta \sin\alpha, \\ Q_{16} &= \frac{1}{2} M_w l \cos\theta \cos\alpha, \\ Q_{26} &= \frac{1}{2} M_w l \sin\theta \cos\alpha. \end{aligned}$$

### 2.1.3 Energia potencjalna.

W opisywanej konstrukcji uwzględniono jedynie energię potencjalną dla podsystemu wahadła ponieważ koła poruszają się po powierzchni ekwipotencjalnej. Energię potencjalną wahadła można obliczyć ze wzoru

$$V_w = -M_w < g, T_0^w R_w > \tag{2.3}$$

gdzie $R_w$ to wektor opisujący położenie środka masy konstrukcji nośnej (wahadła) względem układu lokalnego $X_w Y_w Z_w$. Wektor grawitacji jest wyrażony

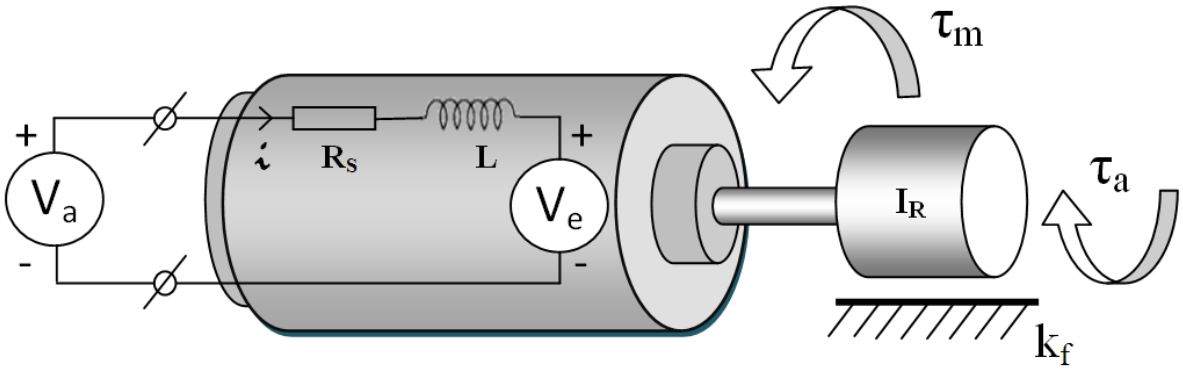
$$g = \begin{pmatrix} 0 \\ 0 \\ -g \\ 0 \end{pmatrix}, \quad R = \begin{pmatrix} -\frac{1}{2}l \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{2.4}$$

Ze wzorów (2.3) i (2.4) otrzymuje się energię potencjalną odwróconego wahadła

$$V_w = \frac{1}{2} M_w l g \cos\alpha.$$

### 2.1.4 Dynamiki napędów kół.

Opisywany robot jest napędzany dwoma silnikami prądu stałego wraz z dwustopniową przekładnią planetarną. Rysunek 2.2 przedstawia model silnika DC, który zostanie uwzględniony w modelu robota. Wejściem silnika jest przyłożone napięcie, wyjściem - moment potrzebny do tego, aby robot utrzymywał równowagę.




Rysunek 2.2 Model silnika prądu stałego.

Napięcie przyłożone do elektrod silnika generuje prąd $i$ płynący przez jego cewki. Z kolei, prąd wytwarza moment obrotowy proporcjonalny do wartości prądu ze stałym współczynnikiem $k_m$ równy

$$\tau_m = k_m i.$$

Cewki silnika, poza indukcyjnością $L$, posiadają także rezystancję. Obydwa te parametry modeluje się jako elementy połączone szeregowo. Należy także zwrócić uwagę na fakt, że cewki wirnika obracają się w polu magnetycznym, które jest wytworzone przez magnesy stałe. Wskutek tego pojawi się, skierowana przeciwnie do napięcia zasilania, zwrotna siła elektromotoryczna $V_e$. Można przyjąć, że jest to liniowa funkcja prędkości obrotowej silnika

$$V_e = k_e \omega.$$

Korzystając z pierwszego prawa Kirchhoffa, można zapisać równanie różniczkowe dla prądu płynącego w obwodzie silnika jako

$$V_a - R_s i - L\frac{di}{dt} - V_e = 0. \tag{2.5}$$

Następnie należy skorzystać z Zasad Dynamiki Newtona, które mówią o tym, że suma wszystkich momentów sił jest liniowo zależna od przyspieszenia osi pomnożonej przez jej bezwładność. Tarcie elementów ruchomych można przybliżyć jako liniową funkcję prędkości obrotowej ze współczynnikiem tarcia $k_f$, jak następuje

$$\sum M = \tau_m - k_f\omega - \tau_a = I_R\frac{d\omega}{dt}, \tag{2.6}$$

gdzie $I_R$ to bezwładność wirnika, zaś $\tau_a$ to moment przyłożony do osi napędu. Z powyższych równań uzyskuje się dwa równania różniczkowe pierwszego rzędu, które są liniowymi funkcjami prądu i prędkości obrotowej

$$\frac{di}{dt} = \frac{V_a}{L} - \frac{R_s}{L}i - \frac{k_e}{L}\omega, \tag{2.7}$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R}i - \frac{k_f}{I_R}\omega - \frac{\tau_a}{I_R}. \tag{2.8}$$

W praktyce często pomija się indukcyjność oraz tarcie w małych silnikach DC. Ponieważ rozważana konstrukcja jest wyposażona właśnie w takie niewielkie silniki, to w modelu konstruowanego robota balansującego powyższe (2.5) i (2.6) upraszczają się do postaci

$$i = \frac{V_a}{R_s} - \frac{k_e}{R_s}\omega, \tag{2.9}$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R}i - \frac{\tau_a}{I_R}. \tag{2.10}$$

Dokonując podstawienia równania (2.9) do (2.10) otrzymano

$$\frac{d\omega}{dt}I_R = \frac{k_m}{R_s}V_a - \frac{k_e k_m}{R_s}\omega - \tau_a. \tag{2.11}$$

Równanie (2.11) można przedstawić w ogólnej postaci modelu przestrzeni stanów, gdzie stanem jest położenie osi koła $\theta$ i jej prędkość $\omega$. Wejścia systemu to napięcia przyłożone do elektrod silnika $V_a$ oraz moment przyłożony do osi napędu $\tau_a$

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{k_e k_m}{I_R R_s} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{k_m}{I_R R_s} & -\frac{1}{I_R} \end{bmatrix} \begin{bmatrix} V_a \\ \tau_a \end{bmatrix},$$

zaś wyjściem jest położenie osi $\theta$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix}.$$

W przedstawionej konstrukcji robota użyto dwóch jednakowych silników prądu stałego i ich identyczne modele włączono do modelu robota balansującego. Oznaczenie pozycji $\theta$, prędkości kątowej $\omega$ i napięcia $V_a$ w modelu $i$-tego silnika dla zgodności z modelem kół w robocie zastąpiono oznaczeniami $\phi$, $\dot{\phi}$ oraz $V_i$. Moment $\tau_a$ przyłożony do osi $k$-tego silnika to moment pochodzący od kół. Wówczas równanie (2.11) przyjmuje postać

$$\tau_a + I_R\ddot{\phi}_i + \frac{k_e k_m}{R_s}\dot{\phi}_i = \frac{k_m}{R_s}V_i.$$

We współrzędnych uogólnionych $q$ robota balansującego, model silników można zapisać jako

$$Q_s\ddot{q} + T\dot{q} = B \begin{bmatrix} V_1 \\ V_2 \end{bmatrix},$$

gdzie macierz bezwładności silnika to

$$Q_s = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_R & 0 & 0 \\ 0 & 0 & 0 & 0 & I_R & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

zaś macierz tarcia to

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{k_e k_m}{R_s} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{k_e k_m}{R_s} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

W niniejszej pracy rozważono dwa przypadki oddziaływania napędów na model dynamiki robota balansującego. W pierwszym z nich przyjęto, że zamocowane do wahadła napędy będą oddziaływały na nie z taką samą siłą, jak na koła, tylko że skierowaną w przeciwnym kierunku. Macierz wejścia ma wtedy postać

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{km}{R_s} & 0 \\ 0 & \frac{km}{R_s} \\ -\frac{km}{R_s} & -\frac{km}{R_s} \end{bmatrix}. \tag{2.12}$$

Drugi przypadek to taki, w którym przyjęto, że napędy oddziałują jedynie na koła robota. Utrzymywanie wahadła w pozycji pionowej uzyskuje się w sposób pośredni czyli np poprzez odpowiednie przemieszczanie się podwozia. Macierz wejścia ma wtedy postać

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{km}{R_s} & 0 \\ 0 & \frac{km}{R_s} \\ 0 & 0 \end{bmatrix}. \tag{2.13}$$

Pierwszy przypadek wykorzystano dla sterowników liniowych, drugi zaś dla nieliniowych.

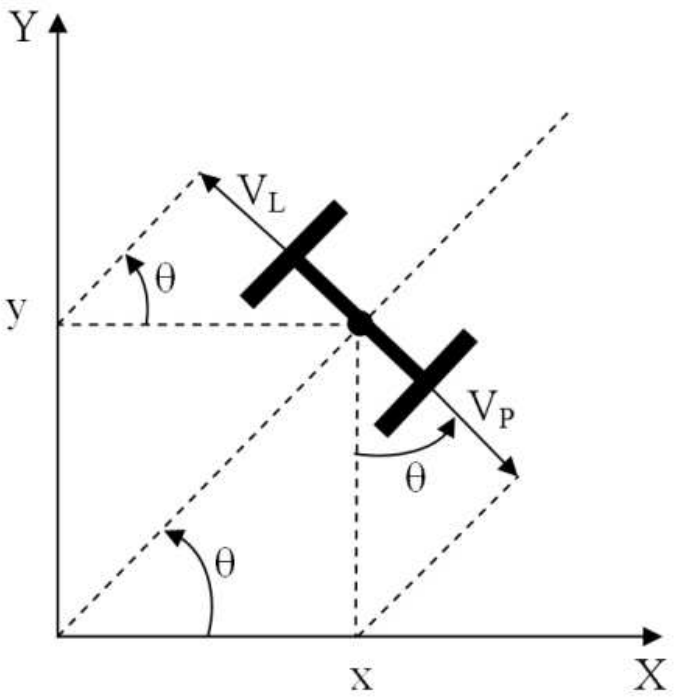
### 2.1.5 Ograniczenia nieholonomiczne.

Odwrócone mobilne wahadło napędzane jest dwoma kołami ustalonymi. Przyjmując założenie, że robot porusza się bez poślizgów kół, można wyprowadzić tzw. ograniczenia fazowe [1] w postaci Pfaffa

$$A(q_k)\dot{q}_k = 0. \tag{2.14}$$

Dla opisywanej dwukołowej konstrukcji przyjęto założenie, że prędkości w punkcie kontaktu kół z podłożem są równe zero. Zatem nie występuje poślizg boczny, poślizg wzdłużny oraz buksowanie kół.

Ruch bez poślizgu bocznego oznacza brak składowej prędkości prostopadłej do kierunku ruchu. Równania składowych prostopadłych mają postać




Rysunek 2.3 Ruch robota po płaszczyźnie.

$$V_L = \dot{x}\sin\theta$$
$$V_P = \dot{y}\cos\theta,$$

a po podstawieniu

$$V_L - V_P = \dot{x}\sin\theta - \dot{y}\cos\theta = 0.$$

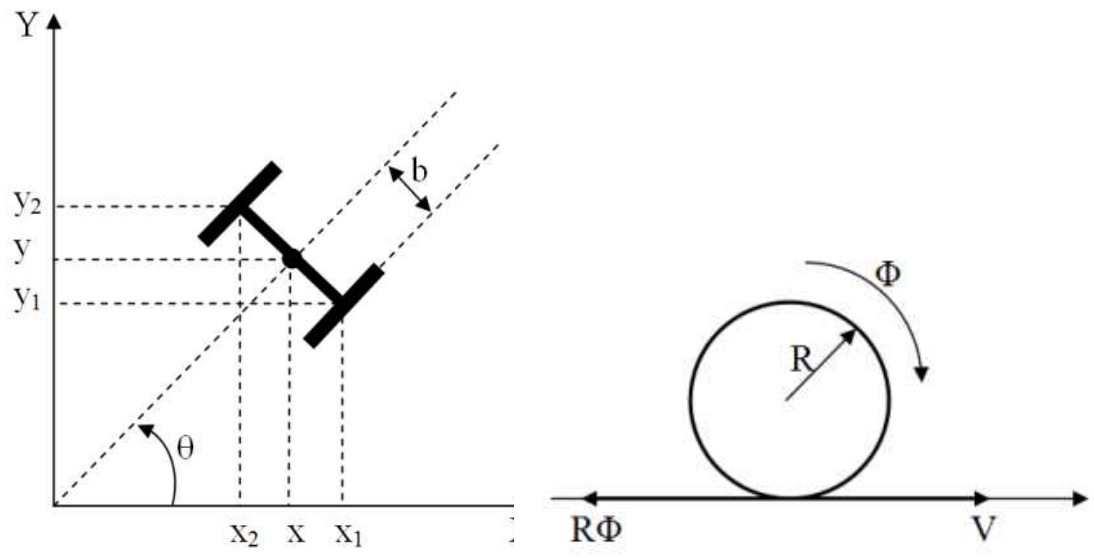
Tak wyprowadzone ograniczenie w postaci Pfaffa wygląda następująco

$$(\sin\theta, -\cos\theta, 0, 0, 0)\dot{q}_k = 0.$$

Brak poślizgu i buksowania kół oznacza brak prędkości w punkcie styku koła z podłożem, co przedstawiono na rysunku 2.4. Zgodnie z rys. 2.4, ograniczenie to zapisano w postaci ogólnej dla $i$-tego koła jako

$$V_i = R\dot{\phi}_i, \tag{2.15}$$
$$\dot{x}_i = V_i\cos\theta, \tag{2.16}$$
$$\dot{y}_i = V_i\sin\theta. \tag{2.17}$$

W przypadku niespełnienia pierwszej z powyższych równości następuje zjawisko




Rysunek 2.4 Ruch robota po płaszczyźnie.

- $V < R\dot{\phi}$ - buksowanie
- $V > R\dot{\phi}$ - poślizg

Prędkość $V$ to suma prędkości wzdłuż osi $x$, $y$ oraz prędkości związanej z orientacją robota $\theta$. Dla koła pierwszego wyprowadzenie ograniczenia wygląda następująco

$$x_1 = x + b\sin\theta, \qquad y_1 = y - b\cos\theta,$$

a po zróżniczkowaniu

$$\dot{x}_1 = \dot{x} + \dot{\theta}b\cos\theta, \qquad \dot{y}_1 = \dot{y} + \dot{\theta}b\sin\theta.$$

Następnie podstawiając do składowych prędkości $\dot{x}_1$ oraz $\dot{y}_1$ równanie (2.15) otrzymuje się

$$\begin{aligned} \dot{x}\cos\theta + \dot{\theta}b\cos^2\theta - R\dot{\phi}_1\cos^2\theta = 0 \\ -\dot{y}\sin\theta - \dot{\theta}b\sin^2\theta + R\dot{\phi}_1\sin^2\theta = 0 \end{aligned} \tag{2.18}$$

oraz przyrównując do siebie równania (2.18) otrzymano ograniczenie mówiące o braku poślizgu wzdłużnego dla pierwszego koła

$$\dot{x}\cos\theta + \dot{y}\sin\theta + b\dot{\theta} - R\dot{\phi}_1 = 0.$$

Ponieważ drugie koło kręci się w przeciwnym kierunku do koła pierwszego ograniczenie dla tego koła ma postać

$$-\dot{x}\cos\theta - \dot{y}\sin\theta + b\dot{\theta} + R\dot{\phi}_2 = 0.$$

Ostatecznie można zapisać wszystkie ograniczenia w postaci macierzowej Pfaffa

$$A(q_k)\dot{q}_k = \begin{bmatrix} \sin\theta & -\cos\theta & 0 & 0 & 0 \\ \cos\theta & \sin\theta & b & -R & 0 \\ -\cos\theta & -\sin\theta & b & 0 & R \end{bmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi}_1 \\ \dot{\phi}_2 \end{pmatrix} = 0.$$

Z warunku Pfaffa (2.14) wynika, że prędkości kół należą do jądra macierzy $A(q_k)$, czyli $\dot{q}_k \in KerA(q_k)$. Należy znaleźć wektory rozpinające te jądro, czyli

$$\dot{q}_k = G(q_k)\eta,$$

gdzie $\eta \in R^m, \quad m = n - l$, $n$-liczba zmiennych stanu, $l$-liczba ograniczeń. Macierz $G(q_k)$ spełnia zatem warunek

$$A(q_k)G(q_k) \equiv 0$$

i można ją wybrać w następujący sposób

$$G(q_k) = \begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ \frac{1}{b} & -\frac{1}{b} \\ \frac{2}{R} & 0 \\ 0 & \frac{2}{R} \end{bmatrix}, \qquad \eta = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix},$$

gdzie $\eta_1$ ma sens przeskalowanej prędkości koła pierwszego, a $\eta_2$ na sens przeskalowanej prędkości koła drugiego. Prędkości $\eta$ noszą nazwę prędkości pomocniczych.

### 2.1.6 Równania dynamiki mobilnego odwróconego wahadła.

Po uzyskaniu wzorów na energię kinetyczną i potencjalną wahadła należy funkcję Lagrange'a równą

$$L(q, \dot{q}) = K_w + K_k - V_w$$

wstawić do zasady d'Alemberta w celu uzyskania równań dynamiki

$$\begin{aligned} &Q_w(q)\ddot{q} + \begin{bmatrix} Q_k & 0 \\ 0 & 0 \end{bmatrix} \ddot{q} + Q_s(q)\ddot{q} + C_w(q,\dot{q})\dot{q} + C_k(q,\dot{q})\dot{q} + C_s(q,\dot{q})\dot{q} \\ &+ T(q)\dot{q} + D(q) = A^T(q)\lambda + Bu \end{aligned} \tag{2.19}$$

gdzie

- $q = (x, y, \theta, \phi_1, \phi_2, \alpha)^T$ - wektor zmiennych stanu,
- $Q_w(q)$ - macierz bezwładności wahadła,
- $Q_k$ - macierz bezwładności kół,
- $Q_s$ - macierz bezwładności silników,
- $C_w(q, \dot{q})$ - macierz Coriolisa wahadła,
- $C_k(q, \dot{q})$ - macierz Coriolisa kół,
- $C_s(q, \dot{q})$ - macierz Coriolisa silników,
- $T$ - macierz tarcia pochodząca od silników,
- $D(q)$ - wektor grawitacji.

Aby uprościć zapis, macierze $Q_w$, $Q_k$ i $Q_s$ dodano do siebie w modelu jako

$$Q_{wks}(q)\ddot{q} + C_w(q,\dot{q})\dot{q} + T\dot{q} + D(q) = A^T(q)\lambda + Bu$$

i otrzymano macierz bezwładności całości

$$Q_{kws}(q) = \begin{bmatrix} Q_{11} & 0 & Q_{13} & 0 & 0 & Q_{16} \\ 0 & Q_{22} & Q_{23} & 0 & 0 & Q_{26} \\ Q_{13} & Q_{23} & Q_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_{55} & 0 \\ Q_{16} & Q_{26} & 0 & 0 & 0 & Q_{66} \end{bmatrix} \tag{2.20}$$

gdzie

$$\begin{aligned} Q_{11} &= 2M_k + M_w, \\ Q_{22} &= 2M_k + M_w, \\ Q_{33} &= 2I_p - \frac{1}{3}M_w l^2(\cos^2\alpha - 1), \\ Q_{44} &= I_{xx} + I_R, \\ Q_{55} &= I_{xx} + I_R, \\ Q_{66} &= \frac{1}{3}M_w l^2, \\ Q_{13} &= -\frac{1}{2}M_w l \sin\theta \sin\alpha, \\ Q_{23} &= \frac{1}{2}M_w l \cos\theta \sin\alpha, \\ Q_{16} &= \frac{1}{2}M_w l \cos\theta \cos\alpha, \\ Q_{26} &= \frac{1}{2}M_w l \sin\theta \cos\alpha, \end{aligned}$$

oraz

- $I_{xx} = \frac{1}{2}M_k R^2$,
- $I_{zz} = \frac{1}{12}M_k w^2 + \frac{1}{4}M_k R^2$,
- $I_p = I_{zz} + M_k b^2$.

Wektor grawitacji ma postać

$$D(q) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\frac{1}{2}M_w l g \sin\alpha \end{pmatrix}$$

Ponieważ macierze bezwładności $Q_k(q)$ oraz $Q_s(q)$ są stałe, to macierze Coriolisa $C_k(q,\dot{q})$ oraz $C_s(q,\dot{q})$ będą równe 0. Zatem należy obliczyć jedynie macierz $C_w(q,\dot{q})$. Korzystając ze wzoru na symbole Christoffela

$$C^{ij} = \sum_k C^i_{kj}\dot{q}_k,$$

gdzie

$$C_{kj}^{i} = \frac{1}{2}\left(\frac{\partial Q^{ij}}{\partial q_k} + \frac{\partial Q^{ik}}{\partial q_j} - \frac{\partial Q^{jk}}{\partial q_i}\right)$$

wyznaczono następującą macierz

$$C_w(q,\dot{q}) = \begin{bmatrix} 0 & 0 & C_{13} & 0 & 0 & C_{16} \\ 0 & 0 & C_{23} & 0 & 0 & C_{26} \\ 0 & 0 & C_{33} & 0 & 0 & C_{36} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{63} & 0 & 0 & 0 \end{bmatrix},$$

której poszczególne elementy są równe

$$\begin{aligned} C_{13} &= -\frac{1}{2}M_w l(\cos\theta\sin\alpha\dot{\theta} + \sin\theta\cos\alpha\dot{\alpha}), \\ C_{16} &= -\frac{1}{2}M_w l(\sin\theta\cos\alpha\dot{\theta} + \cos\theta\sin\alpha\dot{\alpha}), \\ C_{23} &= -\frac{1}{2}M_w l(\sin\theta\sin\alpha\dot{\theta} - \cos\theta\sin\alpha\dot{\alpha}), \\ C_{26} &= \frac{1}{2}M_w l(\cos\alpha\cos\theta\dot{\theta} - \sin\alpha\sin\theta\dot{\alpha}), \\ C_{33} &= \frac{1}{3}M_w l^2\cos\alpha\sin\alpha\dot{\alpha}, \\ C_{36} &= \frac{1}{3}M_w l^2\cos\alpha\sin\alpha\dot{\theta}, \\ C_{63} &= -\frac{1}{3}M_w l^2\cos\alpha\sin\alpha\dot{\theta}. \end{aligned} \tag{2.21}$$

Pozostałe macierze pozostają bez zmian

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{k_e k_m}{R_s} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{k_e k_m}{R_s} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.22}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{km}{R_s} & 0 \\ 0 & \frac{km}{R_s} \\ -\frac{km}{R_s} & -\frac{km}{R_s} \end{bmatrix}, \quad lub \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{km}{R_s} & 0 \\ 0 & \frac{km}{R_s} \\ 0 & 0 \end{bmatrix}, \tag{2.23}$$

$$u = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}. \tag{2.24}$$

Wektor stanu $q$ ma postać

$$q = \begin{bmatrix} x \\ y \\ \theta \\ \phi_1 \\ \phi_2 \\ \alpha \end{bmatrix}. \tag{2.25}$$

## 2.2 Model dynamiki robota balansującego we współrzędnych fazowych

Wyprowadzony w podrozdziale 2.1.6 model dynamiki robota we współrzędnych uogólnionych można wyrazić również w postaci modelu we współrzędnych pomocniczych [4]. Dzięki temu zabiegowi zmienne stanu $q$ zostaną zastąpione zmiennymi sterowalnymi $\eta_1$ i $\eta_2$ (prędkościami pomocniczymi).

Zależność pomiędzy wektorem stanu $\dot{q}_k$, a $\eta$ jest następująca

$$\dot{q}_k = G(q_k)\eta. \tag{2.26}$$

Model dynamiki we współrzędnych uogólnionych

$$Q_{kws}(q)\ddot{q} + C_w(q,\dot{q})\dot{q} + D(q) + T\dot{q} = A^T\lambda + Bu \tag{2.27}$$

gdzie $A$ to macierz ograniczeń Pfaffa można zapisać jako

$$\begin{gathered} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{pmatrix} \ddot{q}_k \\ \ddot{\alpha} \end{pmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{pmatrix} \dot{q}_k \\ \dot{\alpha} \end{pmatrix} + \\ + \begin{bmatrix} 0 \\ D_\alpha \end{bmatrix} + \begin{bmatrix} T_k \\ 0 \end{bmatrix} \begin{pmatrix} \dot{q}_k \\ \dot{\alpha} \end{pmatrix} = \begin{bmatrix} B_k \\ B_\alpha \end{bmatrix} u + \begin{bmatrix} A^T\lambda \\ 0 \end{bmatrix} \end{gathered} \tag{2.28}$$

Aby wyrazić powyższy model we współrzędnych pomocniczych należy wyliczyć wektor $\ddot{q}$, korzystając z zależności (2.26)

$$\ddot{q}_k = \dot{G}\eta + G\dot{\eta}. \tag{2.29}$$

Podstawiając do modelu (2.28) zależności (2.26) i (2.29) i po przemnożeniu przez $G^T$ pierwszego równania macierzowego, otrzymano nowe równanie macierzowe

$$\begin{gathered} \begin{bmatrix} G^TQ_{11}G & G^TQ_{12} \\ Q_{21}G & Q_{22} \end{bmatrix} \begin{pmatrix} \dot{\eta} \\ \ddot{\alpha} \end{pmatrix} + \\ + \begin{bmatrix} G^T(Q_{11}\dot{G} + C_{11}G) & G^TC_{12} \\ Q_{21}\dot{G} + C_{21}G & C_{22} \end{bmatrix} \begin{pmatrix} \eta \\ \dot{\alpha} \end{pmatrix} + \\ + \begin{bmatrix} 0 \\ D_\alpha \end{bmatrix} + \begin{bmatrix} G^TT_kG \\ 0 \end{bmatrix} \begin{pmatrix} \eta \\ \dot{\alpha} \end{pmatrix} = \begin{bmatrix} G^TB_k \\ B_\alpha \end{bmatrix} u \end{gathered} \tag{2.30}$$

Ostatecznie model dynamiki we współrzędnych pomocniczych przyjmuje postać

$$Q^*(q)\begin{pmatrix} \dot{\eta} \\ \ddot{\alpha} \end{pmatrix} + C^*(q,\dot{q})\begin{pmatrix} \eta \\ \dot{\alpha} \end{pmatrix} + D^* + T^*\begin{pmatrix} \eta \\ \dot{\alpha} \end{pmatrix} = B^*u, \tag{2.31}$$

gdzie

- macierz bezwładności to

$$Q^*(q) = \begin{bmatrix} Q^*_{11} & Q^*_{12} & Q^*_{13} \\ Q^*_{21} & Q^*_{22} & Q^*_{23} \\ Q^*_{31} & Q^*_{32} & Q^*_{33} \end{bmatrix}, \tag{2.32}$$

a współczynniki tej macierzy to

$$\begin{aligned} Q^*_{11} =& -\frac{1}{3b^2R^2}(-3b^2R^2M_w - 6b^2R^2M_k - R^2M_wl^2 + R^2M_wl^2\cos^2\alpha, \\ & - 6R^2Ip - 12b^2Ixx - 12b^2Ir), \\ Q^*_{12} =& \frac{1}{3b^2}(3b^2M_w + 6b^2M_k - M_wl^2 + M_wl^2\cos^2\alpha - 6Ip), \\ Q^*_{13} =& \frac{1}{2}M_wl\cos\alpha, \\ Q^*_{21} =& Q_{12}, \\ Q^*_{22} =& Q_{11}, \\ Q^*_{23} =& Q_{13}, \\ Q^*_{31} =& Q_{13}, \\ Q^*_{32} =& Q_{13,} \\ Q^*_{33} =& \frac{1}{3}Ml^2 \end{aligned} \tag{2.33}$$

- macierz sił odśrodkowych i sił Coriolisa

$$C^*(q,\dot{q}) = \begin{bmatrix} C^*_{11} & C^*_{12} & C^*_{13} \\ C^*_{21} & C^*_{22} & C^*_{23} \\ C^*_{31} & C^*_{32} & C^*_{33} \end{bmatrix}, \tag{2.34}$$

ze współczynnikami

$$\begin{aligned} C^*_{11} =& \frac{M_wl^2}{6b^2}\sin(2\alpha)\dot{\alpha}, \\ C^*_{12} =& -\frac{M_wl}{3b^2}\sin\alpha(l\cos\alpha\dot{\alpha} - 3b\dot{\theta}, \\ C^*_{13} =& \frac{M_wl}{6b}\sin\alpha(2l\cos\alpha\dot{\theta} - 3b\dot{\alpha}), \\ C^*_{21} =& -\frac{M_wl}{3b^2}\sin\alpha(l\cos\alpha\dot{\alpha} + 3b\dot{\theta}), \\ C^*_{22} =& C^*_{11}, \\ C^*_{23} =& -\frac{M_wl}{6b}\sin\alpha(2l\cos\alpha\dot{\theta} + 3b\dot{\alpha}), \\ C^*_{31} =& -\frac{M_wl^2}{6b}\sin(2\alpha)\dot{\theta}, \\ C^*_{32} =& -C^*_{31}, \\ C^*_{33} =& 0 \end{aligned} \tag{2.35}$$

- wektor grawitacji

$$D^*(q) = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{2}M_wlg\sin\alpha \end{bmatrix}, \tag{2.36}$$

- macierz tarcia

$$T^* = \begin{bmatrix} \frac{4}{R^2 R_s} k_e k_m & 0 & 0 \\ 0 & \frac{4}{R^2 R_s} k_e k_m & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.37}$$

- macierz wejściowa

$$B^* = \begin{bmatrix} \frac{2}{R R_s} k_m & 0 \\ 0 & \frac{2}{R R_s} k_m \\ -\frac{km}{R_s} & -\frac{km}{R_s} \end{bmatrix}, \quad lub \quad B^* = \begin{bmatrix} \frac{2}{R R_s} k_m & 0 \\ 0 & \frac{2}{R R_s} k_m \\ 0 & 0 \end{bmatrix}, \tag{2.38}$$

gdzie

$$u = \begin{bmatrix} V1 \\ V2 \end{bmatrix}. \tag{2.39}$$

## 2.3 Dane platformy przyjęte podczas symulacji.

W opisywanej konstrukcji można wyróżnić trzy główne składowe elementy: koła oraz część nośną. W części nośnej zainstalowano dwa napędy wraz z przekładniami planetarnymi. Do osi napędów zamocowano koła.

**UWAGA! Poniższe dane przyjęto jedynie w symulacjach i niektóre z nich mogą sie różnić od parametrów modelu wykonanego w praktyce.**

| oznaczenie | wartość | jednostka | opis |
|---|---|---|---|
| $M_k$ | **0,2** | kg | masa koła |
| $M_w$ | **2** | kg | masa wahadła |
| $R$ | **0,06** | m | promień koła |
| $l$ | **0,6** | m | długość wahadła |
| $w$ | **0,005** | m | szerokość koła |
| $b$ | **0,15** | m | odległość koła od środka platformy |
| $k_m$ | **0,006123** | N· m/A | współczynnik wytworzonego momentu do prądu płynącego przez cewki silnika |
| $k_e$ | **0,00692** | V· s/rad | współczynnik wytworzonego napięcia w obracających cewkach silnika |
| $R_s$ | **3** | Ω | rezystancja cewek silnika |
| $g$ | **9,81** | $m/s^2$ | grawitacja |
| $I_{xx}$ | **0,00036** | kg· $m^2$ | moment bezwładności koła względem osi X |
| $I_{zz}$ | **0,00018** | kg· $m^2$ | moment bezwładności koła względem osi Z |
| $I_R$ | **0,005** | kg· $m^2$ | moment bezwładności napędu |
| $I_p$ | **0,0047** | kg· $m^2$ | moment platformy wraz z kołami względem osi Z |

# Rozdział 3

# Sterowanie liniowe

Jak wspomniano we wstępie, większość rozwiązań praktycznych w literaturze opartko na sterowaniu dla układów liniowych. Aby zbadać teoretycznie rozwiązania innych autorów znane z literatury, należy wyznaczyć przybliżenie liniowe robota balansującego w pobliżu punktu równowagi. Następnie do otrzymanych równań przybliżonych można zastosować liniowe regulatory LQR i PP, co pokazano w kolejnych podrozdziałach.

**Należy podkreślić, że aby móc zastosować sterowniki liniowe, macierz $B$ musiała mieć postać (2.12). Zatem w dalszych rozważaniach zakłada się, że napędy oddziaływają zarówno na koła jak i na wahadło.**

## 3.1 Przybliżenie liniowe systemu

Do utrzymywania robota w pozycji pionowej podjęto próbę zaimplementowania sterownika liniowego. Konstrukcja sterowników jest oparta na pierwszej zasadzie Lapunowa [2] i wykorzystuje metody projektowania liniowych sterowników rozmieszczających bieguny oraz minimalno-kwadratowych. Do zaprojektowania sterowników niezbędne jest znalezienie przybliżenia liniowego w pobliżu punktu równowagi systemu o postaci

$$\dot{x} = f(x) + g(x)u. \tag{3.1}$$

### 3.1.1 Wyznaczenie przybliżenia liniowego

Nieliniowy model dynamiki we współrzędnych pomocniczych ma postać

$$Q^*(q)\begin{pmatrix} \dot{\eta} \\ \ddot{\alpha} \end{pmatrix} + C^*(q,\dot{q})\begin{pmatrix} \eta \\ \dot{\alpha} \end{pmatrix} + D^* + T^*\begin{pmatrix} \eta \\ \dot{\alpha} \end{pmatrix} = B^*u,$$

Następnie przekształcono równanie w następujący sposób

$$\begin{pmatrix} \dot{\eta} \\ \ddot{\alpha} \end{pmatrix} = Q^{*-1}\left\{-(C^* + T^*)\begin{pmatrix} \eta \\ \dot{\alpha} \end{pmatrix} - D^*\right\} + Q^{*-1}B^*u = f(x) + g(x)u, \tag{3.2}$$

gdzie $x$ jest nowym wektorem stanu równym

$$\begin{aligned} x_1 &= \phi_1, \\ x_2 &= \dot{\phi}_1 \\ x_3 &= \phi_2, \\ x_4 &= \dot{\phi}_2 \\ x_5 &= \alpha, \\ x_6 &= \dot{\alpha} \end{aligned} \tag{3.3}$$

Widać, że $f(x)$ i $g(x) \in I\!R^3$. Ponieważ w modelu (3.2) korzystamy ze współrzędnych pomocniczych $\eta$, to do poprawnej interpretacji prędkości kół $\dot{\phi}$ należy pierwsze dwa wiersze równania odpowiednio przeskalować

$$\dot{\phi}_i = \frac{2}{R}\eta_i. \tag{3.4}$$

W nowych współrzędnych $x$ równania dynamiki (3.2) maja postać

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{2}{R}f_1(x) \\ x_4 \\ \frac{2}{R}f_2(x) \\ x_6 \\ f_3(x) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2}{R}g_1(x) \\ 0 \\ \frac{2}{R}g_2(x) \\ 0 \\ g_3(x) \end{bmatrix} u = F(x) + G(x)u \tag{3.5}$$

Następnie dokonano przybliżenia liniowego równań wokół punktu równowagi

$$x_0 = [\phi_1, \dot{\phi}_1, \phi_2, \dot{\phi}_2, \alpha, \dot{\alpha}]^T = [0,0,0,0,0,0]^T \quad i \quad u = 0 \tag{3.6}$$

w celu otrzymania ogólnej liniowej postaci równania stanu

$$\dot{x} = Ax + Bu, \text{ gdzie } \quad A = \frac{\partial F}{\partial x}|_{x_0}, \quad B = \frac{\partial G}{\partial x}|_{x_0} \tag{3.7}$$

Obliczeń dokonano w środowisku Matlab korzystając z funkcji *jacobian*. Otrzymano następujące wartości

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0,0723 & 0 & 0,0048 & -63,2450 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0,0048 & 0 & -0,0723 & -63,2450 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0,0051 & 0 & 0,0051 & 34,0117 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0,3352 & 0,0011 \\ 0 & 0 \\ 0,0011 & 0,3352 \\ 0 & 0 \\ -0,0337 & -0,0337 \end{bmatrix} u \tag{3.8}$$

### 3.1.2 Stabilność przybliżenia liniowego

Według pierwszej metody Lapunowa układ dynamiczny jest stabilny wtedy i tylko wtedy, gdy wszystkie wartości własne $\lambda_i$ macierzy systemowej $A$ przybliżenia liniowego mają nie dodatnie części rzeczywiste, a każda wartość własna o zerowej części rzeczywistej jest wartością własną jednokrotna [2].

W przypadku wyznaczonego przybliżenia liniowego system nie jest stabilny. Obliczenia wykonano w środowisku Matlab przy użyciu funkcji *eig*

```
eig(A)
```

$$\begin{aligned} \lambda_1 &= 0 \\ \lambda_2 &= 0 \\ \lambda_3 &= 5.8226 \\ \lambda_4 &= -5.8415 \\ \lambda_5 &= -0.0486 \\ \lambda_6 &= -0.0771 \end{aligned} \tag{3.9}$$

Jak widać $Re(\lambda_3) > 0$ zatem potwierdza się hipoteza o niestabilności układu.

### 3.1.3 Sterowalność przybliżenia liniowego

Układ dynamiczny jest sterowalny wtedy i tylko wtedy, gdy $rank(\Omega) = n$. Macierz $\Omega$ ($\Omega = [B; AB; \ldots; A^{n-1}B]$) to macierz sterowalności Kalmana [2]. Jej wyznaczenia dokonano w środowisku Matlab przy użyciu funkcji *ctrb*

```
ctrb(A,B)
```

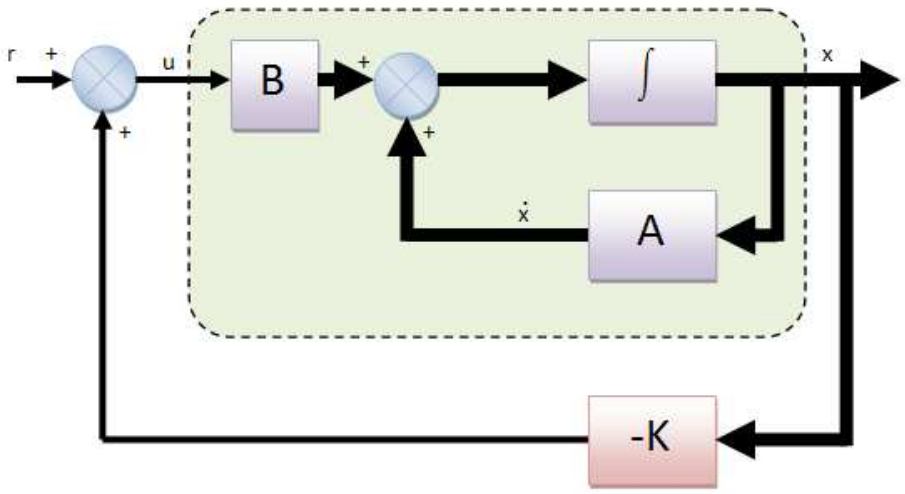
$$rank([B; AB; \ldots; A^{n-1}B]) = 6 \tag{3.10}$$

Jak widać, system (3.8) jest sterowalny.

## 3.2 Projektowanie sterownika liniowego

Na potrzeby sterowania robotem balansującym zaprojektowano sterownik liniowy korzystając z dwóch metod. Pierwszy z nich to sterownik minimalno-kwadratowy, drugi to sterownik rozmieszczający bieguny. Teoria regulacji korzysta z idei sprzężenia zwrotnego. Wektor stanu przemnożony przez odpowiednią macierz wzmocnień $K$ podawany jest na wejście systemu, jak pokazano na rysunku 3.1. Prawo sterowania ma postać

$$u = -Kx. \tag{3.11}$$




Rysunek 3.1 Schemat blokowy sterownika dla liniowej wersji systemu

Przybliżenie liniowe robota balansującego ma postać (3.7)

$$\dot{x} = Ax + Bu.$$

Stosując prawo sterowania (3.11) do modelu (3.7) otrzymano

$$\dot{x} = (A - BK)x = \tilde{A}x, \tag{3.12}$$

Aby układ z zamknięta pętlą sprzężenia zwrotnego był stabilny, musi byc spełniony warunek $\forall_i eig_i(\tilde{A}) < 0$.

### 3.2.1 Sterownik minimalno-kwadratowy (LQR)

Jednym z najczęściej używanych sterowników liniowych jest LQR. Podczas sterowania zakładamy, że system liniowy znajduje się w stanie równowagi. Celem sterowania jest utrzymanie systemu w położeniu równowagi, pomimo oddziałujących na niego zakłóceń. Zakładamy ponadto, że znany jest jego stan początkowy oraz wszystkie zmienne są mierzalne.

Należy wyznaczyć sterowanie $u$, które minimalizuje kwadratowy wskaźnik jakości:

$$\begin{aligned} J = \frac{1}{2}\int_0^T \left(x^T(t)Qx(t) + u^T(t)Ru(t)\right) dt, \\ Q = & Q^T \geq 0, \\ R = & R^T > 0, \end{aligned} \tag{3.13}$$

gdzie Q i R są stałymi diagonalnymi nastawami regulatora. W praktyce dobiera się wartości elementów macierzy $Q$ i $R$ w następujący sposób:

$$\begin{aligned} Q_{ii} &= 1/\text{max. akceptowalna wartość } [x_i^2] \\ R_{ii} &= 1/\text{max. akceptowalna wartość } [u_i^2] \end{aligned} \tag{3.14}$$

Forma kwadratowa $x^TQx$ we wskaźniku jakości $J$ odpowiada kosztowi, jaki ponosi układ ze względu błąd położenia x (opisuje błąd położenia i prędkości w stanie braku równowagi układu). Człon $u^TRu$ opisuje koszt energrtyczny zastosowanego sterowania.

**(UWAGA! Oznaczenia Q,R nie są tymi samymi oznaczeniami jakie występują w poprzednich rozdziałach. Zostały użyte tylko przy opisie sterownika LQR.)**

Prawo sterowania w sterowniku LQR ma postać

$$u = -R^{-1}B^TP, \tag{3.15}$$

gdzie $P$ uzyskuje się poprzez rozwiązanie następującego równania Riccatiego

$$0 = PA + A^TP - PBR^{-1}B^TP + Q. \tag{3.16}$$

Macierz wzmocnień $K$ opisana jest równaniem

$$K = R^{-1}B^TP \tag{3.17}$$

i określa w całości postać sprzężenia zwrotnego dla opisywanego systemu.

W pracy do wyznaczenia macierzy $K$ użyto środowiska Matlab i dostępnej w nim funkcji *lqr*.

```
[K, P, ev] = lqr(A,B,Q,R)
```

gdzie

$K$ −macierz wzmocnień sterownika
$P$ −macierz Lapunowa,
$ev$ −wektor wartości własnych układu zamkniętego.

Przykład wyliczenia nastaw sterownika jest następujący. Za $Q$ i $R$ przyjęto macierze

$$Q = \begin{bmatrix} a_Q & 0 & 0 & 0 & 0 & 0 \\ 0 & b_Q & 0 & 0 & 0 & 0 \\ 0 & 0 & c_Q & 0 & 0 & 0 \\ 0 & 0 & 0 & d_Q & 0 & 0 \\ 0 & 0 & 0 & 0 & e_Q & 0 \\ 0 & 0 & 0 & 0 & 0 & f_Q \end{bmatrix} \tag{3.18}$$

gdzie $a_Q = b_Q = c_Q = d_Q = e_Q = f_Q = 1$, zaś

$$R = \begin{bmatrix} a_R & 0 \\ 0 & b_R \end{bmatrix} \tag{3.19}$$

gdzie $a_R = b_R = 1$

Otrzymana macierz wzmocnień miała postać

$$K = 1.0e3 * \begin{bmatrix} -0.0000 & -0.0006 & -0.0010 & -0.0030 & -1.1322 & -0.2003 \\ -0.0010 & -0.0030 & -0.0000 & -0.0006 & -1.1322 & -0.2003 \end{bmatrix} \tag{3.20}$$

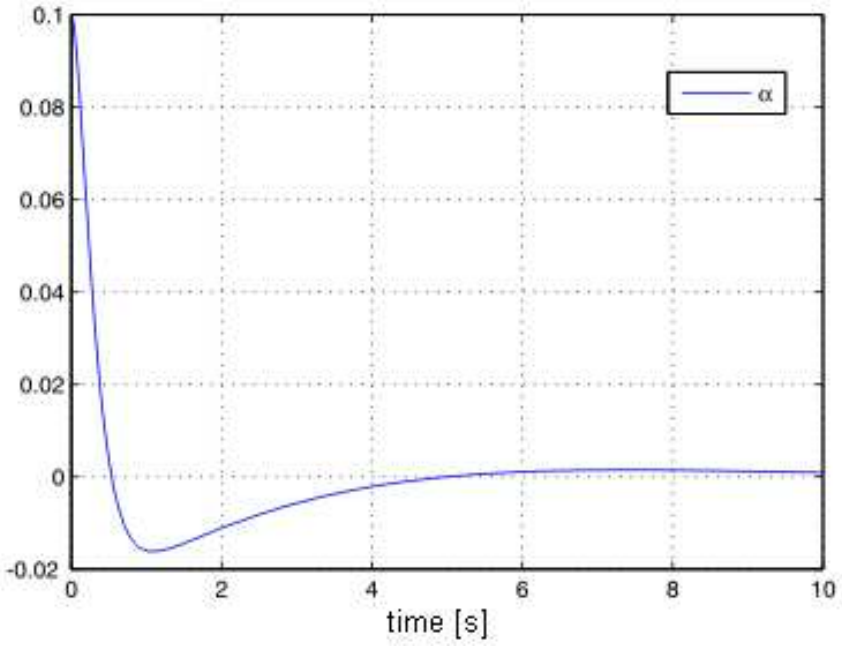
Wartości własne układu z zamkniętą pętlą sprzężenia (3.7) są następujące

$$\begin{aligned} \lambda_1 &= -5.9015, \\ \lambda_2 &= -5.7678, \\ \lambda_3 &= -0.4432 + 0.3710i, \\ \lambda_4 &= -0.4432 - 0.3710i, \\ \lambda_5 &= -0.3427 + 0.3054i, \\ \lambda_6 &= -0.3427 - 0.3054i. \end{aligned} \tag{3.21}$$

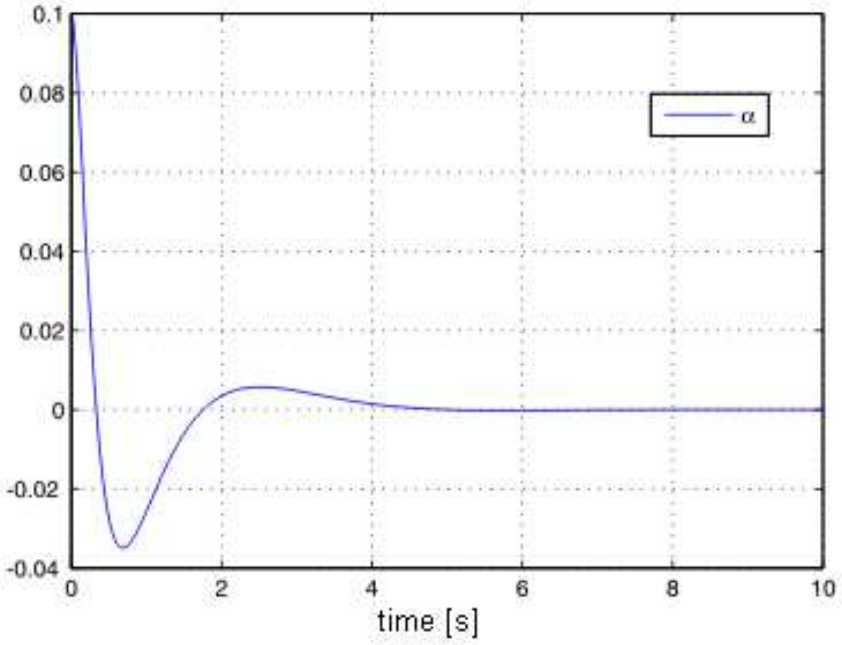
Wyniki symulacji przedstawiono na rysunku 3.2. Aby zaobserwować jak szybko sterownik doprowadza układ do stanu równowagi, przyjęto początkową wartość wychylenia równą

$$\alpha(0) = 0.1 \tag{3.22}$$

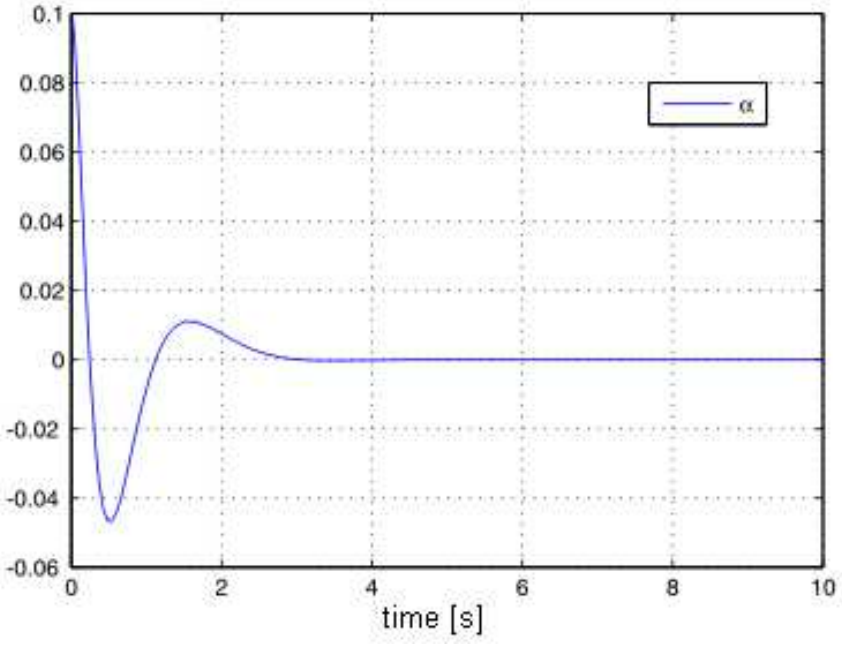
Zaobserwowano, że wraz ze wzrostem współczynników macierzy Q malał czas powrotu do stanu równowagi. Wzrastało także przeregulowanie.



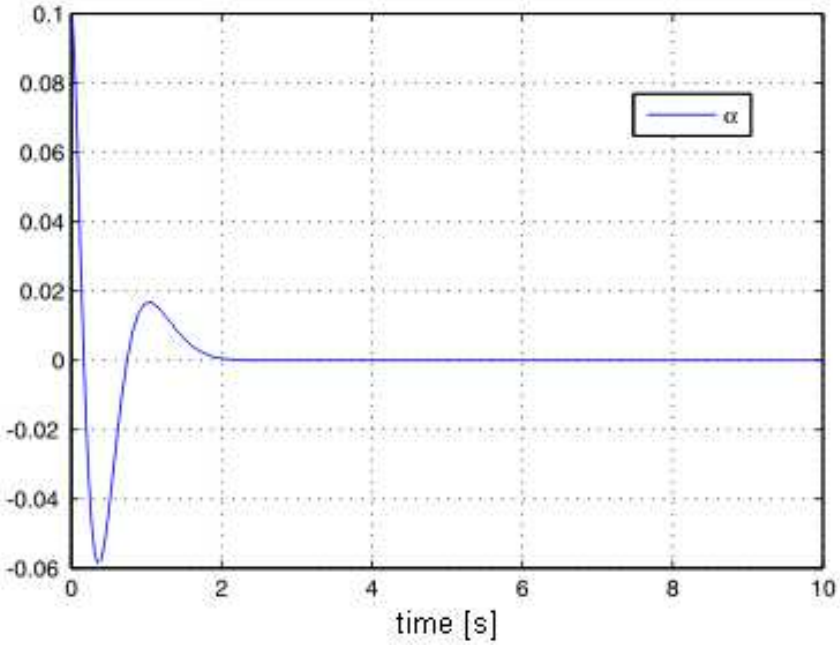

Rysunek 3.2 Wychylenie wahadła, przy użyciu sterownika LQR dla parametrów $Q = I_6$ i $R = I_2$



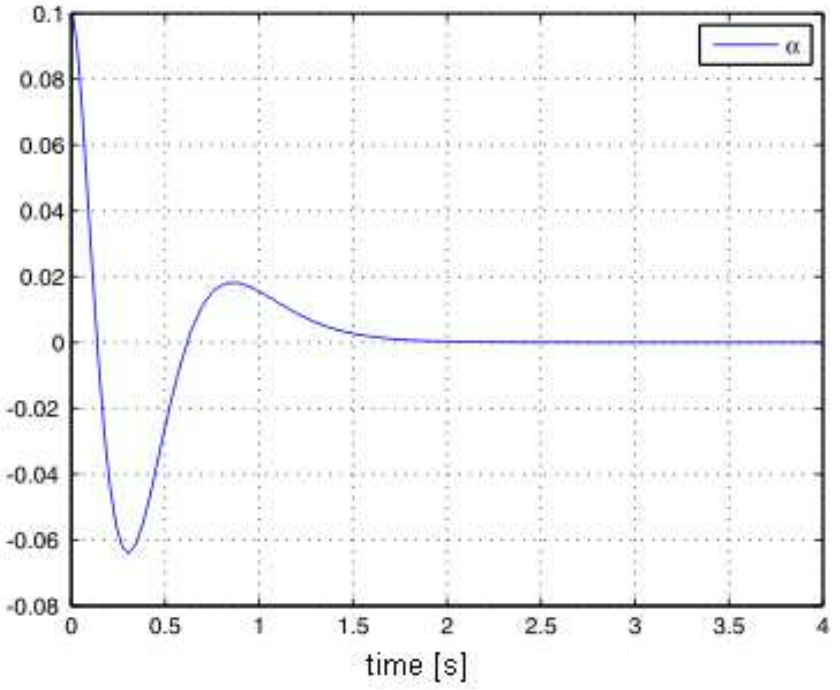

Rysunek 3.3 Wychylenie wahadła,przy użyciu sterownika LQR dla parametrów $a_Q = c_Q = e_Q = 10$, $b_Q = d_Q = f_Q = 1$ oraz $R = 0.1 \cdot I_2$




Rysunek 3.4 Wychylenie wahadła, przy użyciu sterownika LQR dla parametrów $a_Q = c_Q = e_Q = 100$, $b_Q = d_Q = f_Q = 1$ oraz $R = 0.1 \cdot I_2$




Rysunek 3.5 Wychylenie wahadła, przy użyciu sterownika LQR dla parametrów $a_Q = c_Q = e_Q = 1000$, $b_Q = d_Q = f_Q = 1$ oraz $R = 0.1 \cdot I_2$




Rysunek 3.6 Wychylenie wahadła, przy użyciu sterownika LQR dla parametrów $a_Q = c_Q = e_Q = 3000$, $b_Q = d_Q = f_Q = 1$ oraz $R = 0,1 \cdot I_2$

### 3.2.2 Sterownik rozmieszczający bieguny (PP).

Metoda sterowania polega na wyznaczeniu takiej macierzy $K$ danej równaniem (3.7), aby wynikowa macierz $\tilde{A}$, równa

$$\tilde{A} = A - BK$$

miała wartości własne spełniające warunek $\forall_i eig_i(\tilde{A}) < 0$. Ich optymalne położenia wyznacza się metodą prób i błędów. Oceny dokonuje się poprzez obserwację sygnału wyjściowego, który powinien mieć najmniejsze odkształcenie od sygnału wejściowego. Jeżeli celem jest szybka odpowiedź, to bieguny powinny leżeć daleko od osi urojonej charakterystyki fazowo-częstotliwościowej. Trzeba pamiętać, że szybka odpowiedź na zadane wymuszenie będzie wymagać silnych napędów robota.

Macierz $K$ wyznaczono w środowisku Matlab przy użyciu funkcji *place*.

```
K = place(A,B,p)
```

gdzie

$p$ wektor biegunów otrzymanych przy sprzężeniu zwrotnym (p=eig(A-BK))

Przykład wyliczenia nastaw sterownika jest następujący. Za $p$ przyjęto

$$p = [a \quad b \quad c \quad d \quad e \quad f]. \tag{3.23}$$

gdzie $a = -1$, $b = -2$, $c = -3$, $d = -4$, $e = -5$, $f = -6$.

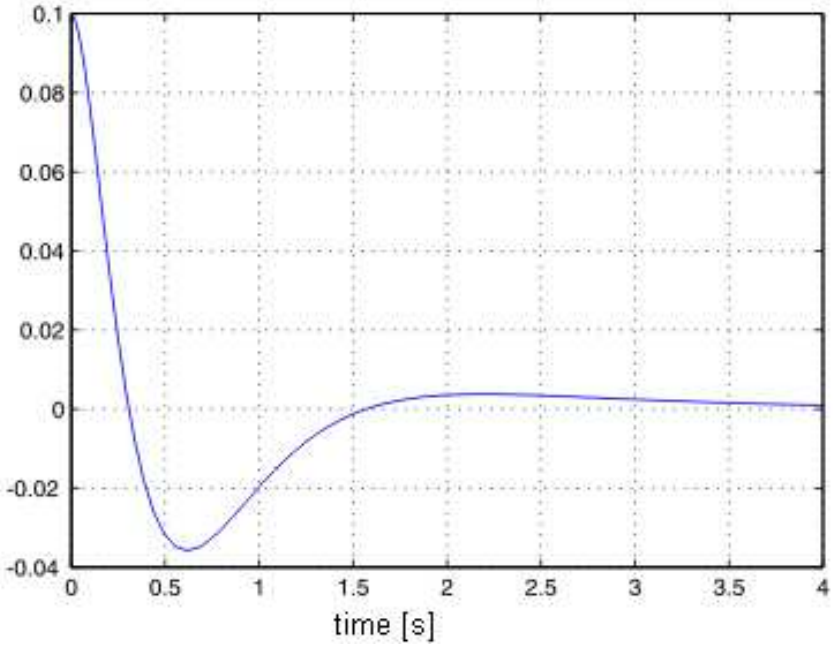
Stabilność systemu będzie zagwarantowana tak długo, jak długo część rzeczywista $eig\tilde{A}$ będzie mniejsza od zera. Macierz wzmocnień $K$ dla powyższych parametrów ma postać:

$$K = 1.0e3 \begin{bmatrix} 0.0168 & 0.0106 & -0.0120 & -0.0093 & -0.7699 & -0.1272 \\ -0.0299 & -0.0298 & 0.0035 & -0.0081 & -2.3509 & -0.4674 \end{bmatrix} \tag{3.24}$$

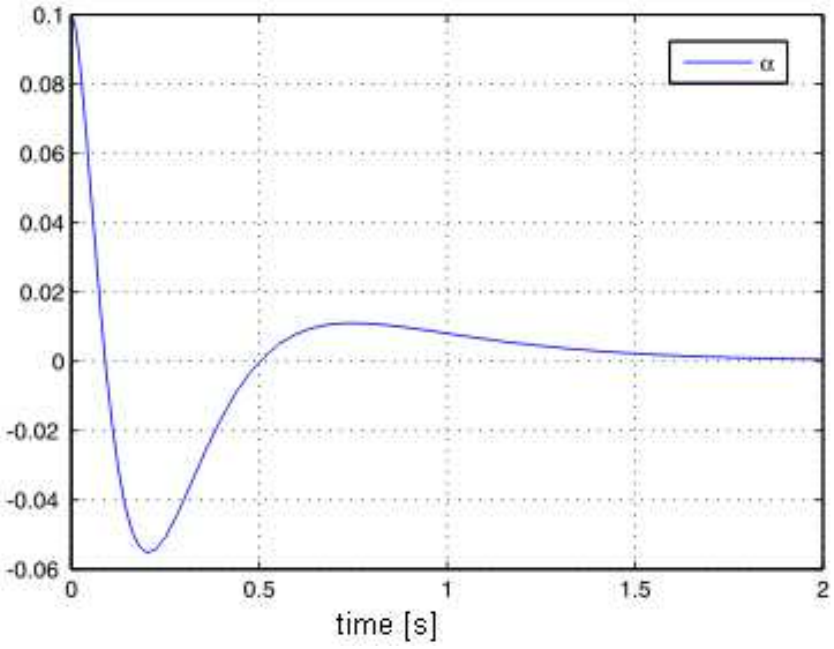
Wyniki symulacji przedstawiono poniżej. Aby zaobserwować jak szybko sterownik doprowadza układ do stanu równowagi, przyjęto wychylenie początkowe

$$\alpha(0) = 0.1 \tag{3.25}$$

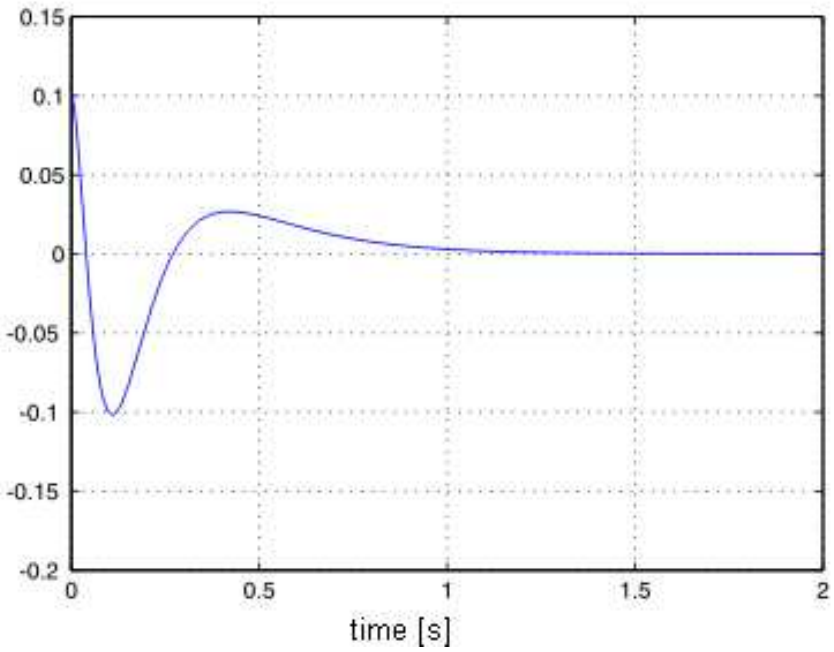
Zaobserwowano, że wraz z oddalaniem biegunów od osi urojonej malał czas powrotu do stanu równowagi. Wzrastało także przeregulowanie.




Rysunek 3.7 Wychylenie wahadła przy użyciu sterownika PP dla biegunów $\tilde{A}$ równych $a = -1$, $b = -2$, $c = -3$, $d = -4$, $e = -5$, $f = -6$




Rysunek 3.8 Wychylenie wahadła przy użyciu sterownika PP dla biegunów $\tilde{A}$ równych $a = -3$, $b = -6$, $c = -9$, $d = -12$, $e = -15$, $f = -18$

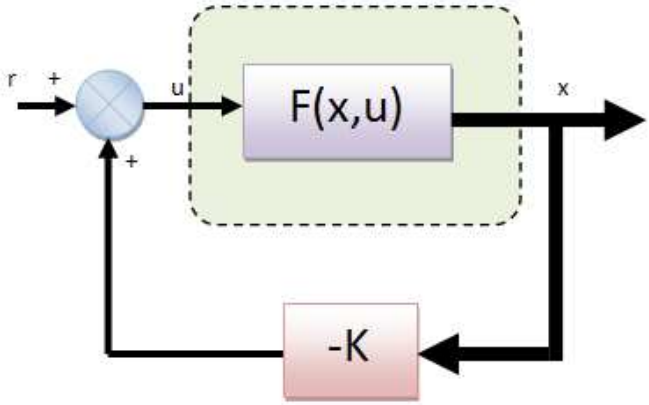



Rysunek 3.9 Wychylenie wahadła przy użyciu sterownika PP dla biegunów $\tilde{A}$ równych $a = -5$, $b = -10$, $c = -15$, $d = -20$, $e = -25$, $f = -30$

### 3.2.3 Zastosowanie liniowego sterownika do nieliniowego modelu robota

Po wyznaczeniu optymalnych sterowników dla systemów liniowych podjęto próbę przetestowania ich na nieliniowym modelu dynamiki robota. Założono, że przy niewielkich odchyleniach wahadła od pionu sterownik powinien pracować prawidłowo.

Na wykresach przedstawiających odchylenie wahadła od pionu zaobserwowano, że strojenie sterownika w kierunku zwiększenia szybkości działania powodowało wzrost przeregulowań. Wzrasta także zapotrzebowanie na moment napędowy aktuatorów. Sterownik rozmieszczający bieguny działał zdecydowanie szybciej. Po czasie 1s robot z niezerowego stanu początkowego odzyskiwał równowagę.

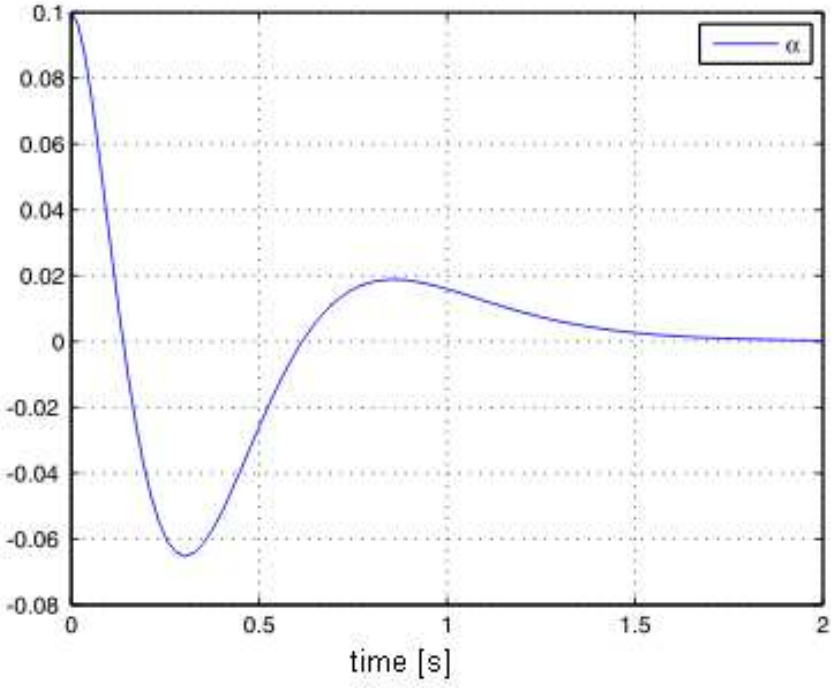



Rysunek 3.10 Schemat blokowy sterownika dla nieliniowego systemu

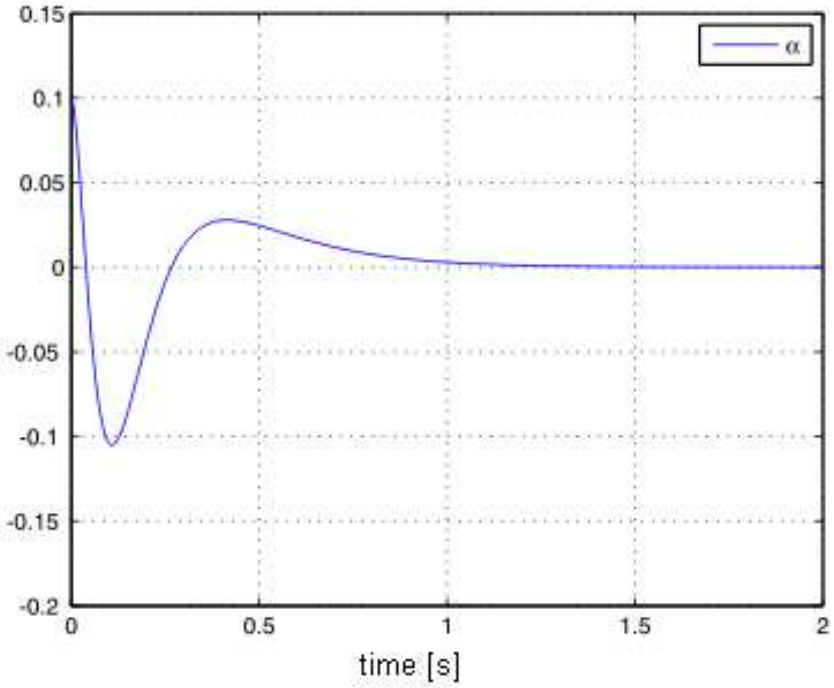
Zgodnie z rysunkiem 3.10 przetestowano sterowniki liniowe na nieliniowym modelu robota balansującego. Wybrano po jednej, najszybciej działającej wersji sterownika zaprojektowanej metodą PP oraz LQR. Aby zaobserwować czy sterownik poprawnie doprowadza układ do stanu równowagi, przyjęto wychylenie początkowe równe

$$\alpha(0) = 0.1 \tag{3.26}$$

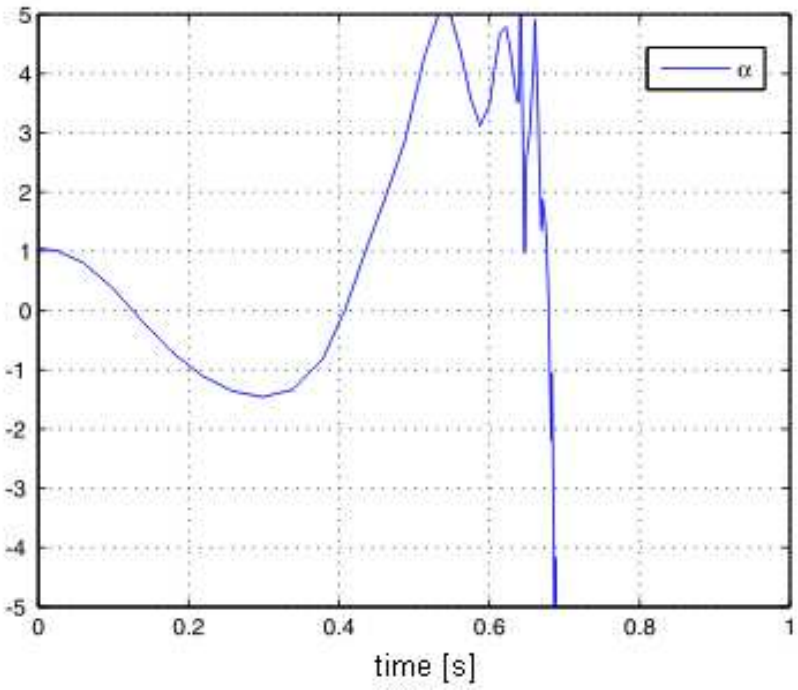
Jak widać z rysunków 3.11, 3.12 oraz 3.13 sterowniki działają poprawnie dla niewielkich odchyleń korpusu robota od pionu. Przy większych odchyleniach np. $\alpha(0) = \frac{\pi}{3}$, robot nie jest w stanie osiągnąć stanu równowagi co pokazano na rysunku 3.13.




Rysunek 3.11 Wychylenie wahadła nieliniowego systemu przy użyciu sterownika LQR dla parametrów $a_Q = c_Q = e_Q = 3000$, $b_Q = d_Q = f_Q = 1$, $a_R = b_R = 0,1$, $\alpha(0) = 0.1$




Rysunek 3.12 Wychylenie wahadła nieliniowego systemu przy użyciu sterownika PP dla biegunów $\tilde{A}$ równych $a = -5$, $b = -10$, $c = -15$, $d = -20$, $e = -25$, $f = -30$




Rysunek 3.13 Wychylenie wahadła nieliniowego systemu przy użyciu sterownika LQR dla parametrów $a_Q = c_Q = e_Q = 3000$, $b_Q = d_Q = f_Q = 1$, $a_R = b_R = 0,1$, $\alpha(0) = \frac{\pi}{3}$

# Rozdział 4

# Nieliniowy sterownik

## 4.1 Projektowanie sterownika nieliniowego

W poprzednim rozdziale zaproponowano sterowniki liniowe, które działają poprawnie jedynie w okolicy punktu równowagi. Założono, że w punkcie tym wszystkie prędkości i przyspieszenia mają wartość zerową. Nie mniej jednak model dynamiki robota balansującego jest jak najbardziej nieliniowy. W rozdziale tym zaprezentowano nieliniowy system sterowania, który pozwoli osiągać punkt równowagi nawet wtedy, gdy korpus robota znacznie się odchyli od pionu przy niezerowych wartościach prędkości i przyspieszeń. Zaprojektowany sterownik umożliwi także śledzenie trajektorii przez odwrócone mobilnego wahadło.

**Ponownie rozważmy model dynamiki robota, tym razem jednak przyjęto, że jest to manipulator mobilny z jednym pasywnym przegubem (stopień swobody pozbawiony napędu) (2.13). Oznacza to, że teraz traktujemy system tak, że napędy oddziaływają jedynie na koła.**

Wprowadzając macierz $B$ pochodzącą z równań (2.13), otrzymamy w ten sposób nieco inny model, częściowo niedosterowany

$$\begin{bmatrix} Q_{11}^* & Q_{12}^* \\ Q_{21}^* & Q_{22}^* \end{bmatrix} \begin{pmatrix} \dot{\eta} \\ \ddot{\alpha} \end{pmatrix} + \begin{pmatrix} F_1^* \\ F_2^* \end{pmatrix} = \begin{bmatrix} B^* \\ 0 \end{bmatrix} \begin{pmatrix} \tau \\ 0 \end{pmatrix}. \tag{4.1}$$

### 4.1.1 Częściowa linearyzacja globalna

W rozważanym modelu rolę pasywnego przegubu spełnia odwrócone wahadło. Dla tego typu obiektów możliwe jest zastosowanie tzw. częściowej linearyzacji globalnej [10], transformującej model, z punktu widzenia sterowania, do wygodniejszej postaci.

Przekształcając drugie równanie (4.1) w celu wydzielenia wyrażenia na $\ddot{\alpha}$

$$\ddot{\alpha} = -(Q_{22}^*)^{-1}(Q_{21}^*\dot{\eta} + F_2^*) \tag{4.2}$$

i wstawiając do pierwszego równania [11] uzyskamy zapis

$$\bar{Q}(q)\dot{\eta} + \bar{F}(q,\dot{q}) = B^*\tau, \tag{4.3}$$

$$\ddot{\alpha} = -(Q_{22}^*), \tag{4.4}$$

gdzie

$$\begin{aligned} \bar{Q}(q) &= Q_{11}^*(q) - Q_{12}^*(q)Q_{22}^*(q)^{-1}Q_{21}^*(q), \\ \bar{F}(q,\dot{q}) &= F_1^*(q,\dot{q}) - Q_{12}^*(q)Q_{22}^*(q)^{-1}F_2^*(q,\dot{q}). \end{aligned}$$

Podstawiając do części sterowanej układu prawo sterowania

$$\tau = (B^*)^{-1}\left\{\bar{Q}(q)u + \bar{F}(q,\dot{q})\right\}, \tag{4.5}$$

otrzymujemy model (4.1) częściowo zlinearyzowany system o postaci

$$\begin{aligned} \ddot{\alpha} &= -(Q_{22}^*)^{-1}(Q_{21}^*\dot{\eta} + F_2^*), \\ \dot{\eta} &= u. \end{aligned} \tag{4.6}$$

Z punktu widzenia sterowania, dynamika została zlinearyzowana, a przekształcone równanie dla wahadła jest rodzajem ograniczeń. System ten jest punktem wyjścia do projektowania nieliniowego algorytmu sterowania pozwalającego nie tylko na utrzymywanie położenia równowagi, ale także na śledzenie zadanej trajektorii.

### 4.1.2 Nieliniowy sterownik

W celu zaprojektowania sterownika należy rozważyć model jako dwa podsystemy, którymi będziemy sterować na poziomie kinematyki i dynamiki. Na poziomie kinematyki uzyskujemy prędkości referencyjne $\eta_r$, które zapewnią śledzenie zadanej trajektorii $\alpha_d(t)$ dla korpusu (odwróconego wahadła). Na poziomie dynamicznym uzyskujemy częściowo zlinearyzowane sterowanie $u$, które reguluje prędkością $\eta$ w taki sposób, aby śledzić prędkości referencyjne $\eta_r$ uzyskanych w sterowniku kinematycznym.

#### Sterownik kinematyczny

Niech równanie sterownika ma następująca postać

$$-(Q_{22}^*)^{-1}(Q_{21}^*\dot{\eta}_r + F_2^*) = \ddot{\alpha}_d - K_d(\dot{\alpha} - \dot{\alpha}_d) - K_p(\alpha - \alpha_d), \qquad K_d, K_p > 0. \tag{4.7}$$

Niestety, powyższe równanie jest skalarne, natomiast $\eta_r$ jest dwuwymiarowe. Jednakże można przyjąć założenie, że robot utrzymując równowag porusza się tylko po linii prostej. Zatem relacja pomiędzy prędkościami jest następująca $\eta_{1r} = \eta_{2r}$. Przy takim założeniu z równania (4.7) można wyznaczyć przyspieszenia referencyjne $\dot{\eta}_r$. W tym celu należy najpierw wyliczyć $\dot{\eta}_{1r} = \dot{\eta}_{2r}$ z równania analitycznego

$$\dot{\eta}_{1r} = \frac{Q_{22}^* K_p(\alpha - \alpha_d) + Q_{22}^* K_d(\dot{\alpha} - \dot{\alpha}_d) - Q_{22}^*\ddot{\alpha} - F_2}{2Q_{21}^*} \tag{4.8}$$

#### Sterownik dynamiczny

W praktyce rzeczywiste prędkości kół różnią się od prędkości referencyjnych uzyskanych w sterowniku kinematycznym.Zastosujmy nowy algorytm sterowania, który zagwarantuje asymptotyczne śledzenie trajektorii we wszystkich współrzędnych mobilnego odwróconego wahadła. Prawo sterowania ma postać

$$u = \dot{\eta}_r - K_m e_\eta, \qquad K_m > 0, \tag{4.9}$$

gdzie $K_m$ jest pewną macierzą regulacji, zaś

$$e_\eta = \eta - \eta_r = \begin{pmatrix} \eta_1 - \eta_{1r} \\ \eta_2 - \eta_{2r} \end{pmatrix}$$

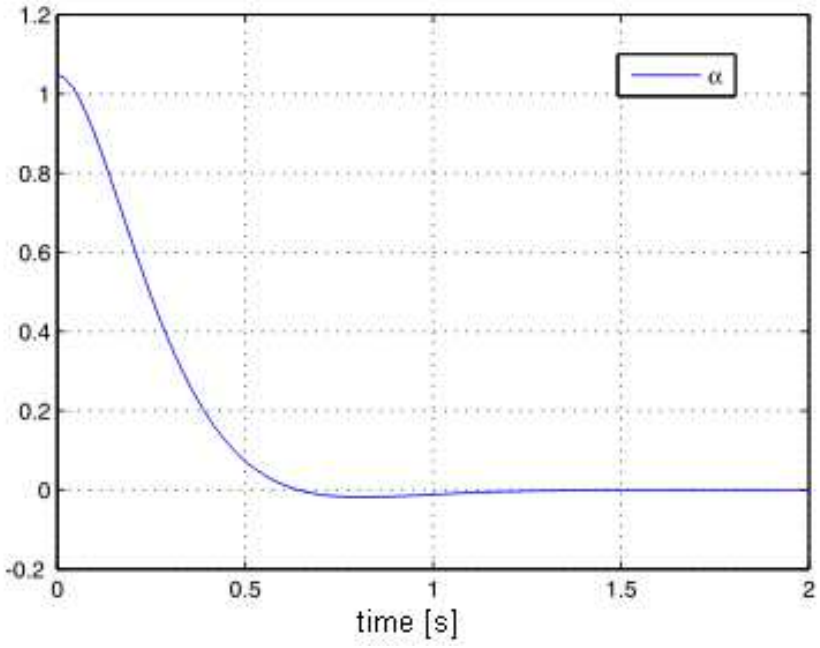
jest błędem występującym na poziomie dynamicznym, kiedy prędkości kół nie są równe prędkościom referencyjnym, np. w pierwszej fazie ruchu, na ogół zachodzi $\eta(0) \neq \eta_r(0)$.

## 4.2 Symulacje

Zaprojektowany nieliniowy sterownik dynamiczny w przeciwieństwie do liniowej wersji, powinien działać poprawnie także przy dużych wartościach kąta $\alpha$. Liniowe sterowniki doprowadzały wahadło do równowagi ze stanu nie większego niż $\frac{\pi}{4}$. Na wartość graniczną $\alpha$ miały także wpływ nastawy regulatorów.

Na rysunku 4.1 przedstawiono wyniki symulacji systemu dla nastaw i warunków początkowych równych

$$\begin{aligned} K_p &= 40, \\ K_d &= 10, \\ K_m &= 10, \\ \alpha_d &= 0, \\ \alpha(0) &= \frac{pi}{3}, \\ \eta_{1r}(0) &= 0, \\ \eta_{2r}(0) &= 0. \end{aligned} \tag{4.10}$$

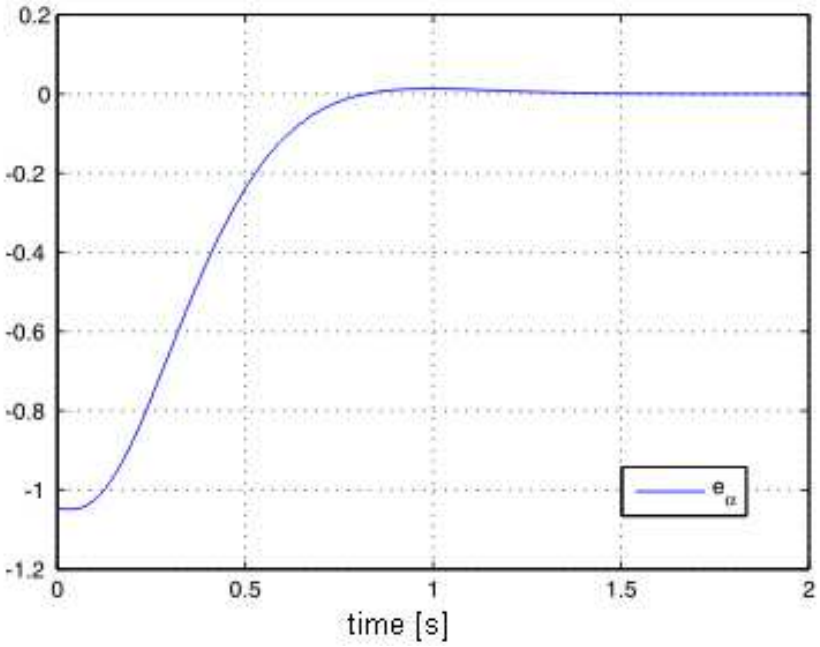



Rysunek 4.1 Wychylenie wahadła (system nieliniowy) przy użyciu nieliniowego sterownika

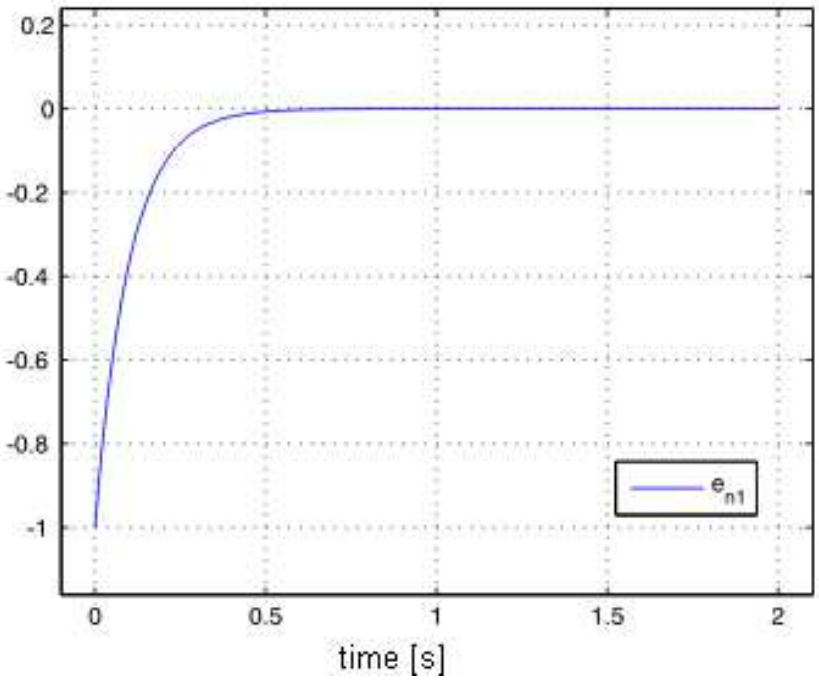
Na rysunkach 4.2 oraz 4.3 wyniki symulacji, w których śledzona jest niezerowa wartość wychylenia wahadła $\alpha_d = \frac{\pi}{3}$. Symulacje przeprowadzono dla nastaw i warunków początko-

wych równych

$$\begin{aligned} K_p &= 40, \\ K_d &= 10, \\ K_m &= 10, \\ \alpha_d &= \frac{\pi}{3}, \\ \alpha(0) &= 0, \\ \eta_{1r}(0) &= 1, \\ \eta_{2r}(0) &= 1. \end{aligned} \tag{4.11}$$



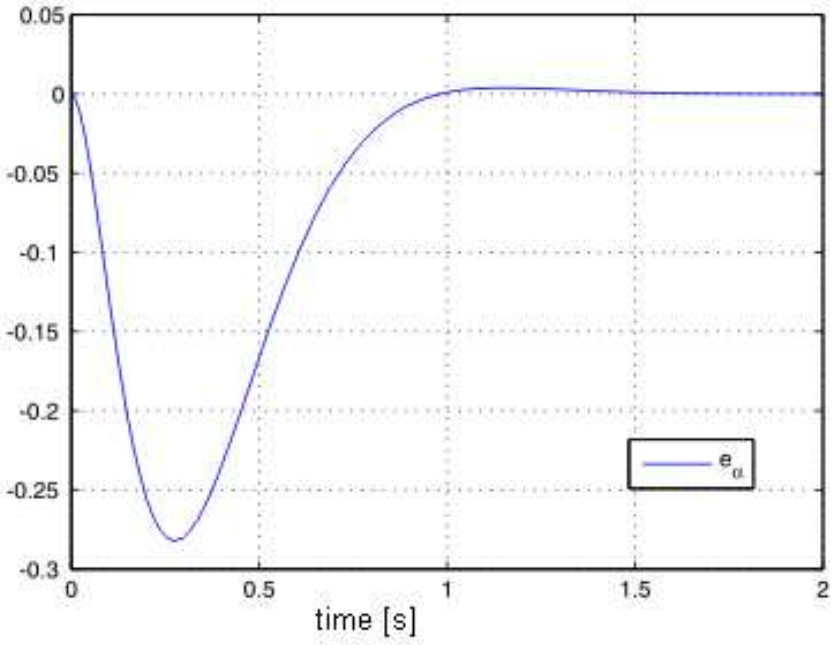

Rysunek 4.2 Błąd śledzenia zadanej stałej wartości kąta $\alpha_d$




Rysunek 4.3 Błąd śledzenia prędkości referencyjnych $e_{\eta 1}$

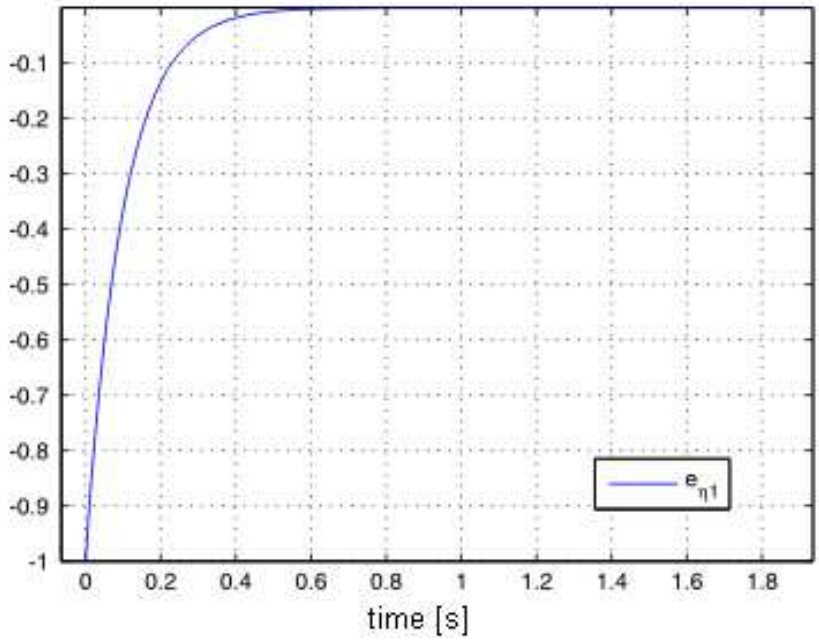
Zaprojektowany sterownik pozwala także na śledzenie kąta $\alpha_d$ zmiennego w czasie (śledzenie trajektorii). Na rysunkach 4.4 i 4.5 zaprezentowano wyniki symulacji opisywanego przykładu dla nastaw i warunków początkowych równych

$$\begin{aligned} K_p &= 40, \\ K_d &= 10, \\ K_m &= 10, \\ \alpha_d &= 0,05 \sin(\frac{t}{10}), \\ \alpha(0) &= 0, \\ \eta_{1r}(0) &= 1, \\ \eta_{2r}(0) &= 1. \end{aligned} \tag{4.12}$$




Rysunek 4.4 Błąd śledzenia zadanej stałej wartości kąta $\alpha_d$

Przeprowadzając powyższe symulacje pokazano, że zastosowanie nieliniowego sterownika jest znacznie szersze. Pozwala on nie tylko na utrzymywanie równowagi robota balansującego, ale także można go stosować do utrzymywania wychylenia ($\alpha_d \neq 0$) lub śledzenia trajektorii ($\alpha_d(t)$ korpusu. Ponadto zaprezentowany w pracy nieliniowy sterownik bez problemu radził sobie z osiągnięciem równowagi nawet, gdy wychylenie początkowe było równe $\alpha(0) = \frac{\pi}{3}$.

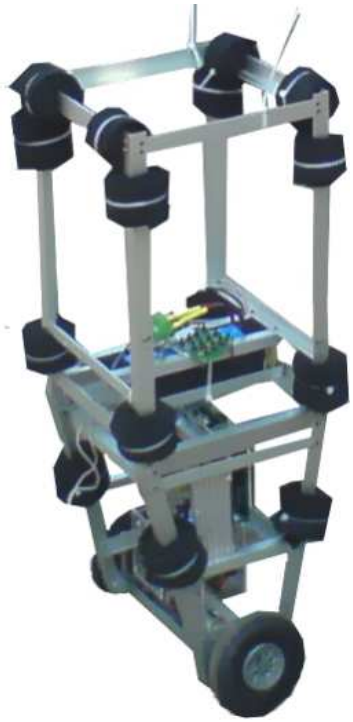



Rysunek 4.5 Błąd śledzenia prędkości referencyjnych $e_{\eta 1}$

# Rozdział 5

# Praktyczna realizacja układu sterowania

W pracy podjęto próbę budowy zamodelowanego robota balansującego. Zmierzono się z wieloma trudnościami w realizacji poszczególnych elementów składowych robota, jak: opracowanie i wykonanie konstrukcji nośnej, odpowiedni dobór napędów, użycie dostatecznie wydajnego sterownika silników prądu stałego, zapoznanie się i uruchomienie wydajnej jednostki centralnej MPC555 firmy Freescale, a także opracowanie odpowiedniego zestawu sensorów, niezbędnych do poprawnej pracy urządzenia. Dzięki temu eksperymentowi, udało się zweryfikować poprawność działania, rozważanego dotychczas jedynie teoretycznego modelu robota. Doświadczenie to dowodzi jak ważna, w procesie urzeczywistnienia teoretycznego modelu jest poprawna identyfikacja fizycznego obiektu. Trudności z pomiarem rzeczywistych wartości wektora stanu, a także niska jakość użytych do budowy robota napędów, pozwoliły jedynie na implementację prostych regulatorów PID.

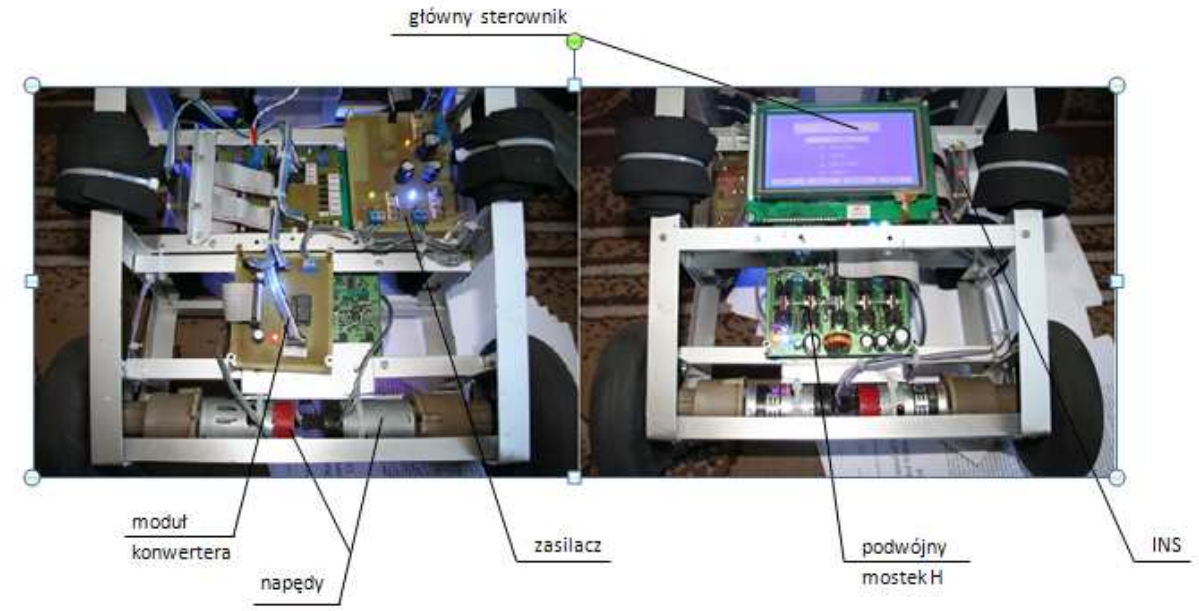


Rysunek 5.1 Robot „Kosmos”

Budowa robota ma charakter modułowy. Główny sterownik został tak zaprojektowany,

aby umożliwić łatwą rozbudowę robota o dodatkowe urządzenia peryferyjne. W pracy zdecydowano o wyposażeniu urządzenia w następujące moduły:

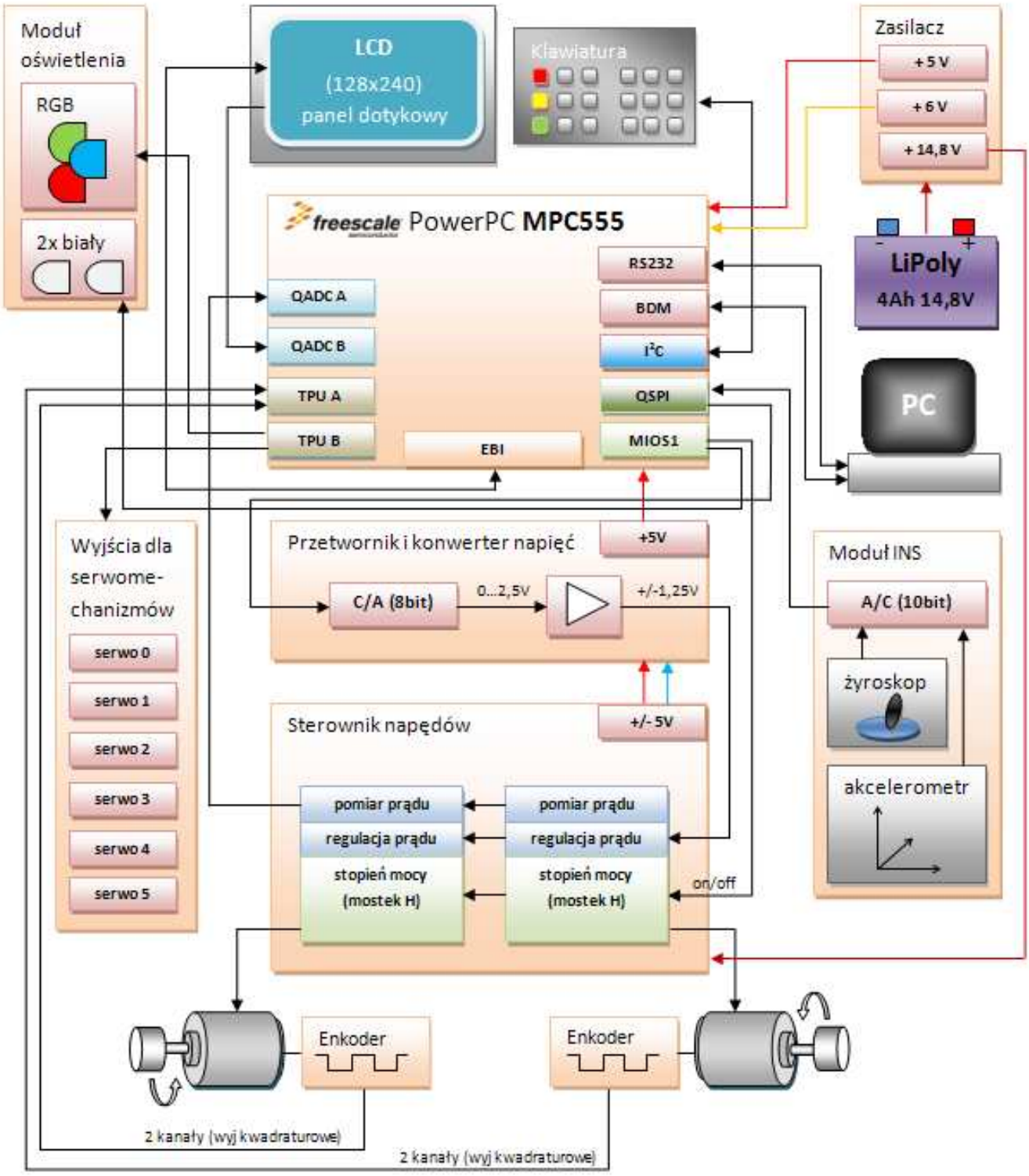
- główna jednostka sterująca CPU MPC555 firmy Freescale wraz z dotykowym, graficznym wyświetlaczem LCD,
- układ konwertera napięć z przetwornikiem cyfrowo-analogowym,
- podwójny mostek typu H z regulacją prądu, wyposażony w przetwornice napięcia +/- 5v,
- moduł zasilacza +5V oraz +6V,
- moduł INS,
- moduł oświetlenia RGB oraz kamery,
- klawiaturę 24-klawisze,
- pakiet zasilający, składający się z czterech ogniw litowo-polimerowych o pojemności 4800mAh,
- dodatkowo istnieje możliwość podłączenia sześciu serwomechanizmów.




Rysunek 5.2 Rozmieszczenie poszczególnych modułów na robocie (widok z tyłu i z przodu)

Schemat blokowy robota „Kosmos" przedstawiono na rysunku 5.3. Warto zwrócić uwagę na to, że do modułu głównego sterownika, doprowadzono aż trzy źródła zasilania. Zabieg ten pozwolił uniknąć propagacji zakłóceń pomiędzy poszczególnymi modułami zainstalowanymi w robocie.

Z uwagi na to, że tematem pracy było głównie zamodelowanie i przeprowadzenie symulacji zdecydowano szczegółowo opisać tylko niektóre moduły robota skonstruowanego na potrzeby eksperymentów. W tekście zamieszczono fragmenty kodu oprogramowania głównego sterownika, które zdaniem autora są najciekawsze. Część praktyczna jest wciąż rozwijana i udoskonalana.

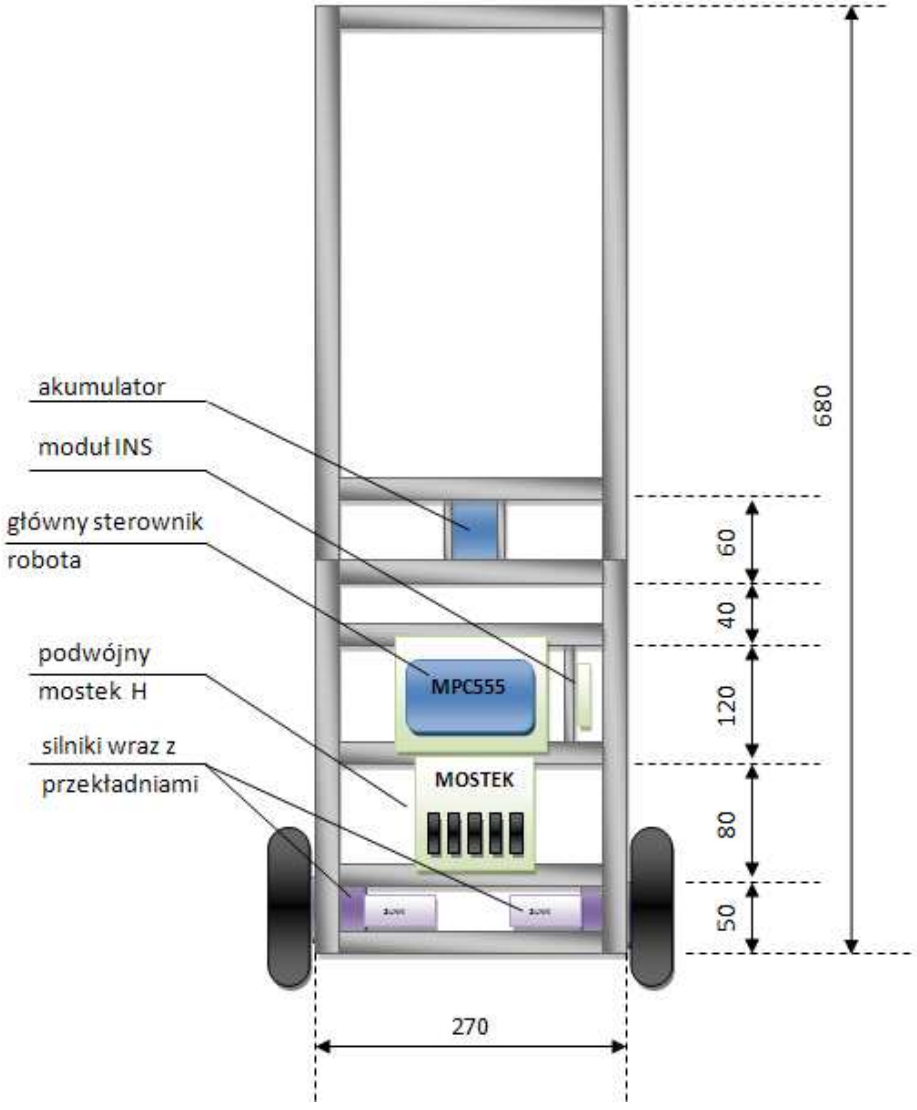



Rysunek 5.3 Schemat blokowy robota „Kosmos”

# 5.1 Konstrukcja mechaniczna

## 5.1.1 Budowa korpusu robota

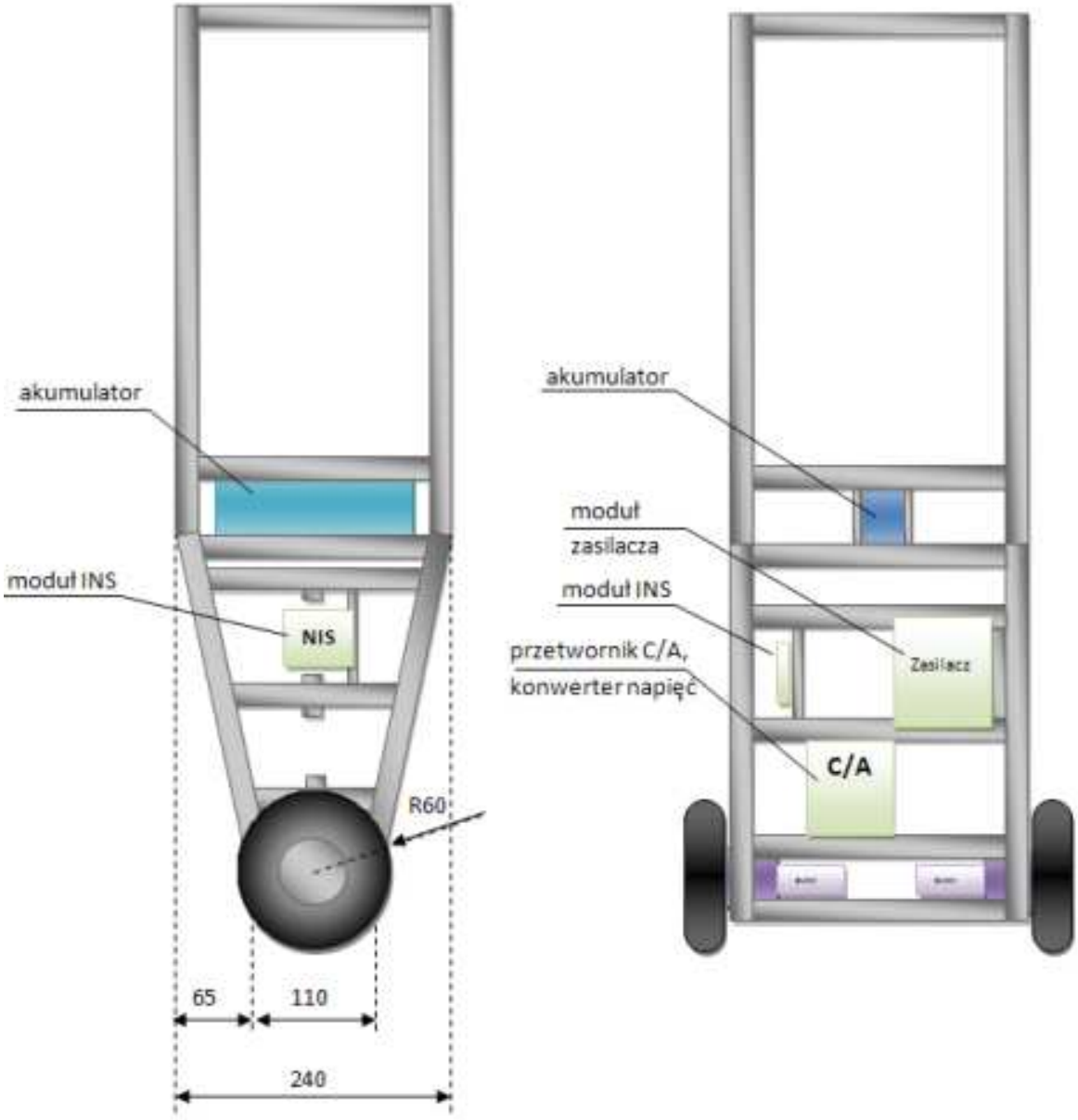
Do budowy robota użyto w głównej mierze kątowników aluminiowych w kształcie „L". Całość poskręcano śrubami o średnicy 3mm. Poszczególne moduły zamocowano na stalowych tulejach dystansowych. W celu chronienia konstrukcji, na stelaż nawinięto odbojniki wykonane z mikrogumy. W dolnej części korpusu zamocowano wszystkie moduły z elektroniką, tj. główny sterownik, moduł przetwornika wraz z konwerterem napięć, moduł INS, moduł zasilacza, podwójny mostek typu H. Najniżej zamocowano napędy wraz z kołami a w połowie korpusu umiejscowiono akumulator. Konstrukcja okazała się bardzo stabilna i wytrzymała na wielokrotne upadki. Wymiary korpusu prezentują rysunki 5.4 i 5.5.




Rysunek 5.4 Rysunek poglądowy (widok z przodu).

Dane fizyczne skonstruowanego modelu nieco się różnią od parametrów przyjętych

w trakcie symulacji w poprzednich rozdziałach. Wyniki symulacji są bardzo podobne, a ich formatowanie jest dość pracochłonne, autor zdecydował o pozostawieniu wyników symulacji, przeprowadzonych na podstawie pierwotnie przyjętych parametrów fizycznych.




Rysunek 5.5 Rysunek poglądowy (widok z boku i z tyłu).

Zmierzone nowe parametry fizyczne platformy (wahadła) oraz kół, które należy uwzględnić w modelu matematycznym:

| oznaczenie | wartość | jednostka | opis |
|---|---|---|---|
| $M_k$ | **0,176** | kg | masa koła |
| $M_w$ | **3.9** | kg | masa wahadła |
| $R$ | **0,06** | m | promień koła |
| $l$ | **0,68** | m | długość wahadła |
| $w$ | **0,035** | m | szerokość koła |
| $b$ | **0,15** | m | odległość koła od środka platformy |

### 5.1.2 Napędy robota

Robot balansujący został wyposażony w dwa silniki prądu stałego niemieckiej firmy Graupner. Użyte modele to Graupner 500E (rysunek 5.6). Jak większość silników modelarskich, charakteryzują się dużą sprawnością. Egzemplarz oznaczony literą „E" należy do silników typu ekonomicznego, czyli takiego, który dostarcza sporych momentów sił przy niewielkim zużyciu energii elektrycznej. Silniki te pracują na znamionowym napięciu 12V. Parametry silnika zamieszczono w tabeli poniżej.


Rysunek 5.6 Silnik 500E firmy Graupner

| opis | wartość | jednostka |
|---|---|---|
| napięcie znamionowe | 6...12 | V |
| prędkość | 12000 | obr/min |
| prąd bez obciążenia | 0,4 | A |
| prąd przy max sprawności | 2 | A |
| prąd w zwarciu | 10 | A |
| sprawność | 67 | % |
| grubość wału | 3,17 | mm |
| masa | 158 | g |
| wymiary (długość x średnica) | 50x35,8 | mm |

Tabela. 5.1 Parametry silnika

Napęd na koło został przeniesiony poprzez podwójną przekładnię planetarną. Przekładnie użyte w robocie, wymontowano z tanich wkrętarek akumulatorowych. Jak się okazało, ich niska jakość znacząco wpłynęła na jakość sterowania platformą i w dużym stopniu uniemożliwiła implementację większości rozważanych teoretycznie sterowników.

Zmierzone nowe parametry fizyczne pojedynczego napędu wraz z przekładnią, który należy uwzględnić w modelu matematycznym:

| oznaczenie | wartość | jednostka | opis |
|---|---|---|---|
| $k_m$ | **0,725** | N· m/A | współczynnik wytworzonego momentu |
| $k_e$ | **0,33** | V· s/rad | współczynnik wytworzonego napięcia |
| $R_s$ | **3** | Ω | rezystancja cewek silnika |
| $I_R$ | **0,0031** | kg· $m^2$ | moment bezwładności napędu |
| $n$ | **1:36** | – | przełożenie |


Rysunek 5.7 Pojedynczy napęd.

## 5.2 Konstrukcja elektroniczna

### 5.2.1 Czujniki pomiaru prędkości obrotowej kół

Na tylnej ściance każdego silnika zainstalowano magnetyczny enkoder AS5040 firmy Austriamicrosystems (rysunek 5.9 i 5.8) [14] [28]. Czujnik ten doskonale nadaje się do zastosowań w robotyce, do pomiarów prędkości obrotowej oraz położenia osi napędowych. Jest jednym z najmniejszych magnetycznych czujników obrotu na świecie. Wewnątrz struktury układu znajduje się procesor, połączony z matryca czujników pola magnetycznego (czujników Hall'a). Dzięki specjalnemu algorytmowi, procesor potrafi określić położenie linii sił pola magnetycznego (biegnących pomiędzy biegunami magnesu) względem owej struktury. Zapewniają bardzo precyzyjny bezkontaktowy pomiar obrotów, gdyż ich rozdzielczość wynosi 1024 impulsy na obrót. W czujniku wykorzystano wyjścia kwadraturowe i podłączono je do głównego sterownika. .

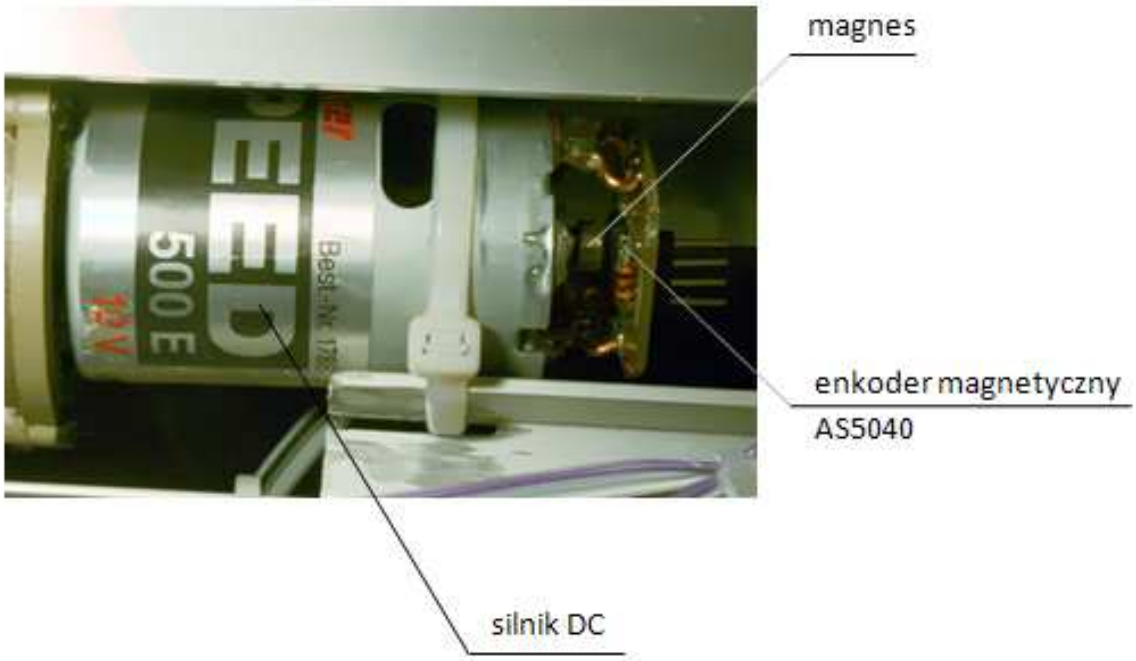
### 5.2.2 Prosty moduł INS

Do budowy modułu INS (Inertial Navigate System) użyto dwóch sensorów: trzyosiowy akcelerometr, jednoosiowy żyroskop, a także szybki, 10-cio bitowy przetwornik A/C. Pierwszy z czujników produkuje firma Freescale i jest to model MMA7260 [24]. Drugi czujnik produkowany jest przez firmę Analog Devices i jest to model ADXRS150 [25]. Do zamiany sygnałów analogowych na cyfrowe wykorzystano przetwornik A/C TLV2556 firmy Texas Instruments [26]. Poniżej przedstawiono główne parametry czujników oraz przetwornika.

| opis | wartość | jednostka |
|---|---|---|
| czułość | 800 | mV |
| wybór czułości | 1,5/2/4/6 | g |
| napięcie znamionowe | 3,3 | V |
| szumy | 4,7 | mV rms |
| typ obudowy | QFN | - |

Tabela. 5.2 Parametry akcelerometru MMA7260

Żyroskop, akcelerometr oraz przetwornik umieszczono na jednej płytce drukowanej. Niestety jeden z czujników wymaga zasilania o napięciu 3V3. Konieczne zatem było użycie dodatkowego stabilizatora, obniżającego napięcie zasilania modułu. Moduł ten

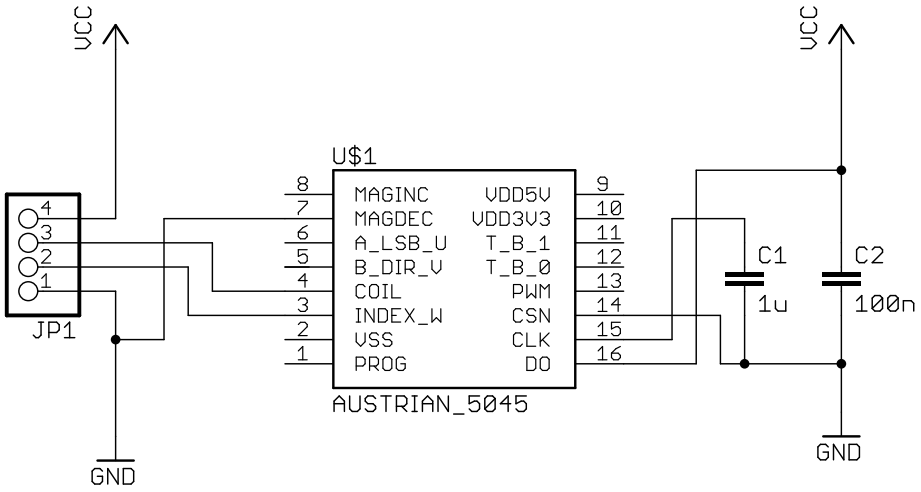



Rysunek 5.8 Mocowanie enkodera

| opis | wartość | jednostka |
|---|---|---|
| czułość | 12,5 | mV/$^o$/s |
| max częstotliwość | 40 | Hz |
| napięcie znamionowe | 5 | V |
| szumy | 0,05 | ± /svHz |
| typ obudowy | BGA | - |

Tabela. 5.3 Parametry żyroskopu ADXRS150

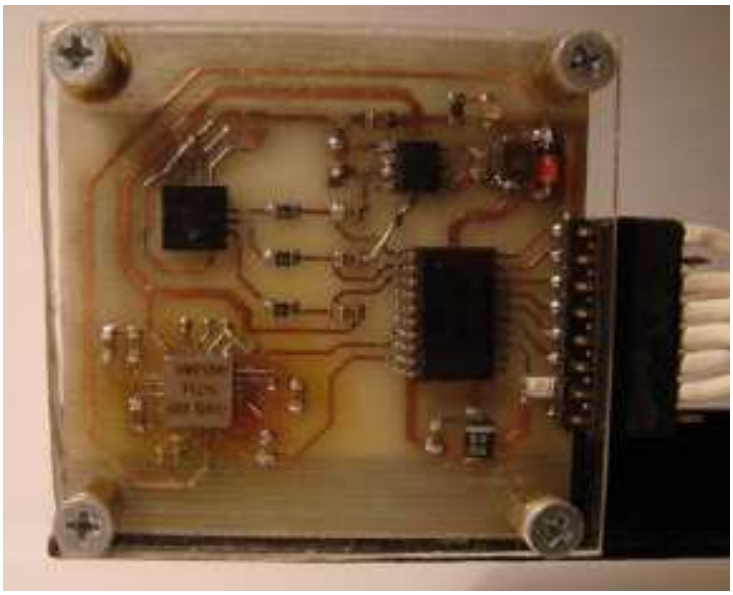
przedstawiono na rys. 5.10. Całość przymocowano do korpusu przy pomocy wkrętów i sprężyn, tak aby możliwa była regulacja kąta, pod którym moduł został zamocowany.



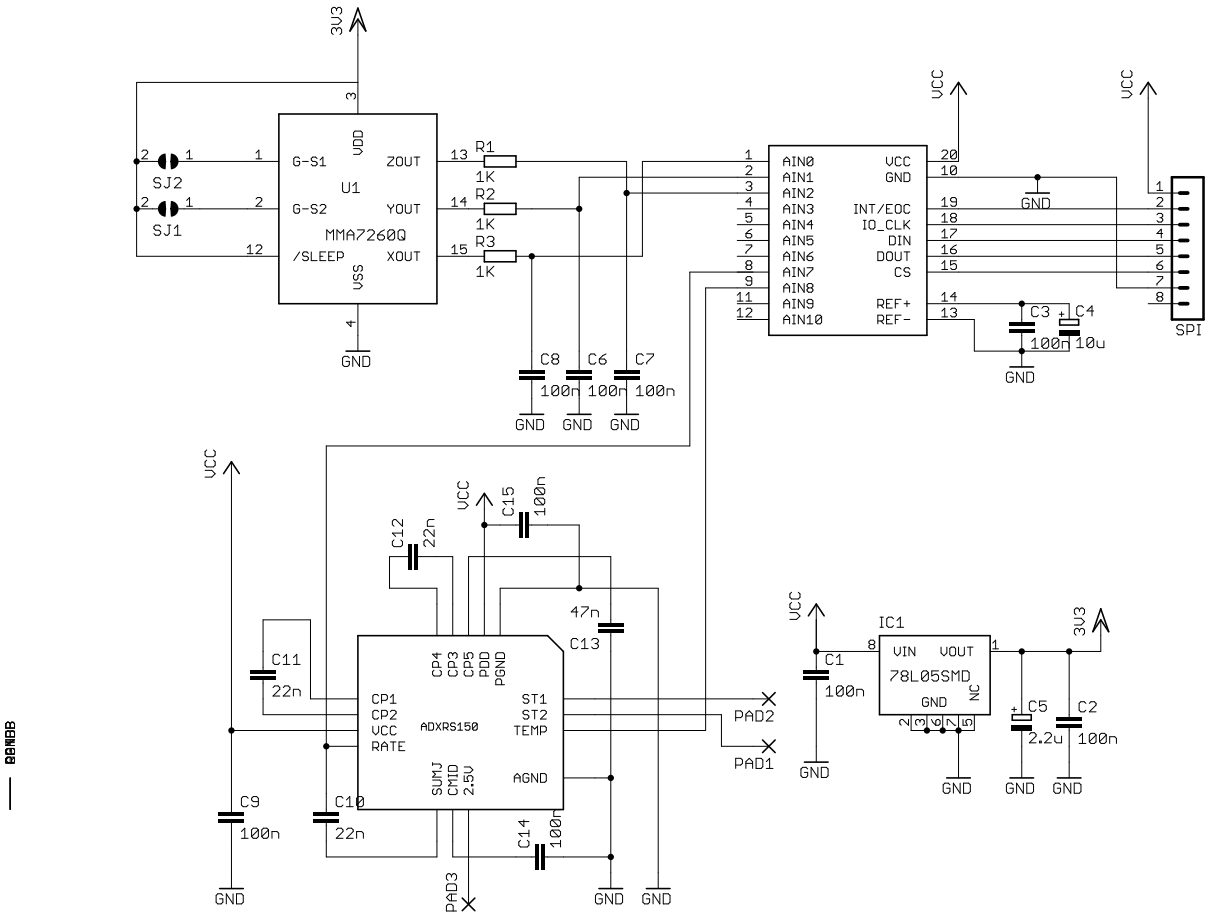

Rysunek 5.9 Schemat ideowy czujnika obrotów

| opis | wartość | jednostka |
|---|---|---|
| max częstotliwość | 200 | KSPS |
| rozdzielczość | 12 | bitów |
| napięcie znamionowe | 5 | V |
| liczba kanałów | 12 | - |
| interfejs | SPI | - |
| max częstotliwość interfejsu | 12 | MHz |
| napięcie referencyjne | wbudowane 4,098 | V |

Tabela. 5.4 Parametry przetwornika A/C TLV2556



Rysunek 5.10 Widok modułu nawigacyjnego INS




Rysunek 5.11 Schemat modułu nawigacyjnego INS

### 5.2.3 Sterownik napędów

Do sterowania napędami zdecydowano o użyciu podwójnego gotowego mostka H [12], [9]. W pracy przedstawiono jedynie schemat blokowy mostka.

Rysunek 5.12 Widok skonstruowanego podwójnego mostka H

Największą zaletą opisywanego mostka jest możliwość pracy z regulacją średniej wartości napięcia na zaciskach silnika lub praca z regulacją prądu przepływającego przez cewki silnika. Ten drugi pozwala na kontrolę momentu siły wytwarzanego przez oś napędu. Dodatkowo w strukturze mostka znajduje się zasilacz napięcia symetrycznego ± 5V. Napięcie dodatnie wytwarzane jest przez zintegrowaną z mostkiem przetwornicę. Parametry mostka przedstawiono w tabeli poniżej.

| opis | wartość | jednostka |
|---:|:---:|:---:|
| max napięcie zasilania | 40 | V |
| min napięcie zasilania | 12 | V |
| max prąd wyjściowy na kanał | 5 | A |
| wyj. pomiaru prądu | tak | - |
| rodzaj wejścia | analogowy lub/i PWM | - |

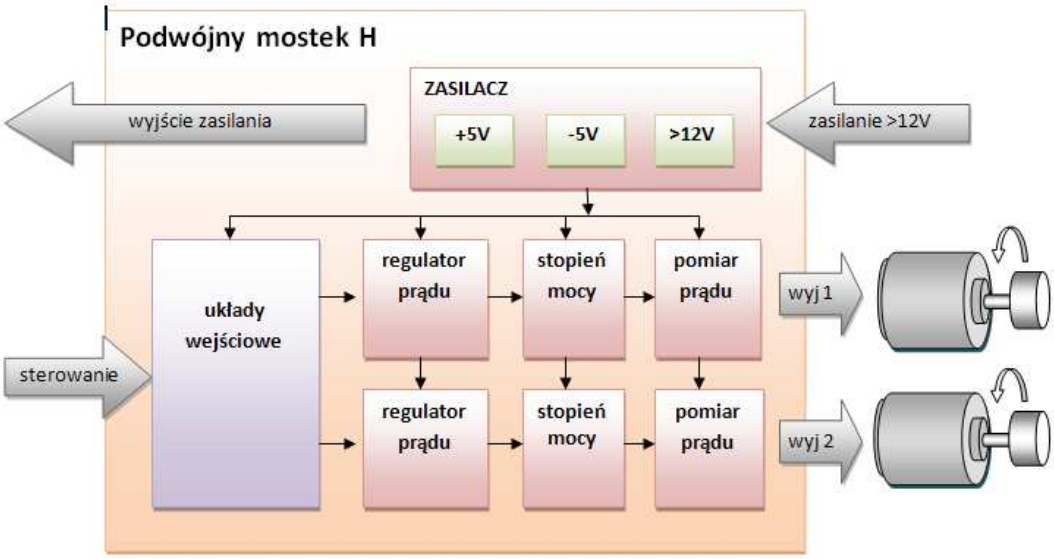
Tabela. 5.5 Parametry podwójnego mostka typu H

W mostku można wyróżnić następujące istotne sygnały:

- wejściowe:
  - napięcie zasilania 12-40V,
  - sygnały PWM (w pracy wykorzystano jako załączenie mostka),
  - sygnał sterujący analogowy, bipolarny,

  - masa,

- wyjściowe:
  - dwa wyjścia do zasilania silników,
  - dwa analogowe wyjścia do pomiaru prądu płynącego przez silniki,
  - napięcie +5V,
  - napięcie -5V,
  - masa.



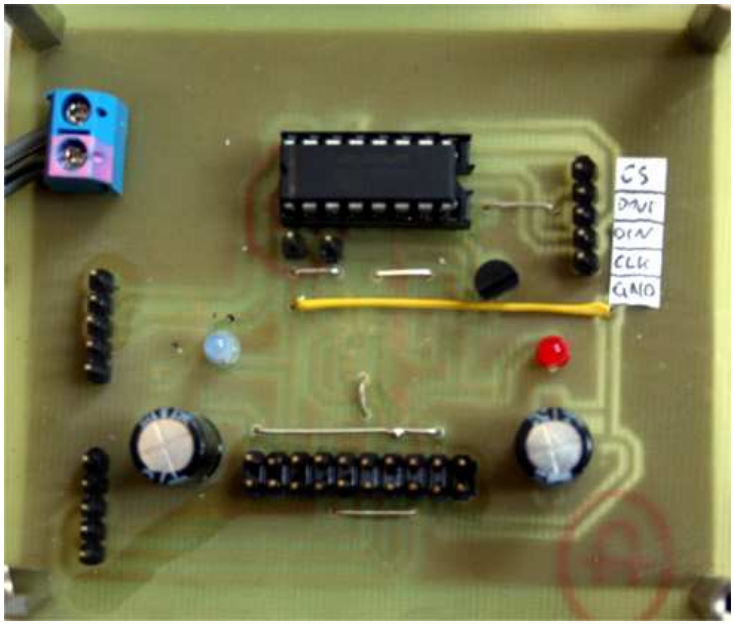

Rysunek 5.13 Schemat blokowy podwójnego mostka H

Więcej o mostku można przeczytać w pracach [12], [9]. Niestety mostek posiada kilka wad. Jedną z nich jest dość spora złożoność konstrukcji. Ponadto do poprawnego sterowania silnikiem w obie strony wymagane jest bipolarne napięcie sterujące. Ten drugi fakt wpłyną na skonstruowanie dodatkowego modułu, wyposażonego w przetwornik C/A oraz konwerter napięć z unipolarnego na bipolarny. Zastosowany przetwornik C/A to model MAX534 firmy MAXIM [27]. Istotne parametry przetwornika zestawiono w tabeli poniżej.

| opis | wartość | jednostka |
|---:|:---:|:---:|
| rozdzielczość | 8 | bitów |
| czas przetwarzania | 8 | $\mu$s |
| napięcie znamionowe | 5 | V |
| liczba kanałów | 4 | - |
| interfejs | SPI | - |
| max częstotliwość interfejsu | 10 | MHz |
| wyjścia | Rail-to-rail | - |

Tabela. 5.6 Parametry przetwornika C/A MAX534

Opisany powyżej mostek wyposażono w wyjścia napięcia symetrycznego co wykorzystano w module konwertera. Widok konwertera przedstawiono na rysunku 5.14.



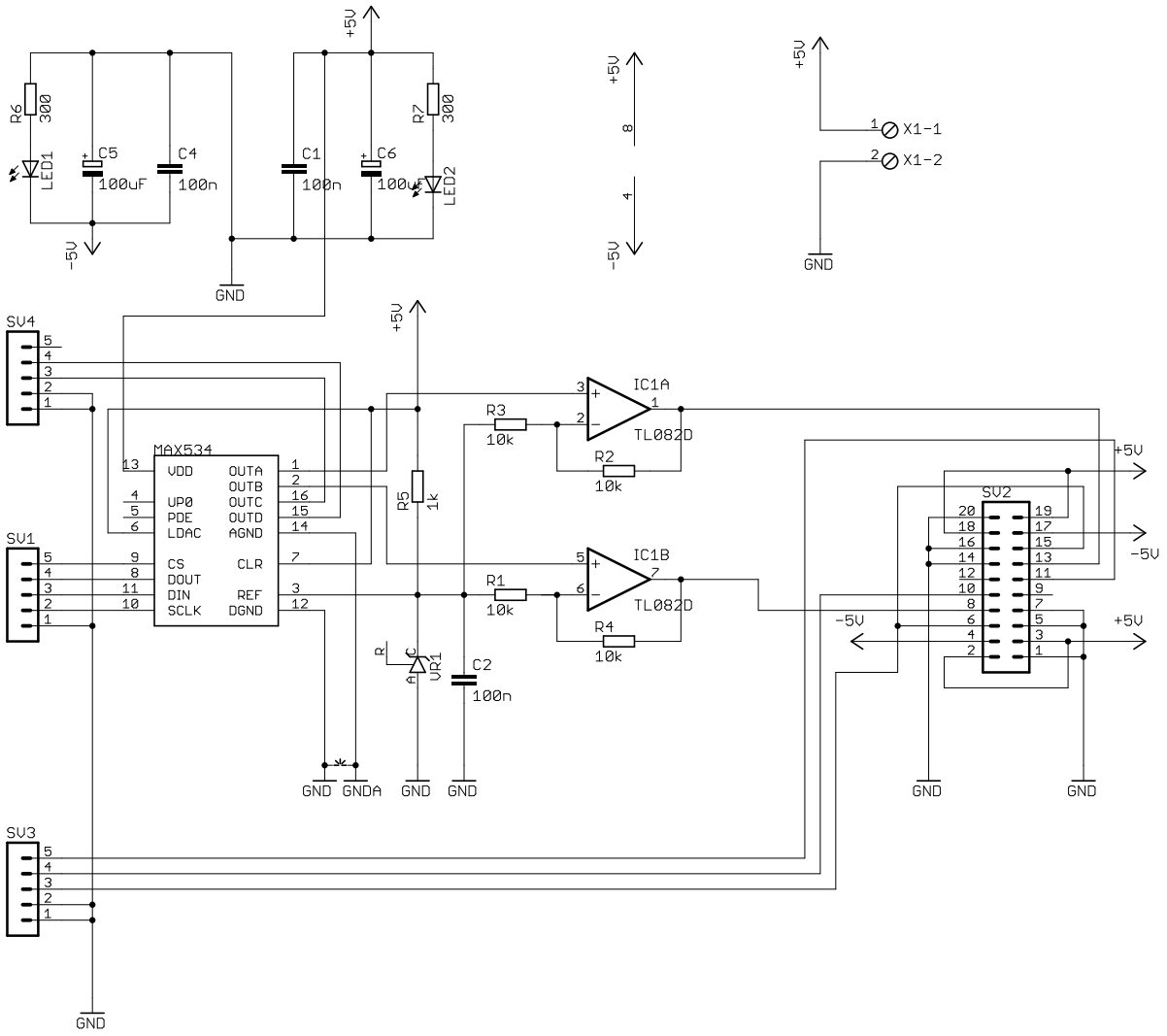

Rysunek 5.14 Widok modułu z przetwornikiem i konwerterem napięć

Z uwagi na to, że konwerter jest zasilany napięciami z mostka, w którym napięcie +5V jest generowane przez zintegrowaną przetwornicę, zdecydowano o wyprowadzeniu tego źródła na płytce oraz podłączeniu go do głównego sterownika. Schemat modułu konwertera pokazano na rysunku 5.15. Zaznajamiając się z wyprowadzeniami mostka opisanymi w jego dokumentacji można zaobserwować, że złącze konwertera jest w pełni zgodne ze złączem mostka. Na wyjściu przetwornika C/A uzyskuje się napięcia 0...2,5V, gdyż wyjścia te sa typu Rail-to-rail, a napięcie referencyjne wynosi 2,5V. Zasilany symetrycznie układ TL082 to podwójny wzmacniacz operacyjny, na którym zrealizowano przesunięcie napięcia, czyli zamianę z unipolarnego na bipolarne ±1,25V. Konwerter, oprócz złącza głównego posiada także trzy mniejsze złącza 5-cio pinowe, którymi podłącza się moduł do głównego sterownika. Jedno z nich to interfejs SPI, drugie służy do doprowadzenia napięć do pomiaru prądu, płynącego przez silniki oraz linii mostka załączających stopnie mocy. Trzecie złącze to pozostałe dwa wyjścia analogowe, które wyprowadzono z czterokanałowego przetwornika.

### 5.2.4 Zasilanie

Głównym źródłem zasilania całego robota jest wysokiej jakości pakiet akumulatorów litowo-polimerowy o pojemności 4800mAh i napięciu 14,8V firmy KOKAM. Głównymi jego zaletami są możliwy duży prąd rozładowywania oraz relatywnie mały ciężar ok. 0,5kg. Warto zwrócić uwagę na to, że napięcie akumulatora przy maksymalnym naładowaniu wynosi 16,8V. Pozwala to na nieprzerwaną, prace robota do 3-4h. Robot został wyposażony w tzw. balancer, który w trakcie ładowania czuwa nad tym, aby każde ogniwo było równo naładowane. Akumulatory tego typu są bardzo drogie, a ich nieodpowiednie ładowanie może doprowadzić do trwałego uszkodzenia jednego z ogniw.

Jak już wspomniano na wstępie, główny sterownik wymaga, aż trzech źródeł zasilania. Jedno z nich (+5V) dostarczone jest z przetwornicy zintegrowanej w sterowniku napędami,




Rysunek 5.15 Schemat modułu z przetwornikiem i konwerterem napięć

dwa pozostałe (+5V oraz +6V) dostarczone są z zasilacza. Wszystkie urządzenia peryferyjne podłącza się jedynie do głównego sterownika. Zasilanie dostarczane jest razem z przewodami sygnałowymi. Dzięki temu zminimalizowano liczbę przewodów w układzie. Napięcie (+5V), które dostarczone jest z mostka służy głównie do zasilania klawiatury, modułu INS, pery-ferii podłączanych do magistral $I^2C$ oraz SPI. Pierwsze napięcie (+5V) dostarczone z zasilacza zasila układy, które wymagają większej wydajności prądowej, np. serwomechanizmy, moduły oświetlenia, czujniki odległości. Drugie zasilanie dostarczone z modułu zasilacza (+6V) służy jedynie do zasilenia płytki mikrokontrolera wraz z dotykowym wyświetlaczem.

Schemat zasilacza przedstawiono na rysunku 5.18. Układ L1084-5V to liniowy stabilizator dużej mocy typu low drop. Maksymalny prąd obciążenia tego układu wynosi 5A. Układ LM7806 to zwykły liniowy stabilizator 6V o wydajności prądowej 1A. Za bezpiecznikami na wyjściu każdego układu zastosowano diodę zenera, której zadaniem jest szybkie spalenie bezpiecznika, na wypadek uszkodzenia któregoś ze stabilizatorów. Dla zwiększenia bezpieczeństwa głównego sterownika wyposażono go w dodatkowe stabilizatory. W zasilaczu wyprowadzono wyjście napięcia 0...4V, które dzięki dwóm diodom zenera jest zredukowaną wartością napięcia akumulatora. Dostarcza ono informacji do głównego sterownika o stanie naładowania akumulatora. Napięcie akumulatora nigdy nie


Rysunek 5.16 Akumulator Li-Poly 4800mAh 14,8V

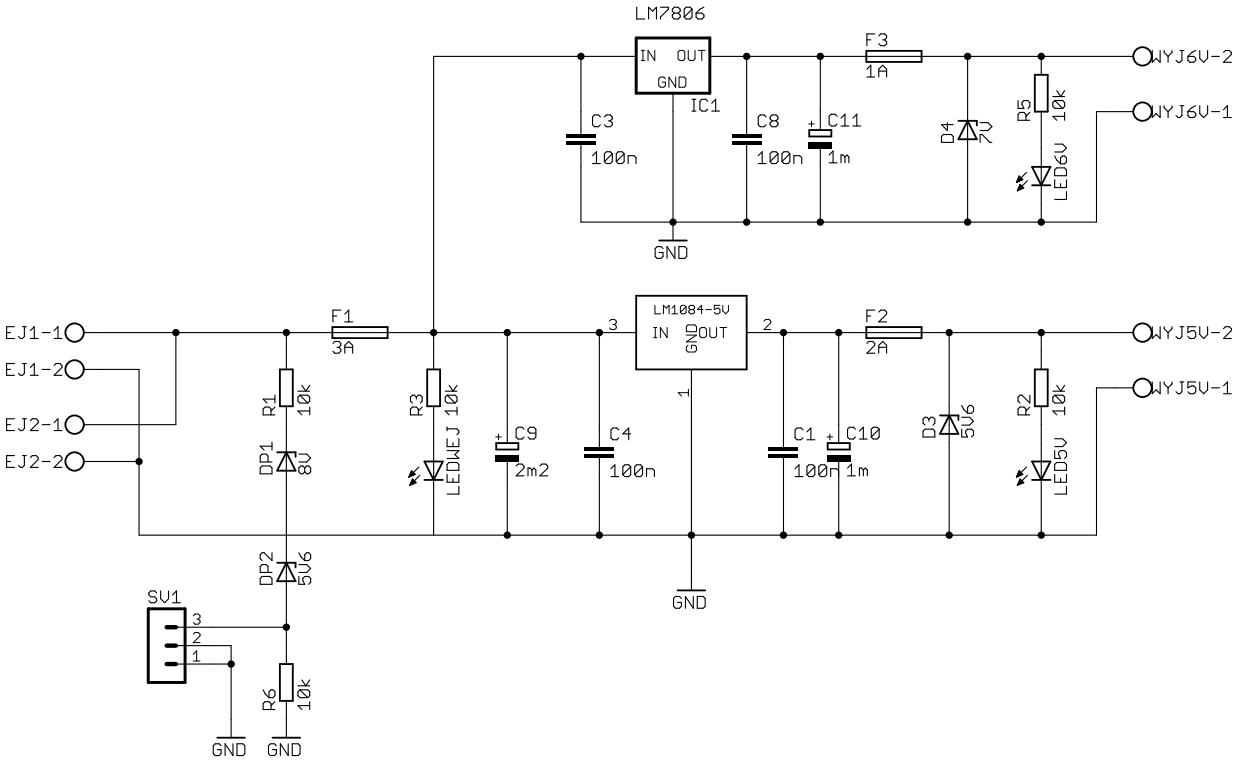
Rysunek 5.17 Widok modułu zasilacza

powinno spaść poniżej 10V. Dla bezpieczeństwa ogniw w oprogramowaniu robota zaimplementowano odłączanie stopni mocy, jeżeli napięcie w układzie spadnie poniżej 12,5V.

### 5.2.5 Główny sterownik

Główny sterownik składa się z trzech piętrowo na sobie zainstalowanych płytek drukowanych. Pierwsza z nich zawiera graficzny wyświetlacz LCD na którym zainstalowano także panel dotykowy. Druga płytka zwiera głównie jednostkę centralną, a trzecia to komplet złącz sterownika.

Płytka wyświetlacza to gotowy moduł, który posiada zintegrowaną przetwornicę ujemnego napięcia, które jest niezbędne do poprawnej pracy wyświetlacza. Istotną cechą całego modułu jest to, że zarówno kontroler jak i układ przetwornicy pracują na napięciu 3,3V, czyli takim jak jednostka centralna. Kontroler sterujący wyświetlaczem to popularny układ T6963 firmy Toshiba [15]. Rozdzielczość wyświetlacza wynosi 240x128 pikseli. Sposób podłączenia wyświetlacza do głównego sterownika pokazano w tabeli 5.2.5.




Rysunek 5.18 Schemat modułu zasilacza

Głównym elementem środkowej płytki [17] jest moduł firmy PHYTEC phyCORE-MPC555 [18]. Jest to wielowarstwowa płytka drukowana, która zawiera mikrokontroler PowerPC MPC555 firmy Freescale [20], układy pamięci ROM 1MB, pamięci RAM 1MB, pamięć FLASH, układ zegara czasu rzeczywistego, sterowniki magistrali RS232 oraz CAN.

Poniżej przedstawiono główne cechy układu PowerPC MPC555:

- 32-bitowa architektura RISC PowerPC,
- zegar 40MHz,
- 64-bitowa jednostka zmiennoprzecinkowa,
- 26 kB SRAM,
- 448 kB FLASH,
- podwójny interfejs UART,
- interfejs QSPI z (kolejka 32 słowa),
- podwójny interfejs CAN 2.0B,
- dwa układy TPU3 po 16 kanałów,
- układ MIOS1 dwa 16 bitowe systemy liczników ,8 kanałów, 16-bitowych układów PWM,
- dwa 10-bitowe QADC (7$\mu$s) po 16 wejść,
- uniwersalne, wielofunkcyjne sygnały I/O,


Rysunek 5.19 Wyświetlacz graficzny LCD

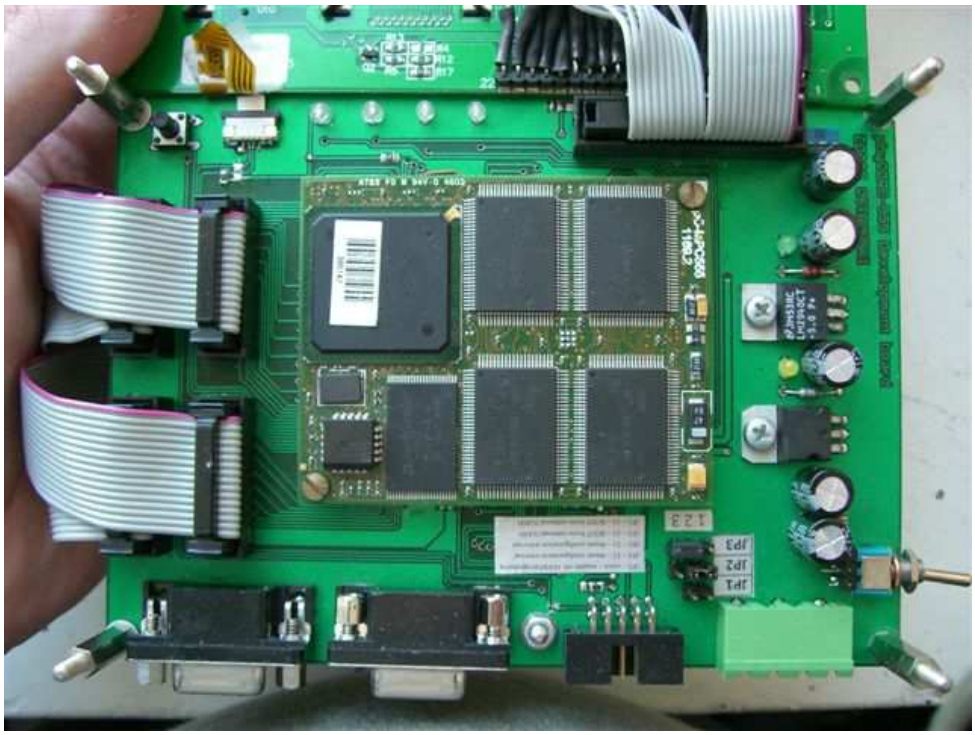
- port testowo-uruchomieniowy JTAG/BDM.

Interfejs wyświetlacza jest typu i80 czyli taki, który posiada osobno linie wyboru zapis i linię wyboru odczyt. Interfejs mikrokontrolera ma typ M68 z pojedynczą linią wyboru odczytu RDNWR, dlatego na płytce zastosowano układ 74HC14, w którym wykorzystano pojedynczy negator. Wyświetlacz zgodnie z tabela, 5.2.5 podłączono tak, aby wykorzystać sprzętowy blok wyboru (dekoder adresów). Dodatkowo na płytce umieszczono cztery diody podłączone poprzez klucze do linii I/O MPIO[11:14]. W panelu dotykowym wykorzy-stano jedynie oś X, gdyż oprogramowanie wyświetlania grafiki napisano tak, że wyświetla „klawisze" (pola) jedynie w poziomie. Moduł z mikrokontrolerem wymaga dwóch napięć zasilania 5V i 3,3V. Na płytce znalazł się także zasilacz, który do poprawnej pracy sterownika musi być zasilony napięciem min +6V. Dodatkowo na płytce umieszczono dwa złącza RS232, złącze BDM oraz złącza którymi podłącza się trzecią płytkę sterownika.

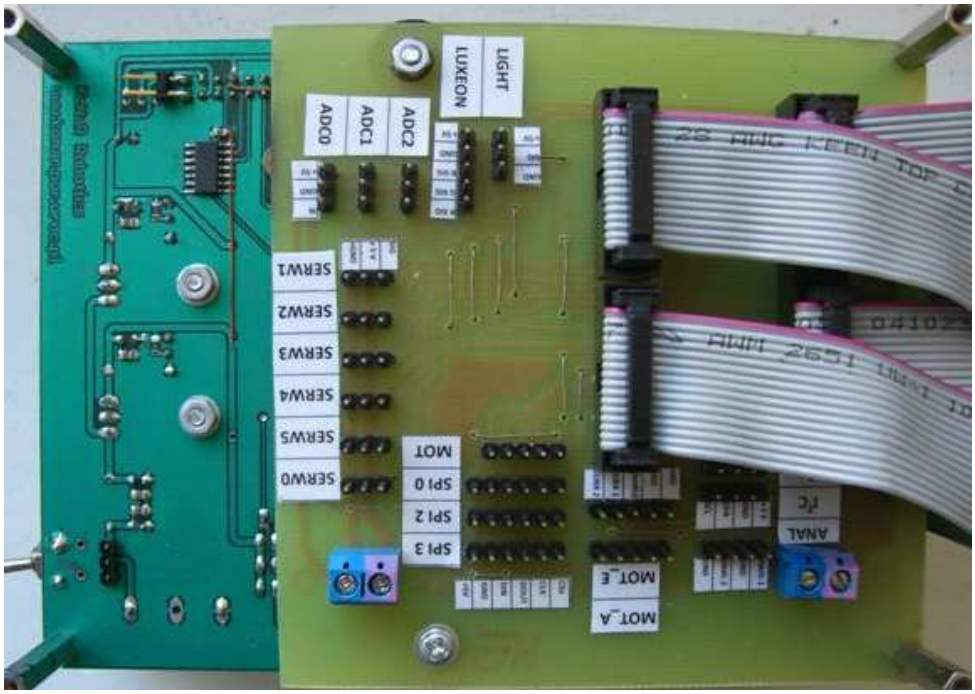
Trzecia płytka sterownika zawiera wszystkie złącza dla peryferii robota. To do niej podłącza się moduł INS, klawiaturę, moduł sterownika silnikami, moduł oświetlenia, serwomechanizmy oraz czujniki odległości. Z uwagi na to, że płytka ta odpowiada także za doprowadzenie napięć zasilania dla poszczególnych peryferii, doprowadzono do niej dwa napięcia +5V.

| płytka LCD | płytka z mikrokontrolerem | opis |
|---|---|---|
| A | +5V | zasilanie podświetlenia |
| K | GND | masa zasilania podświetlenia |
| Vout | do potencjometru | wyj. ujemnego napięcia -15V |
| V0 | do potencometru | wej. napięcia ujemnego (np. z dzielnika) |
| PD | nie podłączono | pin kontrolny |
| GND | GND | masa kontrolera |
| VDD | +3V3 | zasilanie kontrolera |
| VEE | nie podłączono | - |
| /WR | RDNWR | aktywny zapis |
| /RD | /RDNWR | aktywny odczyt (negator) |
| C/E | CS2 | wybór wyświetlacza |
| C/D | A31 | wybór rejestru (danych/sterujący) |
| /RST | MPI015 | reset wyświetlacz |
| DB0...DB9 | D0...D7 | linie danych |
| FS | jumper | wybór czcionki |

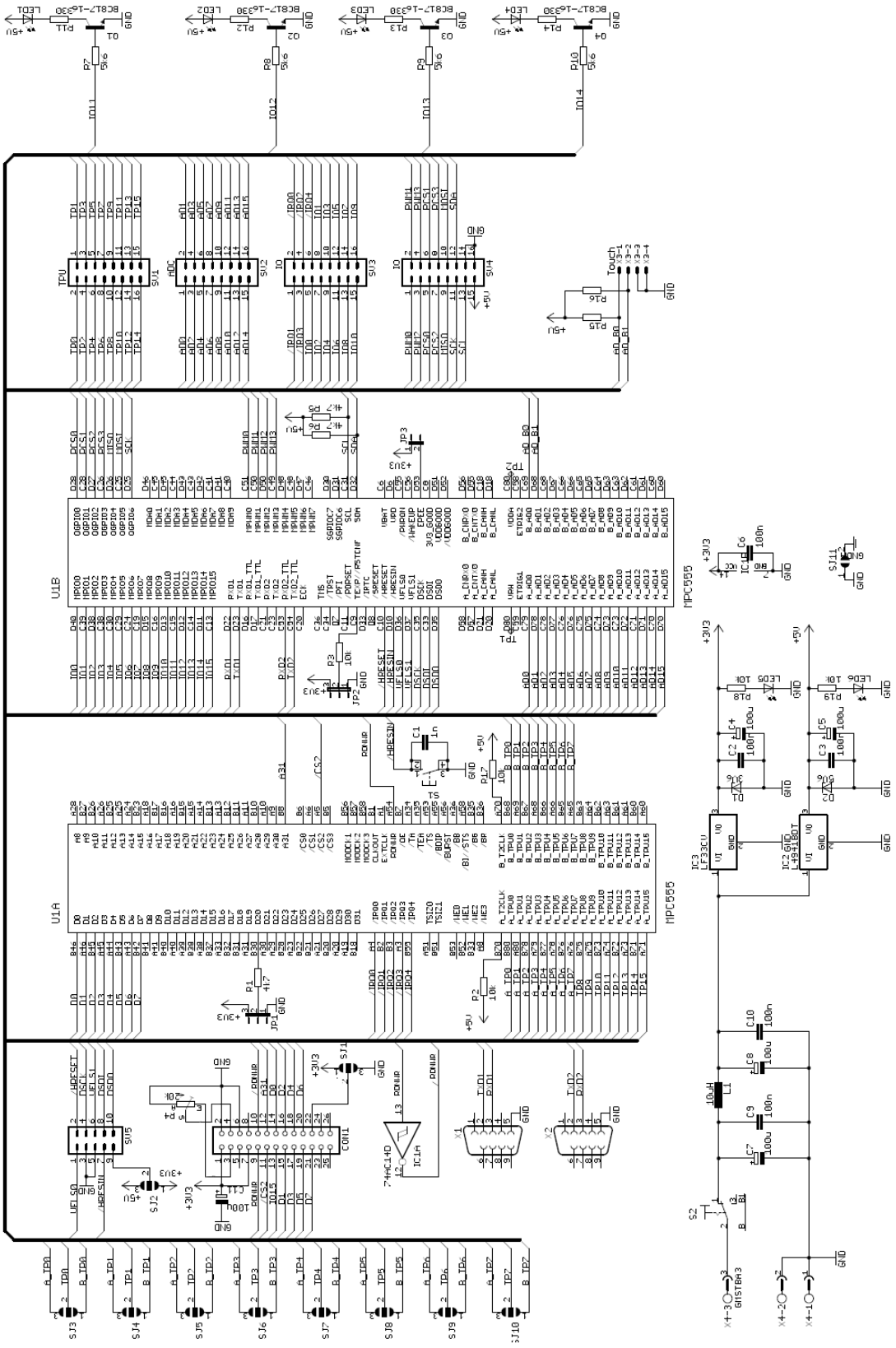
Tabela. 5.7 Podłączenie wyświetlacza LCD



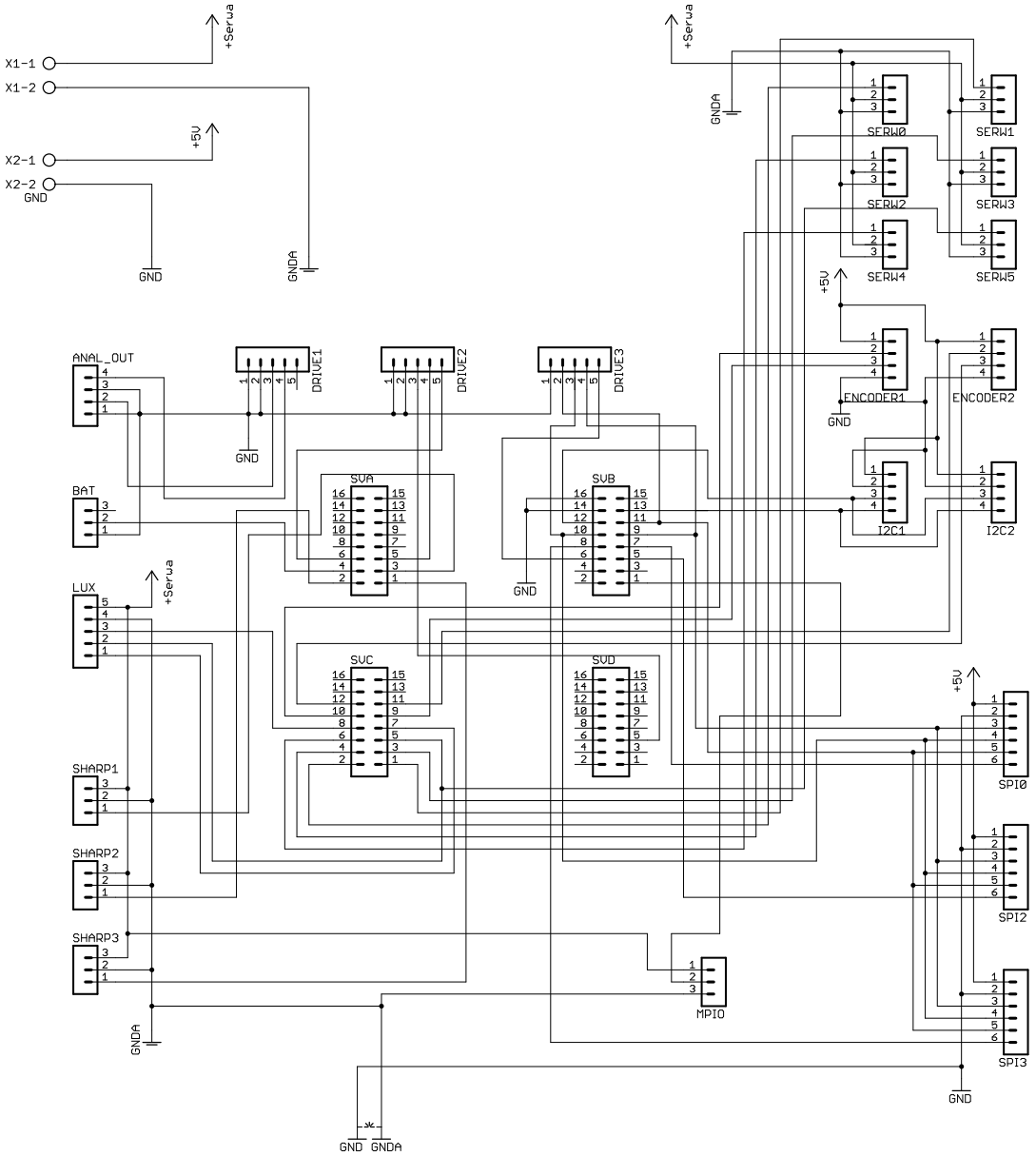
Rysunek 5.20 Widok środkowej płytki sterownika głównego




Rysunek 5.21 Widok dolnej płytki sterownika głównego



Rysunek 5.22 Schemat środkowej płytki sterownika głównego




Rysunek 5.23 Schemat dolnej płytki sterownika głównego

## 5.3 Pomiar odchylenia kątowego

W części pracy poświęconej symulacjom zakładano, że pomiar odchylenia kątowego i jego prędkości kątowej są informacjami dobrze mierzalnymi i pozbawionymi znaczących błędów. W praktyce pozyskanie tych informacji należy do jednych z najtrudniejszych zadań.

Zwykle klasyczne odwrócone wahadło zaczepione jest na ruchomej platformie, która przemieszcza się równolegle do powierzchni, np. wózek na czterech kołach. Zakładając, że porusza się ona jedynie po powierzchniach płaskich, wystarczy zastosować enkoder o dostatecznie dużej rozdzielczości. Robot balansujący nie posiada wyróżnionej platformy, której orientacja względem podłoża byłaby stała, gdyż koła robota są zamocowane bezpośrednio do wahadła (korpusu). Dobrym rozwiązaniem, pozwalającym zmierzyć pożądaną wielkość jest odpowiedni system sensoryczny. W części opisującej praktyczny eksperyment, przedstawiono prosty miniaturowy moduł INS (Inertial Navigation System), który wyposażono w dwa czujniki, trzyosiowy akcelerometr i żyroskop.

Fuzji pomiarów z wymienionych czujników dokonuje się w sposób niezwykle efe-ktywny przy użyciu algorytmu Filtracji Kalmana [16]. Akcelerometry mierzą zarówno przyspieszenia dynamiczne jak i statyczne, np grawitacja. Zdolność tą wykorzystuje się do wyznaczania odchylenia od pionu. Tanie, dostępne na rynku modele charakteryzują się sporymi szumami na wyjściu, dlatego zaleca się stosowanie jeszcze jednego czujnika. Żyroskopy, jak wiadomo, dostarczają informacji o prędkości obrotowej. Niestety ich wadą jest fakt, że z czasem ulegają dryftowi, pomiary stają się więc błędne. Filtr Kalmana [5] doskonale niweluje niepożądane zjawisko dryftu oraz dokonuje przy tym fuzji pomiarów, dostarczając dostatecznie dokładnej i nie zaszumionej informacji o odchyleniu oraz prędkości obrotowej.

### 5.3.1 Wyznaczenie odchylenia od pionu na podstawie pomiaru przyspieszeń

Czujniki przyspieszenia dostarczają informacji o kierunku działania siły grawitacji. Jak wiadomo, robot znajduje się w ciągłym ruchu. Odczytane z osi poziomej i pionowej przyspieszenia będą składowymi mierzonej siły grawitacji oraz przyspieszeń robota. Należy ten fakt uwzględnić w trakcie obliczeń trygonometrycznych. Jedną z metod wyłuskania kąta odchylenia ilustruje rysunek 5.24

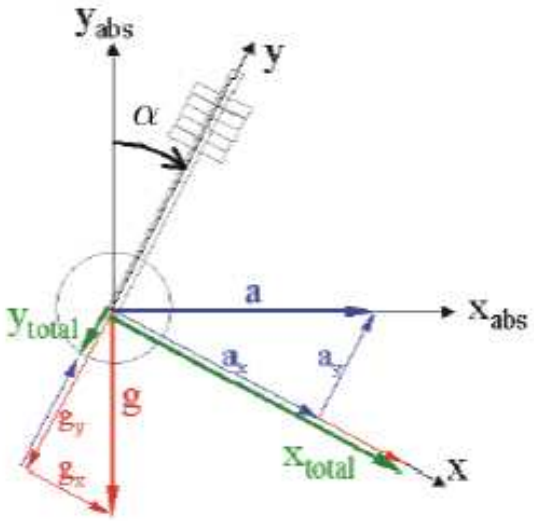
Po odczytaniu czujników przyspieszeń, dokonuje się następujących obliczeń

$$a = +/- \sqrt{x^2 + y^2} - 1 \tag{5.1}$$

„+" jeżeli $a > 0$, „-" jeżeli $a < 0$. Następnie dokonuje się obliczeń

- dla a>0 $\alpha = \arcsin(\frac{x - y \cdot a}{x^2 + y^2})$
- dla a<0 $\alpha = \arcsin(\frac{x + y \cdot a}{x^2 + y^2})$
- dla a=0 $\alpha = \arcsin(x)$

Wszystkie wartości są znormalizowane do działania siły grawitacji (g=1). Jak widać z powyższych równań, kłopotliwe jest oszacowanie kierunku przyspieszenia całego robota $a$. Można tego dokonać znając znak wartości sterowania napędów.




Rysunek 5.24 Uzyskanie kąta odchylenia od pionu na podstawie pomiarów z akcelerometru

### 5.3.2 Filtr Kalmana

W 1960 r. R.E. Kalman opublikował słynną pracę, w której przedstawił metodę filtracji dynamicznej, która stała się jedną z najpopularniejszych metod estymacji statycznie optymalnej. Posługując się tym narzędziem, można wyznaczyć pomiarowo niedostępne zmienne jedynie na podstawie bieżących wartości wielkości pomiarowo dostępnych oraz znajomości modelu matematycznego łączącego ze sobą obydwie te grupy pomiarów.

Filtry Kalmana [7] [5] (z ang. Kalman Filter w skrócie KF) znalazły zastosowanie w wielu dziedzinach nauki. Najczęściej są one stosowane w inżynierii, głównie w układach sensorycznych: robotów, samolotów oraz promów kosmicznych. Stosuje się je także w technikach komputerowych do przetwarzania obrazów oraz w ekonomii do prognozowania wskaźników gospodarczych. Metoda ta jest chętnie wykorzystywana w robotyce, gdzie układy percepcji otoczenia odgrywają istotna rolę. Są często jedynym źródłem informacji o środowisku, w którym znajduje się urządzenie.

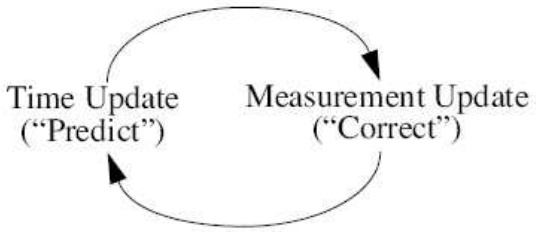
W ramach wprowadzenia warto wspomnieć o własnościach filtru [7].

- KF jest optymalnym estymatorem, gdyż przy konkretnych założeniach może on spełniać pewne kryterium np minimalizacja błędu średniokwadratowego.
- Kolejną cechą, która świadczy o optymalności filtru to fakt, że korzysta on z wszystkich dostępnych pomiarów bez względu na to, z jaką dokładnością i precyzją zostały one wykonane. Ostatecznie na ich podstawie dokonuje najlepszej estymacji stanu.
- FK jest algorytmem typu rekursywnego. Nie przechowuje on wszystkich danych z przeszłości i nie dokonuje on w każdym korku ich przeliczenia. Informacje są przetwarzane sukcesywnie, bazując na wartościach obliczonych w poprzednim kroku.
- Jest to algorytm przetwarzania danych. Znając wejście i wyjście systemu można uzyskać niedostępne (niemierzalne) wartości na podstawie dostępnych (mierzalnych)

danych, np z sensorów. W teorii systemów liniowych mówimy o obserwowalności układu.

- Metodę nazywamy filtrem, gdyż jest on optymalnym estymatorem stanu tzn, że uzyskamy możliwie optymalna wartość, na podstawie wielu pomiarów pochodzących z zaszumionego środowiska.

Filtr Kalmana jest dwufazowym rekursywnym algorytmem (Rysunek 5.25). Pierwsza faza algorytmu nazywana jest predykacją (z ang. predict). Równania wykonywane w trakcie tej fazy nazywane są aktualizacją czasową (z ang. time update). Druga faza nazywa się korekcją (z ang. correct), a jej równania to aktualizacja pomiarowa (z ang. measurement update). W trakcie predykcji, bazując na stanie z poprzedniego kroku, wyznacza się estymowaną wartość stanu $\hat{x}$ oraz jego kowariancję i są to wartości a priori. Pomiar $z$ w drugiej fazie jest pewną formą sprzężenia zwrotnego. Na jego podstawie dokonuje się wyznaczenia wartości a posteriori dla stanu i jego kowariancji.




Rysunek 5.25 Cykl dwufazowego algorytmu FK.

Do opisu zarówno procesu jak i systemu pomiarowego stosuję się modele matematyczne.

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{5.2}$$

$$z_k = Hx_k + v_k \tag{5.3}$$

Pierwsze równanie różnicowe to model procesu, który częściowo jest deterministyczny a częściowo losowy. Jest to powiązanie poprzedniego stanu z aktualnym poprzez macierz $A$. Macierz $B$ to wymuszenie stanu (sterowanie), $w_k$ to tzw. szum procesu (część losowa). Drugie równanie to model pomiaru, gdzie $H$ jest macierzą wiążącą pomiar ze stanem (wyjście filtru), a $v_k$ to zakłócenia. Zarówno $w_k$ jak i $v_k$ reprezentują białe szumy Gausa.

$$\begin{aligned} p(w) &\sim N(0, Q) \\ p(v) &\sim N(0, R) \end{aligned} \tag{5.4}$$

Podobnie zapis prawdopodobieństw warunkowych dla powyższych modeli będzie wyglądał następująco:

$$p(x_k|x_{k-1}) \sim N(Ax_{k-1} + Bu_k, Q) \tag{5.5}$$

$$p(z_k|x_k) \sim N(Hx_k, R) \tag{5.6}$$

Teraz zdefiniujmy błędy szacowania, w których $\hat{x}^-$ to estymowany stan a priori uzyskany z procesu, a $\hat{x}$ to estymowany stan a posteriori uwzględniający pomiar $z_k$.

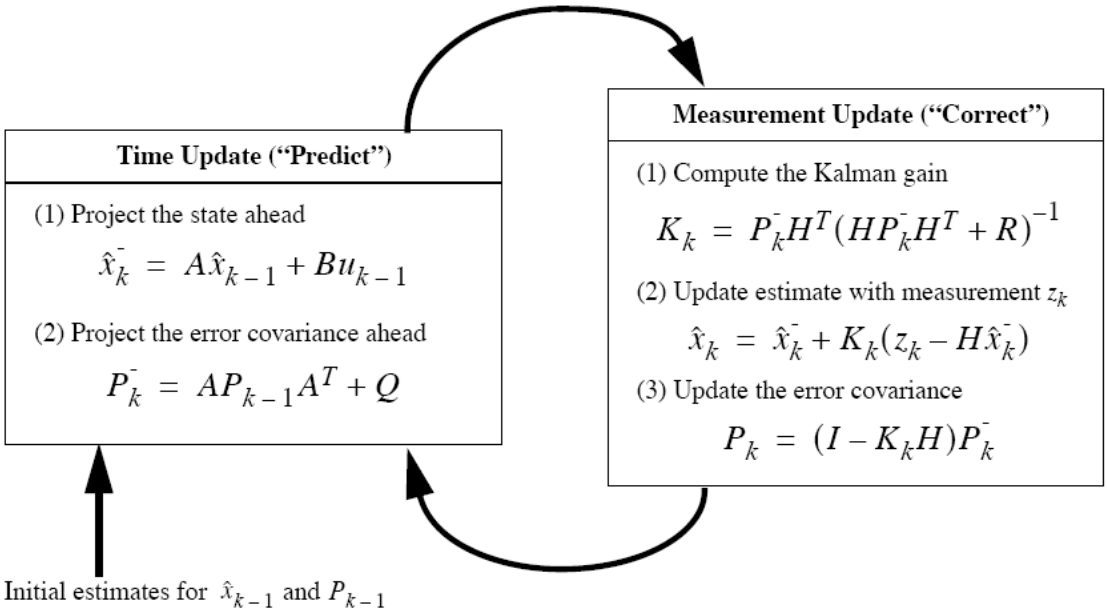
$$e_k^- \equiv x_k - \hat{x}_k^- \tag{5.7}$$

$$e_k \equiv x_k - \hat{x}_k \tag{5.8}$$

gdzie $e_k^-$ to błąd a priori, a $e_k$ to błąd a posteriori. Są to różnice pomiędzy rzeczywistym stanem, a wartością estymowaną. W praktyce rzeczywiste wartości stanu $x_k$ nie są znane. Macierze kowariancji (zależności wariancji składowych wektora stanu) to:

$$P_k^- = E[e_k^-, e_k^-] \tag{5.9}$$
$$P_k = E[e_k, e_k] \tag{5.10}$$

gdzie $P_k^-$ to macierz kowariancji a priori, a $P_k$ to macierz kowariancji a posteriori.




Rysunek 5.26 Kompletny algorytm FK.

Równania pierwszej fazy FK ma postać:

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_{k-1} \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{5.11}$$

gdzie $\hat{x}_k^-$ i $P_k^-$ to prognozowane wartości stanu i kowariancji a priori, $\hat{x}_{k-1}$ i $P_{k-1}$ to optymalne szacowane wartości a posteriori wykonane w poprzednim kroku. Wzmocnienie Kalmana jest czymś w rodzaju wagi z jaką wpłynie faza korekcji na estymowany stan:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{5.12}$$

Równanie, które mówi jakie jest optymalne skorygowanie prognozy w czasie $k$, bazujące na wszystkich dotychczasowych pomiarach będzie miało postać:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{5.13}$$

gdzie $z_k$ to pomiar, a różnica $(z_k - H\hat{x}_k^-)$ nazywa się *measurement innovation*. Pozostaje jeszcze skorygować macierz kowariancji:

$$P_k = (I - K_k H)P_k^- \tag{5.14}$$

gdzie $I$ to macierz jednostkowa.

### 5.3.3 Pomiar odchylenia kątowego, a Filtr Kalmana

Jest to jeden z najczęściej spotykanych przykładów wykorzystania KF [16]. W obiektach, które nie mają stałego punktu odniesienia pomiarów odchylenia dokonuje się przy pomocy inklinometrów, akcelerometrów lub żyroskopów. Niestety żaden z sensorów nie dostarcza dostatecznie dokładnej informacji o odchyleniu kątowym. W praktyce wykorzystuje się kilka czujników dla jednego systemu. Najciekawszym przypadkiem jest inklinometr lub akcelerometr wraz z żyroskopem.

Akcelerometry mierzą przyspieszenia statyczne, np grawitacja, zdolność tą wykorzystuje się do wyznaczenia odchylenia od pionu. Tanie, dostępne na rynku modele charakteryzują się sporymi szumami na wyjściu, dlatego zaleca się stosowanie jeszcze jednego czujnika. Żyroskopy, jak wiadomo, dostarczają informacji o prędkości obrotowej. Niestety ich wadą jest fakt, że z czasem ulegają dryftowi, pomiary stają się więc błędne. Filtr Kalmana doskonale niweluje niepożądane zjawisko dryftu oraz dokonuje przy tym fuzji pomiarów dostarczając dostatecznie dokładnej i nie zaszumionej informacji o odchyleniu (obrocie). Równanie takiego systemu ma postać:

$$\alpha_k = \alpha_{k-1} + (\omega_{k-1} - g_{bias})dt + w_k \tag{5.15}$$

oraz równanie pomiaru

$$z_k = H\alpha_k + v_k \tag{5.16}$$

gdzie $\alpha$ to odchylenie kątowe, $\omega$ to prędkość kątowa, $g_{bias}$ to dryft żyroskopu, $w_k$ to szum żyroskopu, a $v_k$ to szum akcelerometrów. Pomiar prędkości kątowej $\omega$ odczytany z żyroskopu uwzględniony zostanie już w fazie predykcji jako wymuszenie $u$, a pomiar odchylenia uzyskany z akcelerometrów w fazie korekcji.

Można więc zapisać jeszcze raz równania algorytmu KF:
Predykcja:

$$\begin{aligned} \hat{x}_k^- &= Ax_{k-1} + Bu \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{5.17}$$

Korekcja:

$$\begin{aligned} K_k &= \frac{P_k^- H^T}{HP_k^- H^T + R} = P_k^- H^T (HP_k^- H^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\ P_k &= (I - K_k H)P_k^- \end{aligned} \tag{5.18}$$

gdzie poszczególne macierze są równe

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1 & 0 & -dt \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \\ \mathbf{B} &= \begin{bmatrix} dt \\ 1 \\ 0 \end{bmatrix}, \end{aligned} \tag{5.19}$$

wektor stanu

$$x = \begin{bmatrix} \alpha \\ \omega \\ g_{bias} \end{bmatrix} \tag{5.20}$$

wyjście filtru

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{5.21}$$

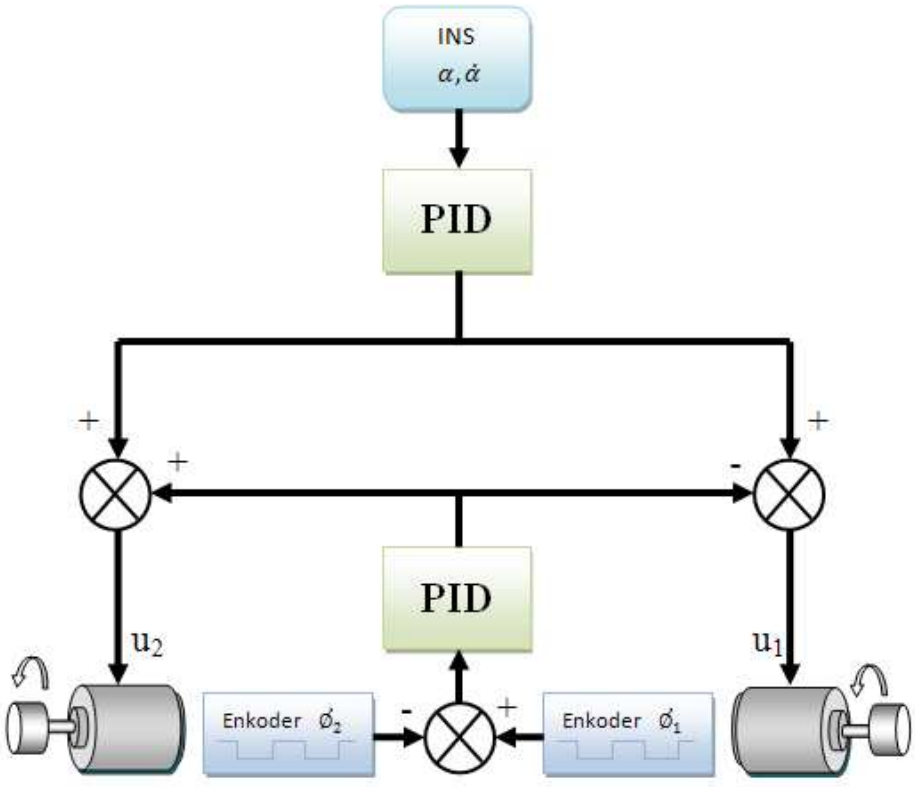
R to macierz $1 \times 1$ (szum akcelerometru), macierz kowariancji $Q$ ma wymiar $3 \times 3$

$$\mathbf{Q} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \tag{5.22}$$

Zarówno $Q$ oraz $R$ należy wyznaczyć eksperymentalnie. Wymiar macierzy $K$ jest $1 \times 3$.

## 5.4 Regulator PID

Po skonstruowaniu robota okazało się, że niektóre parametry fizyczne elementów składowych nieco się różniły od tych, które przyjęto w trakcie symulacji. W trakcie budowy, wielokrotnie zmieniano koncepcję, która przyczyniła się do wymieniany napędów na zdecydowanie mocniejsze. Symulacje przeprowadzono zanim jeszcze powstał fizycznie robot. Niestety autor nie zdołał ponowne wyznaczyć optymalnych sterowników oraz przeprowadzić symulacje. Aby jednak przetestować konstrukcję i zbadać poprawność działania wszystkich modułów, zdecydowano o napisaniu prostego podwójnego regulatora PID.




Rysunek 5.27 Schemat blokowy prostego podwójnego regulatora PID

Głównym celem sterowania było utrzymywanie równowagi. Do tego wystarczyło zastosować prosty regulator PID. Niestety nigdy nie zdarza się tak, aby obydwa napędy

charakteryzowały się takimi samymi parametrami, nawet jeśli są tego samego typu. Niezbędne okazało się zastosowanie drugiego regulatora, który kontrolował, aby obydwa koła kręciły się z taką samą prędkością.

Równanie pierwszego regulatora

$$u_\alpha = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

gdzie

$$e(t) = \alpha_d - \alpha(t) \qquad \alpha_d = 0$$

Warto zwrócić uwagę na to, że jeżeli $\alpha_d = 0$ to błąd człon różniczkującego, to $\frac{de(t)}{dt}$ jest prędkością kątową $\dot{\alpha}$.

Drugie równanie regulatora jest następujące

$$u_{\phi 12}(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

gdzie

$$e(t) = \dot{\phi}_1 - \dot{\phi}_2$$

Można teraz zapisać sterowanie napędów w całości

$$u_1(t) = u_\alpha(t) + u_{\phi 12}(t)$$

$$u_2(t) = u_\alpha(t) - u_{\phi 12}(t)$$

## 5.5 Oprogramowanie

Program dla mikrokontrolera został w całości napisany w środowisku CodeWarrior IDE w wersji 5.1.1090 firmy Metrowerks Corporation [29] w języku ANSI C.

W podrozdziale poświęconemu oprogramowaniu zdecydowano o przedstawieniu tylko niektórych fragmentów kodu. Głównie skupiono się na konfiguracji poszczególnych modułów oraz implementacji Filtru Kalmana oraz podwójnego sterownika PID. Wszystkie przedstawione listingi zostały zaimplementowane i przetestowane.

### 5.5.1 Definicje

Do omówienia konfiguracji poszczególnych modułów niezbędne jest zapoznanie się z niektórymi definicjami przedstawionymi poniżej.

```
   #define XTAL 20000000
   #define PIT_TB (XTAL/256)
// **********************************
// Definicje dla USIU
// **********************************
   #define TPSEC 100 //przerwanie 100Hz
   #define PIT_CONST    ((PIT_TB+(TPSEC>>1))/TPSEC) // oblicz stała dla PIT
// **********************************
// Definicje dla QSM
```

```
// ***********************************
   #define ADCSEL      0x0E     // ADC aktywny LOW dla PCS0 - moduł INS
   #define DACSEL      0x0D     // DAC aktywny LOW dla PCS1 - sterownik silników
   #define QsmDT       0x20
   #define QsmBITSE    0x40
   #define QsmDSCK     0x10
   #define DESELECT    0xF     // no SLAVEs selected
// ***********************************
// Definicje dla TPU
// ***********************************
   #define CHAN0    0   // TPU channel
   #define CHAN1    1   // TPU channel
   #define CHAN2    2   // TPU channel
   #define CHAN3    3   // TPU channel
   #define CHAN4    4   // TPU channel
   #define CHAN5    5   // TPU channel
   #define CHAN6    6   // TPU channel
   #define CHAN7    7   // TPU channel
   #define CHAN8    8   // TPU channel
   #define CHAN10   10  // TPU channel
   #define CHAN11   11  // TPU channel
   #define CHAN12   12  // TPU channel
   #define CHAN13   13  // TPU channel
   #define CHAN14   14  // TPU channel
   #define CHAN15   15  // TPU channel
   #define FSys     40000000 // 40   MHz
   #define A_TCR1CK     ((FSys)/4)   // 10   MHz
   #define A_TCR2CK     ((FSys)/8)   // 5  MHz
   #define B_TCR1CK     ((FSys)/4)   // 10   MHz
   #define B_TCR2CK     ((FSys)/8/4) // 0,625    MHz
   #define PWM_FRQ_SERW 50           // częstotliwość PWM dla serwomechanizmów [Hz]
   #define PWM_PER_SERW ((B_TCR2CK+(PWM_FRQ_SERW>>1))/PWM_FRQ_SERW)
   #define PWM_FRQ_LUX  250          // częstotliwość PWM dla oświetlenia RGB [Hz]
   #define PWM_PER_LUX  ((B_TCR2CK+(PWM_FRQ_LUX>>1))/PWM_FRQ_LUX)
// ***********************************
// Definicje dla QADC
// ***********************************
   #define QADC_CCW_P 0x0200
   #define QADC_CCW_BYP 0x0100
   #define QADC_CCW_QCLK2 0x0000
   #define QADC_CCW_QCLK4 0x0040
   #define QADC_CCW_QCLK8 0x0080
   #define QADC_CCW_QCLK16 0x00C0
   #define QADC_CCW_AN0 0
   #define QADC_CCW_AN1 1
   #define QADC_CCW_AN2 2
   #define QADC_CCW_AN3 3
   #define QADC_CCW_AN48 48
   #define QADC_CCW_AN49 49
```

```
#define QADC_CCW_AN50 50
#define QADC_CCW_AN51 51
#define QADC_CCW_AN52 52
#define QADC_CCW_AN53 53
#define QADC_CCW_AN54 54
#define QADC_CCW_AN55 55
#define QADC_CCW_AN56 56
#define QADC_CCW_AN57 57
#define QADC_CCW_AN58 58
#define QADC_CCW_AN59 59
#define QADC_CCW_VRL 60
#define QADC_CCW_VRH 61
#define QADC_CCW_VRM 62 // (VRH-VRL)/2
#define QADC_CCW_EOQ 63
```

### 5.5.2 Konfiguracja UART

Do komunikacji sterownika z komputerem PC skonfigurowano jeden z dwóch dostępnych portów UART [20]. Wyjścia portów SCI można przy pomocy specjalnych zworek na module skonfigurować jako TTL lub RS232. Pierwszy port (UART1) skonfigurowano do pracy z prędkością 57600kb w trybie RS232.

```
InitPort1();
InitializeUART(kBaud57600);
printf("\n\rUART channel 1 initialized\n\r");
```

### 5.5.3 Konfiguracja modułu USIU

Moduł USIU (Unified System Interface Unit) [20] jest odpowiedzialny za start mikrokontrolera, jego inicjację, tryby pracy, zabezpieczenia, pracę zewnętrznej magistrali (EBI), synchronizację zegara, pamięć, debagowanie oraz system przerwań.

Jednym z bloków modułu USIU jest funkcja cyklicznego przerwania PIT (Periodic Interrupt Timer). W sterowniku skonfigurowano je jako główne przerwanie, w którym realizowane jest sterowanie robota. Częstotliwość tego przerwania ustawiono na 100Hz.

Jak już wspomniano USIU kontroluje także zewnętrzną magistralę (EBI), która wyposażona jest w sprzętowy dekoder adresów. Blok wyboru urządzeń zewnętrznych został skonfigurowany do pracy z wyświetlaczem LCD. W module z mikrokontrolerem linie wyboru CS0 i CS1 są juz zarezerwowane do pracy z pamięciami. Zatem wybór wyświetlacza skonfigurowano dla linii CS2. Dostęp do wyś-wietlacza ustawiono pod adresami: 0x02000000 - rejestr danych, 0x02000001 - rejestr konfiguracyjny.

```
void Usiu_Init()
{
/********************** USIU ******************************/

/********************** PIT ******************************/
// STEP 1: Inicjacja
   USIU.PITC.B.PITC =PIT_CONST;  // wartość do zliczenia
   USIU.PISCR.B.PITF = 1;        // zatrzymanie PIT jeżeli FREEZE
   USIU.PISCR.B.PTE = 1;         // start licznika
```

```
// STEP 2: Ustawienie poziomu przerwań
   USIU.PISCR.B.PIRQ = 0x20;     // Poziom przerwań dla PIT równy 2
// STEP 3: Włącz/wyłącz przerwanie
   USIU.PISCR.B.PIE = 0 ;        // 0-włącz 1-wyłącz
/********************* INTERRUPTS ************************/
   USIU.SIMASK.R = 0xFFFF0000;   // maskuj wszystkie przerwania
/********************* CHIP SELECTS **********************/
/******* LCD - konfiguracja dekodera adresowego ************/
/************************ BR2/OR2 ****************************/
   USIU.BR2.B.BA=0x0400;         //Ustaw adres bazowy na 0x200
//
// UWAGA
// pole BA ma szerokość 17 bitów, a nie 16
// zapisanie liczby 0x0400 odpowiada adresowi 0x0200
//
   USIU.BR2.B.PS=1;        // 8 bitów - szerokość danych
   USIU.BR2.B.V=1;         // bit skończonej konfiguracji
   USIU.OR2.B.AM=0xff80;   // maskuj cały adres
   USIU.OR2.B.SCY=0xF;     // szerokość zegara (najdłuższa)
   USIU.OR2.B.BSCY=2;      // szerokość impulsu zegara
   USIU.OR2.B.ACS=3;       // CS ustaw w w połowie zegara
/************************ POWER & CLOCK **********************/
   USIU.SCCR.B.RTDIV=1;    // Ustaw dzielnik zegara RTC na 256
/******************** END USIU ******************************/
}
```

### 5.5.4 Konfiguracja modułu MIOS1

Moduł MIOS1 (Modular Input/Output Subsystem) [20] zawiera szereg przydatnych funkcji czasowo-licznikowych. Pozwala na skonfigurowanie 32 kanałów jako funkcje PWM, timery, linie I/O, generatory impulsów, IC, OC, pomiar szerokości impulsu. W sterowniku skonfigurowano dwa kanały. Kanał szósty jako cykliczne przerwanie, w którym dokonuje się rekursywnego przeliczania Filtru Kalmana. Częstotliwość tego przerwania wynosi 500Hz. Kanał zerowy został skonfigurowany jako funkcja PWM. Służy do sterowania oświetleniem białym.

```
void Mios1_Init()
{
/************************ MIOS1 ******************************/
// Inicjacja MIOS Timer MMCSM6       // 6 kanał jako timer z
                                     //   przerwaniem 500Hz
   MIOS1.MMCSM6SCR.B.CLS = 0x3;      // ustaw źródło impulsów
                                     //   z dzielnika zegara
   MIOS1.MMCSM6SCR.B.CP = 0xEC;      // ustaw dzielnik zegara 20 - 1MHz
   MIOS1.MMCSM6CNT.R = 0xFFFF-1000;  // ustaw wartość do zliczenia
   MIOS1.MMCSM6MLR.R = 0xFFFF-1000;  // ustaw wartość do załadowania
                                     // po zliczeniu
// Initialization MPWM0 na kanale 0 PWM
   MIOS1.MPWMSM0PERR.R=0x0FFF;       // PWM okres
```

```
   MIOS1.MPWMSM0PULR.R=0x0800;       // PWM wypełnienie
   MIOS1.MPWMSM0SCR.B.EN=1;          // PWM uruchom
// Inicjacja dzielnika częstotliwości
   MIOS1.MCPSMSCR.B.PSL = 2;         // dzielnik częstotliwości
                                     //   2 (40MHz/2=20MHz)
   MIOS1.MCPSMSCR.B.PREN = 1;        // włącz licznik
// Konfiguracja przerwań dla MIOS1
   MIOS1.MIOS1LVL0.B.LVL=1;          // Ustaw poziom przerwań na 1
   MIOS1.MIOS1ER0.B.EN6 = 0;         // Przerwanie  1-włącz 0-wyłącz
/****************** END MIOS1 ******************************/
}
```

### 5.5.5 Konfiguracja modułów TPU

Moduły TPU3 [20] [21] są rozwinięciem dobrze już znanej wersji TPU. Są to niezależne moduły mikrokontrolera, które można w dowolny sposób zaprogramować jako różne funkcje czasowe. Ich największą zaletą jest to, że każda z nich posiada swoją niezależną jednostkę, co pozwala odciążyć główną jednostkę od czasochłonnych operacji czasowo-licznikowych. W sterowniku skonfigurowano obydwa dostępne układy TPU. Moduł TPU_A został wykorzystany głównie do obsługi enkoderów jako dekodery kwadraturowe. Moduł TPU_B odpowiada za sterowanie serwomechanizmami oraz układem oświetlenia RGB.

```
void Tpu_Init()
{
/*********************** TPU ******************************/
/*********************** TPU_A ****************************/
   TPU_A.TPUMCR3.B.PWOD=1;     // Ten bit trzeba ustawić przed konfiguracją
                               //   TCR1P/TCR2P ponieważ pola konfiguracji
                               //   dzielnika w rejestrze MCR domyślnie
                               //   można ustawiać tylko raz
   TPU_A.TPUMCR.B.EMU=1;       // Tryb pracy z indywidualna maska funkcji
   TPU_A.TPUMCR.B.T2CG=1;      // Ustaw źródło zegara TCR2 na wew.
   TPU_A.TPUMCR.B.PSCK=1;      // Dzielnik źródła zegara TCR1 1=/4 0=/32
   TPU_A.TPUMCR.B.TCR1P=0;     // Dzielnik zegara TCR1 0-1 1-2 2-4 3-8
   TPU_A.TPUMCR.B.TCR2P=0;     // Dzielnik zegara TCR2 0-1 1-2 2-4 3-8
   TPU_A.TICR.B.CIRL = 4;      // Ustaw poziom przerwań na 4
   TPU_A.TICR.B.ILBS = 0;      // Wyłącz przerwania
/*********************** TPU_B ************************/
   TPU_B.TPUMCR3.B.PWOD=1;
   TPU_B.TPUMCR.B.EMU=1;
   TPU_B.TPUMCR.B.T2CG=1;
   TPU_B.TPUMCR.B.PSCK=1;      // źródło dzielnika zegara TCR1 1=/4 0=/32
   TPU_B.TPUMCR.B.TCR1P=0;     // Dzielnik zegara TCR1 0-1 1-2 2-4 3-8
   TPU_B.TPUMCR.B.TCR2P=2;     // Dzielnik zegara TCR2 0-1 1-2 2-4 3-8
   TPU_B.TICR.B.CIRL = 5;      // Ustaw poziom przerwań na 4
   TPU_B.TICR.B.ILBS = 0;      // nie używaj przerwań
/*********************** TPU CHAN CONF ********************/
// konfiguracja sterowań dla serwomechanizmów
//ch0
```

```
   tpu_pwm_init(&TPU_B,CHAN0,3,PWM_PER_SERW,PWM_PER_SERW/15,TPU_PWM_TCR2);
//ch1
   tpu_pwm_init(&TPU_B,CHAN1,3,PWM_PER_SERW,PWM_PER_SERW/15,TPU_PWM_TCR2);
//ch2
   tpu_pwm_init(&TPU_B,CHAN2,3,PWM_PER_SERW,PWM_PER_SERW/15,TPU_PWM_TCR2);
//ch3
   tpu_pwm_init(&TPU_B,CHAN3,3,PWM_PER_SERW,PWM_PER_SERW/15,TPU_PWM_TCR2);
//ch4
   tpu_pwm_init(&TPU_B,CHAN4,3,PWM_PER_SERW,PWM_PER_SERW/15,TPU_PWM_TCR2);
// konfiguracja sterowań dla układu oświetlenia RGB
//ch5
   tpu_pwm_init(&TPU_B,CHAN5,3,PWM_PER_LUX,PWM_PER_LUX/2,TPU_PWM_TCR2);
//ch6
   tpu_pwm_init(&TPU_B,CHAN6,3,PWM_PER_LUX,PWM_PER_LUX/2,TPU_PWM_TCR2);
//ch7
   tpu_pwm_init(&TPU_B,CHAN7,3,PWM_PER_LUX,PWM_PER_LUX/2,TPU_PWM_TCR2);
// konfiguracja obsługi enkoderów
// ch8 & ch9
   tpu_qdec_init(&TPU_A,CHAN8,3,0);
// ch10 & ch11
   tpu_qdec_init(&TPU_A,CHAN10,3,0);
   /****************** END TPU ******************************/
}
```

Przykład odczytu wartości zliczonych przez enkodery

```
  result0=(tpu_qdec_position(&TPU_A, CHAN8));
  result1=(tpu_qdec_position(&TPU_A, CHAN10));
```

### 5.5.6 Konfiguracja QSMCM

Moduł QSMCM ( Qued Serial Multi-Channel Module) [20] odpowiada za kolejkową komunikację przy pomocy interfejsu SPI. Komunikację tą wykorzystują dwa podzespoły robota: sterownik silników oraz moduł nawigacyjny INS. Przy obydwu układach wykorzystuje się transmisję o długości 12 bitów.

```
void Qsmcm_Init()
{
/********************** QSPI ****************************/
   QSMCM.DDRQS.R = 0xfe;         // PCSx, SCK, MOSI - wyj, MISO - wej
   QSMCM.PORTQS.R = DESELECT<<3;
   QSMCM.PQSPAR.R = 0x7b;        // Wszystkie piny dla QSPI
   QSMCM.SPCR1.R = 0x1717;       // DSCKL=?us, DTL=?us  ustaw opóźnienia
   QSMCM.SPCR2.R = 0x0700;       // kolejkuj od 0 do 7 (8 words),
   QSMCM.SPCR0.R = 0xB00F;       // master, 12bitów, cpol 0, cpha 0, 3MHz
   QSMCM.TRANRAM[0].R = (VUINT16)0xF00;  // komenda dla kanału 0
   QSMCM.COMDRAM[0].R = QsmDT|  QsmBITSE  | QsmDSCK | ADCSEL;
                                        // 12 bitów  QSPI
   QSMCM.TRANRAM[1].R = (VUINT16)0x000;  // komenda dla kanału 1
   QSMCM.COMDRAM[1].R =  QsmDT  |  QsmBITSE | QsmDSCK | ADCSEL;
```

```
                                        // 12 bitów  QSPI
   QSMCM.TRANRAM[2].R = (VUINT16)0x100;  // komenda dla kanału 2
   QSMCM.COMDRAM[2].R =  QsmDT |  QsmBITSE| QsmDSCK | ADCSEL;
                                        // 12 bitów  QSPI
   QSMCM.TRANRAM[3].R = (VUINT16)0x200;  // komenda dla kanału 3
   QSMCM.COMDRAM[3].R =  QsmDT |  QsmBITSE  | QsmDSCK | ADCSEL;
                                        // 12 bitów  QSPI
   QSMCM.TRANRAM[4].R = (VUINT16)0x700;  // komenda dla kanału 4
   QSMCM.COMDRAM[4].R = QsmDT  |  QsmBITSE | QsmDSCK | ADCSEL;
                                        // 12 bitów  QSPI
   QSMCM.TRANRAM[5].R = (VUINT16)0x800;  // komenda dla kanału 5
   QSMCM.COMDRAM[5].R =  QsmDT |  QsmBITSE  | QsmDSCK | ADCSEL;
                                        // 12 bitów  QSPI
   QSMCM.TRANRAM[6].R = (VUINT16)(0x0300)+(128); // komenda dla kanału 6
   QSMCM.COMDRAM[6].R =  QsmDT |  QsmBITSE  | QsmDSCK | DACSEL;
                                              // 12 bitów  QSPI
   QSMCM.TRANRAM[7].R = (VUINT16)(0x0700)+(128); // komenda dla kanału 7
   QSMCM.COMDRAM[7].R =  QsmDT |  QsmBITSE  | QsmDSCK | DACSEL;
                                              // 12 bitów  QSPI
   /****************** END QSPI *****************************/
}
```

Poniżej przedstawiono przykład uruchomienia transmisji.

```
    QSMCM.SPCR1.B.SPE=1;  // uruchom transmisję
if (QSMCM.SPSR.B.SPIF==1)
   {
   result1=QSMCM.RECRAM[2].R;
   result2=QSMCM.RECRAM[3].R;
   result3=QSMCM.RECRAM[4].R;
   result4=QSMCM.RECRAM[5].R;
   result5=QSMCM.RECRAM[1].R;
   QSMCM.SPSR.B.SPIF=0; // zgaś flagę ukończenia
   QSMCM.SPCR1.B.SPE=1; // uruchom kolejną transmisję
   }
```

### 5.5.7 Konfiguracja QADC64

Moduł QADC ( Qued Analog-To-Digital Converter Module-64) [20] to kolejkowy 10-bitowy, 16 kanałowy, przetwornik analogowo-cyfrowy. MPC555 zawiera dwa niezależne moduły QADC_A oraz QADC_B. W oprogramowaniu sterownika, skonfigurowano pierwszy moduł przetwornika do pomiaru napięcia akumulatora oraz trzech czujników odległości. Drugi moduł odpowiedzialny jest za pomiary dotykowego panelu, zainstalowanego na wyświetlaczu LCD. Wszystkie wyprowadzenia przetwornika mogą także pracować jako zwykłe linie wejściowe, a połowa z nich także jako wyjściowe.

```
void Qadc_Init()
{
/********************** QADC ****************************/
/********************** QADC_A **************************/
```

```
   QADC_A.QADC64MCR.B.STOP = 0;   // włącz zegar dla QADC
   QADC_A.QADC64MCR.B.FRZ  = 0;   // ignoruj sygnał FREEZE
   QADC_A.QADC64MCR.B.SUPV = 1;
   QADC_A.QADC64INT.B.IRL1 = 7;   // poziom przerwań dla pierwszej kolejki
   QADC_A.QADC64INT.B.IRL2 = 7;   // poziom przerwań dla drugiej kolejki
// Control Register 0
   QADC_A.QACR0.B.MUX = 0;        // wybieraj automatycznie
   QADC_A.QACR0.B.PSH = 11;       // High QLCK Time = (PHS + 1) / FSys
   QADC_A.QACR0.B.PSL = 7;        // Low  QLCK Time = (PHL + 1) / FSys
// Control Register 1 (queue 1)
   QADC_A.QACR1.B.CIE1 = 0;       // Wyłącz przerwania od skończonego
                                  //   cyklu pomiarów
   QADC_A.QACR1.B.PIE1 = 0;       // Wyłącz przerwania od pauzy
                                  //   cyklu pomiarów
   QADC_A.QACR1.B.MQ1  = 0x01;    // pojedynczy skan wyzwalany programowo
// konfiguracja kanałów pomiarowych 0...3
   QADC_A.CCW[0].R = QADC_CCW_BYP | QADC_CCW_QCLK16 | QADC_CCW_AN0;
   QADC_A.CCW[1].R = QADC_CCW_BYP | QADC_CCW_QCLK16 | QADC_CCW_AN1;
   QADC_A.CCW[2].R = QADC_CCW_BYP | QADC_CCW_QCLK16 | QADC_CCW_AN2;
   QADC_A.CCW[3].R = QADC_CCW_BYP | QADC_CCW_QCLK16 | QADC_CCW_AN3;
   QADC_A.CCW[4].R = QADC_CCW_EOQ;
/********************** QADC_B ***************************/
   QADC_B.QADC64MCR.B.STOP = 0;
   QADC_B.QADC64MCR.B.FRZ  = 0;
   QADC_B.QADC64MCR.B.SUPV = 0;
   QADC_B.QADC64INT.B.IRL1 = 7;
   QADC_B.QADC64INT.B.IRL2 = 7;
// Control Register 0
   QADC_B.QACR0.B.MUX = 0;
   QADC_B.QACR0.B.PSH = 11;
   QADC_B.QACR0.B.PSL = 7;
// Control Register 1 (queue 1)
   QADC_B.QACR1.B.CIE1 = 0;
   QADC_B.QACR1.B.PIE1 = 0;
   QADC_B.QACR1.B.MQ1  = 0x01;
// konfiguracja kanałów pomiarowych - tylko 0 kanał
   QADC_B.CCW[0].R = QADC_CCW_BYP | QADC_CCW_QCLK16 | QADC_CCW_AN1;
   QADC_B.CCW[4].R = QADC_CCW_EOQ;
/********************** QADC ****************************/
```

Poniżej przedstawiono przykładowy fragment kodu realizujący pojedynczy pomiar (cykl kolejki).

```
void read_analog_inputs()
{
   QADC_A.QACR1.B.SSE1 = 0x01;      // uruchom pojedynczy skan
   while( !(QADC_A.QASR0.B.CF1)) {} // czekaj na koniec przetwarzania
   QADC_A.QASR0.B.CF1 = 0;          // zgaś flagę końca przetwarzania
   result0=QADC_A.RJURR[0].R;       // kopiuj rezultat do zmiennej pomocniczej
   result1=QADC_A.RJURR[1].R;       // kopiuj rezultat do zmiennej pomocniczej
```

```
   result2=QADC_A.RJURR[2].R;       // kopiuj rezultat do zmiennej pomocniczej
   result3=QADC_A.RJURR[3].R;       // kopiuj rezultat do zmiennej pomocniczej
}
```

### 5.5.8 Obsługa przerwań w MPC555

Mikrokontroler MPC555 posiada wektory, tzw zdarzeń (Exceptions) [20] [19], a nie tradycyjnie wektory przerwań. W wektorach tych zapisuje się adres, pod którym znajduje się funkcja obsługująca dane zdarzenie. Wszystkie przerwania od poszczególnych modułów widziane są jako jedno i to samo zdarzenie (External Exception 0x500). Jednym ze sposobów odróżniania źródeł przerwań zewnętrznych jest nadanie każdemu z nich innego poziomu. Innym sposobem może być czytanie kolejno flag wystąpienia przerwania poszczególnych modułów. W sterowniku zrealizowano pierwsze rozwiązanie.

Za kontrolowanie przerwań odpowiedzialny jest moduł USIU, którego konfigurację przedstawiono powyżej. Warto wspomnieć, że bloki modułu USIU, takie jak PIT, PLL, Real-Time Clock itd., mogą zgłaszać przerwania na poziomach 0...7. Dodatkowo dochodzą przerwania od zewnętrznych, linii mikrokontrolera, które nie reprezentują tych samych poziomów przerwań. Daje w sumie 16 poziomów wszystkich przerwań. Moduły podłączone do IMB3 BUS takie jak TPU, MIOS1, QADC, TOUCAN, QSMCM mogą mieć przydzielane poziomy przerwań z zakresu 0...32 przy czym przerwania na poziomie $>7$ są zgłaszane do jednostki jako przerwania o poziomie 7. Ich rozstrzyganiem zajmuje się moduł UIMB.

W zaimplementowanej obsłudze rozpoznawania źródeł przerwań odczytuje się rejestr SIPEND.

```
void InterruptHandler(long cause)
{
UINT32 int_value = 0 ;
//*****************************************************
// Włącznie jednostki zmiennoprzecinkowej
//   ( w przerwaniach domyślnie jest wyłączana}
//*****************************************************
   asm{
   mfmsr r3
   ori  r3,r3,0x00002000
   mtmsr r3
   }
//*****************************************************
   int_value = USIU.SIPEND.R ;
   switch(cause)
   {
      case 0x500: // zewnętrzne przerwanie
         if (int_value&LEVEL1)
         {
            MIOS1.MIOS1SR0.B.FLG6 = 0;  // zgaś flagę przerwania
            Interrupt_MIOS();           // funkcja obsługująca przerwanie
                                        //   cykliczne skonfigurowane
                                        //   w module MIOS1
         }
```

```
        else if (int_value&LEVEL2)
        {
           USIU.PISCR.B.PS = 1;       // zgaś flagę przerwania
           Interrupt_PIT();           // funkcja obsługująca przerwanie PIT
        }
        else return;
   }
}
```

Po rozpoznaniu źródła przerwania następuje zgaszenie flagi oraz skok do funkcji obsługującej dane przerwanie.

### 5.5.9 Filtr Kalmana

Poniżej przedstawiono przykład implementacji filtra dla dowolnego mikrokontrolera [16]. W tym celu zadeklarowano dwie struktury *stan* oraz *kalm*. Pola tych struktur przechowują pomiary, wartości poszczególnych elementów macierzy $P$, $Q$ oraz wektora stanu $x$. Warto zwrócić uwagę na to, że nie ma konieczności obliczania wszystkich elementów macierzy $P$. W procesie filtracji biorą udział jedynie elementy $P_{11}$, $P_{13}$, $P_{21}$, $P_{31}$, $P_{33}$.

```
//*******************************************************
// Definicja struktury stanu - zmienna globalna
//*******************************************************
struct typ_state {
   VFLOAT64  alpha;         // odfiltrowane odchylenie
   VFLOAT64  dalpha;        // odfiltrowana prędkość odchylania
   VFLOAT64  pomiar_alpha;  // pomiar odchylenia
   VFLOAT64  pomiar_omega;  // pomiar prędkość kątowej
   VFLOAT64  phi1;          // obrót pierwszego koła
   VFLOAT64  phi2;          // obrót drugiego koła
   VFLOAT64  dphi1;         // prędkość obrotowa pierwszego koła
   VFLOAT64  dphi2;         // prędkość obrotowa pierwszego koła
   };
extern struct typ_state state;
//*******************************************************
// Definicja struktury dla F.Kalmana - zmienna globalna
//*******************************************************
struct typ_kalman {
   VFLOAT64  Q;       // Wartości przekątnej macierzy Q
   VFLOAT64  R;       // Wariancja pomiaru
   VFLOAT64  alpha;   // odchylenie
   VFLOAT64  omega;   // prędkość kątowa
   VFLOAT64  g_bias;  // dryft żyroskopu
   VFLOAT64  P11;     // wartości macierzy kowariancji P
   VFLOAT64  P13;
   VFLOAT64  P21;
   VFLOAT64  P31;
   VFLOAT64  P33;
   VFLOAT64  K1;      // wartości macierzy K
```

```
   VFLOAT64  K2;
   VFLOAT64  K3;
};
extern struct typ_kalman kalm;
```

W programie głównym należy wszystkie pola zainicjować odpowiednimi wartościami.

```
void Kalman_Init()
   {
// stan
   state.alpha=0;
   state.dalpha=0;
   state.pomiar_alpha=0;
   state.pomiar_omega=0;
// kalman
   kalm.Q=0.0001;
   kalm.R=1;
   kalm.alpha=0;
   kalm.omega=0;
   kalm.g_bias=0;
   kalm.P11=kalm.R;
   kalm.P13=0;
   kalm.P21=0;
   kalm.P31=0;
   kalm.P33=0;
   kalm.K1=0;
   kalm.K2=0;
   kalm.K3=0;
```

Teraz można przystąpić do rekursywnej pracy filtra. Do tego celu wykorzystano przerwanie skonfigurowane w module MIOS1. Wszystkie operacje arytmetyczne są wykonywane na zmiennych typu *float*. Proces obliczeń przebiega dość szybko, gdyż MPC555 wyposażony jest w 64-bitową jednostkę zmiennoprzecinkową.

```
#define dt 0.002 // przerwanie co 2ms, f=500Hz


void Interrupt_MIOS()
{
//***********************************************
// POMIAR
//***********************************************
   state.pomiar_alpha=odczytaj_akcelerometry();
   state.pomiar_omega=odczytaj_żyroskop();
//***********************************************
// FILTR KALMANA
//***********************************************
//////////////////
// PREDYKCJA
//_____________________________________________
```

```
//    |alpha  |   |1 0 -dt| |alpha  |    | dt|
//    |omega  |  =|0 0 -1 |*|omega  |   +| 1 |*gyro
//    |q_bias |k  |0 0  1 | |q_bias |k-1 | 0 |
// xk apriori
//
   kalm.alpha=kalm.alpha+(state.pomiar_omega-kalm.g_bias)*dt;
   kalm.omega=state.pomiar_omega-kalm.g_bias;
// kalm.g_bias=kalm.g_bias;
//-----------------------------------------------
// Pk=AP_{k-1}A^T+Q a priori
//
   kalm.P11=kalm.P11-kalm.P31*dt+kalm.P33*dt*dt-kalm.P13*dt+kalm.Q;
   kalm.P13=kalm.P13-kalm.P33*dt;
   kalm.P21=kalm.P33*dt-kalm.P31;
   kalm.P31=kalm.P31-kalm.P33*dt;
   kalm.P33=kalm.P33+kalm.Q;
//
////////////////////
// KOREKCJA
// -----------------------------------------
// Kk=PkH^T(HPkH^T+R)^-1
//
   kalm.K1=kalm.P11*(1/(kalm.P11+kalm.R));
   kalm.K2=kalm.P21*(1/(kalm.P11+kalm.R));
   kalm.K3=kalm.P31*(1/(kalm.P11+kalm.R));
// -----------------------------------------
// xk=xk-K(zk-Hxk) a posteriori
//
   kalm.apha=kalm.alpha+kalm.K1*(state.pomiar_alpha-kalm.alpha);
   kalm.omega=kalm.omega+kalm.K2*(state.pomiar_alpha-kalm.alpha);
   kalm.g_bias=kalm.g_bias+kalm.K3*(state.pomiar_alpha-kalm.alpha);
// -----------------------------------------
// Pk=(1-KkH)Pk  a posteriori
//
   kalm.P11=(1-kalm.K1)*kalm.P11;
   kalm.P13=(1-kalm.K1)*kalm.P13;
   kalm.P21=kalm.P21-kalm.P11*kalm.K2;
   kalm.P31=kalm.P31-kalm.P11*kalm.K3;
   kalm.P33=kalm.P33-kalm.P13*kalm.K3;
//
////////////////////////////
// AKTUALIZACJA WEKTORA STANU
//
state.alpha=kalm.theta;
state.dalpha=kalm.omega;
//
}
```

### 5.5.10 Sterownik

W robocie zaimplementowano jedynie prosty, podwójny regulator PID. Jego implementacja w układach mikroprocesorowych nie jest zbyt skomplikowana. Poniżej przedstawiono fragment kodu, który odpowiada za dostarczenie odpowiednich sterowań do silników.

Zanim przedstawiony zostanie fragment kodu sterownika, konieczne jest zapoznanie się z niektórymi zmiennymi globalnymi wykorzystywanymi w sterowaniu.

```
//*******************************************************
// Definicja struktury ustawień PID - zmienna globalna
//*******************************************************
struct Settings_tag {
    struct {
      VFLOAT64 P;
      VFLOAT64 I;
      VFLOAT64 D;
      VFLOAT64 angle_int_err;
    } angle_PID;
    struct {
      VFLOAT64 P;
      VFLOAT64 I;
      VFLOAT64 D;
      VFLOAT64 diferential_int_err;
      VFLOAT64 diferential_old_err;
    } diferential_PID;
};
extern struct Settings_tag settings;
//*******************************************************
// Definicja struktury stanu - zmienna globalna
//*******************************************************
struct typ_state {
   VFLOAT64  alpha;         // odfiltrowane odchylenie
   VFLOAT64  dalpha;        // odfiltrowana prędkość odchylania
   VFLOAT64  pomiar_alpha;  // pomiar odchylenia
   VFLOAT64  pomiar_omega;  // pomiar prędkość kątowej
   VFLOAT64  phi1;          // obrót pierwszego koła
   VFLOAT64  phi2;          // obrót drugiego koła
   VFLOAT64  dphi1;         // prędkość obrotowa pierwszego koła
   VFLOAT64  dphi2;         // prędkość obrotowa pierwszego koła
   };
extern struct  typ_state state;
```

Kod sterownika jest następujący

```
void Interrupt_PIT()
{
//*******************************************************
//Prosty podwójny regulator PID
//*******************************************************
```

```
   FLOAT64 ster1=0; //sterowanie pierwszego silnika - zmienna pomocnicza
   FLOAT64 ster2=0; //sterowanie pierwszego silnika - zmienna pomocnicza

   FLOAT64 angle_err=0;  //błąd odchylenia od pionu

   FLOAT64 diferential_err=0;       //błąd różnicy prędkości kół
   FLOAT64 diferential_deriv_err=0; //różnica błędu różnicy prędkości kół

////////////////////////////////////
// Pierwszy regulator PID (odchylenie od pionu)
//
   angle_err=state.alpha;                        //błąd odchylenia od pionu
   settings.angle_PID.angle_int_err+=angle_err; // całkowanie błędu
// Sprawdza granice całkowania
   if (settings.angle_PID.angle_int_err>0.1)
       settings.angle_PID.angle_int_err=0.1;
   if (settings.angle_PID.angle_int_err<-0.1)
       settings.angle_PID.angle_int_err=-0.1;
//
////////////////////////////////////
// Drugi regulator PID (różnica prędkości kół)
//
   diferential_err=state.dphi1-state.dphi2;         // błąd różnicy
   settings.diferential_PID.diferential_int_err+=diferential_err;
                                                     // całkowanie błędu
// Sprawdza granice całkowania
   if (settings.diferential_PID.diferential_int_err>10)
       settings.diferential_PID.diferential_int_err=10;
   if (settings.diferential_PID.diferential_int_err<-10)
       settings.diferential_PID.diferential_int_err=-10;
//
   diferential_deriv_err=diferential_err
   -settings.diferential_PID.diferential_old_err;   // różniczkowanie błędu
//
////////////////////////////////////
// Wyliczanie sterowania dla pierwszego koła
//
   ster1=(angle_err*settings.angle_PID.P
     +state.dalpha*settings.angle_PID.D
     +settings.angle_PID.angle_int_err*settings.angle_PID.I)
     +diferential_err*settings.diferential_PID.P
     +settings.diferential_PID.diferential_int_err*settings.diferential_PID.I
     +diferential_deriv_err*settings.diferential_PID.D;
//
////////////////////////////////////
// Wyliczanie sterowania dla drugiego koła
//
   ster2=(angle_err*settings.angle_PID.P
    +state.dalpha*settings.angle_PID.D
```

```
    +settings.angle_PID.angle_int_err*settings.angle_PID.I)
    -diferential_err*settings.diferential_PID.P
    -settings.diferential_PID.diferential_int_err*settings.diferential_PID.I
    -diferential_deriv_err*settings.diferential_PID.D;
//
////////////////////////////////////
// Zadawanie sterowania na silniki
//
   Control_LR(ster1,ster2);
}
```

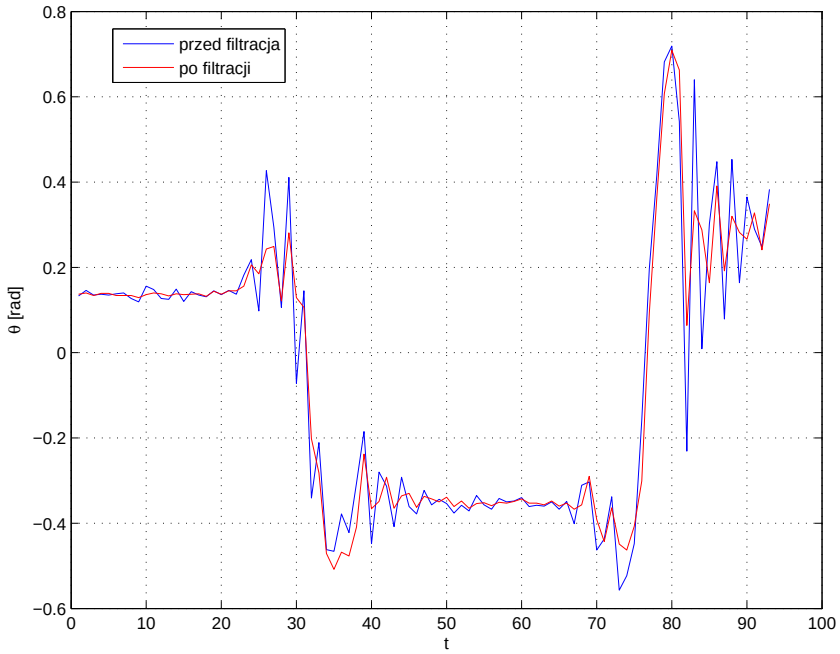
## 5.6 Wyniki badań

Na zbudowanym obiekcie przeprowadzono szereg eksperymentów, w celu weryfikacji teoretycznych rozważań. Zaimplementowany prosty regulator PID sprawdzał się w miarę dobrze. Utrzymywał równowagę robota, a przy tym lekko wahał się i podjeżdżał cyklicznie do przodu i do tyłu. Największym problemem w procesie sterowania okazały się spore luzy na przekładniach. Należałoby w tym przypadku podjąć próbę uwzględnienia tego zjawiska w procesie sterowania, np. wprowadzając histerezę.

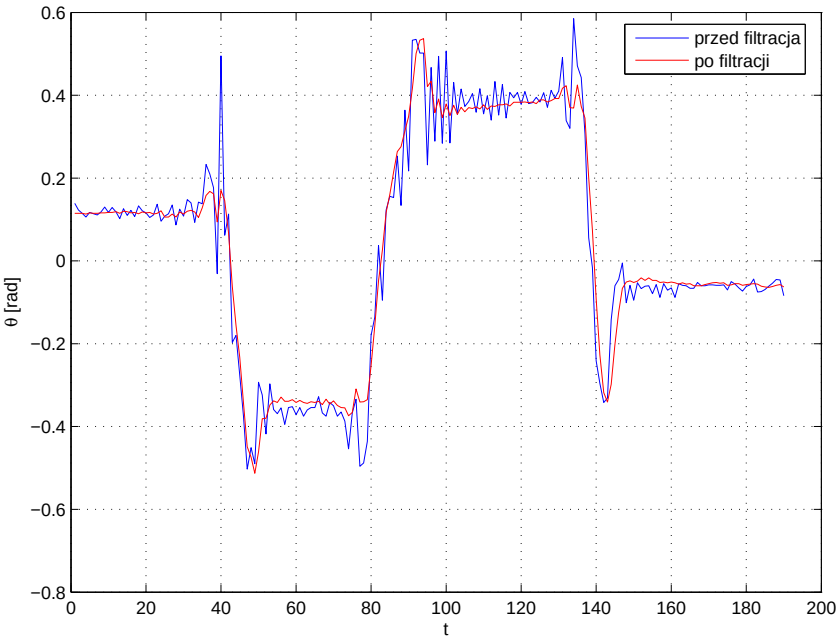
Aby luzy te nie wpływały znacząco na jakość sterowania, w procesie filtracji pomiarów tak dobrano współczynniki, aby luzy te nie były przez system wykrywane. Tak silne ustawienie parametrów filtracji, niestety prowadziło także do zatracenia istotnych informacji, takich jak szybkie zmiany odchylenia od pionu. Zjawisko to m.in. powodowało, że robot cały czas się kołysał. Poniżej przedstawiono kilka wykresów, które ilustrują działanie modułu INS oraz sterownika utrzymującego równowagę. Większość parametrów została dobrana eksperymentalnie.

Układ pomiarowy INS przebadano dla różnych wartości macierzy $Q$ i $R$. Pierwsze trzy wykresy (rysunki 5.28, 5.29, 5.30) uzyskano badając moduł przy dużych wychyleniach od pionu rzędu 0,4 *rad*. Czwarty wykres (rysunek 5.31) pokazuje jak pracuje filtr przy bardzo silnych ustawieniach parametrów. Badanie to przeprowadzono w trakcie pracy robota (utrzymywanie równowagi).

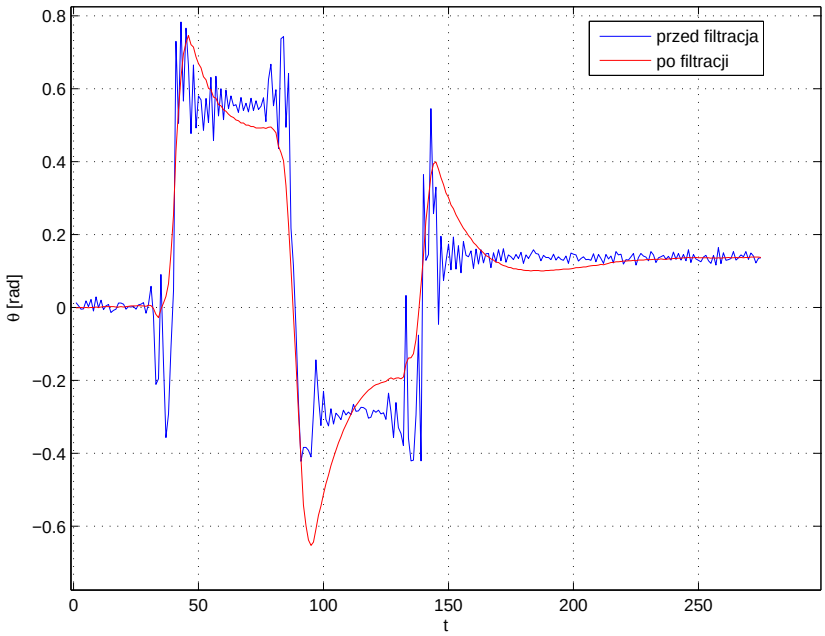
Kolejne wykresy ilustrują działane sterownika. Na pierwszym z nich (rysunek 5.32) przedstawiono odfiltrowaną wartość odchylenia od pionu robota w trakcie normalnej pracy. Można zaobserwować jak robot „kołysze się" wokół zera. Drugi wykres (rysunek 5.33) przedstawia prędkość odchylenia od pionu. Trzeci wykres (rysunek 5.34) to pomiar prędkości kół. Widać, kiedy prędkość ta jest równa zero. Jak wspominano w części teoretycznej, silniki działają siłą nie tylko na koła, ale i na wahadło (korpus) robota. Czas, w którym prędkość jest równa zero to chwile, kiedy moment siły jest niejako odbierany przez wahadło. Dopiero jak wzrośnie odchylenie, następuję obrót koła względem platformy oraz jej przemieszczenie. Ostatni wykres (rysunek 5.35) ilustruje wartości sterowań w trakcie pracy robota.



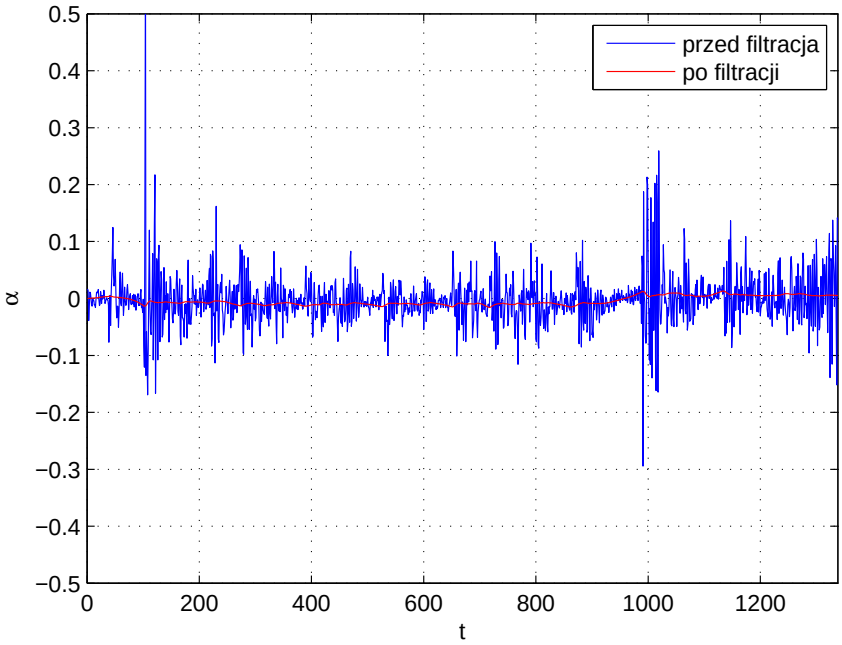

Rysunek 5.28 Pomiar odchylenia kątowego dla parametrów $Q = I$ i $R = 1$



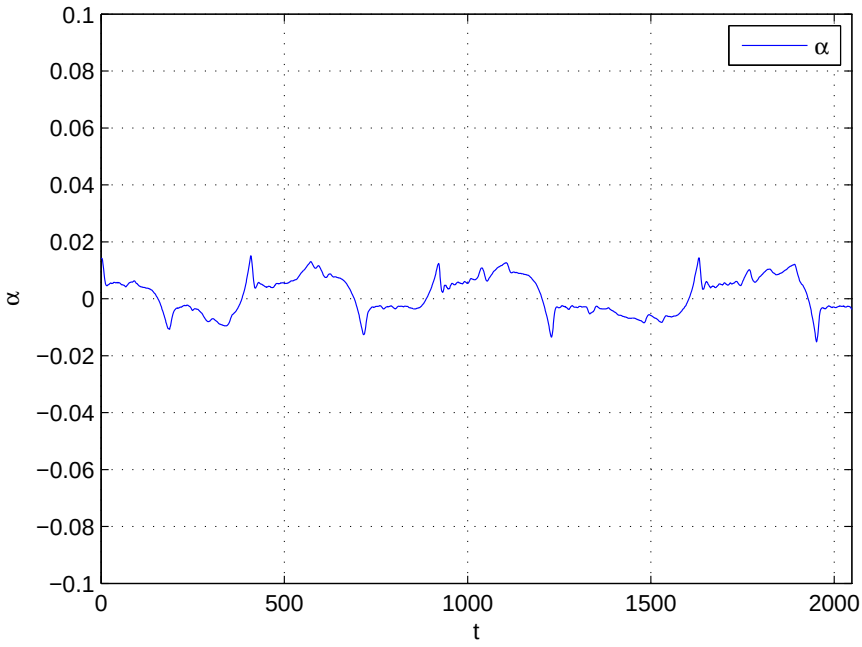

Rysunek 5.29 Pomiar odchylenia kątowego dla parametrów $Q = I \cdot 0.01$ i $R = 10$



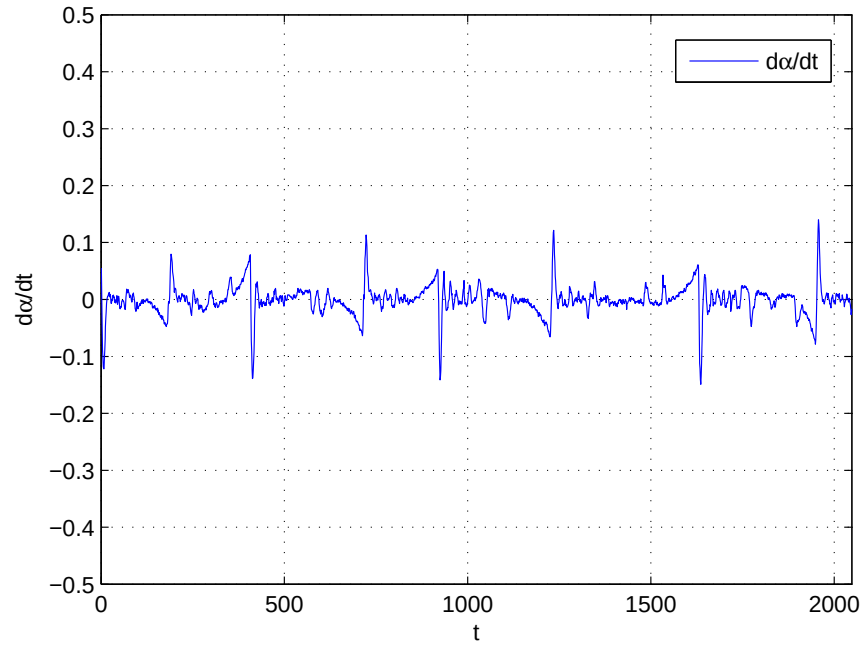

Rysunek 5.30 Pomiar odchylenia kątowego dla parametrów $Q = I \cdot 0.0001$ i $R = 1$



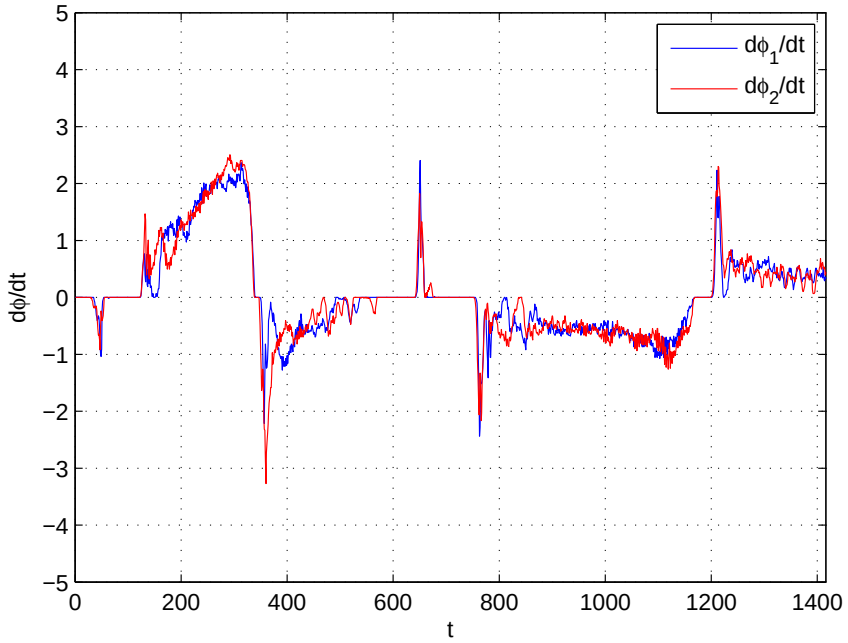

Rysunek 5.31 Pomiar odchylenia kątowego w trakcie pracy robota przy parametrach $Q = I \cdot 0.00001$ i $R = 1000$



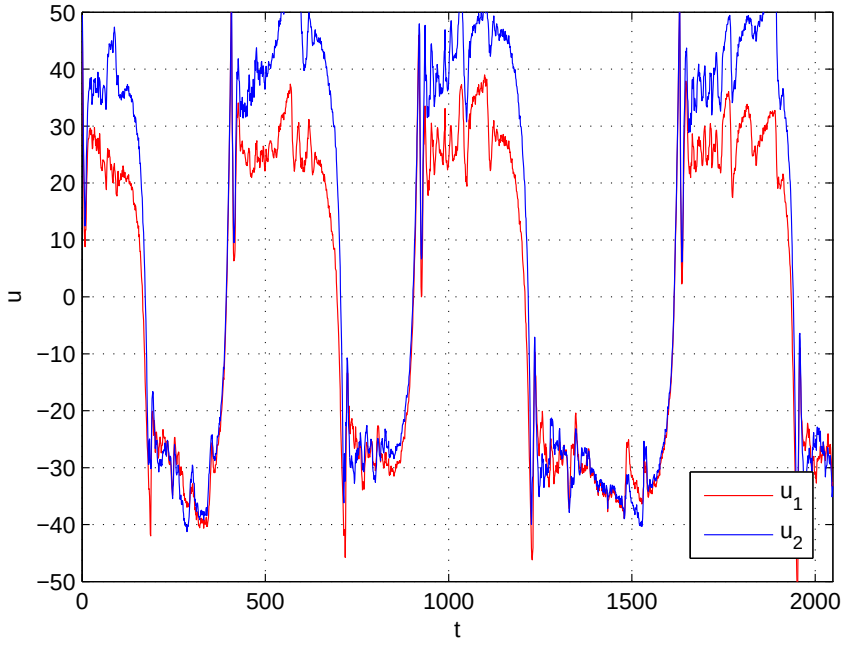

Rysunek 5.32 Pomiar odchylenia kątowego w trakcie pracy robota przy parametrach $Q = I \cdot 0.00001$ i $R = 1000$




Rysunek 5.33 Pomiar prędkości kątowej odchylenia od pionu




Rysunek 5.34 Pomiar prędkości kół w trakcie pracy robota




Rysunek 5.35 Przebiegi sterowań obydwu kół

# Rozdział 6

# Podsumowanie

Praca dotyczyła modelowania, sterowania i praktycznej implementacji robota balansującego (mobilne odwrócone wahadło). Wyliczony nieliniowy model dynamiki robota, w porównaniu z modelami innych podobnych konstrukcji wydaje się być prawidłowy. Po przeprowadzeniu wielu symulacji można stwierdzić, że zarówno sterowniki liniowe jak i nieliniowe dają podobny efekt. Cel sterowania jest spełniony. Warto jednak podkreślić, że nieliniowe sterowniki bez problemu radzą sobie nie tylko z utrzymywaniem równowagi, nawet przy bardzo dużych odchyleniach robota od pionu, ale także pozwalają trajektorię wychylenia. Jednak w praktyce bardzo rzadko doprowadza się do takich skrajnych sytuacji. Porównując jakość sterowania, nieliniowa wersja sterownika wydaje się znacznie lepsza, o szerszych możliwościach. Wynika to z faktu, że brany jest pod uwagę pełny nieliniowy model dynamiki.

Rozważania teoretyczne zilustrowano nie tylko symulacjami, ale również powstał robot balansujący „KOSMOS". Z praktycznego punktu widzenia okazuje się, że zarówno wersja sterownika liniowego jak i nieliniowego nie wymaga systemów o dużej wydajności obliczeniowej. A zatem, wybrana do eksperymentów jednostka centralna MPC555 firmy Freescale z pewnością poradziłaby sobie z realizacją nieliniowego sterownika. W trakcie budowy robota wielokrotnie zmieniano koncepcję. Niektóre parametry fizyczne elementów składowych znacznie różniły się od tych, które przyjęto w trakcie symulacji. Naj-większym mankamentem konstrukcji były użyte do przeniesienia napędu przekładnie. Posiadały one dość spore luzy, których w żadnym opracowanym sterowniku nie uwzględniono. Luzy napędów powodowały, że robot nieustannie wpadał w wibracje. Po wielu próbach z zastosowaniem prostego regulatora PID i silnymi nastawami Filtra Kalmana, udało się doprowadzić do tego, że robot zaczął poprawnie utrzymywać równowagę.

Konstrukcja jest wciąż rozwijana. Obecnie przeprowadzono testy robota z zainstalowanym w korpusie laptopie. W przyszłości zostaną wymienione napędy na precyzyjne oraz zostaną zaimplementowane zasymulowane algorytmy. W trakcie badań nad modułem nawigacyjnym INS autor wnioskuje, że zdecydowanie lepiej jest użyć inklinometr zamiast akcelerometrów. Ten pierwszy czujnik pozbawiony jest błędów związanych z pomiarem poziomych przyspieszeń platformy. Dodatkowo metoda pomiaru odchylenia, na podstawie pomiarów przyspieszeń w dwóch osiach, sprawdza się jedynie podczas ruchu platformy po powierzchniach płaskich lub nachylonych jedynie wzdłuż kierunku jazdy. Gdy robot przechyla się na boki pomiar ten staje się błędny. Po kilku eksperymentach okazało się, im niższe ułożenie środka ciężkości, tym lepiej robot utrzymywał pion.

Roboty to konstrukcje, których ludzkość wciąż się obawia. Badania nad robotami społecznymi wskazują, w którym kierunku powinno się kierować prace nad urzeczywistnianiem mechanicznych towarzyszy. Zbytnie podobieństwo do człowieka nie jest


Rysunek 6.1 Studenci podczas zabawy z robotem przed Wydziałem Elektroniki PWr

wskazane, gdyż czyni maszynę przerażającą [13]. Balansująca platforma pomimo tego, że nie posiada kończyn, w pewnym sensie przypomina napęd kroczący. Jest wyposażona w dwa punkty podparcia oraz także utrzymuje równowagę. W trakcie testów skonstruowanego robota okazało się, że tego typu konstrukcja robota jest milej odbierana przez człowieka. Delikatne ruchy robota, uwidocznione problemy z utrzymywaniem równowagi, zwłaszcza na nierównych powierzchniach niejako odsłaniały niedoskonałości konstrukcji, czyniąc ją bardzo naturalną w zachowaniu i nie sprawiająca zagrożenia.

# Bibliografia


[1] TCHOŃ K., MAZUR A., DULĘBA I., HOSSA R., MUSZYŃSKI R. *Manipulatory i roboty mobilne: modele, planowanie ruchu, sterowanie, Akademicka Oficyna Wydawnicza PLJ*, Warszawa, 2000.

[2] KACZOREK T. *Teoria sterowania i systemów* Wydawnictwo Naukowe PWN.

[3] HOSSA R. *Modele i algorytmy sterowania kołowych robotów mobilnych, rozprawa doktorska,* Politechnika Wrocławska, 1996.

[4] MAZUR A. *Model dynamiki i kinematyki manipulatora mobilnego typu RTR* Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, Wrocław, 1999.

[5] WELCH G.,BISHOP G.: *An Introduction to the Kalman Filter.* University of North Carolina at Chapel Hill Department of Computer Science Chapel Hill, NC 27599-3175

[6] NEGENBORN R.: *Robot Localization and Kalman Filters. On finding your position in a noisy world.* Institute of Information and Computing Sciences in partial fulfilment of the requirements for the degree of Master of Science, specialized in Intelligent Systems

[7] PETER S. MAYBECK: *Stohastic models, estimation and control. Volume 1* Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base Ohio.

[8] CHI OOI R.*Balancing a Two-Wheeled Autonomous Robot* The University of Western Australia School of Mechanical Engineering, Final Year Thesis 2003.

[9] WNUK M. SZLAWSKI R. *Konstrukcja dwukołowego robota mobilnego RoBik* Raport serii SPR nr 12/2004. Instytut Cybernetyki Technicznej Politechniki Wrocławskiej. Wrocław 2004

[10] A. De LUCA, IANNITTI S., ORIOLO G. *Stabilization of the PR planar underactuated robot.* Proc. IEEE International Conference on Robotics and Automation (ICRA 2001).

[11] RATAJCZAK A., TCHOŃ K. *Control of underactuated robotic manipulators: an endogenous configuration space approach* Proc. IEEE Conf. on Methods and Models in Automation and Robotics MMAR 2007, pp. 985?990, Szczecin, 2007.

[12] SZLAWSKI R. *Konstrukcja podwójnego mostka H* Raport 15/10/2004. Wrocław 2007.

[13] WALTERS M.L., SYRDAL D.S., DAUTENHAHN K., BOEKHORST R., LEE KOAY K. *Avoiding the uncanny valley: robot appearance, personality and consistency of behavior in an attention-seeking home scenario for a robot companion* Received: 5 February 2007 / Published online: 20 November 2007.

[14] KĘDZIERSKI J. OSTROWSKI E. *Enkoder magnetyczny AS5040* Raport Koła Naukowego Robotyków „KoNaR" 2007.

[15] KĘDZIERSKI J. OSTROWSKI E. *Sterowanie wyświetlaczem graficznym z kontrolerem firmy Toshiba T6963C* Raport Koła Naukowego Robotyków „KoNaR" 2007.

[16] KĘDZIERSKI J. *Filtr Kalmana - zastosowania w prostych układach sensorycznych* Raport Koła Naukowego Robotyków „KoNaR" 2008.

[17] OSTROWSKI E. *MPC555 Development Board* Raport Koła Naukowego Robotyków „KoNaR" 2008.

[18] *phyCORE-MPC555* Hardware Manual, Edition July 2005.

[19] DUNLOP J., FUCHS J., MIHALIK S. *MPC555 Interrupts* Rev. 0, 26 July 2001.

[20] Freescale Semiconductor, Inc. *MPC555 / MPC556* Revised 15 October 2000.

[21] DEES R., LOELIGER J. *General TPU C Functions for the MPC500 Family* Rev. 0, 10/2002.

[22] http://www.segway.com

[23] *Segbot - Final project for the Introduction to Mechatronics class at the University of Illinois* http://coecsl.ece.uiuc.edu/ge423 /spring04/group9/index.htm, 2004.

[24] *±1.5g - 6g Three Axis Low-g Micromachined Accelerometer* Freescale Semiconductor, Technical Data, Rev 5, 03/2008.

[25] *±150°/s Single Chip Yaw Rate Gyro with Signal Conditioning* Analog Devices Inc.

[26] *12-BIT, 200-KSPS, 11 CHANNEL, LOW POWER, SERIAL ADC WITH INTERNAL REFERENCE* Texas Instruments, DECEMBER 2001.

[27] *+5V, Low-Power, 8-Bit Quad DAC with Rail-to-Rail Output Buffers* MAXIM, 19-1105, Rev 0. 8/96.

[28] *www.austriamicrosystems.com.* „AS5040 DataSheet Rev 1.6 2006".

[29] *CodeWarrior Development Tools IDE 5.1 Userñs Guide*