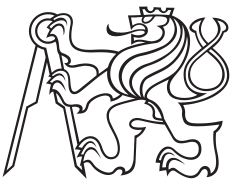
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Využití robota LEGO Mindstorms – návrh speciálních úloh

Michal Stračinský

Květen 2017
Vedoucí práce: Ing. Martin Hlinovský, Ph.D.

Poděkování / Prohlášení

Rád bych poděkoval vedoucímu práce Ing. Martinu Hlinovskému, Ph.D. za spolupráci a zapůjčení stavebnice NXT. Dále děkuji rodině a přátelům za podporu, porozumění a trpělivost.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

.....

Abstrakt / Abstract

Cílem této práce je navrhnout a realizovat inverzní kyvadlo (*segway*) pro stavebnici *LEGO Mindstorms NXT* a také pro něj vytvořit řídící mobilní aplikaci. Stavebnice *NXT* byla programována v jazyce NXC a mobilní aplikace byla napsána pro platformu Android. Regulátor pro stabilizaci inverzního kyvadla byl navržen metodou lineárního kvadratického regulátoru s integrační složkou. Výpočty potřebné pro návrh zpětnovazebné regulace byly provedeny v programu MATLAB. Chování zpětné vazby bylo modelováno v prostředí Simulink.

Klíčová slova: Inverzní kyvadlo, Segway, LEGO Mindstorms NXT, NXC, Mobilní aplikace, Android, Lineární kvadratický regulátor s integrační složkou, LQR, MATLAB, Simulink, Zpětnovazební řízení

The aim of this bachelor's thesis is to design and realize an inverted pendulum (*segway*) using a *LEGO Mindstorms NXT* kit and also create a mobile application for control of the *segway*. The *NXT* kit has been programmed using NXC language and mobile application has been developed for Android. Linear–quadratic regulator with integral action has been used to design feedback control for inverted pendulum. Calculations needed for development of feedback regulation had been done in MATLAB software. The feedback regulation has been modeled in Simulink.

Keywords: Inverted pendulum, Segway, LEGO Mindstorms NXT, NXC, Mobile application, Android, Linear–quadratic regulator with integral action, LQR, MATLAB, Simulink, Feedback control

Title translation: Usage of LEGO Mindstorms kit – desing of special tasks

Obsah

1 Úvod 1
2 Inverzní kyvadlo 2
2.1 Popis systému a matematický model 2
2.2 Stavový popis systému 4
2.3 Diskrétní řízení systému 5
2.4 Zapojení modelu v prostředí Simulink 7
2.5 Lineární kvadratický regulátor s integrační složkou 8
2.6 Hodnoty použitého regulátoru ..9
3 Práce s NXT 11
3.1 Jazyk NXC 11
3.2 Komunikace přes Bluetooth ... 11
3.2.1 Omezení toku dat 12
3.3 Ovládání servomotorů 12
3.4 Obsluha senzorů 12
3.5 Použité senzory 14
3.5.1 Gyroskopický senzor 14
3.5.2 Dotykový senzor 15
3.5.3 Ultrazvukový senzor 16
4 Mobilní aplikace 17
4.1 Volba platformy 17
4.2 Rozmístění ovládacích prvků .. 17
4.3 Komunikace přes Bluetooth ... 18
4.3.1 Třída TalkToNXT 19
5 Realizace speciálních úloh 20
5.1 Inverzní kyvadlo ovládané pomocí mobilního telefonu 20
5.1.1 Část NXT 20
5.1.2 Část Android 20
5.2 Inverzní kyvadlo sledující trajektorii vůdce 20
6 Závěr 22
Literatura 23
A Zadání bakalářské práce 25
B Zkratky a symboly 27

Tabulky / Obrázky

2.1. Souřadnice modelu 3
2.2. Převzaté fyzikální parametry modelu 3
2.3. Identifikované fyzikální parametry modelu 4
3.1. Kódování řídících akcí 12
3.2. Typy senzorů 13
3.3. Módy senzorů 13
3.4. Specifikace standardních senzorů 13
4.1. Veřejné funkce třídy TalkToNXT 19

2.1. První přiblížení k modelu segwaye 2
2.2. Souřadnicový systém modelu 3
2.3. Zapojení modelu v Simulinku . . . 7
2.4. Simulace reakce robotu na skokovou změnu reference 10
3.1. Struktura Bluetooth paketu . . . 11
3.2. Gyroskopický senzor 15
3.3. Dotykový senzor 15
3.4. Ultrazvukový senzor 16
4.1. Rozmístění ovládacích prvků . . 18

Kapitola 1
Úvod

Dánská společnost LEGO Group je světově známým výrobcem tzv. *Lego bricks*, Lego kostiček, od velkých DUPLO kostek pro děti až po řadu Technic, která zahrnuje např. ozubená kolečka a hřídelky. Možnosti kombinování jednotlivých dílů jsou prakticky neomezené a záleží pouze na kreativitě jednotlivce, co vše z nich dokáže sestavit.

V této práci použitá stavebnice *LEGO Mindstorms NXT*[1] obsahuje programovatelnou kostku *NXT*, ke které lze připojit až tři servomotory a čtyři senzory. Dále je možné kostku připojit k počítači pomocí USB nebo Bluetooth. Jednotlivé kostky také mohou komunikovat mezi sebou pomocí Bluetooth. Podrobnější informace ohledně *NXT* rozebírají práce Ing. Tomáše Bělíka[2] a Ing. Dana Martince[3], a proto v této práci nebudeme zacházet do detailů ohledně kostky a uvedeme pouze potřebné informace.

Problém zpětnovazební stabilizace inverzního kyvadla je v dnešní době již dobře prozkoumán, namátkou uveďme projekt Self – Erecting Inverted Pendulum vytvořen na ETH Zürich[4], a jeho řešení má svá praktická využití. Příkladem může být dvoukolový elektrický dopravní prostředek od společnosti Segway[1]. V této práci byl zpětnovazební regulátor navržen pomocí lineárního kvadratického regulátoru s integrační složkou. Díky integrální složce lze v ustáleném stavu mít nulovou regulační odchylku.

Při tvorbě zpětnovazebního řízení provádíme maticové výpočty a využíváme teorii řízení. Obojí podporuje např. program MATLAB od společnosti MathWorks[2]. Vzhledem k jeho oblíbenosti v průmyslu a široké podpoře jsme jej využili k návrhu regulátoru. MATLAB také umožňuje simulování modelů, a to pomocí rozšíření Simulink. Ukázku zapojení modelu v Simulinku můžeme vidět na obr. 2.3.

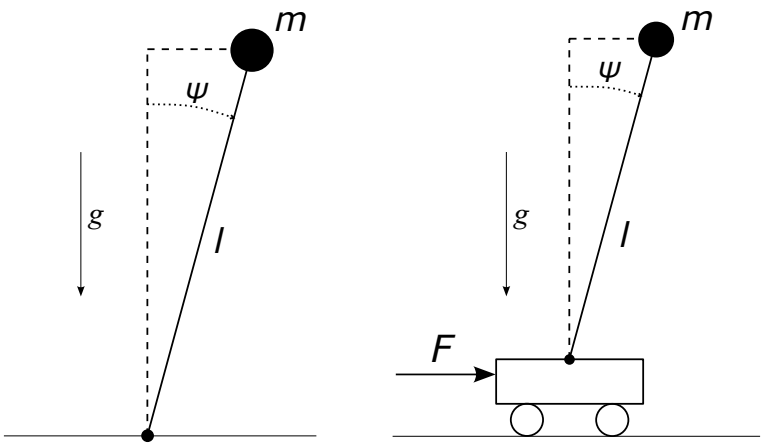
Součástí práce je mobilní aplikace. Uživatelsky přívětivá aplikace by měla mít snadné a intuitivní ovládaní. Rozvržením ovládacích prvků se zabývá část 4.2. Aplikace byla navržena pro platformu Android. Výběr platformy popisuje sekce 4.1.

[1] `http://cz-cs.segway.com/`
[2] `https://www.mathworks.com/`

Kapitola 2
Inverzní kyvadlo

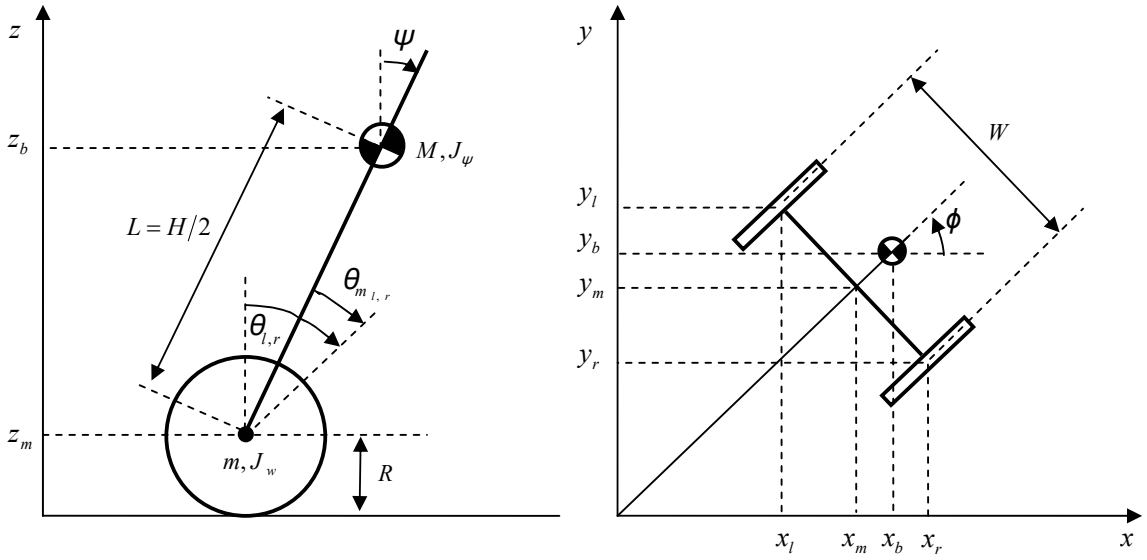
2.1 Popis systému a matematický model

Pod pojmem inverzní kyvadlo si můžeme představit matematické kyvadlo se závažím (hmotným bodem) nad pevným bodem otáčení, viz obr. 2.1 vlevo. Pokud bude úhel ψ roven nule, nedojde bez vnější vlivů k vychýlení kyvadla. V praxi ovšem musíme počítat s okolními vlivy a je potřeba zajistit způsob, jak kyvadlo vrátit do původní polohy. Jedním ze způsobů může být umístění kyvadla na pojízdný vozíček, viz obr. 2.1 vpravo, na který působíme vnější silou tak, aby se kyvadlo příliš nevychýlilo. Tento systém je pouze prvním přiblížením k našemu segwayi. V praxi totiž není vhodné působit vnější silou na vozíček, naopak mnohem snáze je lze pohánět rotačními motory.



Obrázek 2.1. První přiblížení k modelu segwaye.

Velmi realistický model vytvořil Yorihisa Yamamoto[5], kde se ke stabilizaci inverzního kyvadla používají dva rotační motory, stejně jako u segwaye, viz obr. 2.2. Jeho model a ním odvozené pohybové rovnice (3), (4) a (5) byly převzaty pro tuto práci. Některé fyzikální parametry modelu, např. veličiny motoru, kvůli náročnosti na změření a nezávislosti na konstrukci robotu byly také převzaty a uvádí je tabulka 2.2. Zbývající parametry se po identifikaci řádově shodují s výsledky z práce [5] a jsou napsány v tabulce 2.3. Důležité souřadnice z obr. 2.2 jsou popsány v tabulce 2.1.



Obrázek 2.2. Souřadnicový systém modelu.

Značení	Jednotka	Popis
θ_l	$[rad]$	Úhel natočení levého kola
θ_r	$[rad]$	Úhel natočení pravého kola
θ_{m_l}	$[rad]$	Úhel natočení motoru levého kola
θ_{m_r}	$[rad]$	Úhel natočení motoru pravého kola
ψ	$[rad]$	Úhel natočení těla segwaye od kolmice k zemi
ϕ	$[rad]$	Úhel natočení těla segwaye kolem osy z

Tabulka 2.1. Souřadnice modelu.

Fyzikální parametr	Značení	Hodnota	Jednotka
moment setrvačnosti DC motoru	J_m	$1 \cdot 10^{-5}$	$[kgm^2]$
odpor DC motoru	R_m	$6,69$	$[\Omega]$
EMF konstanta DC motoru	K_b	$0,468$	$[Vs/rad]$
momentová konstanta DC motoru	K_t	$0,317$	$[Nm/A]$
koeficient tření mezi tělem a DC motorem	f_m	$0,0022$	$[-]$
koeficient tření mezi kolem a povrchem	f_w	0	$[-]$

Tabulka 2.2. Převzaté fyzikální parametry modelu.

Pohybové rovnice byly odvozeny pomocí Lagrangeovy metody[1] vzhledem k souřadnicovému systému na obr. 2.2. Pro tuto metodu jsou potřeba nezávislé souřadnice $\vec{s}$:

$$\vec{s} = [\,\theta, \quad \psi, \quad \phi\,], \tag{1}$$

kde θ je průměrný úhel natočení levého a pravého kola. Hodnoty úhlů θ a ϕ dopočítáme z natočení kol pomocí rovnic (2), úhel ψ měříme gyroskopickým senzorem, více v sekci 3.5.1.

[1] `https://moodle.fel.cvut.cz/pluginfile.php/38039/mod_resource/content/1/msd_5_intro_to_lagrange_technique_talk.pdf`

Fyzikální parametr	Značení	Hodnota	Jednotka
tíhové zrychlení	g	$9,81$	$[m/s^2]$
hmotnost kola	m	$0,03$	$[kg]$
poloměr kola	R	$0,04$	$[m]$
moment setrvačnosti kola	J_w	$mR^2/2$	$[kgm^2]$
hmotnost těla	M	$0,6$	$[kg]$
šířka těla	W	$0,15$	$[m]$
hloubka těla	D	$0,04$	$[m]$
výška těla	H	$0,144$	$[m]$
vzdálenost těžiště těla od osy kol	L	$H/2$	$[m]$
moment setrvačnosti v ose ψ	J_ψ	$ML^2/3$	$[kgm^2]$
moment setrvačnosti v ose ϕ	J_ϕ	$M(W^2+D^2)/12$	$[kgm^2]$
převodový poměr	n	1	$[-]$

Tabulka 2.3. Identifikované fyzikální parametry modelu.

$$\begin{aligned} \theta &= \frac{1}{2}\cdot(\theta_l+\theta_r), \\ \phi &= \frac{R}{W}\cdot(\theta_r-\theta_l) \end{aligned} \tag{2}$$

Odvození pohybových rovnic a jejich linearizace je podrobněji popsána ve zdroji [5]. Linearizované pohybové rovnice mají tvar:

$$F_\theta = \alpha(v_l+v_r) - 2(\beta+f_w)\dot{\theta} + 2\beta\dot{\psi}, \tag{3}$$

$$F_\psi = -\alpha(v_l+v_r) + 2\beta\dot{\theta} - 2\beta\dot{\psi}, \tag{4}$$

$$F_\phi = \frac{W}{2R}\cdot\alpha(v_r-v_l) - \frac{W^2}{2R^2}\cdot(\beta+f_w)\dot{\phi}, \tag{5}$$

kde parametry α a β jsou:

$$\alpha = \frac{nK_t}{R_m}, \tag{6}$$

$$\beta = \frac{nK_tK_b}{R_m} + f_m. \tag{7}$$

2.2 Stavový popis systému

Základní stavové rovnice:

$$\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}\vec{u} \tag{8}$$

$$\vec{y} = \mathbf{C}\vec{x} + \mathbf{D}\vec{u}, \tag{9}$$

kde $\vec{x}$, $\vec{u}$, $\vec{y}$ jsou sloupcové vektory jednotlivých stavů, vstupů, výstupů a $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ matice vnitřních vazeb systému, vazeb systému na vstupy, vazeb výstupů na stavy, vazeb vstupů na výstupy.

Pro modelování segwaye použijeme dva stavové popisy dle rovnic (10) a (11). První se zabývá stabilizací a druhý natočením robotu. Tomu odpovídají zvolené stavové vektory, které nalezneme v (12) a (13). Jako vstupy použijeme napětí motorů, viz rovnice (14).

$$\begin{aligned} \dot{\vec{x}}_1 &= \mathbf{A}_1\vec{x}_1 + \mathbf{B}_1\vec{u} \\ \vec{y}_1 &= \mathbf{C}_1\vec{x}_1 + \mathbf{D}_1\vec{u} \end{aligned} \tag{10}$$

$$\begin{aligned} \dot{\vec{x}}_2 &= \mathbf{A}_2\vec{x}_2 + \mathbf{B}_2\vec{u} \\ \vec{y}_2 &= \mathbf{C}_2\vec{x}_2 + \mathbf{D}_2\vec{u} \end{aligned} \tag{11}$$

$$\vec{x}_1 = [\,\theta, \quad \psi, \quad \dot{\theta}, \quad \dot{\psi}\,]^T \tag{12}$$

$$\vec{x}_2 = [\,\phi, \quad \dot{\phi}\,]^T \tag{13}$$

$$\vec{u} = [\,v_l, \quad v_r\,]^T \tag{14}$$

Matice $\mathbf{A}_1$, $\mathbf{B}_1$, $\mathbf{A}_2$, $\mathbf{B}_2$ dostaneme úpravou pohybových rovnic. Podrobný popis úprav a symbolické vyjádření hledaných matic najdeme ve zdroji [5]. Vyčíslené matice jsou uvedeny v (15) a (16). Matice $\mathbf{C}_1$, $\mathbf{D}_1$, $\mathbf{C}_2$, $\mathbf{D}_2$ doplníme tak, abychom měli na výstupu všechny stavy.

$$\begin{aligned} \mathbf{A}_1 &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -409,7184 & -162,1273 & 162,1273 \\ 0 & 269,6273 & 78,1496 & -78,1496 \end{bmatrix}, \\ \mathbf{B}_1 &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 157,5798 & 157,5798 \\ -75,9576 & -75,9576 \end{bmatrix}, \\ \mathbf{C}_1 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{D}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{15}$$

$$\begin{aligned} \mathbf{A}_2 &= \begin{bmatrix} 0 & 1 \\ 0 & -96,2033 \end{bmatrix}, \mathbf{B}_2 = \begin{bmatrix} 0 & 0 \\ -49,8693 & 49,8693 \end{bmatrix}, \\ \mathbf{C}_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{D}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{16}$$

2.3 Diskrétní řízení systému

Námi uvažovaný model je řízen spojitě, čehož pomocí *NXT* nemůžeme dosáhnout. Musíme tedy při návrhu počítat s diskrétním řízením a vhodně zvolit vzorkovací periodu. Ta musí být dostatečně malá, abychom splnili vzorkovací teorém, viz rovnice (17), a přitom dostatečně velká, abychom stíhali napočítat řídící napětí pro servomotory. ω_s je vzorkovací frekvence a ω_{max} maximální frekvence v měřeném signálu.

$$\omega_s > 2\omega_{max} \tag{17}$$

Po částečné implementaci jsme provedli měření a průměrná doba běhu řídící smyčky se pohybovala kolem 1 ms. Abychom měli jistotu, že po přidání dalších funkcí programu nepřekročíme vzorkovací periodu T_s, zvolili jsme 10 ms. Tedy podle (17) maximální frekvence měřeného signálu může být 50 Hz, což je více než dostačují pro naši aplikaci.

Dále budeme pracovat s diskrétním stavovým popisem. Pravidla pro diskretizaci popisuje [6]. Rovnice (10) a (11) upravíme na:

$$\begin{aligned} \vec{x}_1(k+1) &= \mathbf{A}_{1_d}\vec{x}_1(k) + \mathbf{B}_{1_d}\vec{u}(k), \\ \vec{y}_1(k) &= \mathbf{C}_1\vec{x}_1(k) + \mathbf{D}_1\vec{u}(k), \end{aligned} \tag{18}$$

$$\begin{aligned} \vec{x}_2(k+1) &= \mathbf{A}_{2_d}\vec{x}_2(k) + \mathbf{B}_{2_d}\vec{u}(k), \\ \vec{y}_2(k) &= \mathbf{C}_2\vec{x}_2(k) + \mathbf{D}_2\vec{u}(k). \end{aligned} \tag{19}$$

Během přechodu k diskrétnímu času musíme přepočítat stavové matice. Převod nám ukazují rovnice (20) a (21).

$$\begin{aligned} \mathbf{A}_{1_d} &= e^{\mathbf{A}_1 \cdot T_s} \\ \mathbf{B}_{1_d} &= \left(\int_0^{T_s} e^{\mathbf{A}_1 \cdot t} dt \right) \mathbf{B}_1 \end{aligned} \tag{20}$$

$$\begin{aligned} \mathbf{A}_{2_d} &= e^{\mathbf{A}_2 \cdot T_s} \\ \mathbf{B}_{2_d} &= \left(\int_0^{T_s} e^{\mathbf{A}_2 \cdot t} dt \right) \mathbf{B}_2 \end{aligned} \tag{21}$$

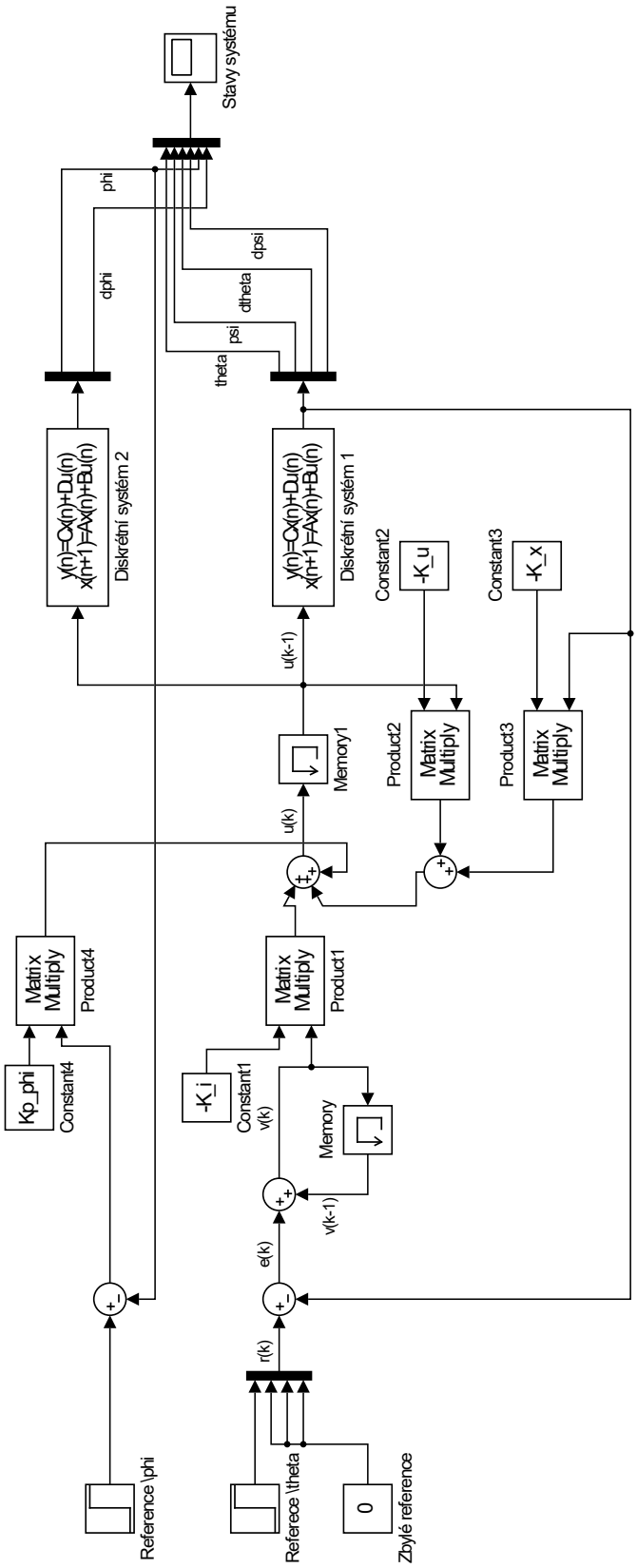
V MATLABu si můžeme usnadnit práci a využít pro převod příkaz `c2d`[1] aplikovaný na objekt `ss`[2]. Jeho použití demonstruje následují ukázka:

```
% A, B, C, D matice spojiteho systemu
% Ts vzorkovaci perioda
sys = ss(A,B,C,D); % vytvori objekt typu ss popisujici system
sysd = c2d(sys,Ts); % zdiskretizuje system

% pristoupeni k vypocitanym maticim
Ad = sysd.a
Bd = sysd.b
Cd = sysd.c
Dd = sysd.d
```

[1] `https://www.mathworks.com/help/control/ref/c2d.html`
[2] `https://www.mathworks.com/help/control/ref/ss.html`

2.4 Zapojení modelu v prostředí Simulink



Obrázek 2.3. Zapojení modelu v Simulinku.

Na obr. 2.3 můžeme vidět kompletní zapojení modelu v Simulinku. Vzhledem k použití stavové zpětné vazby u diskrétního systému 1, popsaného maticemi $\mathbf{A}_{1_d}$, $\mathbf{B}_{1_d}$, $\mathbf{C}_{1_d}$, $\mathbf{D}_{1_d}$, je potřeba nastavovat referenční hodnoty pro všechny stavy systému 1. To můžeme například pomocí bločků `Step`[1], kterým simulujeme skokovou změnu, nebo pomocí `Constant`[2]. Návrh zpětné vazby je popsán v sekci 2.5. Výsledky pak uvádí sekce 2.6.

Diskrétní systém 2, charakterizovaný maticemi $\mathbf{A}_{2_d}$, $\mathbf{B}_{2_d}$, $\mathbf{C}_{2_d}$, $\mathbf{D}_{2_d}$, je řízený P regulátorem, který je potřeba navrhovat až po implementaci zpětné vazby u systému 1. Samotný návrh nebyl nijak složitý a výsledek najdeme v sekci 2.6.

2.5 Lineární kvadratický regulátor s integrační složkou

Podrobnosti o lineárním kvadratickém regulátoru (zkráceně LQR) můžeme dohledat v práci [7], uvádíme pouze potřebné informace.

Základním cílem LQR je minimalizace tzv. nákladové funkce J:

$$J = \frac{1}{2}\vec{x}^T(T_N)\mathbf{M}\vec{x}(T_N) + \frac{1}{2}\int_0^{T_N}[\vec{x}^T(t)\mathbf{Q}\vec{x}(t) + \vec{u}^T(t)\mathbf{R}\vec{u}(t)]dt, \tag{22}$$

ve které $\mathbf{M}$, $\mathbf{Q}$, $\mathbf{R}$ jsou pozitivně definované váhové matice. Optimální řešení spočívání v nalezení předpisu pro $\vec{u}(t)$ tak, abychom minimalizovali hodnotu J.

Vzhledem k použití digitálního řízení je potřeba přejít k diskrétnímu času. Rovnici (22) diskretizujeme:

$$J = \frac{1}{2}\vec{x}^T(N)\mathbf{M}\vec{x}(N) + \frac{1}{2}\sum_{k=0}^{N-1}[\vec{x}^T(k)\mathbf{Q}\vec{x}(k) + \vec{u}^T(k)\mathbf{R}\vec{u}(k)] \tag{23}$$

V MATLABu je definována funkce `lqr`[3] pro lineární kvadratický regulátor a funkce `dlrq`[4] pro jeho diskrétní verzi. Než ovšem `dlrq` použijeme, je důležité se podívat, s jakou nákladovou funkcí MATLAB pracuje:

$$J(u) = \sum_{n=0}^{\infty}[\vec{x}^T(n)\mathbf{Q}\vec{x}(n) + \vec{u}^T(n)\mathbf{R}\vec{u}(n) + 2\vec{x}(n)^T\mathbf{N}\vec{u}(n)], \tag{24}$$

kde $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{N}$ jsou opět váhové matice.

Porovnáním rovnic (23) a (24) vidíme, že verze z MATLABu nepoužívá matici $\mathbf{M}$ a naopak přidává $\mathbf{N}$. Zvolíme proto kompromis, jelikož funkci `dlqr` lze volat i bez matice $\mathbf{N}$:

$$J(u) = \sum_{n=0}^{\infty}[\vec{x}^T(n)\mathbf{Q}\vec{x}(n) + \vec{u}^T(n)\mathbf{R}\vec{u}(n)], \tag{25}$$

LQR použijeme pro navrhnutí zpětné vazby diskrétního systému 1. Váhy matic $\mathbf{Q}$ a $\mathbf{R}$ se nastavují na základě zkušeností a zpravidla se využívá pouze vah na diagonále. Budeme–li mít m stavů a n vstupů, pak rozměr Q bude $m \times m$ a R $n \times n$. Naší prioritou je udržet segway kolmo k zemi, a proto úhlu naklonění ψ přidělíme větší váhu než úhlu natočení kol θ. Pokud bychom dělali regulátor bez integrační složky,

[1] `https://www.mathworks.com/help/simulink/slref/step.html`
[2] `https://www.mathworks.com/help/simulink/slref/constant.html`
[3] `https://www.mathworks.com/help/control/ref/lqr.html`
[4] `https://www.mathworks.com/help/control/ref/dlqr.html`

stačilo by nyní zavolat příkaz `[K,S,e] = dlqr(A1d,B1d,Q,R)` a do proměnné `K` bychom dostali hledané K pro stavovou zpětnou vazbu a do `e` vlastní čísla (póly) systému. Abychom regulátor rozšířili o integrální člen a vliv zpožděného buzení servomotorů, není již potřeba měnit nákladovou funkci J, ale musíme se zaměřit na stavový vektor $\vec{x}_1$ a matice $\mathbf{A}_1$ a $\mathbf{B}_1$.

Stavový vektor rozšíříme na:

$$\hat{\vec{x}}_1(k) = [\vec{v}^T(k), \vec{x}_1^T(k), \vec{x}_u^T(k)]^T, \tag{26}$$

kde $\vec{v}(k)$ reprezentuje integrované chyby od referencí a $\vec{x}_u(k)$ vliv zpožděného buzení servomotorů.

$$\vec{v}(k) = [v_\theta(k), v_\psi(k), v_{\dot{\theta}}(k), v_{\dot{\psi}}(k)]^T \tag{27}$$

$$\vec{x}_u(k) = [x_{u_{v_l}}, x_{u_{v_r}}]^T \tag{28}$$

Změnou stavového vektoru změníme i stavové matice. Podrobný popis úprav nalezneme v [7], kde je uveden předpis pro úpravu matic:

$$\hat{\mathbf{A}}_1 = \begin{bmatrix} \mathbf{I} & -\mathbf{C}_{1_d}\mathbf{A}_{1_d} & -\mathbf{C}_{1_d}\mathbf{B}_{1_d} \\ \mathbf{0} & \mathbf{A}_{1_d} & \mathbf{B}_{1_d} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{29}$$

$$\hat{\mathbf{B}}_1 = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{30}$$

Nyní můžeme použít příkaz `[K,S,e] = dlqr(A1s,B1s,Q,R)`, kde `A1s`, `B1s` jsou matice $\hat{\mathbf{A}}_1$, $\hat{\mathbf{B}}_1$. V proměnné `K` jsou hledané K_i, K_x, K_u. V MATLABu tedy:

```
A1s = [     eye(4), -C1d * A1d, -C1d * B1d;
          zeros(4),        A1d,        B1d;
        zeros(2,4), zeros(2,4),   zeros(2)];

B1s = [ zeros(4,2);
        zeros(4,2);
           eye(2)];

Q = diag(44,93377,1344,44366,262,7071,3567,44268,3935,3935);
R = diag(28695,28695);

[K,S,e] = dlqr(A1s,B1s,Q,R);

K_i = K(:,1:4);
K_x = K(:,5:8);
K_u = K(:,9:10);
```

2.6 Hodnoty použitého regulátoru

Váhové matice pro lineární kvadratický regulátor s integrační složkou:

$$\mathbf{Q} = diag(44, 93377, 1344, 44366, 262, 7071, 3567, 44268, 3935, 3935), \tag{31}$$

$$\mathbf{R} = diag(28695, 28695). \tag{32}$$

Parametry zpětné vazby systému 1:

$$\mathbf{K}_i = \begin{bmatrix} 0,0172130966647 & 0,0000070284628 \\ 0,0172130966647 & 0,0000070284628 \\ 0,0000026391018 & -0,0000014890196 \\ 0,0000026391018 & -0,0000014890196 \end{bmatrix}, \tag{33}$$

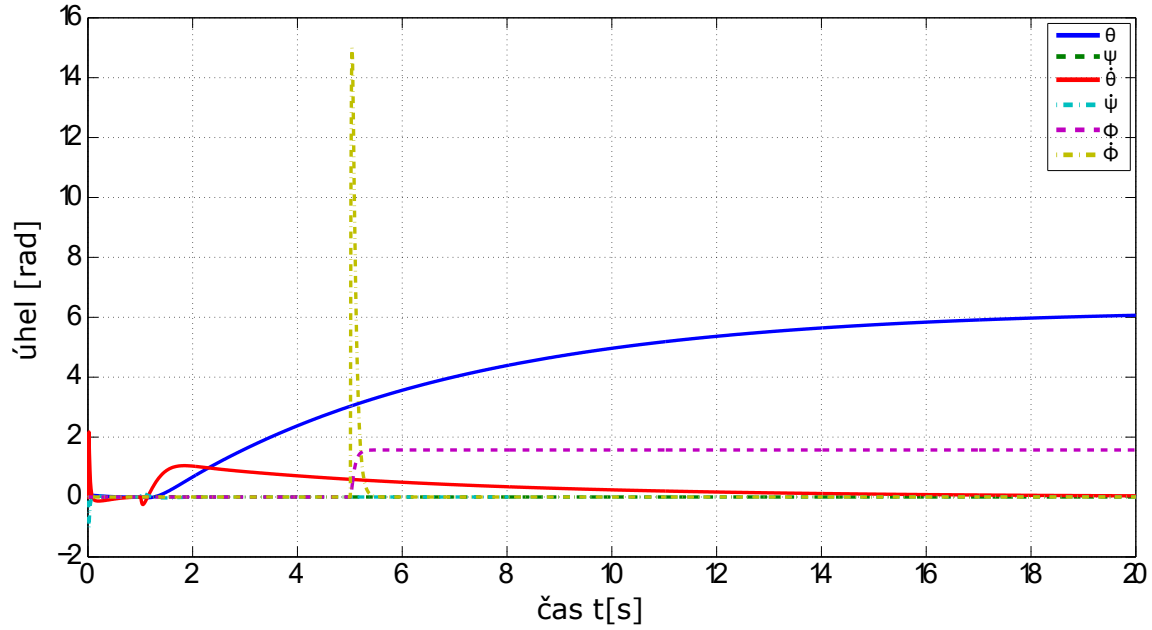
$$\mathbf{K}_x = \begin{bmatrix} -10,1565243105389 & -104,8150460330595 \\ -10,1565243105389 & -104,8150460330595 \\ -4,4973553022519 & -9,6674970403028 \\ -4,4973553022519 & -9,6674970403028 \end{bmatrix}, \tag{34}$$

$$\mathbf{K}_u = \begin{bmatrix} 0,2489142349134 & 0,2489142349134 \\ 0,2489142349134 & 0,2489142349134 \end{bmatrix}. \tag{35}$$

Parametry zpětné vazby systému 2:

$$\mathbf{Kp}_\phi = \begin{bmatrix} -10 & 10 \end{bmatrix}. \tag{36}$$

Návrh regulátorů ověříme simulací. Pomocí bločků `Step` v čase 1 s nastavíme referenční úhel θ na 2π, tedy požadujeme popojetí robotu o jednou otáčku kol. V čase 5 s pak změníme referenční úhel ϕ na $\frac{\pi}{2}$, neboli otočíme segwayem vlevo o $\frac{\pi}{2}$. Na obr. 2.4 vidíme výsledek simulace. Oba úhly dosáhnou své referenční hodnoty. θ nabíhá velmi dlouho. Příčinou je to, že reference pro $\dot{\theta}$ je nastavena na 0. Regulátor se ji snaží dodržet a nedovolí segwayi nabrat vyšší rychlost.



Obrázek 2.4. Simulace reakce robotu na skokovou změnu reference.

Kapitola 3
Práce s NXT

3.1 Jazyk NXC

Not eXactly C[1], neboli NXC, je programovací jazyk vyšší úrovně podobný jazyku C. Vytvořil jej John Hansen v roce 2006. Jazyk je postaven nad NBC překladačem. NBC, Next Byte Codes, je nízkoúrovňový programovací jazyk se syntaxí podobnou Assembleru. Pro kompilaci NXC programů se používá NBC překladač, který vygeneruje spustitelný strojový kód pro *NXT*.

K psaní programů můžeme využít vývojové prostředí Bricx Command Center.[2] Vzniklé programy mají příponu `.nxc`. Návod pro práci s jazykem najdeme na stránce [8].

Každý program musí obsahovat `task` s názvem `main`[3], volně přeloženo jako hlavní úloha. Tato úloha se vykoná při spuštění programu. Uživatel může definovat další úlohy, které mohou běžet v různém pořadí. Také lze spustit více úloh najednou.

3.2 Komunikace přes Bluetooth

NXT kostka umožňuje spojení pomocí Bluetooth s ostatními kostkami, počítačem, mobilním telefonem a jinými zařízeními. Podrobný popis komunikace je popsán v Appendix 2-LEGO MINDSTORMS NXT Direct commands.pdf dostupné na stránce [9][4]. Během komunikace se posílají Bluetooth zprávy, jejichž strukturu názorně vystihuje obr. 3.1.

Obrázek 3.1. Struktura Bluetooth paketu.

Zpráva se dělí na dvě části. První část obsahuje dva byty s údajem o velikosti posílaného příkazu a druhá část příkaz samotný. Je nutné poznamenat, že zmiňované první dva byty se do velikosti posílaného příkazu nepočítají, tedy velikost celé zprávy je o dva byty větší než velikost samotného příkazu. Byte třetí určuje o jaký typ příkazu se jedná, zpravidla se používá `0x00` pro přímý příkazový telegram vyžadující odpověď, anebo `0x80` pro přímý příkazový telegram bez nutné odpovědi. Čtvrtý byte specifikuje příkaz samotný. Další byty se liší podle zvoleného příkazu.

Příkladem posílané zprávy může být z mobilního telefonu vyslané bytové pole:

```
{0x06, 0x00, 0x80, 0x09, 0x00, 0x02, 0x01, 0x00}.
```

Čtvrtý byte říká, že se jedná o příkaz `MESSAGEWRITE`, kdy posíláme string do *NXT* kostky. Pátý byte specifikuje číslo schránky (0–9), šestý pak velikost posílaného stringu.

[1] `http://bricxcc.sourceforge.net/nbc/`
[2] `http://bricxcc.sourceforge.net/`
[3] `http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/task.html`
[4] Dříve tato dokumentace byla dostupná přímo od společnosti LEGO. S nástupem nové řady EV3 už bohužel není.

Od sedmého bytu je posílaný řetězec, který musí být ukončen bytem `0x00`. V tomto příkladu tedy posíláme jedničku ve formátu string. Na straně *NXT* pak přijmeme zprávu pomocí příkazu:

```
ReceiveRemoteString(cislo_schranky, true/false, prijaty_string);
```

První parametr určuje číslo schránky, druhý jestli se má po přijetí zpráva vymazat ze schránky a do třetího se uloží přijatý string.

3.2.1 Omezení toku dat

Řízení robotu, popsané v sekci 2.4, pracuje s referencemi. Ty bychom mohli spravovat v mobilní aplikaci a jejich hodnoty posílat pomocí Bluetooth do *NXT*, tedy odesílat šestici referencí. Druhá možnost je evidovat reference v *NXT* a posílat pouze informaci a dané akci. Kvůli omezení toku dat byla zvolena druhá varianta a kódování akcí uvádí tabulka 3.1.

Akce	Kód
Pohyb dopředu	1
Pohyb dozadu	2
Otočení doleva	3
Otočení doprava	4
Zastavení	5

Tabulka 3.1. Kódování řídících akcí.

3.3 Ovládání servomotorů

Pro přístup k jednotlivým servomotorům se v jazyce NXC používají konstanty ve tvaru `OUT_X`, kde za `X` doplníme buď písmeno `A`, `B`, `C`, či jejich různé kombinace.

Servomotory jsou vybaveny inkrementálním senzorem otáček s přesností $\pm 1^o$. Díky tomu lze poměrně přesně měřit úhel natočení. K této informaci přistoupíme pomocí funkce `MotorRotationCount(OUT_X)`. Pokud budeme chtít čítač vynulovat, stačí zavolat příkaz `ResetRotationCount(OUT_X)`. Pro zjištění aktuální rychlosti otáčení motoru použijeme `MotorActualSpeed(OUT_X)`.

K rozpohybování servomotorů slouží příkaz `OnFwd(OUT_X, vykon)`, kde `vykon` určuje rychlost otáčení a může nabývat hodnot z intervalu $< -100, 100 >$, kde záporný parametr otáčí motorem opačným směrem. V případě, že umístíme servomotor obráceně, a nebudeme chtít měnit znaménko, je možné použít funkci `OnRev(OUT_X, vykon)`, která motorem otáčí obráceně. Pokud bude `vykon` proměnnou, je důležité si ošetřit, aby se její hodnota opravdu nacházela v uvedeném intervalu, jinak může dojít k neočekávanému chování servomotoru.

3.4 Obsluha senzorů

Stavebnice *NXT* obsahuje řadu čidel vyráběných přímo firmou LEGO, například dotykový, světelný či ultrazvukový senzor. Také podporuje používání senzorů od jiných výrobců. Příkladem může být v tomto projektu použitý gyroskopický senzor od společnosti HiTechnic[1]. K připojení čidel slouží čtyři vstupní porty `IN_1` až `IN_4`.

[1] `http://www.hitechnic.com/`

V jazyce NXC před vyčítáním dat ze senzoru je třeba specifikovat jeho typ a mód. Typů je celkem 18 a nejvýznamnější z nich najdeme v tabulce 3.2. Seznam všech osmi módů uvádí tabulka 3.3, častěji se používají módy `SENSOR_MODE_RAW` a `SENSOR_MODE_BOOL`. Pokud budeme senzory používat podle návodu od výrobce, nemusíme přemýšlet nad ručním nastavením typů či módů a můžeme využít předem připravené funkce, více v 3.4.

Typ	Senzor
`SENSOR_TYPE_TOUCH`	NXT nebo RCX dotykový senzor
`SENSOR_TYPE_LOWSPEED`	NXT I2C digitální senzor
`SENSOR_TYPE_COLORFULL`	NXT 2.0 barevný senzor s přísvitem
`SENSOR_TYPE_COLORNONE`	NXT 2.0 barevný senzor bez přísvitu
`SENSOR_TYPE_SOUND_DB`	NXT zvukový senzor s měřením v dB

Tabulka 3.2. Typy senzorů.

Mód	Popis chování
`SENSOR_MODE_RAW`	Vrací hodnotu z intervalu $< 0, 1023 >$.
`SENSOR_MODE_BOOL`	Dvoustavová logika, hodnota nad 562 vrací 1, jinak 0.
`SENSOR_MODE_PULSE`	Počítá počet náběžných hran.
`SENSOR_MODE_EDGE`	Počítá počet náběžných i sestupných hran.
`SENSOR_MODE_PERCENT`	Vrací hodnotu měřené veličiny v procentech.
`SENSOR_MODE_CELSIUS`	Pro RCX teplotní senzor, vrací hodnotu ve ^{o}C.
`SENSOR_MODE_FAHRENHEIT`	Pro RCX teplotní senzor, vrací hodnotu ve ^{o}F.
`SENSOR_MODE_ROTATION`	Pro RCX rotační senzor, 16 tiků za otáčku.

Tabulka 3.3. Módy senzorů.

Příkaz	Popis chování
`SetSensorTouch(IN_X);`	Nastaví vstup `IN_X` jako dotykový spínač.
`SetSensorLight(IN_X,bool);`	Nastaví vstup `IN_X` jako světelný senzor, `true` zapne přísvit.
`SetSensorSound(IN_X,bool);`	Nastaví vstup `IN_X` jako zvukový senzor, `true` dB škálování, jinak dBA.
`SetSensorLowspeed(IN_X,bool);`	Nastaví vstup `IN_X` pro I2C komunikaci, `true` napájený senzor, jinak není nenapájený.[1]

Tabulka 3.4. Specifikace standardních senzorů.

3.5 Použité senzory

3.5.1 Gyroskopický senzor

Gyroskopický senzor není standardní součástí stavebnice *NXT*[1], vyrábí jej společnost HiTechnic. Tento senzor měří úhlovou rychlost v jedné ose a zvládá měřit rychlost do $\pm 360\ ^o s^{-1}$. Při nastavení senzoru do módu `SENSOR_MODE_RAW` vyčítáme jeho hodnotu ve $^o s^{-1}$, která je posunutá o určitý offset. Podle výrobce je offset kolem 620, u našeho senzoru byl experimentálně zjištěn kolem 579. Proto je důležité si offset před každým používáním změřit a doporučuje se jej i v průběhu používání upravovat.

Během měření offsetu jsme objevili interakci mezi buzením servomotorů a gyroskopickým senzorem. Pokud byly motory aktivně buzeny, i když se třeba neotáčely, vracel senzor jinou hodnotu, než když motory neběžely. Při stabilizaci budou servomotory potřeba a proto jsme offset získaly následovně:

```
void getGyroOffset()    // inicializace offsetu pro gyroskop
{
    OnFwd(OUT_BC, 0); // Abychom dostali valadni vysledek,
                      // ale robot nikam neodjel.

    float gSum;
    int  i, gMin, gMax, g;

    do {
        gSum = 0.0;
        gMin = 1000;
        gMax = -1000;
        for (i=0; i<100; i++) {   // Provedeme vice mereni.
            g = SensorRaw(GYRO_PORT);

            if (g > gMax)
                gMax = g;
            if (g < gMin)
                gMin = g;

                gSum += g;
            Wait(5);
       }
    } while ((gMax - gMin) > 1); // Pokud bylo mereni nepresne
      // Napriklad kdyz s~robotem nekdo pohnul

    gOffset = gSum / 100;   // Vysledek zprumerujeme.
}
```

Při měření úhlu natočení je nutné dodržet pravidla numerické integrace. Jako příklad uvedeme řešení aplikované v našem projektu. Jeden cyklus řídící smyčky probíhá 10 ms a v každém běhu se ptáme na hodnotu senzoru. Měříme tedy 100× za sekundu. Funkci `calculateGyro` počítající úhel naklonění opatříme klíčovým slovíčkem `inline`. Tím dáme překladači vědět, že v každém místě volání `calculateGyro` se má místo volání podprogramu funkce přímo zapsat tělo funkce. Díky tomu dojde ke zrychlení naší smyčky. Numericky integrál nahradíme následovně:

[1] Novější řada *LEGO Mindstorms EV3* již obsahuje gyroskopický senzor.

```
float gOffset;

inline void calculateGyro(float &psi, float &dpsi)
{
    long rawG;

    // Do promenne rawG vycteme hodnotu ze senzoru
    // filteredValue = zmerena hodnota nezatizena offsetem
    float filteredValue = (rawG = SensorRaw(GYRO_PORT)) - gOffset;

    // Aktualizace offsetu po kazdem mereni
    gOffset = gOffset * 0.9995 + rawG * 0.0005;

    // Merime rychlost zmeny uhlu v~m.s^-1, nutne prevest na rad.s^-1
    dpsi = filteredValue * PI/180;

    // Pravidlo numericke integrace
    // Merime 100x za sekundu, je tedy potreba jednotlivy prirustek
    // podelit 100.
    psi += dpsi / 100;
}
```

Obrázek 3.2. Gyroskopický senzor[10].

3.5.2 Dotykový senzor

Stavebnice *NXT* obsahuje dotykový senzor, který se zpravidla používá jako tlačítko se dvěma stavy „sepnutý“ a „rozepnutý“. Pokud bychom chtěli přesnější měření, pak můžeme ze senzoru vyčítat surová data v intervalu $< 0, 1023 >$, kde hodnota 1023 reprezentuje zcela volný senzor a hodnota kolem 50 zcela stlačený.

Obrázek 3.3. Dotykový senzor[11].

3.5.3 Ultrazvukový senzor

Ultrazvukový senzor je standardní součástí stavebnice *NXT* pracující na principu sonaru. Umožňuje měřit vzdálenosti v rozsahu od 0 cm do 255 cm s přesností ±3 cm. Senzor podporuje i měření v palcích. Dále záleží na objektu, od kterého vzdálenost měříme. Zpravidla pokřivené či kulaté předměty způsobují problémy a za jistých okolností je senzor nemusí vůbec zaznamenat. Také více senzorů pohromadě může spolu interferovat a znepřesnit výsledky.

Obrázek 3.4. Ultrazvukový senzor[12].

Kapitola 4
Mobilní aplikace

4.1 Volba platformy

Před tvorbou mobilní aplikace bylo důležité rozhodnout, na které platformě poběží. Hlavními kandidáty byl Android, iOS a Windows Phone. Výhodou platformy Android je pokrytí široké škály mobilních telefonů v různých cenových kategoriích. V případě nasazení aplikace pro výuku by bylo potřeba zakoupit několik telefonů. U naší aplikace se předpokládá, že bude mít nízké požadavky na výpočetní výkon zařízení, stačilo by tedy zakoupit levnější telefony. A právě proto jsme zvolili Android, kde máme možnost širší volby.

4.2 Rozmístění ovládacích prvků

Obrazovka s ovládacími prvky je rozdělena horizontálně do tří základních částí. V první nalezneme textové pole pro zadání MAC adresy *NXT* kostky a tlačítko připojit. Zmíněné textové pole je uděláno pomocí prvku `editText`[1] a tlačítko pomocí komponenty `button`.[2] Pro ukázku formátu zápisu MAC adresy je po spuštění aplikace uvedena adresa na testovací *NXT*.

Ve druhé části jsou dvě tlačítka. Levé s nápisem `STOP` pošle příkaz k zastavení segwaye a pravé s popiskem `ČTVEREC` jej uvede do módu jízdy ve čtvercové formaci.

Poslední část obsahuje čtyři tlačítka se směrovými šipkami. Jak už vzhled napovídá, tyto tlačítka slouží k ručnímu řízení robotu. Šipky byly upraveny tak, aby podporovaly dlouhodobé stisknutí. Díky tomu je řízení pohodlnější.

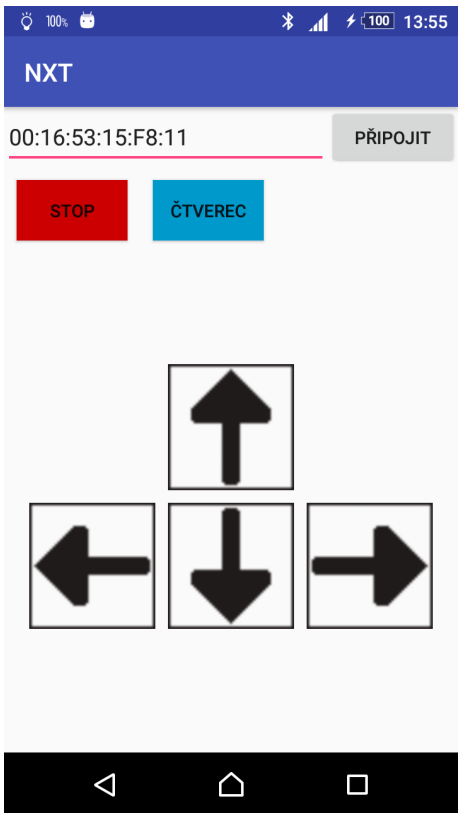
Orientace aplikace byla zafixována v pozici `portrait` (orientace na výšku) a její celkový vzhled můžeme vidět na obr. 4.1.

Ovládací prvky upozorní uživatele krátkou zprávou, pokud se pokusí provést nepodporovanou akci pomocí třídy `Toast`[3]. Taková situace může nastat třeba v případě, kdy nejsme k *NXT* připojeni a pokusíme se robotem pohnout. Důležité je ošetřit, aby se varovná hláška nevypisovala při každém zmáčknutí tlačítka, ale pouze v situaci, kdy k neplatnému stisku došlo a varování není zobrazeno. Jinak by mohlo dojít k situaci, že uživatel vícekrát klikne a upozornění se bude zobrazovat velmi dlouho i v případě ukončení aplikace.

Oprava této chyby byla provedena následujícím způsobem. Do privátní proměnné `mToastMustConFirst` si při vzniku hlavní aktivity vytvoříme `Toast` s varovnou hláškou. Při neplatném stisku se zeptáme:

[1] `https://developer.android.com/reference/android/widget/EditText.html`
[2] `https://developer.android.com/reference/android/widget/Button.html`
[3] `https://developer.android.com/guide/topics/ui/notifiers/toasts.html`

```
try{
    // pokud není Toast zobrazen
    if(!mToastMustConFirst.getView().isShown()) {
        mToastMustConFirst.show();  // zobraz Toast
    }
} catch (Exception exception) { // jinak vloz zapis do Logu
    Log.d(msg, "mToastMustConFirst is already shown");
}
```



Obrázek 4.1. Rozmístění ovládacích prvků.

4.3 Komunikace přes Bluetooth

Spojení navazujeme ze strany mobilního telefonu. K připojení potřebujeme znát MAC adresu *NXT* kostky. Uživatel ji zadá do přichystaného textového pole viz obr. 4.1. Pomocí třídy `BluetoothAdapter`[1] a zadané adresy získáme objekt typu `BluetoothDevice`[2] reprezentující *NXT* zařízení. Nyní stačí navázat spojení pomocí *RFCOMM* protokolu s doporučeným *UUID* „00001101-0000-1000-8000-00805F9B34FB“.

Jakmile naváženeme spojení, můžeme začít posílat příkazy do *NXT* a to ve formě bytových polí, jak bylo popsáno v sekci 3.2. Abychom *NXT* zbytečně nezatěžovali, používáme kódování akcí popsané v sekci 3.2.1. Pro zachování plynulost řízení byla zvolena 100 ms čekací doba mezi jednotlivými zprávami. Po ukončení řízení je potřeba uzavřít komunikační kanál.

[1] `https://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html`
[2] `https://developer.android.com/reference/android/bluetooth/BluetoothDevice.html`

Název	Popis
`isConnected`	Vrací `true` pokud existuje spojení s *NXT*, jinak `false`.
`connectToNXT`	Pokusí se připojit k *NXT*, vrací `true`/`false`.
`disconnect`	Pokusí se odpojit od *NXT*, vrací `true`/`false`.
`setNXT_addressAndCheck`	Zkontroluje zadanou MAC adresu. Uloží platnou adresu a vrátí `true`, jinak `false`.
`sendToNXT`	Pošle příkaz do *NXT* a vrátí `true`. V případě chyby vrátí `false`.
`getmBluetoothAdapter`	Vrátí používaný `BluetoothAdapter`.

Tabulka 4.1. Veřejné funkce třídy TalkToNXT.

4.3.1 Třída TalkToNXT

O komunikaci s *NXT* se stará třída `TalkToNXT`. Definuje veřejné funkce popsané v tabulce 4.1.

Kapitola 5
Realizace speciálních úloh

Základní požadavky na konstrukci robotu jsou následují. Těžiště musí být umístěno nad osou otáčení servomotorů při pohledu z boku. Také by mělo být stejně vzdálené od obou motorů při pohledu zepředu. Dále stavba musí unést váhu *NXT* kostky s akumulátorem, přibližně tedy 0,6 kg. Výhodou by byla robustnost a jednoduchost konstrukce.

Tvorbou inverzního kyvadla s využitím stavebnice *NXT* se zabývalo více projektů. Příkladem uveďme projekty NXTway-GS[5] a HTWay[13]. Použitá konstrukce byla inspirována projektem HTWay. Zásadní inovací je záměna infračerveného senzoru za dotykový, z důvodu implementace „softwarové brzdy“ popsané v následující části.

5.1 Inverzní kyvadlo ovládané pomocí mobilního telefonu

5.1.1 Část NXT

Řídící program využívá dvou po sobě jdoucích úloh. První s názvem `main` nastaví senzory a přiřadí jim vstupní porty, resetuje čítače otáček použitých motorů a změří offset gyroskopického senzoru funkcí `getGyroOffset` uvedenou v sekci 3.5.1.

Druhá úloha `balance` počká na připojení mobilního telefonu. Následně uživatele upozorní výpisem na displej a zvukovým projevem, aby segway postavil. Poté spustí řídící smyčku s periodou 10 ms. Během každého cyklu této smyčky se přeměří úhel naklonění robotu funkcí `calculateGyro`, popsanou v části 3.5.1, aktualizují se stavové veličiny a odchylky od referencí. Dále se určí akční zásah a servomotory jej začnou vykonávat. Při každém desátém cyklu, tedy každých 100 ms, se zkontroluje Bluetooth schránka a vyčtou se z ní příkazy k pohybu. Po zpracování příkazů se nastaví nové reference, kterých se robot snaží dosáhnout.

Při ladění programu byla implementována „softwarová brzda“ spustitelná dotykovým senzorem. Pomocí ní je možné segway zastavit, třeba v případě neočekávaného chování. Brzda nejdříve zastaví servomotory, poté vynuluje reference, čítače otáček servomotorů, stavové veličiny a odchylky od referencí. Pokud bychom chtěli segway vzít do ruky, můžeme brzdu využít k tomu, aby se koly zbytečně neotáčelo.

5.1.2 Část Android

Využívá mobilní aplikaci popsanou v sekci 4.

5.2 Inverzní kyvadlo sledující trajektorii vůdce

Řídící algoritmus spuštěný v *NXT* vychází z programu popsaného v části 5.1.1. Hlavní změnou je způsob nastavení referenčních hodnot. Ty nezadává uživatel pomocí mobilní aplikace, ale jsou generovány robotem na základě měření vzdálenosti ultrazvukovým

senzorem. Změření vzdálenosti trvá déle než jeden cyklus řídící smyčky, a proto je spuštěné paralelně se smyčkou. Při měření se vyslaný signál může špatně odrazit a nevrátit se k přijímači, čímž dojde k výrazné chybě. Abychom tento efekt redukovali, vzdálenost určujeme pomocí průměru posledních tří měření.

Pokud robot ztratí svého vůdce, natočí se doleva a pokusí se jej najít. Jestliže se mu to nepodaří, zkusí hledat vpravo. V případě, že by ho stále nenalezl, zopakuje hledání.

Kapitola 6
Závěr

Cílem této práce bylo navrhnout a implementovat dvě speciální úlohy s využitím stavebnice *LEGO Mindstorms NXT*[1]. Stavebnice byla programována v jazyce NXC. První z úloh bylo inverzní kyvadlo ovládané z mobilní aplikace. Zpětnovazební řízení bylo navrženo lineárním kvadratický regulátorem s integrační složkou. Regulátor s integrační složkou nám umožnil získat nulovou regulační odchylku. Klíčové bylo nalezení váhových matic **Q** a **R** a ověření regulátoru v programu Simulink.

Při pokusu o komunikaci s *NXT* přes Bluetooth jsme narazili na nedostatky v jazyce NXC, který neumožňuje zpracování standardních datových typů (např. `int`, `String`) odesílaných z Androidu. Řešením bylo nastudování dokumentace k Bluetooth komunikaci, což nám umožnilo odesílat příkazy ve formátu bytových polí, kterým *NXT* porozumí a dokáže je vykonat. Dále jsme provedli optimalizaci komunikace pomocí kódování příkazů, což značně zredukovalo množství posílaných dat.

Mobilní aplikace byla napsána pro platformu Android. Umožňuje připojení k robotu a jeho následné řízení pomocí směrových šipek. Další možností je autonomní jízda segwaye po čtvercové trajektorii. Vizí do budoucna je větší flexibilita zadání sledované trajektorie. Což by umožnilo řešit mnoho složitějších úloh, například jízdu předem definovaným bludištěm.

Během měření offsetu gyroskopického senzoru jsme narazili na nedokumentovanou interakci mezi senzorem a servomotory. Ta ovlivňovala návratovou hodnotu senzoru. Řešením bylo určovat offset s vybuzenými motory, abychom zahrnuli jejich vliv na měření offsetu.

Návod na sestavení konstrukce robotu nalezneme v elektronické příloze ve složce *Konstrukce*. Díky němu je možné si vytvořit podrobnou představu o stavbě segwaye. Také zjednodušuje znovupostavení robotu.

Celý projekt je dokumentován na webových stránkách, které nalezneme v elektronické příloze ve složce *Web*.

Literatura

[1] LEGO Mindstorms NXT. *LEGO shop* [online]. 2016 [cit. 2017-05-26]. Dostupné z: https://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547.

[2] Tomáš Bělík: Využití robota LEGO Mindstorms - návrh a realizace speciálních úloh, Bakalářská práce, ČVUT FEL v Praze, 2010.

[3] Dan Martinec: Využití robota LEGO Mindstorms při výuce předmětu A3B99RO Roboti, Bakalářská práce, ČVUT FEL v Praze, 2010.

[4] *Self-Erecting Inverted Pendulum* [online]. Automatic Control Laboratory, ETH Zürich, 2014 [cit. 2017-05-26]. Dostupné z: http://control.ee.ethz.ch/~ifa-fp/wiki/uploads/Main/ExperimentsSummer/IFA_2_2_manual_270214.pdf.

[5] *NXTway-GS (Self-Balancing Two-Wheeled Robot) Controller Design* [online]. 2009 [cit. 2017-05-26]. Dostupné z: https://www.mathworks.com/matlabcentral/-fileexchange/19147-nxtway-gs–self-balancing-two-wheeled-robot–controller-design .

[6] *Diskrétní modely spojitých systémů* [online]. 2017 [cit. 2017-05-26]. Dostupné z: http://www.polyx.com/_ari/slajdy/Bas-ARI-21-Sampled.pdf.

[7] *A Linear-Quadratic Regulator with Integral Action Applied to PWM DC-DC Converters* [online]. 2006 [cit. 2017-05-26]. Dostupné z: https://www.researchgate.net/ publication/224697009.

[8] BENEDETTELLI, Daniele. *Programming LEGO NXT Robots using NXC* [online]. 2007 [cit. 2017-05-26]. Dostupné z: http://bricxcc.sourceforge.net/nbc/nxcdoc/.

[9] *Index of /lego/mindstorms/nxt* [online]. [cit. 2017-05-26]. Dostupné z: - http://files.lechnology.com/lego/mindstorms/nxt/.

[10] *Gyro sensor* [online]. In: . 2014 [cit. 2017-05-26]. Dostupné z: http://sid.cps.unizar.es/ FACILITIES/others.html.

[11] *Touch Sensor* [online]. In: . Inc. 555 Taylor Road, Enfield, CT 06082, USA: LEGO Systems, 2016 [cit. 2017-05-26]. Dostupné z: https://shop.lego.com/en-US/Touch-Sensor-9843.

[12] *Ultrasonic Sensor* [online]. In: . Inc. 555 Taylor Road, Enfield, CT 06082, USA: LEGO Systems, 2016 [cit. 2017-05-26]. Dostupné z: https://shop.lego.com/en-US/Ultrasonic-Sensor-9846.

[13] *HTWay – A Segway type robot* [online]. 2010 [cit. 2017-05-26]. Dostupné z: http://www.hitechnic.com/blog/gyro-sensor/htway/.

Příloha A
Zadání bakalářské práce

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Michal Stračinský

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Využití robota LEGO Mindstorms - návrh speciálních úloh

Pokyny pro vypracování:

1. Navrhněte a realizujte speciální úlohy pro LEGO Mindstorms NXT nebo EV3 s návrhem řízení ve Vámi vybraném programovacím prostředí.
2. Jedna speciální úloha bude inverzní kyvadlo (segway) ovládané (změna směru dopředu, dozadu, otáčení doprava nebo doleva) pomocí mobilního telefonu (iPhone, Androidu atd.) s možností zadání trajektorie pohybu robota a druhá pak inverzní kyvadlo (segway) sledující trajektorii prvního inverzního kyvadla nebo inverzní kyvadlo (segway) zdolávající schody.
3. Zpracujte dokumentaci konstrukce robotů (vytvořené v LEGO designéru) a způsobu jejich programování.
4. Vytvořte speciální podrobné webové stránky k těmto úlohám.

Seznam odborné literatury:

[1] Tomáš Bělík: Využití robota LEGO Mindstorms - návrh a realizace speciálních úloh, Bakalářská práce, ČVUT FEL v Praze, 2010
[2] Dan Martinec: Využití robota LEGO Mindstorms při výuce předmětu A3B99RO Roboti, Bakalářská práce, ČVUT FEL v Praze, 2010

Vedoucí bakalářské práce: Ing. Martin Hlinovský, Ph.D.

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2017

Příloha B
Zkratky a symboly

DC	Stejnosměrný elektrický proud
dlqr	Funkce MATLABu pro lineární kvadratický regulátor v diskrétním čase
DUPLO	Typ stavebních kostek pro děti
EMF	Elektromotorické napětí
EV3	3. generace programovatelných LEGO kostek
HTWay	Typ segway robota s vlastní regulací pohybu
LEGO	Hraj si dobře–dánská firma na výrobu hracích kostek pro děti
LQR	Lineární kvadratický regulátor
MAC	Media Access Control, jednoznačný identifikátor síťového zařízení
MATLAB	Interaktivní programové prostředí a skriptovací jazyk
NBC	Next Byte Codes, nízkoúrovňový programovací jazyk se syntaxí podobnou Assembleru
NXC	Not eXactly C, programovací jazyk vyšší úrovně podobný jazyku C
NXT	Programovatelná kostka
NXTway-GS	Samostatně balancující robot na 2 kolech postavený pomocí stavebnice LEGO Mindstorms NXT
RCX	Robotic Command Exploder, verze programovatelné kostky vyráběné do roku 2009
RFCOMM	Jednoduchá množina příkazů pro přenosový protokol
USB	Universal Serial Bus, typ sériové sběrnice a konektoru
UUID	Univerzální jedinečný identifikátor pro identifikaci informačních systémů