České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídicí techniky

DIPLOMOVÁ PRÁCE

Platforma pro řízení podaktuovaných mechanických systémů

Praha, 2012 **Autor: Ondřej Rott**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

Poděkování

Děkuji především vedoucímu diplomové práce Milanovi Anderlemu za ochotu a příkladné vedení diplomové práce. Dále děkuji ostatním kolegům, především Jaroslavovi Žohovi a Kamilovi Dolinskému, za cenné rady. Také děkuji své rodině a přátelům za podporu.

Abstrakt

Cílem diplomové práce je návrh hardwaru a softwaru pro laboratorní model podaktuovaného kráčejícího robota za účelem experimentálního ověření algoritmů pro jeho řízení a odhadování stavů. Práce popisuje návrh a realizaci veškeré elektroniky pro model podaktuovaného kráčejícího robota. Elektronika je realizována distribuovaně a skládá se ze čtyř nezávislých desek plošných spojů pro řízení jednotlivých linků robota. Každá deska obsahuje procesor, senzory pro měření všech potřebných veličin a komponenty pro výkonové buzení připojených motorů. Komunikace mezi jednotlivými deskami probíhá po sběrnici CAN. Práce se dále zabývá návrhem a implementací algoritmů pro měření stavů jednotlivých linků robota a pro jejich řízení z Matlabu v reálném čase. Část řídicích algoritmů je naprogramována v jazyce C v procesorech umístěných na jednotlivých deskách robota. Druhá část algoritmů je implementována v Matlabu/Simulinku a běží na PC. Komunikace mezi PC a deskami robota probíhá rovněž po sběrnici CAN.

Abstract

The main aim of the master thesis is to design hardware and software for a laboratory model of an underactuated walking robot. The main purpose of this laboratory model is an experimental testing of algorithms designed for control and estimating states of underactuated walking robots. Thesis describes a design and a realisation of whole electronics used on the robot. Electronics is distributed amongst four print circuit boards (PCB). Each PCB controls one link of the robot and contains a microprocessor, sensors and a power-driving of a motor. Comunication among PCB is realised using CAN bus. Consequently, this work introduces a design and an implementation of algorithms for measuring and estimating all states of the robot and for control of all links of the robot from the Matlab in a real-time. One part of algorithms is written in C language and programmed in microprocessors which are located at all PCB. The second part of algorithms is implemented in the Matlab/Simulink and runs on PC. Comunnication between PC and PCB is also realised using CAN bus.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Ondřej Rott**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Platforma pro řízení podaktuovaných mechanických systémů**

Pokyny pro vypracování:

Cílem práce je navrhnout hardware a řídicí software pro laboratorní model podaktuovaného kráčejícího robota za účelem ověření algoritmů pro jeho řízení a odhadování stavů.

Hlavní úkoly práce jsou:
1) upevnění dodané desky s elektronikou na model Acrobota a připojení senzorů a akčních členů (laserové čidlo pro měření vzdálenosti, stejnosměrný motor s převodovkou a inkrementálním snímačem),
2) návrh a implementace řídicího algoritmu pro řízení jednoho linku a měření jeho stavů, real-time řízení z Matlabu,
3) návrh a výroba původní/nové desky s elektronikou pro řízení zbývajících linků, komunikace jednotlivých desek po sběrnici CAN,
4) identifikace modelu Acrobota,
5) návrh a implementace řídicího algoritmu pro experimentální ověření a porovnání navržených pozorovatelů pro model kráčejícího robota.
Důkladně dokumentujte zdrojový kód např. s využitím volně dostupného prostředí DoxyGen.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Milan Anderle

Platnost zadání: do konce letního semestru 2011/2012

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 28. 4. 2011

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Cílem této diplomové práce je navrhnout hardware a řídicí software pro laboratorní model podaktuovaného kráčejícího robota za účelem experimentálního ověření dodaných algoritmů pro jeho řízení a odhadování stavů.

Hlavními úkoly práce je navrhnout a vyrobit desky plošných spojů a veškerou elektroniku pro řízení kráčejícího robota. Navržené a vyrobené desky plošných spojů potom společně se senzory a akčními členy připojit na mechanický model robota. Mechanický model jsem dostal k dispozici od vedoucího práce. Dalším cílem práce je implementovat algoritmy pro měření stavů jednotlivých linků robota a pro jejich řízení z Matlabu v reálném čase. Řízení robota přímo z Matlabu bylo zadáno vedoucím práce z důvodu možnosti pozdějšího využití při výuce a z důvodu snadné implementace regulátorů. Posledním cílem práce je ověření algoritmů navržených pro řízení podaktuovaných kráčejících robotů na fyzickém modelu robota. Tyto algoritmy byly vedoucím práce navrženy a otestovány pouze na simulacích v Matlabu. Mým úkolem je tyto algoritmy otestovat na fyzickém modelu kráčejícího robota.

1.1 Popis jednotlivých kapitol

V kapitole 2 je uveden matematický popis modelu robota. Robot je modelován jako tzv. acrobot, což je jeden z nejjednodušších případů podaktuovaných mechanických systémů.

Popis fyzického modelu robota je v kapitole 3. V této kapitole jsem také popsal identifikaci parametrů reálného robota. Znalost těchto parametrů je nutná pro sestavení správného matematického modelu, který je nutný pro správně fungující řízení chůze ro-

bota.

Veškerou použitou elektroniku robota jsem popsal v kapitole 4. V této kapitole jsem popsal desky plošných spojů, které jsem navrhl pro čtení dat ze senzorů robota a pro ovládání jeho kloubů. Také je zde uveden podrobný popis použitých součástek.

Řídicí software, který jsem navrhl a implementoval, je popsán v kapitolách 5, 6 a 7. V kapitole 5 je uveden popis programů v jazyce C, které jsou naprogramovány v procesorech robota. V kapitole 6 je potom popsán způsob řízení robota v reálném čase přímo z Matlabu. Také je zde popsána komunikace mezi robotem a PC po sběrnici CAN. Implementace algoritmů pro řízení robota a jejich experimentální ověření je popsáno v kapitole 7.

V poslední kapitole 8 je uvedeno shrnutí výsledků mé práce. Je zde uvedeno srovnání a zhodnocení testovaných algoritmů pro odhadování stavů robota a pro jeho řízení. Také jsem zde uvedl nápady pro budoucí zlepšení činnosti robota, které jsem nestihl zrealizovat.

Kapitola 2

Matematický popis modelu

Tato kapitola stručně popisuje matematický model nejjednoduššího typu kráčejícího robota, tzv. acrobota, který reprezentuje dodaný mechanický model kráčejícího robota. Dále stručně popisuje algoritmy pro řízení chůze a odhadování stavů modelu acrobota, které budou ověřeny na reálném modelu podaktuovaného kráčejícího robota. Přesné odvození včetně důkazů stability je uvedeno v citovaných článcích.

2.1 Model acrobota

Acrobot je jeden z nejjednodušších podaktuovaných mechanických systémů. Má dva stupně volnosti a pouze jeden akční člen v kloubu, který spojuje obě nohy robota, viz obr. 2.1. Úhel v bodě, kde se acrobot dotýká podložky, není možné ovládat. Z tohoto důvodu se acrobot řadí mezi podaktuované mechanické systémy.

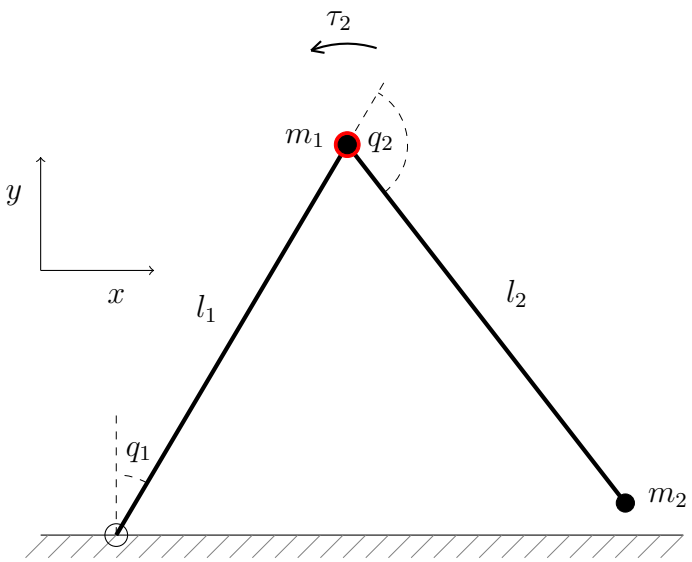
Při popisu chování tohoto mechanického modelu jsou použity Eulerovy-Lagrangeovy rovnice, které vedou na známou dynamickou rovnici mechanického systému ve tvaru

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u, \tag{2.1}$$

kde $D(q)$ je matice setrvačnosti, matice $C(q,\dot{q})$ obsahuje Coriolisovy a odstředivé síly, matice $G(q)$ obsahuje gravitační členy a u je vektor vnějších sil.

V případě acrobota matice $D(q)$, $C(q,\dot{q})$ a $G(q)$ mají tvar

$$D(q) = \begin{bmatrix} \theta_1 + \theta_2 + 2\theta_3 \cos q_2 & \theta_2 + \theta_3 \cos q_2 \\ \theta_2 + \theta_3 \cos q_2 & \theta_2 \end{bmatrix}, \tag{2.2}$$



Obrázek 2.1: Model acrobota.

$$C(q,\dot{q}) = \begin{bmatrix} -\theta_3 \sin q_2 \dot{q}_2 & -(\dot{q}_2 + \dot{q}_1)\theta_3 \sin q_2 \\ \theta_3 \sin q_2 \dot{q}_1 & 0 \end{bmatrix}, \tag{2.3}$$

$$G(q) = \begin{bmatrix} -\theta_4 g \sin q_1 - \theta_5 g \sin(q_1 + q_2) \\ -\theta_5 g \sin(q_1 + q_2) \end{bmatrix}, \tag{2.4}$$

kde 2-dimenzionální konfigurační vektor (q_1, q_2) obsahuje úhly znázorněné na obr. 2.1. Parametry θ_1, θ_2, θ_3, θ_4, θ_5 mají tvar

$$\begin{gathered} \theta_1 = (m_1 + m_2)l_1^2 + I_1,\ \theta_2 = m_2 l_2^2 + I_2, \\ \theta_3 = m_2 l_1 l_2,\ \theta_4 = (m_1 + m_2)l_1,\ \theta_5 = m_2 l_2. \end{gathered} \tag{2.5}$$

Podrobnější odvození lze najít např. v (Fantoni, I. and Lozano, R., 2002).

2.2 Metoda exaktní linearizace

Metodou exaktní linearizace je původní 4-dimenzionální model acrobota převeden do exaktně linearizovaného tvaru (2.6) v nových souřadnicích ξ, který se skládá ze 3-dimenzionálního lineárního systému a 1-dimenzionální nulové dynamiky.

$$\begin{aligned}\dot{\xi}_1 &= d_{11}(q_2)^{-1}\xi_2, \;\; \dot{\xi}_2 = \xi_3, \;\; \dot{\xi}_3 = \xi_4, \\ \dot{\xi}_4 &= \alpha(q,\dot{q})\tau_2 + \beta(q,\dot{q}) = w. \end{aligned} \tag{2.6}$$

Pro potřebu generování referenční trajektorie v linearizovaných souřadnicích ξ je zaveden referenční systém, který je shodný se skutečným systémem. Skutečný model acrobota pak sleduje referenční trajektorii, kterou generuje tento referenční systém

$$\dot{\xi}_1^{ref} = d_{11}^{-1}(q_2^{ref})\xi_2^{ref}, \;\; \dot{\xi}_2^{ref} = \xi_3^{ref}, \;\; \dot{\xi}_3^{ref} = \xi_4^{ref}, \;\; \dot{\xi}_4^{ref} = w^{ref} = 0. \tag{2.7}$$

Odečtením obou systémů a jejich úpravou lze dostat dynamiku odchylky e sledované referenční trajektorie ve tvaru

$$\begin{aligned}\dot{e}_1 &= \mu_2(t)e_2 + \mu_1(t)e_1 + \mu_3(t)e_3 + o(e), \\ \dot{e}_2 &= e_3, \;\; \dot{e}_3 = e_4, \\ \dot{e}_4 &= w - w^{ref} = K_1e_1 + K_2e_2 + K_3e_3 + K_4e_4, \end{aligned} \tag{2.8}$$

kde $\mu_1(t), \mu_2(t), \mu_3(t)$ jsou definovány v (Čelikovský, S. and Zikmund, J. and Moog, C., 2008), kde lze také najít exaktní postup včetně důkazu stability.

Návrh zesílení $K_{1,2,3,4}$ je proveden metodou LMI, viz. (Anderle, M. and Čelikovský, S. and Henrion, D. and Zikmund, J., 2009).

2.3 "High gain" pozorovatel

Prvním typem pozorovatele použitým pro odhad stavů acrobota je tzv. "high gain" observer. Tento pozorovatel je založen na měření stavů q_2 a $\dot{q}_1$. Úhel q_2 je snadné měřit pomocí rotačního inkrementálního senzoru. Úhlovou rychlost $\dot{q}_1$ je možné měřit pomocí digitálního gyroskopu. Na reálném modelu acrobota je pro měření úhlové rychlosti $\dot{q}_1$ použit gyroskop Adis16260 společnosti *ANALOG DEVICES*.

Použitím podobné změny souřadnic, jako v případě návrhu regulátoru pomocí exaktní linearizace, lze získat nový systém souřadnic

$$\dot{\eta}_1 = d_{11}^{-1}(q_2)\,\eta_2 - \dot{q}_1, \quad \dot{\eta}_2 = \eta_3, \quad \dot{\eta}_3 = \eta_4, \quad \dot{\eta}_4 = \beta(\eta) + \alpha(\eta_1,\eta_3)\,\tau_2. \tag{2.9}$$

Pozorovatel pak má tvar

$$\begin{aligned}
\dot{\hat{\eta}}_1 &= -L_1(\eta_1 - \hat{\eta}_1) + d_{11}^{-1}(q_2)\,\hat{\eta}_2 - \dot{q}_1,\\
\dot{\hat{\eta}}_2 &= -L_2(\eta_1 - \hat{\eta}_1) + \hat{\eta}_3,\\
\dot{\hat{\eta}}_3 &= -L_3(\eta_1 - \hat{\eta}_1) + \hat{\eta}_4,\\
\dot{\hat{\eta}}_4 &= -L_4(\eta_1 - \hat{\eta}_1) + \beta(\hat{\eta}) + \alpha(\eta_1, \hat{\eta}_3)\,\tau_2.
\end{aligned} \tag{2.10}$$

Po zavedení chyby odhadu $e = \hat{\eta} - \eta$ má pozorovatel tvar

$$\begin{aligned}
\dot{e}_1 &= L_1 e_1 + d_{11}^{-1}(q_2)\,e_2,\\
\dot{e}_2 &= L_2 e_1 + e_3,\\
\dot{e}_3 &= L_3 e_1 + e_4,\\
\dot{e}_4 &= L_4 e_1 + \beta(\hat{\eta}) - \beta(\eta) + (\alpha(\eta_1, \hat{\eta}_3) - \alpha(\eta_1, \eta_3))\,\tau_2.
\end{aligned} \tag{2.11}$$

Zesílení $L_{1,2,3,4}$ lze navrhnout pomocí tzv. "high-gain" postupu, pokud vezmeme jakékoliv $\widetilde{L}_{1,2,3,4}$ takové, aby byla matice

$$\begin{bmatrix} \widetilde{L}_1 & 1 & 0 & 0 \\ \widetilde{L}_2 & 0 & 1 & 0 \\ \widetilde{L}_3 & 0 & 0 & 1 \\ \widetilde{L}_4 & 0 & 0 & 0 \end{bmatrix} \tag{2.12}$$

Hurwitzovská. Platí

$$L_1 = \Theta\widetilde{L}_1, \quad L_2 = \Theta^2\widetilde{L}_2, \quad L_3 = \Theta^3\widetilde{L}_3, \quad L_4 = \Theta^4\widetilde{L}_4, \tag{2.13}$$

více viz (Anderle, M. and Čelikovský, S., 2010*a*).

2.4 Redukovaný pozorovatel

Redukovaný pozorovatel je druhým typem použitého pozorovatele stavů. Návrh redukovaného pozorovatele úhlové rychlosti $\dot{q}_1, \dot{q}_2$ je založen na schopnosti měřit úhel v bodech q_1 a q_2. Znalost proměnných q_1 a q_2 odpovídá znalosti proměnných ξ_1 a ξ_3 v (2.6). Definice proměnných ξ_1 a ξ_3 je v (Čelikovský, S. and Zikmund, J. and Moog, C., 2008). Je tedy nutné odhadovat pouze stavy ξ_2 a ξ_4.

Úhel q_2 je možné měřit pomocí rotačního senzoru. V případě, že bychom měli robota s chodidlem, mohli bychom podobně měřit i úhel q_1. V našem případě acrobot žádné chodidlo nemá, proto tento úhel musíme měřit nepřímo.

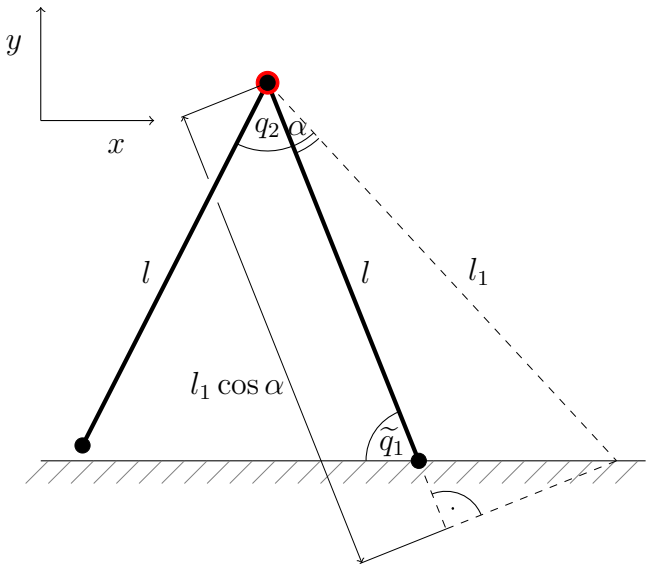
U našeho laboratorního modelu jsme se rozhodli na tělo robota připevnit zařízení pro optické měření vzdálenosti. Úhel mezi směrem laserového paprsku a stojící nohou robota je roven nějakému vhodně zvolenému úhlu α.

Abychom spočítali úhel q_1 definovaný v obr.2.1, potřebujeme znát úhel $\widetilde{q}_1$ definovaný v obr.2.2, 2.3, protože platí $q_1 = \widetilde{q}_1 - \pi/2$. Vzdálenost l_1 mezi laserovým senzorem a zemí se měří laserovým senzorem ODSL8 společnosti *Leuze electronic*.

Z trigonometrických vztahů lze odvodit, že neznámý úhel $\widetilde{q}_1$ je následující funkcí $\widetilde{q}_1(l_1)$ vzdálenosti l_1:

$$\widetilde{q}_1(l_1) = \begin{cases} \arcsin \frac{l_1 \sin\alpha}{\sqrt{l^2+l_1^2-2ll_1\cos\alpha}}\,, & l_1 \geq \frac{l}{\cos\alpha}\,, \\ \pi - \arcsin \frac{l_1 \sin\alpha}{\sqrt{l^2+l_1^2-2ll_1\cos\alpha}}\,, & l_1 \leq \frac{l}{\cos\alpha}\,. \end{cases} \tag{2.14}$$

Tyto dva případy jsou odděleně zobrazeny na obrázcích 2.2, 2.3.



Obrázek 2.2: Měření úhlu $\widetilde{q}_1$ pomocí laserového paprsku pro $l_1 \geq \frac{l}{\cos\alpha}$.

Na základě známých stavů $\xi_{1,3}$ lze odhadovat stavy $\xi_{2,4}$ v rovnici (2.6).

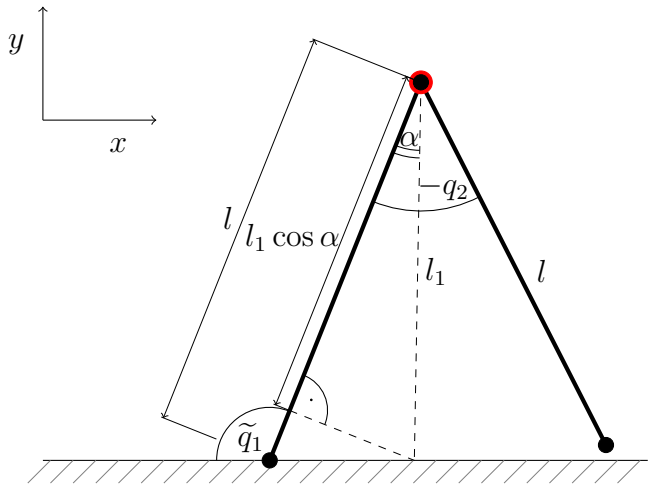
Odhad stavu ξ_2

$$\widehat{\xi}_2 = \widetilde{\xi}_2 + k_2\xi_1, \tag{2.15}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\widetilde{\xi}_2\right) = \xi_3 - k_2^2\xi_1 d_{11}^{-1}(q_2) - k_2\widetilde{\xi}_2 d_{11}^{-1}(q_2). \tag{2.16}$$

S chybou odhadu e_2

$$e_2 = \widetilde{\xi}_2 - \xi_2 + k_2\xi_1. \tag{2.17}$$



Obrázek 2.3: Měření úhlu $\widetilde{q}_1$ pomocí laserového paprsku pro $l_1 \leq \frac{l}{\cos\alpha}$.

Pro $k_2 > 0$ a $t \to \infty$, jde $e_2 \to 0$. Potom je $\widetilde{\xi}_2 + k_2\xi_1$ je skutečně odhadem ξ_2.

Pro odhad stavu ξ_4 je postup podobný. Odhad stavu ξ_4

$$\widehat{\xi}_4 = \widetilde{\xi}_4 + k_4\xi_3, \tag{2.18}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\widetilde{\xi}_4\right) = \alpha(q)\tau_2 + \beta(q, \dot{\widehat{q}}) - k_4^2\xi_3 - k_4\widetilde{\xi}_4, \tag{2.19}$$

kde q je vektor měřených úhlů a $\dot{\widehat{q}}$ je vektor odhadovaných úhlových rychlostí. Chyba odhadu e_4 je definována

$$e_4 = \widetilde{\xi}_4 - \xi_4 + k_4\xi_3. \tag{2.20}$$

Pro dostatečně velké $k_4 > 0$ jde $e_4 \to 0$ pro $t \to \infty$. Potom je $\widetilde{\xi}_4 + k_4\xi_3$ skutečným odhadem stavu ξ_4. Podrobný popis odvození viz (Anderle, M. and Čelikovský, S., 2010*c*).

Výhoda výše popsaného redukovaného pozorovatele je především v tom, že je možné použít malé zesílení u proměnných $k_{2,4}$. Tím se předejde problémům způsobeným při použití "high gain" pozorovatele, který velmi zesiluje šum měření.

2.5 Referenční trajektorie

Pro dosažení chůze robota je třeba generovat referenční trajektorii. Chůze robota je dosaženo tak, že se navržený regulátor snaží sledovat referenční trajektorii generovanou referenčním modelem 2.7. Pro tuto trajektorii platí $w^r \equiv 0$. To znamená, že systém

vyjádřený v exaktně linearizovaných souřadnicích má nulový vstup. Ve skutečných souřadnicích točivý moment není nulový, je roven $\tau_2 = -w\alpha(q, \dot{q})/\beta(q, \dot{q})$

Vygenerovaná referenční trajektorie s $w^r \equiv 0$ zajistí pohyb těžiště robota vpřed s konstantní rychlostí.

Shrnutí kapitoly

V této kapitole byly shrnuty všechny podstatné výsledky z (Čelikovský, S. and Zikmund, J. and Moog, C., 2008), (Anderle, M. and Čelikovský, S. and Henrion, D. and Zikmund, J., 2009), (Anderle, M. and Čelikovský, S., 2010*c*), (Anderle, M. and Čelikovský, S., 2010*b*). Tyto teoretické výsledky byly zatím ověřeny jen na simulační úrovni. V dalších kapitolách budou tyto výsledky ověřeny na skutečném modelu podaktuovaného kráčejícího robota.

Kapitola 3

Popis mechanického modelu

V této kapitole je popsán reálný model kráčejícího robota. Tento model bez elektroniky jsem dostal k dispozici od vedoucího práce. Reálný model se od matematického modelu, který je popsaný v kapitole 2, liší ve větším počtu kloubů (stupňů volnosti). Fyzický model robota má celkem 4 klouby, viz obr. 3.1. Dva klouby představují kyčle robota a dva klouby kolena. Matematický model, který je použit pro řízení chůze, popisuje robota jako tzv. 2-link, tj. jako mechanický systém skládající se ze dvou pevných tyčí spojených jedním kloubem s jedním akčním členem. Tento model tedy popisuje robota tak, jako by měl trvale napnutá kolena. Dva horní klouby, které tvoří kyčle robota, jsou v modelu uvažovány jako jeden kloub. Díky těmto zjednodušením je robot v kapitole 2 modelován tak, jako by měl pouze 2 stupně volnosti. Během chůze robota je třeba, aby byla kolena pokud možno pořád propnutá a tím je potom zajištěna co největší podobnost s matematickým modelem použitým pro řízení. Jedinou výjimkou je noha, která se právě pohybuje. Tu je třeba pokrčit v momentě, kdy míjí stojící nohu robota, aby pohybující se noha nezavadila o zem. Toto pokrčení nohy, které je nutné pro chůzi robota, je v matematickém modelu systému zanedbáno. Pro lepší regulaci jsem toto pokrčení později zohlednil jako skokovou změnu momentu setrvačnosti pohybující se nohy robota. V budoucnu je možno využít většího počtu kloubů robota a pro řízení sestavit model, který bude popisovat robota jako systém s více stupni volnosti.

Obrázek 3.1: Fotografie robota bez elektroniky

3.1 Identifikace parametrů robota

Pro vytvoření správného matematického modelu a tím pádem i pro správně fungující řízení robota je potřeba znát co nejpřesněji všechny jeho parametry. V této podkapitole je popsána identifikace potřebných parametrů. Některé z nich lze získat měřením. Například délky nohou robota nebo hmotnosti jednotlivých části lze snadno změřit a zvážit. Parametry použitých motorů a senzorů lze vyčíst v katalogu výrobce. Některé další parametry, např. momenty setrvačnosti nebo tření, lze získat pouze pomocí experimentů. Rozměry a hmotnosti jednotlivých částí robota jsem získal měřením a vážením.

3.1.1 Parametry získané měřením

Zde je uveden seznam parametrů získaných pomocí měření a vážení.

Níže jsou parametry získané vážením:

1. Noha robota (2 motory, 2 klouby, 2 tyče, chodidlo, šrouby) [694g]

   (a) Spodní část (motor, kloub, tyč, chodidlo) [347g]

(b) Horní část (motor, kloub, tyč) [348g]

i. kloub [147g]
ii. motor [129g]
iii. tyč [56g]
iv. chodidlo [15g]

2. Trup (tyč, tyčky pro upnutí skříňky) [185g]

(a) tyč [82g]
(b) tyčky k skříňce [77g]

Níže jsou parametry získané měřením:
Celá noha robota (2 tyče, 2 klouby, chodidlo) = 0.535 [m]

1. tyč nohy = 0.250 [m]
2. rozestup mezi horní a dolní částí nohy (kloub) = 0.017 [m]
3. délka přečnívajícího chodidla = 0.018 [m]

3.1.2 Parametry použitých motorů

Zde jsou uvedeny parametry použitých motorů Maxon A-max 22. Tyto motory jsem dostal k dispozici společně s mechanickým modelem robota. Motory jsou vybaveny planetovou převodovkou a inkrementálním senzorem. Seznam všech důležitých parametrů motoru je uveden v tabulce 3.2. Seznam parametrů převodovky je uveden v tabulce 3.1.

Převod	1 : 270	
Max.moment	1800	mN m
Účinnost	49	%
Moment setrvačnosti	0.4	g cm^2

Tabulka 3.1: Parametry použité převodovky Maxon GP 22 C (výpis z datasheetu)

Nominální napětí	12	V
Otáčky bez zátěže	9160.0	min^{-1}
Proud bez zátěže	40.5	mA
Max. moment	6.45	mN m
Max. proud	0.571	A
Počáteční proud	1.61	A
Odpor vinutí	7.45	Ω
Indukčnost vinutí	445.0	μH
Momentová konstanta	12.1	$\mathrm{mN\,mA}^{-1}$
Rychlostní konstanta	790.0	$\mathrm{min}^{-1}\mathrm{V}^{-1}$
Mechanická konstanta	21.4	m s
Setrvačnost rotoru	4.19	$\mathrm{g\,cm}^2$
Počet pólových dvojic	1.0	

Tabulka 3.2: Parametry použitého motoru Maxon A-max 22 (výpis z datasheetu)

3.1.3 Identifikace momentu setrvačnosti

Zde je popsána identifikace momentu setrvačnosti nohy robota. Identifikaci jsem prováděl v Matlabu jako odhad parametrů nelineárního grey box modelu, kde modelovaným systémem byl elektromotor spojený s mechanickým kyvadlem (3.3), (3.4). Identifikaci jsem prováděl na datech, které jsem experimentálně změřil na modelu robota a tato data jsem dále zpracoval v Matlabu. Samotnou identifikaci jsem prováděl v Matlabu pomocí funkce `idnlgrey` ze *System Identification Toolboxu*. Zdrojové kódy z Matlabu použité pro identifikaci jsou na přiloženém CD.

3.1.3.1 Identifikační experiment

Pro získání dat důležitých pro identifikaci momentu setrvačnosti jsem nejprve potřeboval provést vhodný identifikační experiment. Identifikaci jsem prováděl pro levou nohu robota. Tuto nohu jsem zafixoval v napnuté poloze. Také jsem zafixoval zbytek modelu robota ve stojanu, aby se nemohl samovolně pohybovat. Pro změření dat pro identifikaci jsem se rozhodl provést experiment, kdy bude motor v levé kyčli robota pokud možno náhodně hýbat levou nohou robota dopředu a dozadu. Po konzultacích s kolegy jsem se rozhodl motor, který ovládá pohyb nohy v levé kyčli robota, budit pomocí `prbs`(pseudorandom

binary) signálu. Jedná se o obdélníkový signál s náhodnými okamžiky změny polarity. Tímto signálem bylo napětí přivedené na motor s hodnotou ±6V [1], viz obr. 3.4. Okamžiky změny polarity jsem omezil tak, aby se polarita napětí přivedeného na motor měnila s frekvencí maximálně 2Hz. Signál jsem do motoru pouštěl přímo z Matlabu pomocí rozhraní popsaného v kapitole 6. Změřená data jsem přijímal do Matlabu a ukládal do souboru. Při experimentu jsem měřil napětí na motoru, proud motorem, úhel v kyčli robota a úhlovou rychlost. Na obr. 3.2, 3.3, 3.4 a 3.5 jsou průběhy měřených veličin během experimentu.

3.1.3.2 Grey box identifikace

Identifikaci momentu setrvačnosti jsem prováděl v Matlabu na datech, která jsem experimentálně změřil na modelu robota. Moment setrvačnosti jsem identifikoval pomocí funkce `idnlgrey` jako odhad parametru nelineárního grey box modelu.

Pro použití tohoto typu odhadu je třeba nejprve sestavit model odhadovaného systému ve formě soustavy nelineárních diferenciálních rovnic prvního řádu (3.1),(3.2).

$$\frac{dx(t)}{dt} = F(t, x(t), u(t), p_1, p, \ldots p_N), \tag{3.1}$$

$$y = H(t, x(t), u(t), p_1, p_2, \ldots p_N) + e(t). \tag{3.2}$$

Po sestavení rovnic modelovaného systému je třeba v Matlabu vytvořit funkci, která reprezentuje tyto rovnice a která musí být ve tvaru `[dx,y]=sys(t,x,u,p1,...,pN)`. Výstupem vytvořené funkce jsou výstupy odhadovaného systému a derivace vnitřních stavů. V případě identifikace momentu setrvačnosti nohy robota jsou to níže uvedené rovnice, kde (3.3) je rovnice elektromotoru a (3.4) je rovnice kyvadla. Níže je ukázka implementace této funkce v matlabu.

$$\frac{di(t)}{dt} = U - Ri - \frac{U_{ind}}{L}, \tag{3.3}$$

$$\ddot{\varphi} = \frac{\tau - mgl_c sin\left(\frac{\varphi}{2}\right)}{I + ml_c{}^2}. \tag{3.4}$$

[1] Pro experiment jsem zvolil tuto hodnotu napětí, která je poloviční oproti nominálnímu napětí motoru 12V z toho důvodu, že je motor velmi zatížen a tím pádem je vysoká hodnota proudu, který jde do motoru. Mechanická zátěž, kterou je noha robota, je tak velká, že i při použití polovičního napětí 6V dosahuje odebíraný proud místy cca 70% maximálního povoleného proudu. Při použití velmi malého napětí, např. 3V, by zase motor nevyvinul potřebný točivý moment, aby dokázal pohnout nohou robota. Proto jsem zvolil tuto hodnotu napětí.

```
function [dxdt, y] = link_and_motor(t,x,U,R,L,Km,Kg,g,l,m,lc,I)
      % Output equation.
      y = x;
      % State equations. Parametrized by physical parameters.
       tau_m = Km*x(1);                % Torque of the motor shaft
       tau_g = Kg*tau_m*0.49;
       U_ind = Kg*Km*x(3);                % Back electromotive force

  dxdt = [
      (U-R*x(1)-U_ind)/L;
                x(3)
      (tau_g-F-m*g*lc*sin(x(2)))/(I+m*lc^2);
  ];
end
```

Po vytvoření souboru s funkcí popisující odhadovaný systém se spuštěním funkce `idnlgrey` provede samotná grey box identifikace. Mezi důležité parametry pro identifikaci pomocí `idnlgrey` patří jméno funkce, obsahující model systému, dále řád systému (počet vstupů, stavů a výstupů), odhadované parametry systému a počáteční podmínky. Výstupem funkce je objekt reprezentující odhadnutý nelineární systém. Níže je ukázka zdrojového kódu pro identifikaci pomocí této funkce.

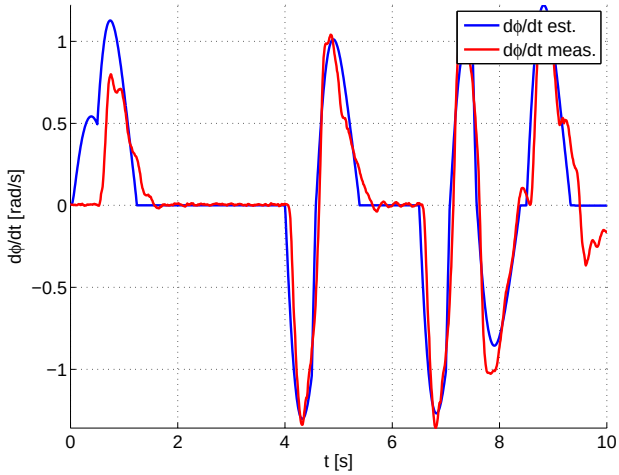
```
%nastaveni parametru
  FileName = 'link_and_motor'; % soubor s implementovanym modelem systemu
  Order = [3 1 3]; % rad systemu [ny nu nx]
  Parameters = [R,L,Km,Kg,Img,g,l,mc,lc,I,b0,b1,b2,a1]; % parametry
  InitialStates = [0; angle(1); 0]; % pocatecni podminky [i; phi; dphi]
  Ts = 0.01; % vzorkovani

% samotna identifikace
  nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts, 'Name', 'One_link');
```

Výsledky identifikace

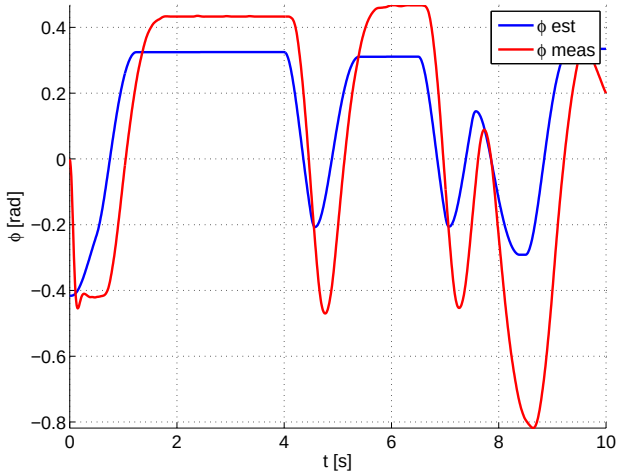
Pomocí uvedeného postupu jsem provedl nelineární grey box identifikaci. Hodnota momentu setrvačnosti, získaná pomocí této identifikace, je $\mathrm{I} = 0.35\,\mathrm{kg\,m^2}$.[2] Na obr. 3.2, 3.3 a 3.5 jsou červenou barvou znázorněny průběhy měřených veličin během identifikačního experimentu. Také jsou zde pro porovnání modře znázorněny průběhy stejných veličin získané z modelu (3.3), (3.4) použitého pro identifikaci (motor a kyvadlo) při přivedení

[2] Tato hodnota momentu setrvačnosti v sobě zahrnuje i tření.

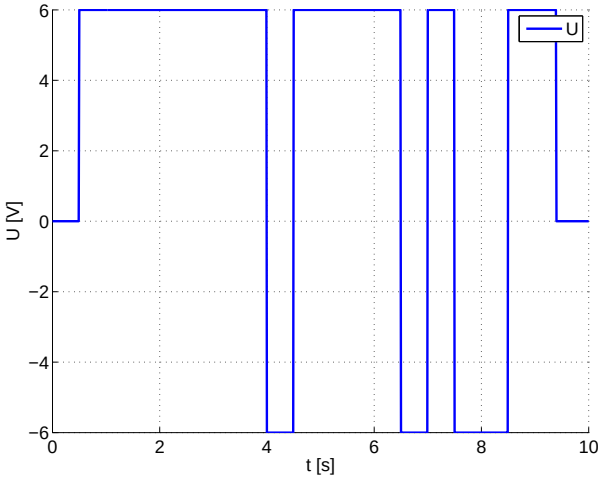
stejného napětí, jaké bylo použité pro identifikační experiment. Při identifikaci jsem čerpal z knihy (Ljung, L., 1999).



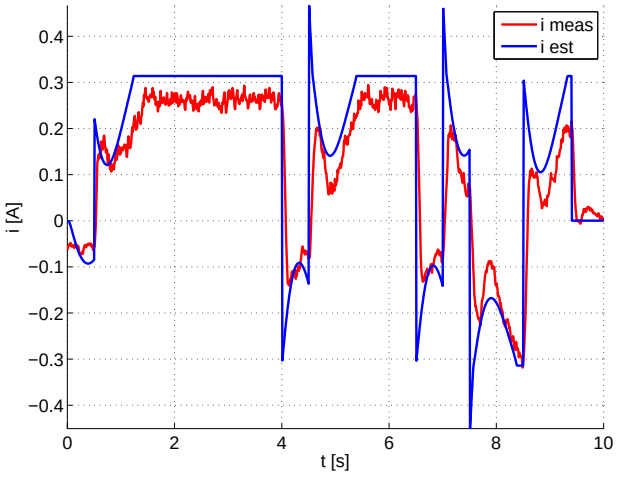
Obrázek 3.2: Úhlová rychlost



Obrázek 3.3: Poloha v kloubu robota



Obrázek 3.4: Napětí na motoru



Obrázek 3.5: Proud tekoucí do motoru

Kapitola 4

Popis měřicí desky

V této kapitole je popsána elektronika na modelu kráčejícího robota. Je zde uveden popis použitého procesoru, senzorů, motorů a dalších elektronických součástek. Také je zde popsáno schéma zapojení jednotlivých desek plošných spojů.

Elektroniku robota jsem navrhl vzhledem k jeho mechanické konstrukci a vlastnostem. Celé tělo kráčejícího robota se skládá ze dvou nohou, kde každá noha má dva klouby. Celkem je tedy potřeba ovládat čtyři klouby. K tomu je třeba měřit všechny potřebné veličiny pro řízení chůze robota. Konkrétně úhly ve všech kloubech a úhlové rychlosti jednotlivých článků. Dále je třeba měřit proud, který je odebírán jednotlivými motory, aby nedošlo k přetížení a zničení motorů. Také bylo třeba vybavit jednotlivé klouby robota koncovými dorazy, aby nemohlo dojít k nárazu jednotlivých článků robota. Dále byla vedoucím práce požadována taková koncepce elektroniky, aby byl robot osazen celkem čtyřmi deskami plošných spojů, kde každá deska nezávisle na ostatních má na starost jeden z kloubů robota.

Vzhledem k výše uvedeným požadavkům je robot osazen čtyřmi deskami plošných spojů. Každá z těchto desek ovládá jeden kloub robota a zároveň měří všechny potřebné veličiny, jako úhel natočení příslušného kloubu, úhlovou rychlost, odebíraný proud atd. Komunikaci mezi jednotlivými deskami jsem realizoval po sběrnici CAN. Použití této sběrnice je výhodné, protože stačí pro propojení všech desek mezi sebou pouze dva vodiče. Navíc je možné po této sběrnici ovládat jednotlivé desky z PC a zároveň přijímat všechny potřebné informace.

4.1 Popis významných součástek

Zde je uveden stručný popis nejdůležitějších elektronických součástek na deskách plošných spojů modelu robota. Níže uvedené součástky jsou osazené na všech deskách modelu robota. Podrobnější informace k jednotlivým součástkám je možné nalézt v datasheetech k uvedeným součástkám.

4.1.1 Gyroskopy

Pro měření úhlových rychlostí jednotlivých článků robota jsem použil jednoosé gyroskopy ADIS16260 společnosti *ANALOG DEVICES*. Tyto gyroskopy vyrobené technologií MEMS pracují na principu Coriolisovy síly. Hlavní parametry gyroskopů jsou uvedeny níže, viz. tabulka 4.1.

Hlavní parametry gyroskopu
Rozsah měření $\pm 80°/s$,$\pm 160°/s$ nebo $\pm 320°/s$
Komunikace po sběrnici SPI
Možnost měření relativního úhlu natočení
Rychlost měření 256 vzorků/s
Rozlišení 14 bitů

Tabulka 4.1: Parametry gyroskopu

Princip MEMS gyroskopu

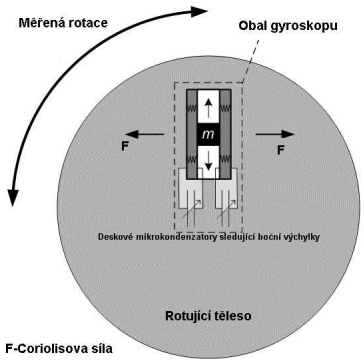
Pod pojmem MEMS (Micro-Electro-Mechanical Systems) se míní umístění elektronických a mikro-mechanických prvků na křemíkovou bázi. Použitý gyroskop ADIS16260 pracující na principu Coriolisovy síly umí měřit úhlovou rychlost pouze v jednom směru - kolmém na plochu čipu. Pro jiné směry je nutné zajistit správné natočení a umístění součástky.

Coriolisova síla F_c je zdánlivá síla, která působí na objekty které se pohybují v rotující neinerciální vztažné soustavě. Je definována jako vektorový součin rychlosti pohybujícího se tělesa v a úhlové rychlosti otáčení vztažné soustavy ω. Viz rovnice (4.1).

$$F_c = 2mv \times \omega \tag{4.1}$$

Základ senzoru tvoří rezonující struktura upevněná v rámu, která se vlivem vlastní mechanické rezonance, zde reprezentované pružinami, pohybuje v uvedeném směru -

kolmém na směr otáčení viz obr. 4.1. Přitom vzniká Coriolisova síla úměrná úhlové rychlosti otáčení, která stlačí vnější pružiny rámu a způsobí vzájemný posuv měřících plošek fungující jako elektrody vzduchových kondenzátorů. Výstupem je tedy změna kapacity úměrná úhlové rychlosti otáčení.



Obrázek 4.1: Princip MEMS gyroskopu

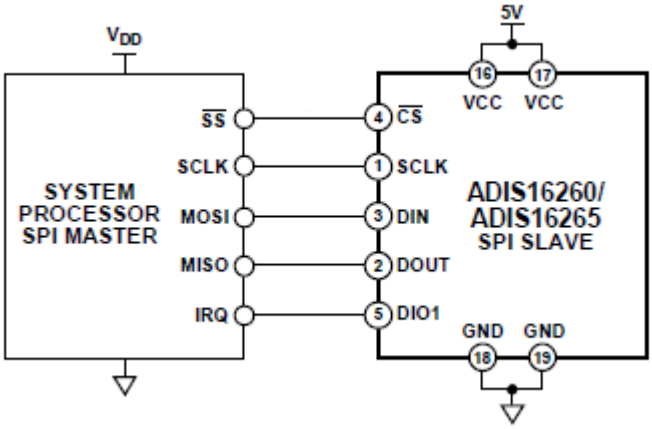
Komunikace s gyroskopem probíhá přes rozhraní SPI(Serial Peripheral Interface). Jedná se o sériové periferní rozhraní pracující na principu master/slave. Schématické propojení je znázorněno na obr. 4.2. Komunikace je realizována pomocí čtyř vodičů:

- **SCK** - hodinový signál pro synchronizaci
- **MISO** - datový signál. Výstup z gyra(slave) směrem k zařízení master
- **MOSI** - datový signál. Vstup do gyra ze zařízení master. Většinou požadavek na data.
- **CS** - signál pro výběr zařízení slave.

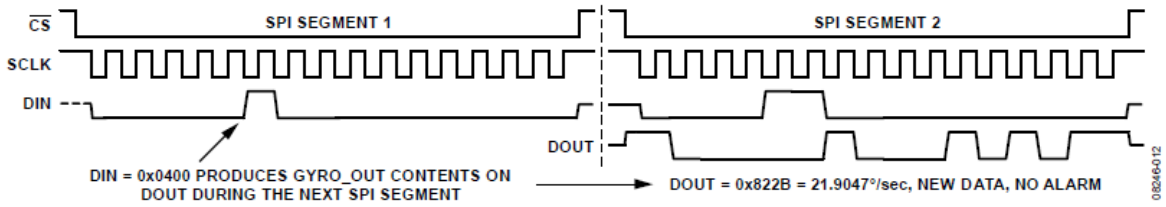
Pokud chce zařízení master(procesor) komunikovat se zařízením slave(gyroskop), pak pomocí CS signálu vybere konkrétní zařízení. Pak začne vysílat hodinový synchronizační signál CLK a požadavek na data. Slave(gyroskop) začne vysílat zpět požadovaná data. Pro komunikaci s gyroskopem se používá 16-bitových rámců. Na obr. 4.3 je znázorněný příklad sekvence pro čtení úhlové rychlosti.

4.1.2 Potenciometry

Pro měření absolutní polohy jednotlivých kloubů jsou použity rotační potenciometry, pracující na principu odporového děliče napětí. Původně jsem chtěl pro měření polohy



Obrázek 4.2: Propojení gyroskopu se zařízením typu master (datasheet Adis16260)



Obrázek 4.3: Příklad čtení po sběrnici SPI(datasheet Adis16260)

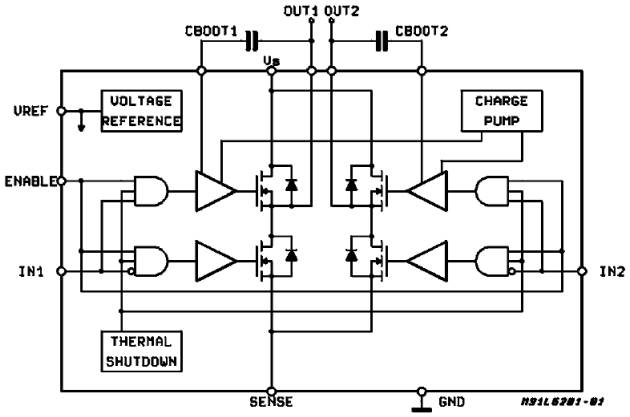
použít magnetický senzor polohy AM8192, ale to z finančních důvodů nebylo možné. Proto jsem použil levnou a podstatně méně přesnou metodu měření v podobě potenciometrů. Na potenciometry je přivedeno z příslušné desky napětí $3.3V$. Výstupem z každého potenciometru je napětí v rozsahu 0 až $3.3V$, které je přivedeno na A/D převodník procesoru. V průběhu vývoje robota jsem mezi potenciometr a A/D převodník procesoru přidal RC filtr typu dolní propust pro filtrování šumu.

4.1.3 H-můstek

Pro ovládání motorů robota pomocí PWM signálu jsem použil plný H-můstek L6201. Tento integrovaný obvod převádí PWM signál z řídicího procesoru na výkonový PWM signál o velikosti 12V, který je přiveden přímo na motor. Blokové schéma obvodu L6201 je na obr. 4.4. Základ můstku tvoří čtyři spínací DMOS tranzistory. Přes vnitřní logiku obvodu jsou tranzistory spínány pomocí signálů PWM signálů IN1 a IN2 přivedených z procesoru LPC2368. Hlavní parametry obvodu viz tabulka 4.2.

Parametry H-můstku
Napájecí napětí až do 48V
Maximální odebíraný proud (střední hodnota) 4A
Maximální proudová špička 5A
Odpor při sepnutí 0.3 Ω
Max. spínaná frekvence 100kHz
Ochrana proti přehřátí

Tabulka 4.2: Parametry H-můstku L6201



Obrázek 4.4: Zapojení plného H-můstku(datasheet L6201)

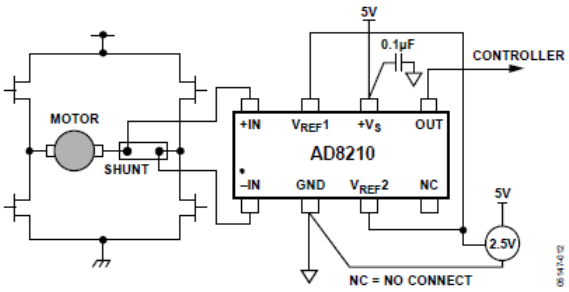
4.1.4 Měření proudu

Pro měření proudu odebíraného motorem, které je potřebné pro ochranu proti přetížení motoru a také pro realizaci proudové zpětné vazby, jsem použil integrovaný obvod AD8210. Jedná se o diferenciální zesilovač, ideální pro zesilování malých úbytků napětí. Zesilovač je připojený na rezistor, který je připojený v sérii s motorem podle schématu na obr. 4.5. Polarita proudu protékaného motorem(a tedy i rezistorem) je úměrná úbytku napětí na rezistoru. V případě vhodně zvoleného referenčního napětí (2.5V) odpovídá nulovému proudu výstupní napětí rovné polovině napěťového rozsahu. Toto výstupní napětí je pak přímo připojeno na A/D převodník procesoru. Odpor rezistoru, použitého pro měření proudu, je třeba dobře zvolit. Odpor by měl být zanedbatelný vzhledem k odporu vinutí motoru. Velikost odporu také ovlivňuje rozsah měřeného proudu a přesnost měření. Diferenciální zesilovač AD8210 je schopný zesilovat úbytky napětí v rozsahu $\pm 125mV$. Maximální povolený proud odebíraný motorem je $570mA$. Hodnotu odporu

jsem proto zvolil podle výpočtu (4.2) $R = 0.2\ \Omega$.

$$R = \frac{U_{max}}{I_{max}} = \frac{125mV}{570mA} = 0.21\ \Omega \approx 0.2\ \Omega \tag{4.2}$$

Při takto zvoleném odporu je možné měřit proud v rozsahu $\pm 625mA$ s přesností $1.6mA$. Tato přesnost je dána rozlišením A/D převodníku procesoru. Velikost zvoleného odporu je také zanedbatelná vzhledem k velikosti odporu vinutí motoru, které má odpor přibližně 20Ω.



Obrázek 4.5: Princip měření proudu pomocí AD8210

4.1.5 Napájení

K zajištění chodu všech použitých součástek bylo třeba zajistit napájení desky několika úrovněmi napětí. Zde je uveden pouze stručný popis. Podrobně je vše znázorněno ve schématu zapojení desky.

- **3.3V** pro napájení procesoru a CPLD
- **5V** pro napájení gyroskopu a měřiče proudu
- **12V** pro napájení H-můstku
- **24V** pro napájení laserového měřiče vzdálenosti

Napětí v rozsahu 12-15V z laboratorního zdroje je přivedeno na spínaný stabilizátor napětí LM2576HV. Výstupem stabilizátoru je napětí 12V, vedoucí na H-můstek a DC/DC měniče TSR 1-2465 s výstupem 6.5V a IU2412S s výstupem 24V pro napájení laserového měřiče vzdálenosti. Napětí 6.5V z DC/DC měniče TSR 1-2465 je dále přivedeno na napěťový stabilizátor TPS73HD. Obvod TPS73HD by mohl být přímo připojen na napětí 12V, ale je to lineární stabilizátor a proto úbytek napětí na něm a tím pádem

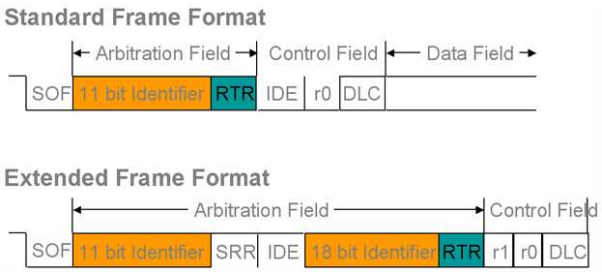
i ztrátový výkon by byl příliš velký a mohlo by dojít ke zničení stabilizátoru vlivem ztrátového výkonu. Proto je napájecí napětí vedeno přes spínaný měnič TSR 1-2465, který má malý ztrátový výkon. Ze stabilizátoru TPS73HD je vyvedeno napětí 3.3V pro napájení procesoru a CPLD a dále 5V pro napájení gyroskopu a měřiče proudu. Stabilizátor navíc obsahuje resetovací výstup pro resetování procesoru při poklesu výstupního napětí[1]. Podrobnější informace o zde uvedených součástkách použitých pro napájení je možné nalézt v datasheetech k těmto součástkám, které jsou na přiloženém CD.

4.1.6 CAN

Komunikace všech desek mezi sebou a s PC probíhá po sběrnici CAN. Použití sběrnice CAN je výhodné, protože umožňuje pomocí pouze dvou vodičů propojit více než dvě zařízení.

CAN je sériová datová sběrnice vyvinutá firmou Bosch. Síťový protokol detekuje a opravuje přenosové chyby vzniklé od okolních elektromagnetických polí. Dovoluje snadné nastavení (konfiguraci) systému a umožňuje centrální diagnostiku. Vysílaná data nemají žádnou adresu, obsah zprávy je dán identifikátorem (ID), který je v celé síti jedinečný. Tento identifikátor definuje obsah přenášené zprávy a zároveň i prioritu zprávy při pokusu o její odeslání na sběrnici. Vyšší prioritu mají zprávy s nižší hodnotou identifikátoru. Příjem zpráv může být mnohonásobný (jedna zpráva může být přijata několika zařízeními). Maximální rychlost přenosu je na sběrnici 1Mbit/sec.

CAN protokol podporuje dva rámcové formáty zpráv, které se liší podle délky identifikátoru. Ve standardním formátu je délka ID 11 bitů a v prodlouženém formátu (extended) je délka označení zprávy 29 bitů. Já jsem pro komunikaci mezi deskami použil pouze zprávy ve standardním formátu. Formát zpráv je znázorněn na obr. 4.6.



Obrázek 4.6: Formát CAN zprávy (zdroj www.softing.com)

[1] Při poklesu napětí a vrácení na původní hodnotu procesor nemusí pracovat správně, nedojde-li k jeho vyresetování.

Zpráva začíná počátečním bitem "Start Of Frame". Dále následuje rozhodovací pole, které obsahuje identifikátor a "RTR"bit, který ukazuje, zda jde o rámec dat nebo požadavek rámce bez bitů. Kontrolní pole obsahuje "IDE"bit, který označuje buď standardní formát, nebo prodloužený formát, dále pak bit "R0"rezervovaný pro budoucí prodloužení a informaci o délce datového pole (DLC). Datové pole je v rozmezí od 0 do 8 bytů a je následováno CRC polem, které se používá pro detekci chyb v přenosu.

Pro připojení jednotlivých desek robota na sběrnici CAN jsem použil budič PCA82C250. Budič v každé desce funguje jako rozhraní mezi procesorem příslušné desky a CAN sběrnicí.

Parametry CAN budiče
Rychlost přenosu až 1 Mbaud
Možnost připojení až 110 uzlů
Nízký odebíraný proud
Tepelná ochrana
Plně kompatibilní s "ISO 11898" standardem

Tabulka 4.3: Parametry CAN budiče PCA82C250

4.1.7 Procesor LPC 2368

Pro řízení jednotlivých desek robota bylo třeba vybrat vhodný procesor. Použitý procesor musí zajistit potřebný výpočetní výkon. Dále podporovat komunikaci se všemi použitými senzory tj.podpora rozhraní SPI pro čtení dat z gyroskopu, schopnost komunikace po sběrnici CAN, možnost řízení pohonů robota pomocí PWM signálu a obsahovat analogově-digitální převodníky pro měření proudu a pro měření úhlů v kloubech pomocí potenciometru.

S ohledem na výše uvedené požadavky byl vybrán procesor LPC 2368 řady ARM7. Jedná se o 32-bitový procesor s RISC architekturou. Procesor byl vybrán také kvůli dobrým zkušenostem kolegů s tímto procesorem.

Hlavní parametry procesoru:

- Max. CPU frekvence až 72 MHz
- 512KB on-chip Flash ROM

- 58KB RAM
- UART rozhraní - použité pro programování procesoru, pro komunikaci s PC
- Komunikace po sběrnici CAN - komunikace s PC a s ostatními deskami
- Komunikace po sběrnici SSP/SPI - komunikace s gyroskopem
- Jednotka pro generování PWM signálu - pro řízení otáček motoru
- 10-bitové A/D převodníky - pro čtení analogového napěťového signálu z potenciometrů, z měřiče proudu a z laserového senzoru vzdálenosti
- I/O piny - použité pro přivedení externího přerušení, ovládání kontrolních LED a dále jako paralelní sběrnice pro komunikaci s CPLD

Pro programování procesoru jsem použil volně dostupné vývojové prostředí Eclipse. Po přeložení z jazyka C do binárního souboru se program z Eclipse nahraje po sběrnici UART do procesoru. Popis programů jednotlivých desek je v samostatné kapitole 5.

4.1.8 CPLD

Pro možnost čítání impulsů z inkrementálních senzorů, umístěných v motorech robota, jsem se rozhodl použít CPLD obvod Xilinx XC95144XL. Pokud by tyto signály byly přivedeny přímo na piny procesoru LPC 2368, byl by procesor zahlcen pouze čítáním impulzů z těchto signálů. Při použití CPLD je možné ušetřit výpočetní výkon procesoru a předzpracovat signály z inkrementálních senzorů v CPLD obvodu. Pro komunikaci mezi CPLD a procesorem je použita paralelní 8-bitová sběrnice.

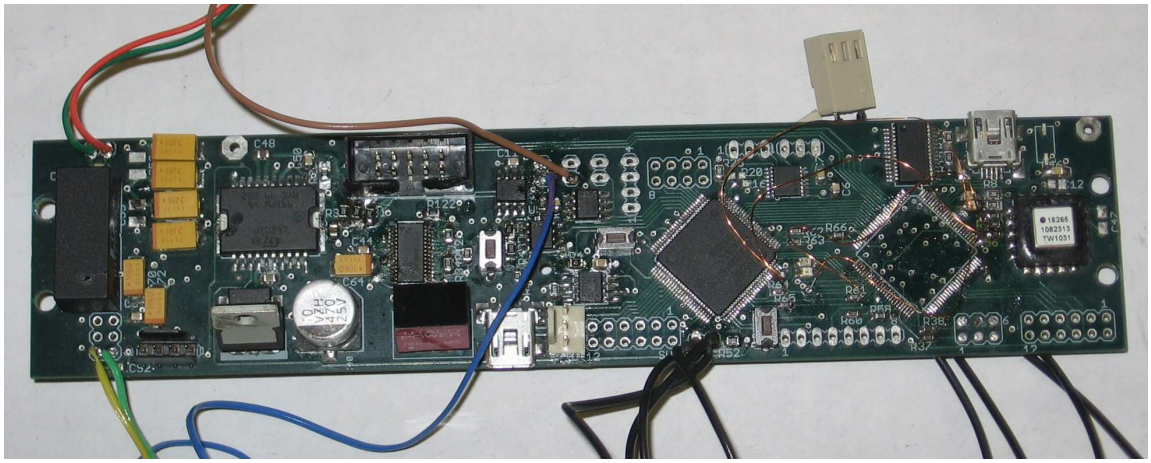
4.2 Vývoj elektroniky

V této části je popsán vývoj elektroniky robota. V průběhu práce na vývoji kráčejícího robota byly celkem navrženy dvě verze desky plošných spojů(DPS). První verze byla určena jako testovací deska pro otestování použitých součástek popsaných v části 4.1 a pro začátek vývoje programu pro řízení robota. Druhou verzi DPS jsem navrhl s drobnými změnami na základě zkušeností získaných při práci s první verzí desky.

Návrhy desek jsem prováděl v programu Eagle. Všechny desky jsou navrženy jako čtyřvrstvé. Desky byly vyrobeny firmou Pragoboard. Všechny součástky jsem na ně sám ručně osadil a desky oživil. U druhé verze jsem gyroskopy a součástky s velmi malými vývody, které byly náročné na osazení a pájení(procesor LPC2368, CPLD) osadil ve spolupráci s katedrou elektrotechnologie. Podklady pro výrobu DPS z programu Eagle jsou na přiloženém CD. Schéma zapojení desky z programu Eagle je v příloze.

4.2.1 První verze

Zde je popsána první verze DPS, použitá v začátku vývoje elektroniky robota. V této části je popsáno pouze hardwarové zapojení. Činnost a programy jednotlivých desek jsou popsány v následující kapitole.



Obrázek 4.7: Fotografie první verze desky plošných spojů

Nejdůležitější částí desky je procesor LPC2368, který řídí veškeré procesy, jako je čtení dat ze všech senzorů, řízení motoru a komunikace s ostatními deskami a PC. Procesor je napájen napětím 3.3V z obvodu TPS73HD a taktován krystalovým oscilátorem s frekvencí 12MHz.

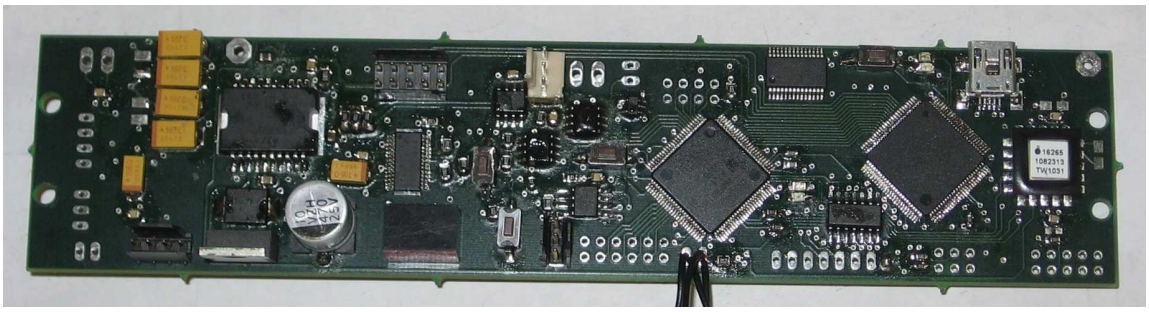
Pro komunikaci procesoru s okolím jsou vyvedeny rozhraní 3x UART, USB, JTAG, CAN, SSP/SPI a I2C. Pro programování jsou použity signály RX0 a TX0 sběrnice UART. Tyto signály jsou přivedeny na převodník FT232R. Jedná se o převodník sériového rozhraní UART na rozhraní USB. Při připojení převodníku USB kabelem k PC operační systém vnímá připojený obvod FT232R jako virtuální sériové rozhraní, kterému přiřadí COM port(většinou COM4). Přiřazený COM je možné ručně změnit. Zbývající dvě UART rozhraní jsou vyvedeny na obvod MAX232 a dále na samostatné konektory. Rozhraní

sběrnice CAN je z procesoru přivedeno na budič PCA82C250. Z budiče jsou signály CAN sběrnice dále přivedeny na třípinový dubox konektor. Rozhraní JTAG,USB a I2C jsou vyvedeny přímo z procesoru na samostatné konektory. Rozhraní SSP/SPI použité pro komunikaci s gyroskopem je přivedeno na CPLD a z něj dále na gyroskop.

Analogové napěťové signály z potenciometrů, laserového senzoru vzdálenosti a měřiče proudu jsou přivedeny na CMOS zesilovače AD8656. Zde jsou přeškálovány na rozsah 0 až 3V a přivedeny na A/D převodníky procesoru.

4.2.2 Druhá verze

Zde je popsána druhá verze desek a v době psaní této diplomové práce také poslední verze. Při návrhu druhé verze jsem na základě práce s první deskou a získaných zkušenostech provedl drobné úpravy a vylepšení. V této části jsou pouze popsány změny v návrhu DPS. Ostatní věci jsou stejné jako u první verze desky.



Obrázek 4.8: Fotografie druhé verze desky plošných spojů

U druhé verze jsem odebral nepotřebné konektory. Rozhraní USB a dvě rozhraní UART jsem odebral, protože jsem je nijak nevyužíval. Pro programování procesoru a pro komunikaci s PC postačilo pouze jedno UART rozhraní.

Na DPS jsem přidal externí EEPROM paměť 24LC02B firmy MIKROCHIP. Jedná se o externě programovatelnou paměť s kapacitou 2kbit. Zařízení je organizováno jako blok 256-ti 8-bitových pamětí připojených pomocí dvouvodičového sériového rozhraní I2C. Paměť je připojena přímo na I2C rozhraní procesoru LPC2368. Externí paměť na desce je výhodná pro možnost uložení konstant a parametrů důležitých pro chod programu procesoru. Pokud budou např. na této externí paměti uloženy konstanty regulátoru, je možné ladit takový regulátor pomocí zápisu do EEPROM po sběrnici CAN bez nutnosti neustálého přeprogramování procesoru.

Další změnou bylo vedení SPI sběrnice použité pro komunikaci mezi procesorem a gyroskopem přímo do procesoru a ne přes CPLD obvod. Druhá SPI sběrnice je navíc vyvedena přes budič 74HC126 na samostatný konektor pro možnost budoucího připojení magnetického senzoru polohy AM8192.

Při práci s první verzí DPS jsem zjistil, že analogový signál z potenciometru do A/D převodníku procesoru je zatížen velkým šumem. Řešení se nabízela dvě. Nejlepší by bylo potenciometr vůbec nepoužít a místo toho použít senzor AM8192. Další řešení bylo filtrovat analogový signál z potenciometru pomocí dolní propusti. Jak už bylo výše řečeno, z finančních důvodů nebylo možné použít senzory AM8192. Proto jsem mezi potenciometr a zesilovač AD8656 přidal RC filtr typu dolní propust prvního řádu s hodnotami $R = 10k\Omega,\ C = 100nF$. Takto navržený filtr má podle vztahu (4.3) mezní frekvenci $f_c = 160Hz$.

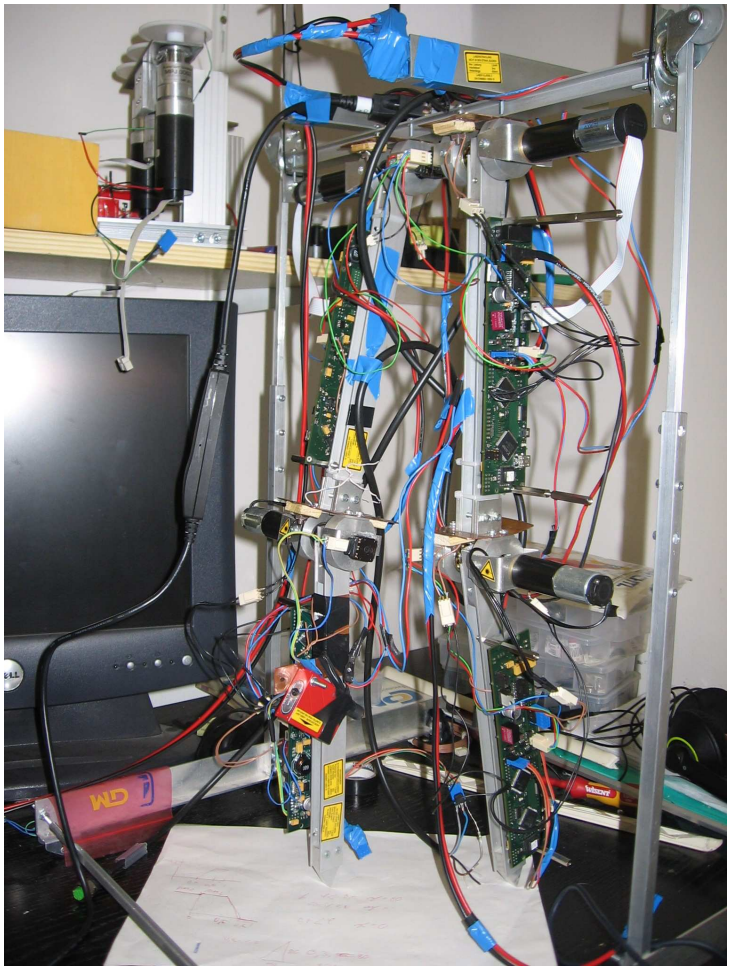
$$f_c = \frac{1}{2\pi RC} \tag{4.3}$$

Tento filtr reaguje dostatečně rychle na změny polohy kloubu robota a zároveň odfiltruje nežádoucí šum.

Druhá verze DPS s výše uvedenými změnami byla vyrobena v počtu 4 kusů, kde každý kus ovládá jeden kloub robota. Plošné spoje byly vyrobeny ve firmě Pragoboard. Součástky jsem osadil a desky oživil vlastními silami. Gyroskopy, CPLD a procesory jsem osadil ručně a přiletoval v pájecí peci ve spolupráci s Ing. Pelikánovou z katedry elektrotechnologie. Gyroskopy mají totiž piny umístěné takovým způsobem, že je téměř nemožné je ručně přiletovat. Ostatní součástky jsem osadil a přiletoval sám.

Shrnutí kapitoly

V této kapitole byl popsán návrh elektroniky kráčejícího robota a všechny důležité součástky, které jsem použil. Jako konečná verze elektroniky byla použita výše popsaná druhá verze vyvinutých DPS. Tato verze byla vyrobena v počtu 4 kusů, vlastními silami osazena a připevněna na fyzický model robota. V budoucnu by bylo možné na další verze desek přidat analogový filtr pro filtrování měřeného proudu. Toto měření obsahuje velký šum. Pro filtraci jsem v programu pro procesory desek implementoval digitální IIR filtr, ale lepší by bylo použití analogového filtru. Pro přidání tohoto filtru jsem se rozhodl v době, kdy už byla vyrobena druhá verze DPS. Na vyrobené a osazené desky už nebylo možné



Obrázek 4.9: Robot s osazenou elektronikou(pracovní verze)

tento filtr přidat. Pro lepší a přesnější měření poloh v kloubech robota by v budoucnu bylo vhodné použité potenciometry nahradit magnetickými senzory polohy AM8192. To prozatím z finančních důvodů nebylo možné.

Kapitola 5

Implementace řídicího programu

V této kapitole je uveden popis algoritmu zajišťující řízení modelu robota. V části 5.1 je obecně popsán koncept, jak jsem se rozhodl celého robota řídit. Dále jsou zde popsány režimy činnosti robota. V části 5.3 je podrobný popis programů pro procesory jednotlivých desek robota. V podkapitole 5.2 je popsáno programovací prostředí, postup přeložení a nahrání programu do cílového procesoru LPC 2368.

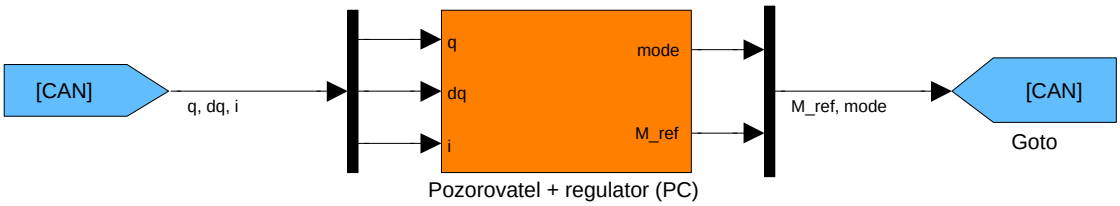
5.1 Koncepce řízení robota

Celý proces řízení modelu robota by se dal rozdělit na dvě části(nebo také dva oddělené a na sobě nezávislé algoritmy).

První částí je algoritmus, který ve zpětné vazbě řídí chůzi modelu robota. Tento algoritmus je realizovaný jako nelineární pozorovatel a regulátor, který je teoreticky popsaný v kapitole 2. Jedná se o hlavní algoritmus pro řízení chůze robota. Tento nelineární regulátor a pozorovatel stavů, který zajišťuje řízení chůze robota, stabilizaci robota a odhad neměřených stavů, je kvůli velké náročnosti na výpočetní výkon realizován v Matlabu na PC a se samotným robotem komunikuje po sběrnici CAN. Hlavní algoritmus pro řízení chůze jsem realizoval v Matlabu také kvůli požadavku vedoucího práce, aby mohl být model robota použit při výuce a studenti mohli snadno měnit parametry regulátoru přímo v Matlabu.

Pozorovatel a regulátor v Matlabu přijímá z jednotlivých desek robota pomocí sběrnice CAN potřebná data (úhly v kloubech robota, úhlové rychlosti, proudy odebírané motory). Z těchto dat vypočítá potřebný moment, který je třeba vyvinout v horních kloubech

("kyčlích"robota, pokud pro lepší názornost použiji přirovnání s lidským tělem), viz obr. 5.1. Matematický model, podle kterého regulátor počítá potřebný akční zásah (kroutící moment) popisuje robota zjednodušeně, jako systém se dvěma stupni volnosti a pouze jedním akčním zásahem, tj. jako robota, který nemá "kolena", nebo jako robota, který má tyto "kolena"trvale napnutá. Fyzicky má robot 5 stupňů volnosti(4 klouby - dvě kolena a dvě kyčle a další stupeň volnosti v bodě, kde se dotýká země nohou, na které právě stojí) a 4 akční zásahy(4 motory). Během chůze jsou kolena robota propnutá. Jedinou výjimkou je noha, která se právě pohybuje.[1] V momentě, kdy se pohybující se noha přiblíží k noze, na které robot právě stojí, se pohybující noha pokrčí, aby nezavadila o zem. Koleno pohybující se nohy se propne, jakmile se pohybující noha vzdálí od nohy, na které robot stojí. Toto pokrčení nohy, které je nutné pro chůzi robota, je v matematickém modelu systému zanedbáno. Pro lepší regulaci jsem toto pokrčení později zohlednil jako skokovou změnu momentu setrvačnosti pohybující se nohy robota. Hlavní regulátor implementovaný v Matlabu v PC je podrobně popsán zvlášť v kapitole 6. V kapitole 6 je také podrobně popsána komunikace mezi PC a robotem.



Obrázek 5.1: Hlavní pozorovatel a regulátor v PC

Druhá část algoritmu pro řízení robota, jsou vnitřní regulační smyčky, implementované v deskách robota. Tyto regulační smyčky jsou implementované distribuovaně v jednotlivých deskách robota. Každá deska obsahuje vlastní regulační smyčku pro řízení svého kloubu. Programy pro řízení jednotlivých desek jsou napsány v jazyce C a mají za úkol čtení dat ze senzorů příslušné desky a ovládání kloubů pomocí servomotorů. V programech jsou implementovány PID regulátory pro regulaci momentu a polohy v jednotlivých kloubech robota. Horní desky robota, ovládají pohyb v horních kloubech pomocí proudového regulátoru, který zajistí požadovaný točivý moment. Zde jsem využil toho, že u použitého motoru je výrobcem garantována lineární závislost mezi vyvinutým

[1] V literatuře, zabývající se kráčejícími roboty, se pohybující noha označuje jako "swing leg"a noha, na které robot stojí pojmem "stance leg".

momentem a odebíraným proudem. Podrobný popis programů pro horní desky je uveden níže v části 5.3. Spodní desky robota zajišťují ovládání dolních kloubů(pokrčování "kolen") pomocí regulátoru polohy. Programy spodních desek jsou popsány také v části 5.3.

5.1.1 Popis režimů činnosti robota

Pro ovládání robota jsem se rozhodl použít systém různých režimů činnosti.Tyto režimy jsou ve zdrojových kódech k procesorům robota označeny proměnnou `mode`. Hlavní režimy jsou **STOP**, **LEFT-STEP**, **RIGHT-STEP** a **CALIB**. Tyto režimy jsou použité pro řízení chůze robota. Potom existují ještě pomocné režimy CURR, ANGLE, EXT a CASCADE, které jsem vytvořil v průběhu vývoje pro ladění regulátorů. Přepínání mezi jednotlivými režimy je řízeno po sběrnici CAN a ovládáno z PC.

STOP: Režim STOP je nastaven ve všech procesorech robota při přivedení napájení na jednotlivé desky. Do režimu STOP je také uveden každý ze čtyř procesorů robota při restartu příslušného procesoru jeho resetovacím tlačítkem umístěným na desce(tlačítko označené kódem TL_2 v podkladech z programu Eagle). Na všech deskám je přivedeno nulové napětí na motory robota. Veškeré senzory dále měří proudy tekoucí do motorů, polohu v kloubech robota a také úhlové rychlosti získané z gyroskopů. Naměřená data všechny desky vysílají po sběrnici CAN s frekvencí 200Hz.

CALIB: Tento režim se používá pro kalibraci senzorů robota. Při přepnutí do režimu CALIB všechny desky přivedou na motory nulové napětí a přečtou několik vzorků dat z měřiče proudu a z potenciometru a z těchto dat spočítají aritmetický průměr. Tuto vypočítanou hodnotu proudu a polohy potom uloží do paměti jako offset senzorů. Takto získaný offset proudu a polohy se potom v programu odečítá z dat změřených ze senzorů. Následně je nastavena změna režimu z CALIB na STOP.

LEFT-STEP/RIGHT-STEP: V těchto režimech je robot během chůze. V režimu LEFT-STEP je, když provádí krok levou nohou vpřed a stojí na pravé noze. V režimu RIGHT-STEP je, když provádí krok pravou nohou vpřed a stojí na levé noze. Regulaci kolene stojící nohy v napnutém stavu obstará procesor v dolní desce stojící nohy, umístěné na lýtku robota. Pro regulaci polohy napnuté nohy, která je během kroku zatížena hmotností celého robota je v procesoru implementován nelineární regulátor polohy popsaný v části 5.4.2. Pokrčování kolene pohybující se nohy robota během kroku zajišťuje procesor v dolní desce příslušné nohy umístěné na lýtku robota. Pro pokrčování kolene je v procesoru implementován PID regulátor polohy. Referenční polohu počítá program

v procesoru pohybující se nohy z úhlu q_2, který je mezi nohami robota. Regulátor pro pokrčování kolene je popsán v části 5.4. Pohyb v kyčlích robota zajišťují procesory horních desek. V programech těchto procesorů je implementován regulátor momentu, který zajistí dosažení požadovaného kroutícího momentu v kyčlích robota. Referenční moment pro tento regulátor je generován v hlavním regulátoru v PC a horní desky robota jej přijímají po sběrnici CAN. Regulátor momentu je podrobně popsán v části 5.3.2.

5.2 Způsob programování procesoru

Pro programování použitých procesorů LPC 2368 jsem využil vývojové prostředí Eclipse. Eclipse je open source vývojová platforma, která je pro většinu lidí známa jako vývojové prostředí(IDE) určené primárně pro programování v jazyce Java. Toto prostředí lze snadno rozšířit pomocí pluginů např. o podporu jazyka C, C++ nebo PHP. Velikou výhodu je podpora většiny operačních systémů(Windows, Linux, Mac). Pro použití tohoto IDE jsem se rozhodl kvůli dobrým zkušenostem kolegů a také kvůli možnosti snadného programování ARM procesorů.

Postup přeložení a nahrání programu do LPC 2368

1. Nejprve je třeba připojit procesor k sériovému portu PC(pokud je jím počítač vybaven). Pokud PC není vybaveno sériovým portem, je třeba použít nějaký převodník UART-USB. Například obvod FT232R, který jsem použil.

2. Označí se projekt, který se má nahrát do procesoru, a v menu se klikne na: *Project* $->$ *Build project*. Eclipse následně přeloží vybraný projekt a vytvoří binární .hex soubor.

3. Označí se vygenerovaný .hex soubor a klikne se na: *Run* $->$ *External Tools* $->$ *Load*.

4. Objeví se okno, které žádá zadání čísla použitého COM portu. Pokud je použit převodník UART-USB, je možné v OS Windows toto číslo zjistit v záložce *Zařízení a tiskárny*, kde se po připojení převodníku objeví virtuální COM port. Windows převodníku většinou přiřadí COM4. Po zadání použitého COM portu je vše připraveno k nahrání programu do procesoru. Nyní Eclipse čeká, až se u použitého procesoru nastaví režim pro programování.

5. Posledním krokem je uvedení procesoru do režimu programování. Toho je dosaženo přivedením správné sekvence logických signálů na piny ISPSEL a RESETN. Pro tento účel je deska procesoru vybavena tlačítky, které po stisknutí přivedou na piny log 0. Následuje sekvence pro uvedení procesoru do režimu programování: Stisk tlačítka TL_2 na příslušné desce. Tlačítko je třeba držet stisknuté. Během té doby stisknout tlačítko TL_1, poté pustit TL_2 a následně pustit TL_1. Poté Eclipse nahraje do procesoru nový program.

5.3 Program jednotlivých desek robota

V této části jsou podrobně popsány programy pro procesory horních i dolních desek robota. Programy jsou velice podobné, proto je popíšu najednou. Jak už bylo uvedeno, horní desky ovládají pohyb v horních kloubech (kyčlích) robota a dolní desky v dolních kloubech (kolenech). Všechny desky čtou data z potenciometrů, které jsem použil pro měření úhlu v kloubech robota, proudy tekoucí do servomotorů a úhlové rychlosti z gyroskopu. Programy pro jednotlivé desky se od sebe liší v ID vysílaných CAN zpráv a v implementovaných regulátorech. V příloze diplomové práce se nachází vývojové diagramy, znázorňující činnost programů. Na pravé noze robota je navíc připevněn laserový senzor vzdálenosti. Proto je v programu pravé dolní desky robota navíc zpracování dat z tohoto senzoru.

Zdrojové kódy je možné nalézt na přiloženém CD. Zdrojový kód pro levou horní desku robota se jmenuje `acrobot_left_up`, kód pro pravou horní desku robota se jmenuje `acrobot_right_up`, pro levou dolní desku `acrobot_left_down` a kód pro pravou dolní `acrobot_right_down`.

5.3.1 Popis programu

Při startu programu se nejprve provede inicializace vstupů a výstupů a použitých sběrnic pomocí registrů PINSEL. Provede se inicializace sběrnice CAN na nejvyšší rychlost 1Mbit/s a zaregistruje se funkce pro obsluhu přerušení při přijetí nové CAN zprávy. Přijetí CAN zprávy zajišťuje funkce `can_rx`. Provede se inicializace SPI sběrnice pro komunikaci s gyroskopem. Délka datového rámce je nastavena na 16 bitů, procesor je nastaven jako zařízení typu master. Hodinový signál pro SPI je škálován na rychlost 600kHz. Dále je

provedena inicializace výstupního PWM signálu pro spínání motoru pomocí H-můstku. Frekvence PWM signálu je 40kHz. Pro ochranu robota před nárazem jsem použil externí přerušení. Provede se inicializace časovače a zaregistrování funkce `callback` pro obsluhu přerušení časovače. Dalším krokem je výpočet offsetu u měření proudu a dále kalibrace potenciometru pro měření polohy.

Poté je nastaven pracovní režim STOP a referenční hodnoty pro regulátor proudu jsou nastaveny na nulovou hodnotu. Jsou načteny parametry pro diskrétní filtry a PID regulátory a provede se jejich inicializace. V programu pro řízení horních desek jsem celkem použil dva IIR filtry prvního řádu pro filtrování měřených dat a jeden PID regulátor pro řízení servomotoru. Po výše popsané inicializaci program začne vykonávat hlavní smyčku.

Hlavní smyčka programu

V hlavní nekonečné smyčce program čte data ze senzorů. Tato naměřená data zpracuje a vysílá je s frekvencí 200Hz po sběrnici CAN ostatním deskám a do PC. V hlavní smyčce program také počítá akční zásah PID regulátoru.

Program nejprve přečte úhlovou rychlost gyroskopu. To provede vysláním binární sekvence DIN=0x0400 po sběrnici SPI. Gyroskop následně odpoví a pošle zpět do procesoru data. Tato změřená data jsou v 14-bitovém formátu. Pro další použití těchto dat je program ještě převede do 16-bitového integeru.

Poté program přečte a zpracuje data z A/D převodníků použitých pro měření polohy pomocí potenciometrů. U čtení dat z A/D převodníků občas docházelo ke kolizi a úplnému zaseknutí programu, když v průběhu čtení nastalo přerušení od časovače, který každých 10ms měří proud tekoucí motorem pomocí dalšího A/D převodníku[2]. V momentě, kdy je procesor uprostřed činnosti převodu dat z převodníku a nastane přerušení, které používá také jakýkoliv z A/D převodníků procesoru, dojde k zaseknutí procesoru, které lze vyřešit pouze restartem. Tento problém jsem nakonec vyřešil použitím semaforu. Pomocí A/D převodníku měřím pouze dvě veličiny, úhel v kloubu příslušné desky a proud odebíraný motorem. Data z převodníků jsou následně přepočítána na jednotky SI.

Měření proudu je bohužel zatížené velkým šumem. Proto jsem se rozhodl měřený proud filtrovat. K tomu jsem použil digitální IIR filtr prvního řádu, který jsem v programu

[2]Použitý procesor obsahuje celkem 6 A/D převodníků. Na desce jsou vyvedeny a použity 3 A/D převodníky

implementoval. Filtr je ve tvaru (5.1).

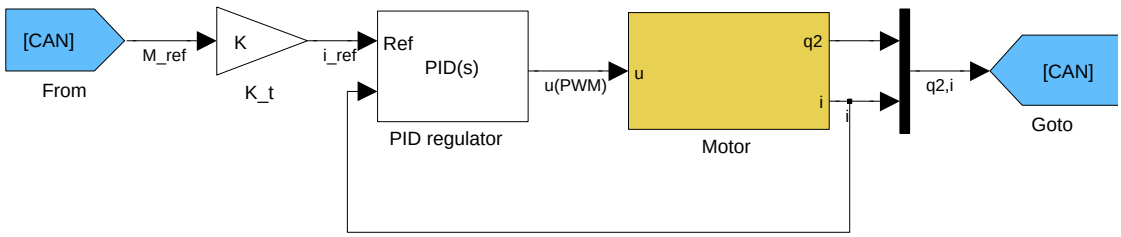
$$F(z) = \frac{b_0 z^n + b_1 z^{n-1} + \ldots + b_n}{a_0 z^n + a_1 z^{n-1} + \ldots + a_n} \tag{5.1}$$

Filtr jsem navrhoval v Matlabu. Polynomy a,b použitého filtru jsou $a = \begin{pmatrix} 1, & -0.9691 \end{pmatrix}$, $b = \begin{pmatrix} 0.0155, & 0.0155 \end{pmatrix}$. Navržený filtr dobře odfiltruje šum a zároveň rychle reaguje na změny velikosti proudu.

Změřená a zpracovaná data program vyšle po sběrnici CAN. Při frekvenci vysílání nových dat 200Hz a použité rychlosti sběrnice 1Mbit/s je zatížení sběrnice cca 10%.

Dalším krokem programu je výpočet akčního signálu pro motor pomocí PID regulátoru (případně kaskády PID regulátorů). Regulátory jsou v programech implementované podle vzorového zdrojového kódu v jazyce C z knihy (Åström, K.J. a Wittenmark, B., 1984), str.318. Výpočet akčního zásahu regulátoru je prováděn s frekvencí 100Hz.

5.3.2 Proudový regulátor



Obrázek 5.2: Regulátor momentu

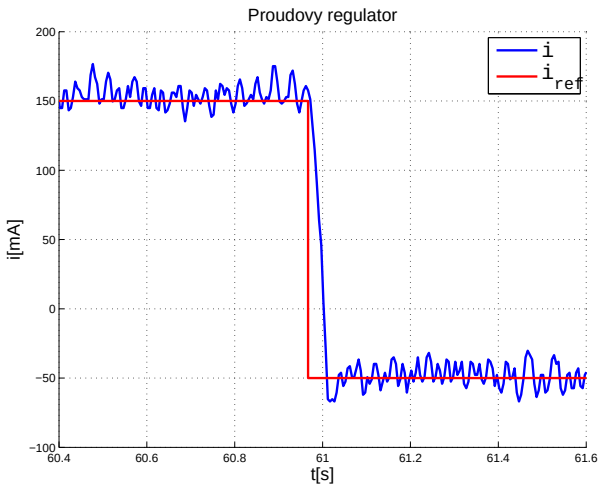
Pro dosažení žádaného momentu v kyčlích robota jsem implementoval proudový regulátor. Využil jsem zde toho, že v katalogu použitých motorů výrobce garantuje lineární závislost mezi odebíraným proudem a točivým momentem motoru. Momentová konstanta pro vztah mezi odebíraným proudem a točivým momentem motoru je $k_t = 12.1\text{mNmA}^{-1}$. Platí $M = k_t i$, kde M je moment motoru a i proud tekoucí do motoru. Proudový regulátor je implementovaný v programech pro horní desky robota. Referenční hodnotu točivého momentu program přijme po sběrnici CAN od hlavního nelineárního regulátoru v PC. Poté provede přepočet na proud podle vztahu $M = k_t i$. Měřený proud použitý v proudové zpětné vazbě je navíc filtrován IIR filtrem, popsaným v (5.1). Blokové schéma regulátoru, nakreslené v Simulinku, je na obr. 5.2.

Pro ladění regulátoru jsem použil grafické uživatelské rozhraní popsané v kapitole 6. Regulátor jsem ladil experimentálně přímo na robotu. Regulátor jsem naladil jako regulátor typu PI. Použitím PI regulátoru pro řízení proudu zavedeme pól na mezi stability. Pól na mezi stability nevadí díky omezení akčního zásahu do motoru(12V). Naladěné konstanty regulátoru viz tabulka 5.1.

K	0.065	N	10
T_i	45	u_{low}	-12V
T_d	0	u_{high}	12V
T_t	1	T_s	0.01s

Tabulka 5.1: Konstanty proudového regulátoru

Během vývoje regulátoru se objevil problém související s velmi velkým počátečním odběrem motoru při velké změně napětí. Při velké skokové změně napětí na motoru má motor tak velký proudový odběr, že poklesne napětí na celé desce a napájecí stabilizátor restartuje procesor. Proto jsem do programu navíc implementoval omezení změny akčního zásahu. Omezení jsem nastavil tak, že během jednoho cyklu výpočetní smyčky regulátoru (5ms) je změna PWM signálu do motoru maximálně 5%. Tuto hodnotu jsem nastavil experimentálně a motor při této velikosti omezení reaguje dostatečně rychle a proudové špičky se omezily. Odezva naladěného proudového regulátoru skokovou změnu referenční hodnoty je znázorněna na obr. 5.3. Proudový regulátor dosáhne velmi rychle referenční hodnoty proudu.



Obrázek 5.3: Proudový regulátor - odezva na změnu reference

5.4 Pokrčování kolen robota

V této části je podrobně popsán mechanismus, jakým jsem realizoval pokrčování kolen robota během chůze. Pokrčování kolen je úplně nezávislé na hlavním regulátoru a zajišťují jej programy v procesorech spodních desek robota. Struktura programu pro spodní desky je stejná jako struktura programu pro procesory horních desek, popsaná v části 5.3. Jediný rozdíl je v implementovaném regulátoru. Regulátor pro ovládání kolen robota má 3 hlavní části. Algoritmus pro pokrčování kolen robota během chůze je schematicky znázorněn na obr. 5.4.

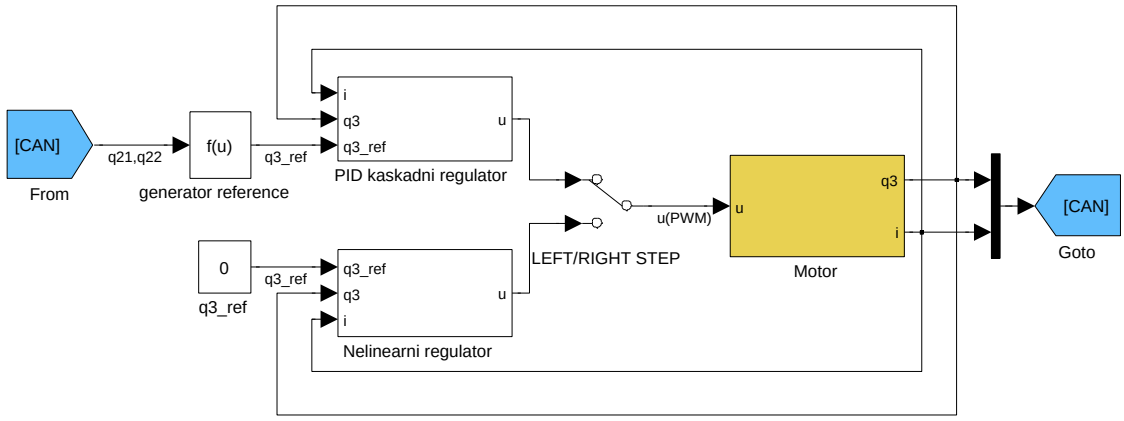
První částí algoritmu je regulátor pro pohybující se nohu. Tento regulátor, zajišťující pokrčování pohybující se nohy během kroku robota, je podrobně popsaný v části 5.4.1. Druhou částí je regulátor pro stojící nohu robota. Tento regulátor, popsaný v části 5.4.2, zajišťuje trvalé propnutí stojící nohy během kroku. Pro regulaci každé nohy jsem použil jiný regulátor z důvodu různé dynamiky nohou robota. Během vývoje robota jsem nejprve použil pro regulaci stojící nohy robota stejný regulátor, jako pro regulaci pohybující se nohy. Stojící noha je ale během kroku zatížena hmotností celého robota a velmi se zde projeví nelinearita systému. Proto jsem pro stojící nohu implementoval jiný typ regulátoru, než pro pohybující se nohu. Během chůze robota potom program v obou nohou robota přepíná mezi těmito dvěma regulátory. Přepínání mezi těmito dvěma regulátory se provádí podle režimů činnosti, popsaných v podkapitole 5.1.1. V režimu LEFT-STEP je v programu levé dolní desky aktivní první typ regulátoru pro pokrčování kolene a v pravé dolní desce druhý typ regulátoru, pro zachování kolene v napnutém stavu. Během režimu RIGHT-STEP je tomu naopak.

Třetí částí algoritmu je funkce pro výpočet referenční polohy, v jaké má být koleno pohybující se nohy. Tato funkce na základě polohy v kyčlích robota počítá referenční polohu pro koleno pohybující se nohy. Tato reference je použita pouze pro první typ regulátoru, který ovládá pohybující se nohu. Druhý typ regulátoru má referenční polohu nulovou (reference pro napnutou nohu).

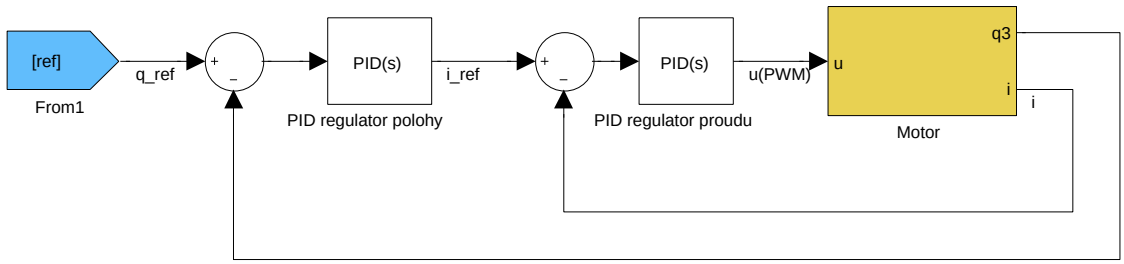
5.4.1 Kaskádní regulátor polohy

Pro ohýbání kolene pohybující se nohy během chůze jsem do programů pro procesory spodních desek robota implementoval kaskádní regulátor polohy a proudu. Struktura kaskádního uspořádání regulátoru je znázorněna na obr. 5.5.

Jedná se o kaskádu dvou za sebou spojených PID regulátorů. Ve vnitřní smyčce



Obrázek 5.4: Struktura algoritmu pro ovládání kolen robota



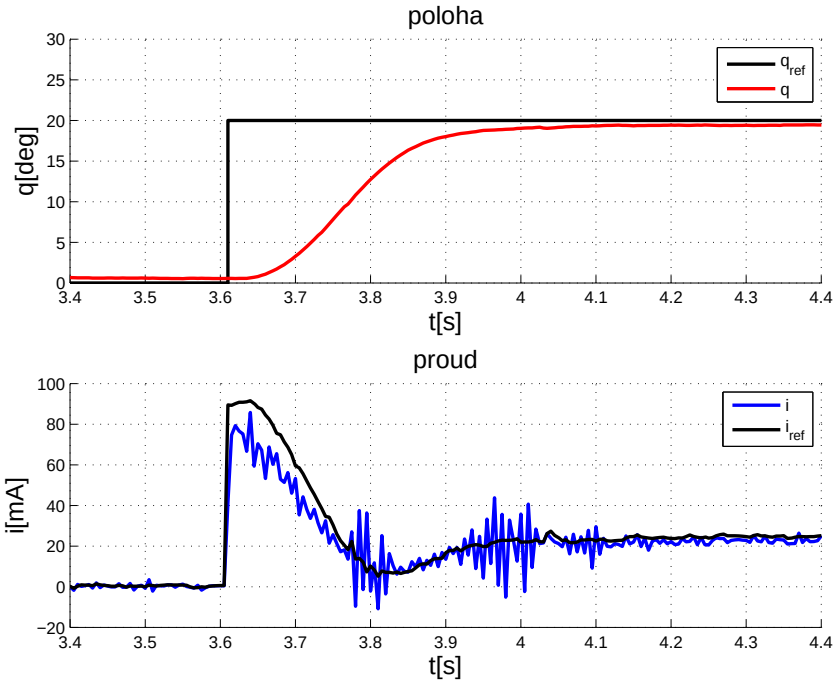
Obrázek 5.5: Kaskádní regulátor polohy pro pohybující se nohu

je použitý PID regulátor proudu popsaný v části 5.3.2. Proudový regulátor má stejné parametry, jako proudový regulátor v programech pro řízení horních kloubů robota, viz tabulka 5.1. Ve vnější smyčce je regulátor polohy. Výstupem PID regulátoru polohy je referenční hodnota proudu pro vnitřní regulátor. Použití kaskádního regulátoru je výhodné pro omezení proudových špiček při rozběhu motoru. Kaskádní regulátor také lépe reguluje pokrčení nohy robota na referenční polohu než samotný PID regulátor polohy, který jsem implementoval jako první typ regulátoru pro ohýbání kolene. Parametry vnějšího regulátoru polohy viz tabulka 5.3. Omezení referenční hodnoty proudu jsem nastavil na $\pm 210\,\mathrm{mA}$. Toto omezení zajistí ochranu motoru před přetížením.

K	5	N	10
T_i	100	$i_{ref\ low}$	-210mA
T_d	0.1	$i_{ref\ high}$	210mA
T_t	0.2	T_s	0.01s

Tabulka 5.2: Konstanty kaskádního regulátoru polohy

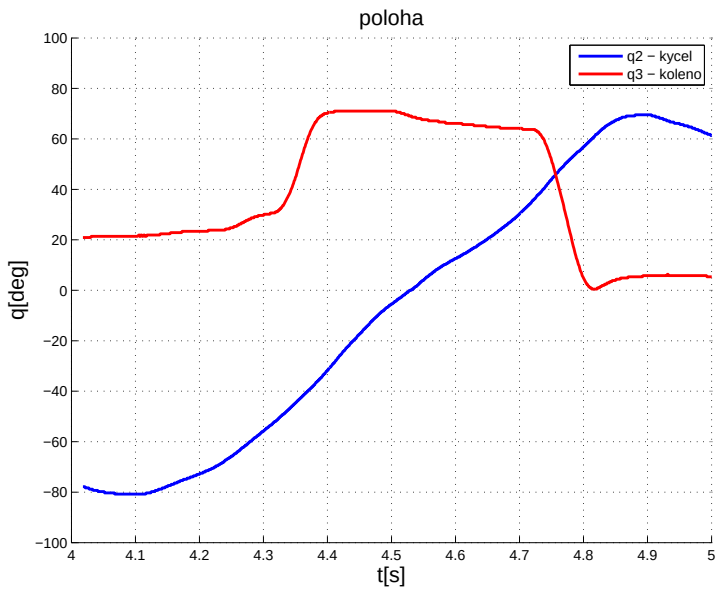
Na obr. 5.6 je znázorněna odezva kaskádního regulátoru polohy na změnu reference. V horní části obrázku je vidět pokrčení kolene do požadované polohy. V dolní části grafu je potom vidět proudová reference generovaná vnějším regulátorem polohy pro vnitřní regulátor proudu a reakce proudového regulátoru na tuto referenci.



Obrázek 5.6: Kaskádní regulátor polohy - odezva na změnu reference

Během kroku je třeba generovat referenční polohu pro regulátor pohybující se nohy. Program spodní desky robota přijímá po CAN sběrnici zprávy obsahující data ze senzorů horních desek robota. Mezi těmito daty také polohu v horních kloubech (kyčlích) robota měřenou pomocí potenciometrů. Z úhlů v kyčlích robota q_{left}, q_{right} program spočítá úhel mezi nohami robota $q_2 = q_{left} + q_{right}$. Z takto získané polohy nohou robota program spočítá referenci pro polohový regulátor kolene.

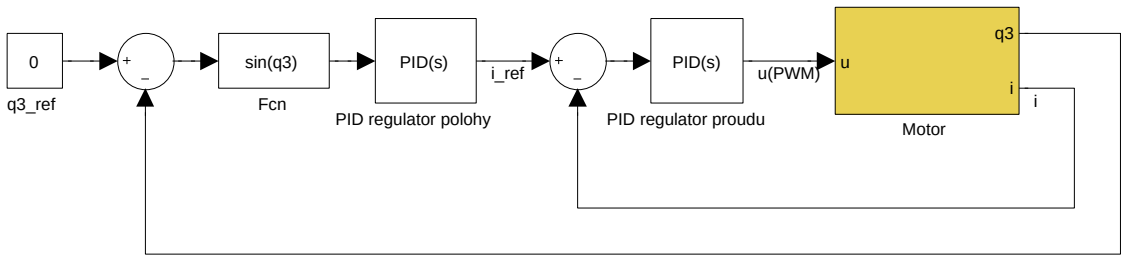
Na počátku kroku, když je pohybující se noha za stojící nohou ($q_2 < 0$), je reference nulová. V momentě, když se pohybující se noha přiblíží přes určitou mez k stojící noze, začne funkce implementovaná v programu zvětšovat referenci polohy. Tato reference je největší v momentě, když jedna noha míjí druhou. Jakmile se pohybují noha dostane před stojící nohu ($q_2 > 0$), začne se reference polohy opět snižovat až na nulovou hodnotu. Vše je dobře vidět na obr. 5.7, kde jsou znázorněny úhly v kyčlích robota q_2 a v koleni pohybující se nohy q_3 během jednoho kroku.



Obrázek 5.7: Pokrčování kolene během kroku - reakce na změnu úhlu v kyčlích robota

5.4.2 Nelineární kaskádní regulátor polohy

Dalším použitým regulátorem je regulátor pro řízení stojící nohy v napnutém stavu. Nejprve jsem chtěl pro tento účel použít stejný regulátor jako v části 5.4.1. To se ukázalo jako problém, protože zatížená noha nesoucí celou hmotnost robota má úplně jinou dynamiku, než nezatížená pohybující se noha. Kromě jiné dynamiky se zde mnohem více projeví nelinearita celého systému se kterou si lineární regulátor už nedokáže poradit.



Obrázek 5.8: Regulátor polohy a kompenzací nelinearity pro stojící nohu

Když jsem pro regulaci použil stejný kaskádní regulátor jako v podkapitole 5.4.1, nereagoval tento regulátor při pokrčení kolene dostatečně rychle. Následkem toho se koleno robota vlivem zatížení pokrčilo tolik, že motor ovládající pohyb kolene neměl sílu zpět napnout nohu robota. Když jsem na tuto situaci reagoval použitím regulátoru s větším zesílením, docházelo ke kmitání. Hlavním problémem zde bylo to, že točivý mo-

ment v koleni zatížené nohy roste nelineárně s rostoucím úhlem pokrčení podle (5.2), kde K je konstanta závisející na hmotnosti robota a délce nohy. U regulátoru pro nezatíženou pohybující se nohu to tolik nevadí z důvodu malé hmotnosti lýtka robota.

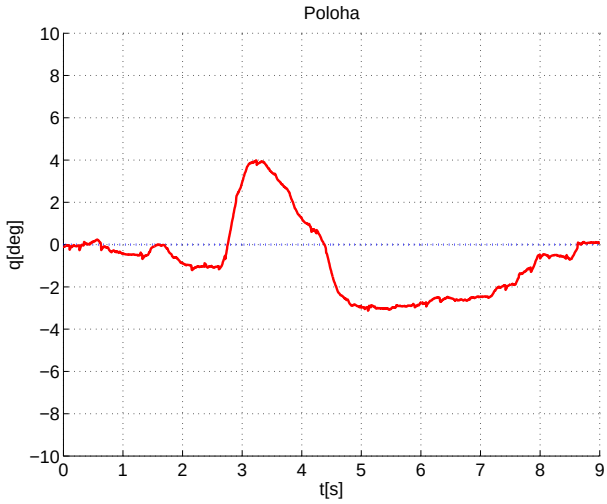
$$M = K \sin\left(\frac{\alpha}{2}\right) \tag{5.2}$$

Tuto nelinearitu jsem se v použitém regulátoru rozhodl kompenzovat. Do kaskádního struktury regulátoru na obr. 5.5 jsem přidal před vstup regulátoru polohy funkci, kompenzující nelinearitu (5.2). Pro regulátor s touto kompenzací jsem naladil nové parametry regulátoru, uvedené v tab 5.3.

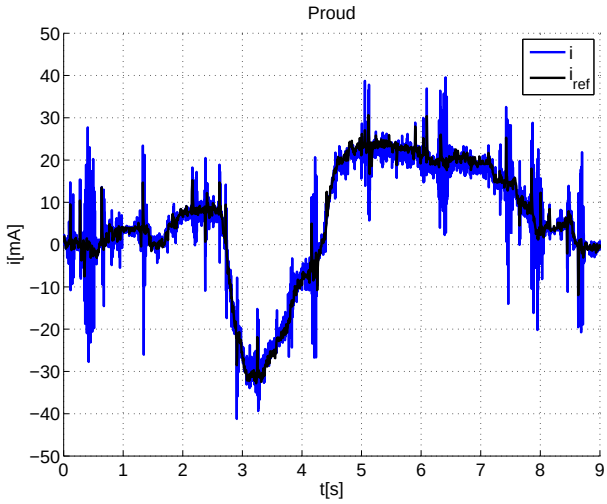
K	300	N	10
T_i	10000	$i_{ref\ low}$	-360mA
T_d	10	$i_{ref\ high}$	360mA
T_t	0.2	T_s	0.01s

Tabulka 5.3: Konstanty pro regulátor polohy stojící nohy

Na obr. 5.9 je znázorněn průběh regulace tímto regulátorem s kompenzací nelinearity. Referenční hodnota úhlu, na kterou se regulátor snaží dostat je vždy nulová (napnutá noha). Na obrázku je zobrazena regulace kolene v nataženém stavu při zátěži. Zde jsem při ladění regulátoru provedl experiment, kdy jsem robota postavil celou jeho hmotností na řízenou nohu a navíc ještě vrchní část modelu robota zatížil rukou. Tím jsem simuloval podmínky, kdy je stojící noha robota během chůze zatížena hmotností robota.



Obrázek 5.9: Regulace stojící nohy - poloha



Obrázek 5.10: Regulace stojící nohy - proud

Shrnutí kapitoly

V této kapitole jsem nejprve popsal koncepci řízení robota, které je rozděleno na hlavní regulátor, implementovaný v PC v Matlabu a potom na vnitřní regulační smyčky, distribuované v procesorech robota. Soustředil jsem se zde hlavně na popis programů pro procesory robota, které realizují vnitřní regulační smyčky. Vývojové diagramy, znázorňující činnost programů, jsou v příloze. Uvedl jsem zde také popis jednotlivých regulátorů, které jsem implementoval pro regulaci momentu a polohy. Pro regulaci polohy jsem použil kaskádní strukturu regulátorů. Všechny výše popsané regulátory jsem naladil a otestoval experimentálně přímo na modelu robota. Podrobnější informace o ladění kaskádních regulátorů lze nalézt v (Valentine, R., 1998). Pro programování v jazyce C jsem použil knihy (Herout, P., 2001*a*),(Herout, P., 2001*b*).

Kapitola 6

Komunikace s PC

V této kapitole je podrobně popsán způsob komunikace jednotlivých desek modelu robota s PC a také mezi sebou. Komunikaci mezi deskami a PC jsem realizoval pomocí sběrnice CAN, jak už bylo uvedeno v předchozích kapitolách. Popis hardwarové části sběrnice CAN a popis formátu zpráv je uveden v kapitole 4 v části 4.1.6. V této kapitole jsem se soustředil na popis komunikace ze softwarového hlediska a hlavně na podrobnější popis samotných zpráv. V podkapitole 6.2 je popsáno grafické uživatelské rozhraní, které jsem použil pro komunikaci s PC a pro diagnostiku. Způsob, jakým jsem realizoval řízení robota přímo z Matlabu v reálném čase, je popsán v podkapitole 6.3.

6.1 Komunikace po sběrnici CAN

Jednotlivé desky robota mezi sebou komunikují po sběrnici CAN. Tato komunikace je důležitá pro veškerou činnost robota. V části 6.1.1 je uveden stručný popis použitého CAN převodníku, zajišťujícího připojení PC ke sběrnici. V části 6.1.2 je potom podrobný popis jednotlivých zpráv. V tabulce 6.1 je uveden seznam všech používaných CAN zpráv. Pro snadnější označení jednotlivých desek v tabulce jsem použil níže uvedené zkratky:

- Deska ovládající levý dolní kloub robota (levé koleno) - **LD** (left down)
- Deska ovládající levý horní kloub robota (levý kyčel) - **LU**(left up)
- Deska ovládající pravý dolní kloub robota (pravé koleno) - **RD** (right down).
- Deska ovládající pravý horní kloub robota (pravý kyčel) - **RU** (right up).

6.1.1 Použitý CAN převodník

Pro připojení PC na sběrnici CAN je třeba použít nějaký CAN převodník. Já jsem se rozhodl pro použití převodníku Kvaser Leaf Light HS. Převodník umožňuje připojení PC k jednomu CAN kanálu pomocí rozšířeného rozhraní USB. K převodníku je na internetových stránkách firmy Kvaser zdarma k dispozici software *Kvaser Can King*.

Software *Kvaser Can King* umožňuje provádět základní diagnostiku sběrnice, jako zobrazení zpráv posílaných po sběrnici nebo kontrola zatížení sběrnice.

Parametry převodníku Kvaser Leaf Light HS:

- podpora USB rozhraní
- Vysílání a přijímání standardních i rozšířených CAN zpráv
- Rychlost přenosu 5 - 1000 kbit/s
- Maximální rychlost přenosu 8000 zpráv/s.

Obrázek 6.1: Kvaser Leaf Light HS (zdroj www.kvaser.com)

6.1.2 Popis zpráv sběrnice CAN

Zde je uveden podrobnější popis zpráv, které jsem použil pro komunikace mezi deskami robota. Všechny zprávy sběrnice je možné rozdělit do 3 základních skupin. První skupinou jsou zprávy posílané z PC do robota, druhou jsou zprávy posílané z robota do PC a poslední třetí skupinou zprávy pro komunikaci desek mezi sebou. Následuje podrobný popis těchto zpráv.

Zprávy posílané z PC do robota:
Do této skupiny patří zprávy provádějící změnu režimů činnosti, zprávy nesoucí informace o referencích pro vnitřní regulátory robota a zprávy pro nastavení nových parametrů regulátorů a filtrů.

Zprávy provádějící změnu režimů činnosti popsaných v části 5.1.1 jsou vysílány z PC buď pomocí GUI nebo pomocí Matlabu. Zpráva o změně režimu činnosti se neposílá periodicky, ale pouze jednorázově v momentě, kdy dochází ke změně režimu. Zpráva obsahuje pouze jeden datový byte, udávající režim činnosti podle tabulky. Jednotlivé režimy zde nebudu popisovat. Podrobný popis je v kapitole zabývající se programem robota v části 5.1.1. Jedná se o zprávy s ID = 0x201.

Mezi další typ použitých zpráv patří zprávy nesoucí informace o referencích pro vnitřní regulátory robota implementované v programech jednotlivých desek, tj. reference točivého momentu pro horní desky robota. Při ladění regulátorů robota jsem kromě referenčního momentu také posílal deskám reference pro polohu v jednotlivých kloubech. Pro regulaci chůze robota stačí vysílat pouze referenční moment pro horní desky. Tyto zprávy jsou vysílány z PC buď pomocí GUI nebo pomocí Matlabu. Zprávy nesoucí reference mají ID = 0x210 a obsahují celkem 8 datových bytů. Datová část zprávy může obsahovat celkem čtyři reference typu `int16`. Reference jsou vysílány periodicky podle vzorkovací frekvence hlavního regulátoru v Matlabu v PC.

Posledním typem zpráv z PC do robota jsou zprávy pro nastavení nových parametrů regulátorů a filtrů. Tyto zprávy jsou vysílány z PC pouze pomocí GUI a používají se pouze při ladění regulátorů a filtrů. Zprávy s novými parametry obsahují v datové části informace o desce, pro kterou jsou určeny a o regulátoru nebo filtru, pro který jsou určeny. Zpráva nesoucí nové parametry regulátorů má ID = 0x230. Zprávy vyžadující čtení parametrů určitého regulátoru nebo filtru určité desky mají ID = 0x235.

Zprávy vysílané z robota do PC:
Sem patří zprávy, zajišťující přenos všech dat potřebných pro řízení robota do PC. Patří sem zprávy s informacemi o úhlových rychlostech jednotlivých částí robota, o polohách ve všech kloubech robota, o proudech tekoucích do motorů robota a také zprávy s daty z laserového senzoru, určeného pro měření úhlu mezi zemí a stojící nohou robota. Jedná se zprávy uvedené v prvních 9 řádcích tabulky 6.1.

Zprávy posílané mezi deskami:
Do této skupiny patří pouze zprávy vysílané z horních desek a nesoucí informaci o poloze v kyčlích robota. Tyto zprávy jsou určeny pro desku právě se pohybující nohy. Algoritmus pro řízení pokrčování kolene pohybující se nohy má díky těmto zprávám informaci o úhlu

q_2, který mezi sebou svírají obě nohy robota. Zpráva s informací o poloze v levém horním kloubu q_{left} má ID = 0x142. Zpráva s informací o poloze v pravém horním kloubu q_{right} má ID = 0x144. Více informací o generování reference pro regulátory řídící pokrčování kolen na základě poloh v kloubech q_{left},q_{right} je uvedeno v kapitole 5 v části 5.4.1.

ID(hex)	ID(dec)	zdroj zprávy	popis
0x101	257	deska 1 - LD	úhlová rychlost
0x102	258	deska 2 - LU	úhlová rychlost
0x103	259	deska 3 - RD	úhlová rychlost
0x104	260	deska 4 - RU	úhlová rychlost
0x141	321	deska 1 - LD	poloha, proud odebíraný motorem
0x142	322	deska 2 - LU	poloha, proud odebíraný motorem
0x143	323	deska 3 - RD	poloha, proud odebíraný motorem
0x144	324	deska 4 - RU	poloha, proud odebíraný motorem
0x108	264	deska 3 - RD	úhel změřený pomocí laseru
0x236	566	všechny desky	vyslání parametrů konkrétní desky do PC
0x201	513	PC	nový režim činnosti
0x230	560	PC	nastavení nových parametrů pro konkrétní desku
0x235	565	PC	požadavek na čtení parametrů konkrétní desky
0x210	528	PC	referenční data pro regulátory jednotlivých desek

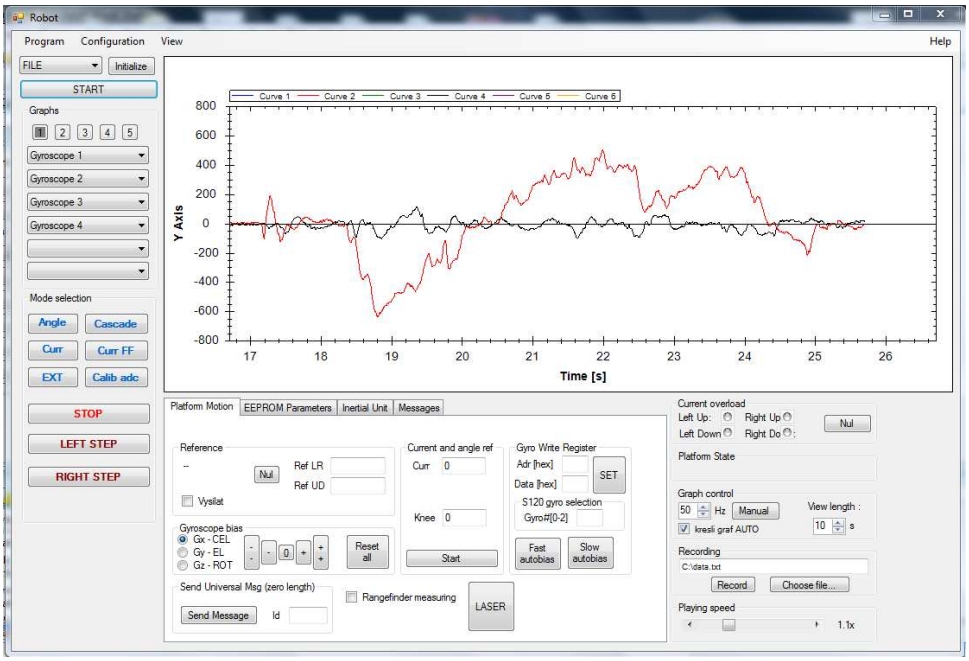
Tabulka 6.1: Seznam zpráv sběrnice CAN

6.2 GUI

V této podkapitole je popsáno grafické uživatelské rozhraní (GUI), které jsem použil pro zobrazování průběhů měřených veličin, ladění regulátorů a ovládání robota z PC. Toto GUI jsem nepoužil pro zpětnovazební řízení chůze robota. Řízení chůze robota jsem realizoval v Matlabu-Simulinku a je popsáno v části 6.3. GUI jsem používal pouze pro vývoj, diagnostiku a testování robota.

Jako základ pro GUI jsem použil uživatelské rozhraní od kolegů z katedry řídicí techniky, které používají v projektu *Stabilizované kamerové základny*. Jejich uživatelské rozhraní jsem si dále upravil pro své potřeby, abych jej mohl použít pro komunikaci

s robotem. GUI je realizované v jazyce C# v Microsoft Visual Studiu a je vytvořeno jako samostatná aplikace.

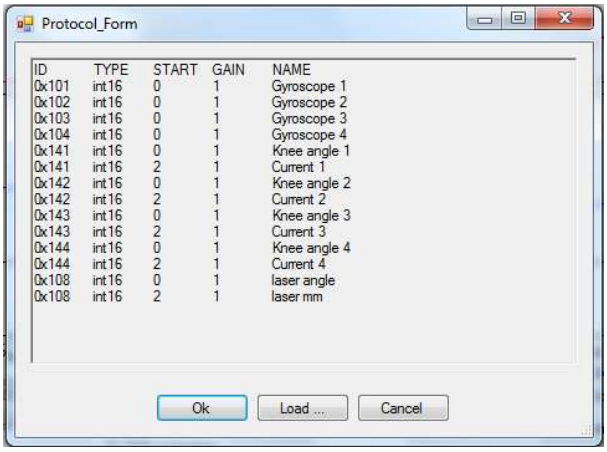


Obrázek 6.2: Grafické uživatelské rozhraní

Po spuštění se zobrazí hlavní okno aplikace, které je možné vidět na obr. 6.2. V záložce *Configuration* se nejprve nastaví parametry sběrnice CAN (rychlost, použitý kanál) a protokol se seznamem použitých zpráv viz obr. 6.3. Poté je možné začít pracovat v jednom ze dvou hlavních režimů. Buď je možné online komunikovat po sběrnici CAN s robotem a nebo offline přehrávat dříve naměřená data. Mezi těmito dvěma režimy je možné přepínat v záložce FILE/CAN v levé horní části okna. Na obr. 6.2 je nastaven režim FILE pro přehrávání dříve změřených dat. Vedle této záložky se nachází tlačítko *Initialize* pro počáteční inicializaci CAN sběrnice a pod ním tlačítko START/STOP pro spuštění a zastavení komunikace. Pod těmito tlačítky se nachází záložky pro nastavení dat zobrazovaných na grafu, který je hlavní součástí GUI. Pomocí těchto záložek je možné nastavit vykreslování až šesti možných průběhů. Na obrázku je znázorněn průběh úhlových rychlostí čtených z gyroskopů.

V levé dolní části grafu se nachází tlačítka pro přepínání mezi režimy činnosti robota popsanými v kapitole 5 v části 5.1.1. V dolní části okna aplikace pod grafem se nachází záložky pro nastavování referencí a ladění vnitřních regulátorů robota. Nejdůležitější jsou první dvě záložky *Platform motion* a *EEPROM Parameters*.

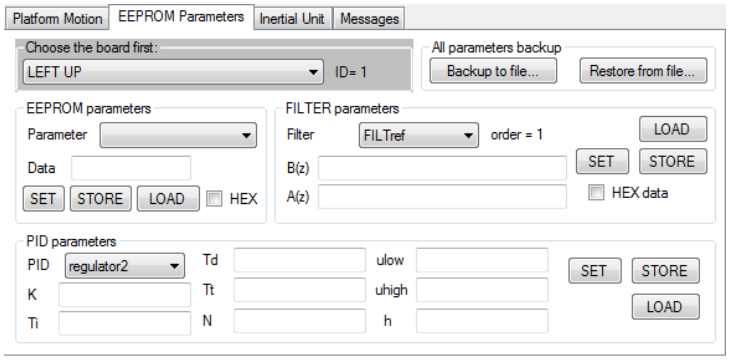
První záložka *Platform motion* slouží pro nastavování referencí pro regulátor proudu



Obrázek 6.3: Protokol přijímaných CAN zpráv

(položka *curr*) a regulátor polohy (položka *knee*). Dále se zde nachází tlačítka pro nastavení rozsahu měření a offsetu gyroskopu. První záložka je vidět na obrázku celé aplikace obr. 6.2.

Druhá záložka *EEPROM Parameters* znázorněná na obr. 6.4 slouží pro ladění vnitřních regulátorů a filtrů a pro ukládání parametrů do externí EEPROM paměti. Nejprve je třeba vybrat desku, do které se budou ukládat nová data. To se provede vybráním příslušné desky v levé horní části této druhé záložky (*Choose the board first*). Na obr. 6.4 je takto vybrána deska *LEFT UP*. Dále je možné načíst z vybrané desky parametry vybraných regulátorů a filtrů pomocí tlačíka LOAD. Pro nastavení a uložení nových parametrů stačí pouze napsat nové hodnoty a ty nahrát do RAM pamětí procesoru vybrané desky pomocí tlačítka SET.



Obrázek 6.4: Formulář pro ladění regulátorů a filtrů

6.3 Komunikace s Matlabem

V této podkapitole je popsáno, jak jsem realizoval propojení robota po sběrnici CAN s Matlabem/Simulinkem pro použití při real-time řízení. V této podkapitole není uveden popis hlavního regulátoru pro řízení chůze, ale pouze princip propojení s tímto regulátorem. Hlavní regulátor a pozorovatel je popsán v kapitole 7. Jak už bylo uvedeno v předcházejících kapitolách, hlavní regulátor a pozorovatel stavů, který zajišťuje řízení chůze robota, stabilizaci robota a odhad neměřených stavů, je kvůli velké náročnosti na výpočetní výkon realizován v Matlabu na PC. Pro realizaci real-time řízení bylo třeba najít způsob, jak zajistit komunikaci mezi Matlabem a samotným robotem. Navíc bylo nutné zajistit, aby algoritmus v Matlabu pracoval v reálném čase. Nabízely se dvě cesty, jak toto řízení v reálném čase realizovat. První možností bylo použít Real-Time Workshop + Real-Time Windows Target. Druhou možností bylo použít Real-Time Toolbox firmy Humusoft.

6.3.1 Způsob komunikace

Pro řízení robota z Matlabu v reálné čase jsem zkoušel použít dva různé přístupy. Použít buď Real-Time Windows Target nebo Real-Time Toolbox. Obě možnosti jsem testoval a objevil jejich výhody a nevýhody.

Real-Time Workshop + Real-Time Windows Target:
Během vývoje softwaru robota jsem nejprve zkoušel použít pro řízení robota Real-Time Windows Target. Real-Time Windows Target je software, umožňující v počítači spouštět modely vytvořené v Simulinku v reálném čase. Práce s tímto softwarem probíhá tak, že je třeba nejprve pomocí Simulink kodéru (Real-Time Workshop Simulink Coder) přeložit simulinkový model do jazyka C. Poté se vygenerovaný kód nahraje do Real-Time Windows Targetu, kde se spustí. Takto spuštěný kód potom běží v PC jako proces s nejvyšší prioritou. Tato vysoká priorita zajistí, že algoritmus vytvořený v simulinku probíhá v reálném čase, což je výhodou tohoto přístupu. Velikou nevýhodou ovšem je, že při provádění velmi složitých výpočtů může nastat kolaps operačního systému vlivem nedostatku operační paměti. Procesy Real-Time Windows Targetu mají totiž vyšší prioritu než operační systém počítače. Další nevýhodou je poměrně omezené množství periferií, s kterými je schopen Real-Time Windows Target pracovat. Není zde např. podpora funkcí pro použití CAN sběrnice. Proto jsem při testování řízení robota pomocí tohoto softwaru používal pouze propojení přes UART. Také je velmi omezené množství funkcí, které je

možné použít v simulinkovém modelu, ze kterého se potom generuje kód pro Real-Time Windows Target. Real-Time Windows Target podporuje pouze bloky ze základní knihovny Simulinku. Bloky a funkce z většiny toolboxů nepodporuje.

Real-Time Toolbox:
Druhý způsob testovaného real-time řízení byl pomocí Real-Time Toolboxu firmy Humusoft. Při použití tohoto toolboxu stačí vložit do simulinkového modelu blok *RTSync*, který zajistí vykonávání výpočtů v reálném čase s pevně zadanou periodou. Dále je třeba správně nastavit konfigurační parametry použitého modelu. Výhodami je možnost použít libovolné funkce a bloky z Matlabu-Simulinku, podpora většiny toolboxů a periferií (včetně CAN) a také to, že pro řízení robota byla postačující demoverze Real-Time Toolboxu, která je zdarma k dispozici na webových stránkách firmy Humusoft. Jedinou nevýhodou tohoto přístupu je menší garance provedení výpočtu v reálném čase.

Po několika experimentech s oběma typy real-time řízení jsem se nakonec rozhodl použít druhý přístup, tj. použít **Real-Time Toolbox**. Pro toto rozhodnutí jsem měl několik důvodů. Hlavním důvodem byla stabilita operačního systému. Při testování řízení robota pomocí Real-Time Windows Targetu docházelo velmi často ke kolapsu OS. Také by nebylo možné použít komunikaci mezi PC a robotem po sěrnici CAN. Jedinou možností bylo při použití Real-Time Windows Targetu propojení po sériové lince.

6.3.2 Rozhraní pro real-time řízení

Zde je popsáno rozhraní, které jsem vytvořil pro propojení robota s Matlabem/Simulinkem pro řízení robota v reálném čase. Při vývoji tohoto rozhraní jsem potřeboval vyřešit způsob komunikace Matlabu po sběrnici CAN a také způsob jak zajistit, aby algoritmus pro řízení robota probíhal v reálném čase. Rozhraní pro real-time řízení jsem realizoval v Simulinku jako samostatné bloky pro přijímání zpráv z robota a pro vysílání zpráv pro robota. Tyto bloky je možné vložit do jakéhokoliv simulinkovského modelu a propojit s libovolným regulátorem, který může běžný uživatel v simulinku lehce sestrojit.

6.3.2.1 Real-Time Toolbox

Vykonávání regulátoru v reálné čase je zajištěno pomocí Real-Time toolboxu. Z tohoto toolboxu stačí použít pouze blok *RT Sync*, který je třeba vložit do použitého modelu, který se má simulovat v reálném čase. Pro zajištění real-time chodu modelu je třeba v bloku *RT Sync* nastavit vzorkovací frekvenci. Tuto frekvenci je nutné správně nastavit

vzhledem k náročnosti výpočtů použitého modelu a výpočetního výkonu PC. Při nastavení větší frekvence, než je počítač schopen zvládnout, dojde k tomu, že Matlab nestihne provést potřebné výpočty v požadovaném čase. V mém případě jsem pro řízení robota pomocí nelineárního regulátoru byl nucen vzorkovací frekvenci omezit na 200 Hz. V bloku lze také nastavit maximální počet "nestihnutých kroků"(Maximum tick missed). Pokud Matlab nestihne provést určitý počet kroků, který se na nastaví v proměnné `maximum tick missed`, tak se simulace ukončí.

Dále je třeba pro vykonávání kódu v reálném čase správně nastavit konfigurační parametry modelu v záložce *Simulation − > Configuration Parameters*. Zde je třeba nastavit v části *Solver* parametry:

- Type: Fixed-step
- Solver: ode3(Bogacki-Shampine)
- Fixed-step size: Auto

Při tomto nastavení zajistí Real-Time toolbox vykonávání kódu v reálném čase s periodou, která je nastavena v bloku *RT Sync*.

6.3.2.2 Rozhraní pro komunikaci pomocí CAN

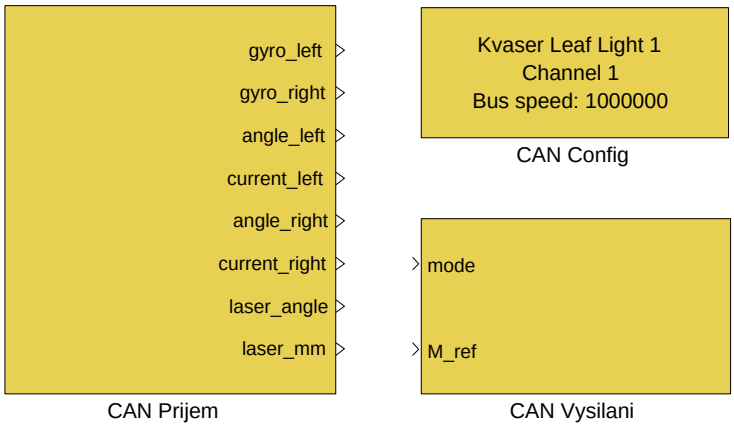
Zde je uveden popis rozhraní, které jsem vytvořil pro komunikaci Matlabu s robotem po sběrnici CAN. Rozhraní je vytvořeno jako samostatné bloky pro Simulink a skládá se z bloku pro příjem zpráv (CAN Prijem), z bloku pro vysílání zpráv (CAN Vysilani) a bloku pro nastavení CAN sběrnice (CAN Config). Při tvorbě tohoto rozhraní jsem použil *Vehicle Network Toolbox*, který obsahuje funkce pro práci s CAN sběrnicí. Následuje popis jednotlivých bloků.

CAN Config

V tomto bloku se nastaví parametry CAN sběrnice. Nastaví se zde rychlost sběrnice a použitý kanál.

CAN Prijem

Tento blok zajišťuje přijímání dat z desek robota do Simulinku. Jednotlivé desky robota vysílají cyklicky každých 5 ms změřená data ze senzorů. Blok *CAN Prijem* nejprve pomocí funkce *CAN Receive* přijme všechny nové CAN zprávy. Tyto zprávy jsou následně pomocí funkce *CAN Unpack* převedeny na konkrétní data. Funkce *CAN Unpack* se volá několikrát pro jednotlivé zprávy. Pomocí parametrů ID a DLC, které udávají identifikátor (ID) konkrétní zprávy (viz tabulka 6.1) a dále délky datové části zprávy(DLC) a zadaných



Obrázek 6.5: Rozhraní pro komunikaci po sběrnici CAN

datových typů funkce *CAN Receive* pozná, o jaká data se jedná a ta potom převede na datový typ `double`. Následně je uloží do konkrétní proměnné. U všech přijímaných dat jsem navíc nastavil ukládání do Workspace Matlabu, pro možnost pozdějšího použití.

CAN Vysilani

Tento blok zajišťuje vysílání dat ze Simulinku do desek robota. Blok *CAN Vysilani* pracuje přesně naopak než blok *CAN Prijem*. Blok nejprve převede data z typu `double` na `int16`. Potom pomocí funkce *CAN Pack* z dat sestaví celkem dvě CAN zprávy, kterým přiřadí ID z tabulky 6.1. Jedná se o zprávu udávající režim činnosti robota a pak zprávu s referencemi pro vnitřní regulátory robota. Tyto zprávy následně vyšle pomocí funkce *CAN Transmit*.

Shrnutí kapitoly

V kapitole byla nejprve popsána komunikace mezi PC a jednotlivými deskami robota po sběrnici CAN včetně podrobného popisu CAN zpráv. Dále zde bylo popsáno grafické uživatelské rozhraní, které jsem použil pro ladění regulátorů implementovaných v procesorech robota a pro zobrazování měřených stavů. Nakonec jsem zde popsal způsob, jakým jsem realizoval řízení robota z Matlabu v reálném čase a rozhraní, které jsem vytvořil pro propojení robota s Matlabem/Simulinkem. V budoucnu je díky propojení s Matlabem možné, aby i uživatel bez zkušeností s elektronikou a programováním mikroprocesorů mohl vyvíjet nové algoritmy pro řízení modelu kráčejícího robota.

Kapitola 7

Experimentální porovnání pozorovatelů

V této kapitole jsem popsal testování řízení chůze robota z Matlabu. Zaměřil jsem se hlavně na otestování dvou typů pozorovatelů, určených pro odhadování stavů Acrobota. Jedním ze zadaných cílů mé práce bylo otestovat dodané algoritmy pro řízení chůze Acrobota a odhadování stavů. Od vedoucího práce jsem dostal k dispozici v Matlabu vytvořený model simulující řízení Acrobota. Součástí modelu byl matematický model samotného Acrobota a pro něj navržený regulátor a pozorovatel. Jedná se o algoritmy popsané v kapitole 2 a otestované zatím pouze v simulacích, ale ne na reálném systému. Mým úkolem bylo tyto algoritmy otestovat na reálném modelu. Na reálném modelu jsem testoval celkem dva typy pozorovatelů stavů. Prvním typem je high gain pozorovatel teoreticky popsaný v kapitole 2 v části 2.3. Otestování tohoto pozorovatele je popsáno v části 7.3. Druhým typem je redukovaný pozorovatel, teoreticky popsaný v kapitole 2 v části 2.4. Experimentální ověření je v této kapitole v části 7.4. Experimentální ověření pro regulátor a oba typy pozorovatelů jsem prováděl pouze na jednom kroku robota.

7.1 Popis řídícího algoritmu

V této podkapitole je uveden stručný popis hlavního řídícího algoritmu. Na obr. 7.1 je zjednodušené schéma hlavního algoritmu pro řízení chůze robota. Skutečný regulátor a pozorovatel, vytvořený v Simulinku, je velmi nepřehledný. Pro lepší přehlednost jsem uvedl pouze principiální schéma nakreslené v Simulinku. Skutečný řídicí algoritmus vytvořený

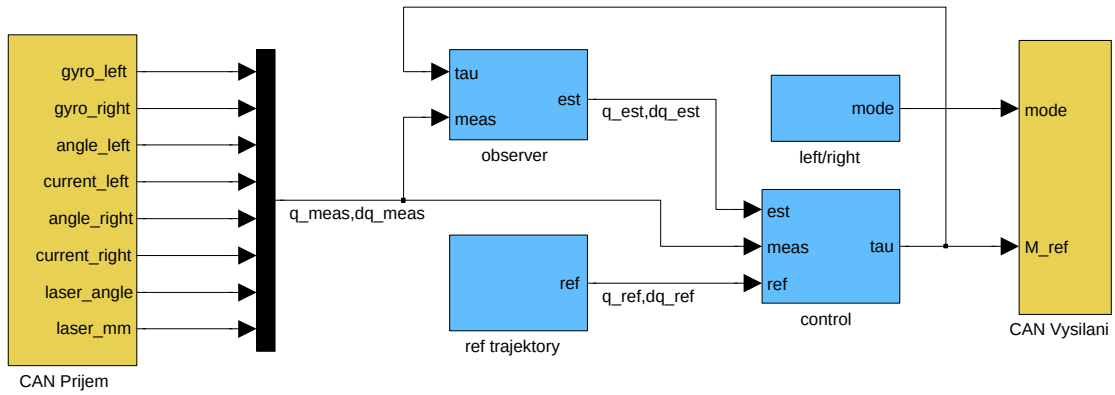
v Simulinku je na přiloženém CD. Algoritmus pro řízení chůze se skládá z několika částí:

Rozhraní pro CAN: Toto rozhraní zajišťuje komunikaci s reálným modelem robota a samostatně jsem ho popsal v části 6. Jsou to bloky *CAN Prijem* a *CAN Vysilani*.

Referenční systém: Tento systém generuje referenční trajektorii chůze a je popsaný v kapitole 2 v části 2.5. Referenční systém generuje referenční hodnoty pro všechny stavy robota, tj. q_{1ref}, q_{2ref}, $\dot{q}_{1ref}$, $\dot{q}_{2ref}$. Jedná se o blok *Ref trajektory* na obr. 7.1.

Nelineární regulátor: Cílem regulátoru je dosáhnout referenční trajektorie. Na základě znalosti stavů systému q_1, q_2, $\dot{q}_1$, $\dot{q}_2$ a referenčních hodnot q_{1ref}, q_{2ref}, $\dot{q}_{1ref}$, $\dot{q}_{2ref}$ regulátor vypočítá potřebný akční zásah (točivý moment) pro motory ovládájících pohyb v horních kloubech robota (kyčlích). Jedná se nelineární regulátor označený jako *Control* na obr. 7.1 a popsaný v kapitole 2.

Nelineární pozorovatel: Pozorovatel na základě měřených stavů a točivého momentu odhaduje neměřené stavy. Pro odhad stavů může být použit high gain pozorovatel, který na základě měřených stavů q_2 a $\dot{q}_1$ odhaduje stavy q_1 a $\dot{q}_2$ a nebo redukovaný pozorovatel, který na základě měřených stavů q_1 a q_2 odhaduje stavy $\dot{q}_1$ a $\dot{q}_2$. Tyto pozorovatele a jejich otestování jsem podrobně popsal v samostatných podkapitolách 7.3 a 7.4.



Obrázek 7.1: Regulátor a pozorovatel stavů použitý pro řízení chůze robota

7.2 Průběh experimentu

V této podkapitole je uveden popis experimentu, který jsem provedl pro ověření regulátoru a pozorovatele stavů. Experiment jsem prováděl vždy pouze pro jeden krok robota. Robot je připraven provádět více kroků. Chůze robota s více kroky zatím není realizovaná z následujících důvodů:

- Redukovaný pozorovatel, popsaný dále v podkapitole 7.4, potřebuje laserový senzor vzdálenosti pro měření úhlu q_1. Tento senzor, který je upevněný na stojící noze, jsem měl k dispozici pouze jeden. Pro to jsem mohl provádět pro ověření tohoto pozorovatele pouze jeden krok.
- Z důvodu konečné délky kabelů (externí napájení a řízení pomocí PC).
- Robot se dokáže stabilizovat pouze ve 2-D rovině (vpřed-vzad a nahoru-dolů, ale ne vlevo-vpravo). Pro realizaci chůze je třeba vyrobit nějaký typ základny, která bude umožňovat volný pohyb robota ve 2-D rovině a zároveň ho bude stabilizovat tak, aby nespadl na stranu.
- Není sestaven model impaktu[1] pro případ reálného robota, který je nutný pro výpočet vícekrokové trajektorie.

Jak už jsem uvedl, pro budoucí chůzi s více kroky je robot připraven. Pro více kroků stačí pouze přepínat mezi režimy LEFT-STEP a RIGHT-STEP a posílat deskám robota referenční moment pro horní klouby. Pro přepínání mezi režimy je třeba detekovat konec kroku, kdy robot dopadne pohybující se nohou na zem. To je možné realizovat dvěma způsoby, případně kombinací obou. První možný způsob detekce je pomocí nespojitosti úhlové rychlosti. Při dopadu nohy na zem nastane skoková změna úhlové rychlosti $\dot{q}_1$ měřené pomocí gyroskopu. Druhým způsobem je dopad nohy robota detekovat ze znalosti úhlů q_1 a q_2. Úhel q_2 vždy měřen. Úhel q_1 je buď měřen přímo laserovým senzorem a nebo je odhadován pomocí pozorovatele. Ze znalosti těchto úhlů a ze znalosti délek nohou robota je pak možné pomocí trigonometrie spočítat okamžik, kdy noha robota dopadne na zem.

[1] Situace při dopadu nohy robota, kdy při dopadu pohybující se nohy robota na zem prudce poklesnou úhlové rychlosti robota. Polohy v jednotlivých kloubech se nemění. Namodelování tohoto jevu, který se nazývá impakt, je nutné pro realizaci vícekrokové trajektorie. Model impaktu je zobrazení $[\dot{q}^+] = \Phi_{Imp}[\dot{q}^-]$, které na základě stavů acrobota na konci jednoho kroku vypočítá počáteční stavy acrobota na počátku dalšího kroku. Více viz (Anderle, M. and Čelikovský, S., 2010*d*)

Pro experimentální ověření chůze robota jsem vytvořil k tomu zvlášť určený Simulinkový model. Jako základ jsem použil model, který jsem dostal od vedoucího a který byl určen pouze pro simulace. Pomocí rozhraní, které jsem vytvořil pro ovládání fyzického modelu robota v reálné čase, jsem potom ovládal chůzi robota přímo z Matlabu.

Nejprve jsem s pomocí vedoucího připravil model robota do výchozí pozice, kdy byla jedna noha robota vpředu a druhá vzadu. Během začátku kroku bylo třeba model robota držet, aby neupadl. Také bylo nutné nastavit co nejpřesněji počáteční podmínky. Jako náročné se ukázalo nastavení správných počátečních podmínek pro úhlové rychlosti robota. Referenční trajektorie je totiž modelována jako periodická funkce pro stav, kdy se robot již pohybuje a provádí jednotlivé kroky. Tím pádem jsou reference na počátku kroku pro úhlové rychlosti nenulové. Stejně bylo nutné po konci kroku robota chytit. Z toho důvodu, že jsem měl k dispozici pouze jeden laserový senzor, jsem experiment prováděl vždy tak, že robot prováděl krok levou nohou vpřed. Robot tak pracoval v režimu LEFT-STEP. Na začátku kroku byla levá noha vzadu a pravá v předu. Obě kolena robota byla propnutá. Poté jsem v Simulinku spustil připravený model. Robot následně provedl krok levou nohou vpřed a dopadl na ni. Během kroku jsem měřil data ze všech senzorů. S pomocí vedoucího jsme provedli větší množství těchto experimentů, kdy jsme testovali oba typy pozorovatelů. Výsledky experimentů pro oba pozorovatele jsou napsané níže.

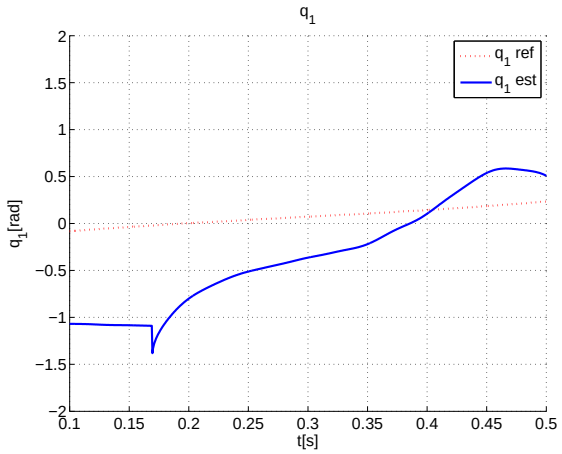
7.3 High gain pozorovatel

Zde je popsáno testování high gain pozorovatele, který na základě měřených stavů q_2 a $\dot{q}_1$ odhaduje stavy q_1 a $\dot{q}_2$. Princip High gain pozorovatele je teoreticky popsaný v kapitole 2 v části 2.3. Při použití tohoto typu pozorovatele stavů je úhel q_2 měřen pomocí potenciometru a úhlová rychlost $\dot{q}_1$ pomocí gyroskopu. High gain pozorovatel, implementovaný v Matlabu, má strukturu (2.10). Zesílení $L_{1,2,3,4}$ je možné navrhnout pomocí tzv. "high-gain"postupu, pokud se vezme jakékoliv $\widetilde{L}_{1,2,3,4}$ tak, aby matice (2.12) byla Hurwitzovská. Pomocné zesílení $\widetilde{L}_{1,2,3,4}$ a parametr Θ jsem po několika experimentech zvolil

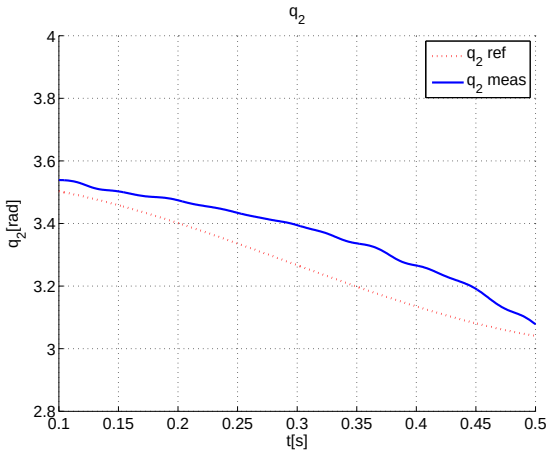
$$\widetilde{L}_{1,2,3,4} = \begin{pmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{pmatrix} = \begin{pmatrix} -46 \\ -790 \\ -6030 \\ -17000 \end{pmatrix}, \quad \Theta = 2. \tag{7.1}$$

Pro testování pozorovatele na kroku robota jsem pro hlavní regulátor, popsaný v části 7.1 zvolil frekvenci vzorkování 200Hz. Tato frekvence je použita pro výpočet hlavního nelineárního regulátoru a pozorovatele a také pro komunikaci po sběrnici CAN. Pro vyšší frekvenci není počítač schopen stihnout výpočet regulátoru. Tato vzorkovací frekvence by měla být co největší. Pro nižší frekvence není schopen pozorovatel odhadovat neměřené stavy.

Dále se objevil problém u měřených stavů. Pozorovatel typu high gain je totiž velmi citlivý na šum a na skokové změny měřených veličin. Proto jsem musel udělat úpravu v hlavním regulátoru a měřené stavy q_2 a $\dot{q}_1$ filtrovat a teprve filtrované hodnoty použít pro výpočet pozorovatele. Provedl jsem několik experimentů a kroků s robotem, kdy jsem měřil a ukládal potřebná data a testoval pozorovatel stavů společně s regulátorem. Níže jsou na obr. 7.2, 7.3, 7.4 a 7.5 průběhy měřených a odhadovaných veličin.

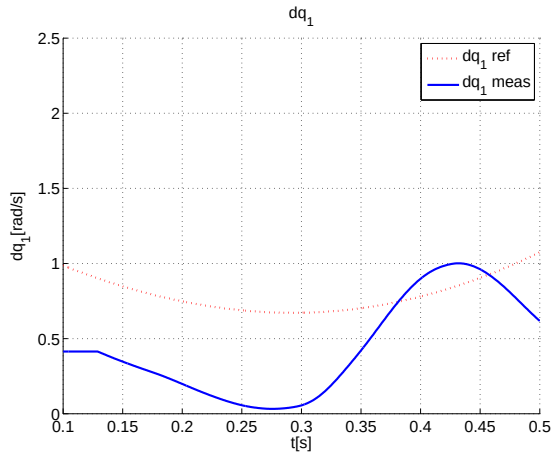


Obrázek 7.2: High gain pozorovatel - odhadovaný úhel q_1

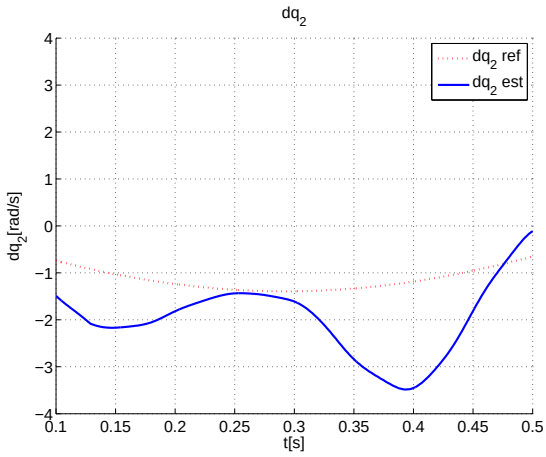


Obrázek 7.3: High gain pozorovatel - měřený úhel q_2

Největším problémem se u tohoto typu pozorovatele během testování ukázala citlivost na správný model systému. Matematický model, podle kterého pozorovatel odhaduje neměřené stavy, by měl být co nejpřesněji popsaný a všechny parametry by měly být co nejlépe identifikované. Pokud se matematický model odhadovaného systému mírně liší od reálného systému, pozorovatel potom při použití vyšších hodnot parametru Θ odhaduje nesmyslné hodnoty. Proto jsem musel použít nižší hodnotu tohoto parametru, než je možné u simulace. Na druhou stranu by měla být hodnota tohoto parametru co možná nejvyšší. Při nízké hodnotě je potom odhad nepřesný, případně při špatných počátečních podmínkách dlouho trvá, než pozorovatel dosáhne správného odhadu. Při experimentech jsem došel k závěru, že kvůli výše uvedeným důvodům **pro odhad stavů kráčejícího**



Obrázek 7.4: High gain pozorovatel - měřená úhlová rychlost $\dot{q}_1$



Obrázek 7.5: High gain pozorovatel - odhadovaná úhlová rychlost $\dot{q}_2$

robota high gain pozorovatel není příliš vhodný.

V grafech jsou vidět průběhy všech stavů během jednoho kroku robota. Ve stejných grafech jsou také zobrazeny referenční hodnoty těchto stavů, kterých se regulátor během kroku robota snaží dosáhnout. Skutečné hodnoty se od referenčních výrazně liší z několika důvodů. Prvním důvodem je omezení akčního zásahu motoru[2] a tím pádem maximálního možného točivého momentu na cca $0.4Nm$.

Druhým důvodem je potom neshoda počátečních podmínek referenčního modelu a fyzického modelu. Referenční model totiž generuje referenční trajektorii pro situaci, kdy robot již kráčí. Proto jsou, jak jsem již uvedl výše, referenční hodnoty úhlových rychlostí na počátku kroku robota nenulové (z pohledu referenčního modelu robot právě dokončil krok). Nastavit správně při experimentu počáteční podmínky reálného modelu robota pro tyto nenulové rychlosti je obtížné. Z tohoto důvodu se počáteční podmínky pro oba modely tolik liší. Kvůli výše popsaným důvodům nestihne robot dosáhnout během jednoho kroku shody mezi referenčními a skutečnými hodnotami regulovaných stavů.

[2]Pokud toto omezení nastavím v simulaci, kde jsou ideální podmínky a ne na reálném modelu, kde jsou šumy měření a omezené vzorkování, tak i v simulaci se během prvního kroku robota stavy velmi liší od referenčních stavů. V průběhu dalších kroků se konvergence k referenční trajektorii zlepšuje a až po několika krocích skutečná trajektorie přesně sleduje referenční trajektorii.

7.4 Redukovaný pozorovatel

V této podkapitole je popsáno testování redukovaného pozorovatele. Tento typ pozorovatele na základě měřených stavů (úhlů) q_1 a q_2 odhaduje stavy (úhlové rychlosti) $\dot{q}_1$ a $\dot{q}_2$. Pozorovatel je popsaný v kapitole 2 v části 2.4. Tento pozorovatel jsem testoval stejným způsobem, jako výše popsaný high gain pozorovatel a to pouze na jednom kroku robota. Testování na jednom kroku bylo také z důvodu, že jsem měl k dispozici pouze jeden laserový senzor vzdálenosti. Při použití tohoto pozorovatele je úhel $\dot{q}_1$ měřen nepřímo pomocí laserového senzoru vzdálenosti podle vztahu (2.14). Úhel $\dot{q}_2$ je měřen přímo pomocí potenciometru. Úhlové rychlosti $\dot{q}_1$ a $\dot{q}_2$ jsou odhadovány.

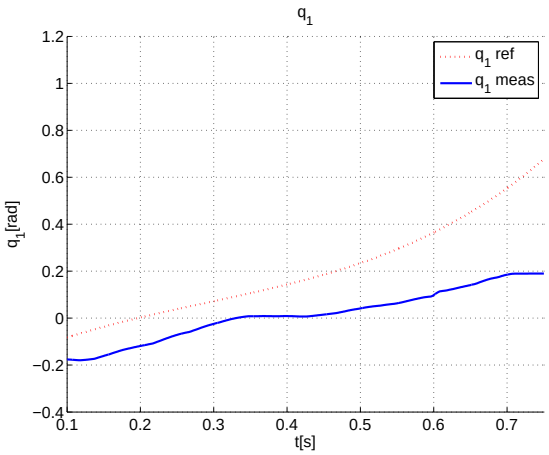
Redukovaný pozorovatel má jinou strukturu než high gain pozorovatel a skládá se ze dvou skalárních diferenciálních rovnic (2.15), (2.18). Výhodou této struktury je to, že můžeme použít malé parametry zesílení $k_{2,4}$. Tyto parametry jsem po několika testech zvolil $\mathrm{k}_{2,4} = 12$. Pro tento pozorovatel jsem provedl také množství experimentů na několika krocích robota. Vzorkovací frekvence hlavního regulátoru byla také 200 Hz jako u předchozího typu pozorovatele. Pozorovatel je méně citlivý na šum měření než výše popsaný high gain pozorovatel. Pro jistotu jsem měřené stavy také filtroval, jako u high gain pozorovatele. Redukovaný pozorovatel je také méně citlivý na přesný matematický model systému oproti high gain pozorovateli. Níže jsou na obr. 7.6, 7.7, 7.8 a 7.9 průběhy měřených a odhadovaných veličin.

Regulované stavy robota se liší od referenčních ze stejných důvodů jako v předcházející části u high gain pozorovatele a to kvůli omezení akčního zásahu, který dokáže použitý motor vyvinout a také kvůli neshodě mezi počátečními podmínkami referenčního modelu a reálného modelu robota na začátku kroku.

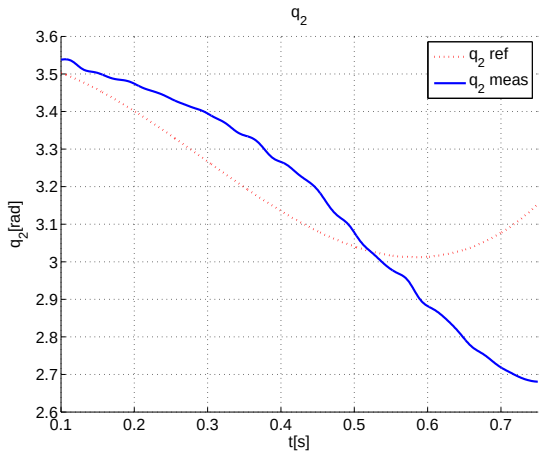
Z grafů je vidět o něco lepší průběh měřených a odhadovaných stavů vzhledem k referenčním hodnotám než u high gain pozorovatele. Při testování jsem došel k závěru, že **pro odhadování stavů kráčejícího robota je vhodnější redukovaný pozorovatel.**

Shrnutí kapitoly

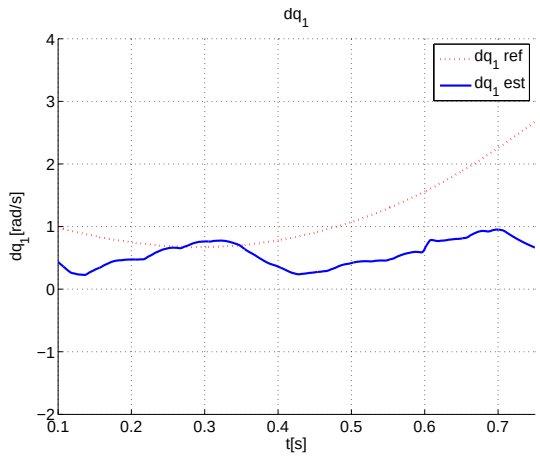
Během experimentů jsem otestoval dva typy pozorovatelů - *redukovaný pozorovatel* a *high gain pozorovatel*. Oba pozorovatele jsem testoval v rámci jednoho kroku robota. Výsledkem experimentů je závěr, že pro odhadování stavů je lepší použít redukovaný pozorovatel namísto high gain pozorovatele a to hlavně kvůli vnitřní struktuře redukovaného po-



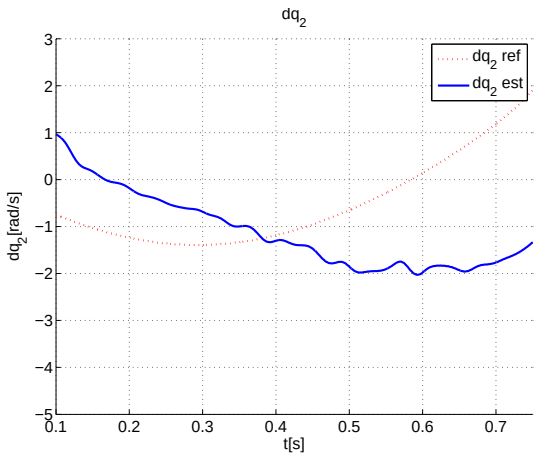
Obrázek 7.6: Redukovaný pozorovatel - měřený úhel q_1



Obrázek 7.7: Redukovaný pozorovatel - měřený úhel q_2



Obrázek 7.8: Redukovaný pozorovatel - odhadovaná úhlová rychlost $\dot{q}_1$



Obrázek 7.9: Redukovaný pozorovatel - odhadovaná úhlová rychlost $\dot{q}_2$

zorovatele, kde můžeme použít malé parametry zesílení $k_{2,4}$. Tím se předejde nežádoucímu zesilování šumu, které se projevuje v případě high gain pozorovatele.

Během experimentů se výrazně lišily průběhy měřených a odhadovaných stavů od jejich referenčních trajektorií. To je způsobeno omezením akčního zásahu motoru a také z důvodu rozdílných počátečních podmínek pro referenční model a reálný model. Při velkém rozdílu mezi referenčními hodnotami stavů robota a skutečnými stavy a současném velkém omezení akčního zásahu trvá i několik kroků robota, než algoritmus použitý pro řízení dosáhne shody stavů s referencemi. Dalším omezením byla limitovaná hodnota maximálního vzorkování. Toto omezení bylo dáno rychlostí CAN sběrnice a také výpočetní náročností řídicího algoritmu.

Kapitola 8

Závěr

Výsledkem této diplomové práce je elektronikou osazený prototyp kráčejícího podaktuovaného robota. Pro tento prototyp jsem po osazení robota veškerou elektronikou a naprogramování řídicích algoritmů otestoval v rámci jednoho kroku dva typy pozorovatelů, navržených pro odhadování stavů podaktuavonaého kráčejícího robota. Díky rozhraní, které jsem vytvořil pro přímé ovládání robota z Matlabu/Simulinku v reálném čase, může tento prototyp robota sloužit jako platforma pro snadný budoucí vývoj a testování algoritmů, určených pro řízení podaktuovaných mechanických systémů.

Činnost obou typů algoritmů určených pro odhadování stavů robota byla několikrát otestována v rámci jednoho kroku robota. Výsledkem experimentů je závěr, že pro odhadování stavů kráčejícího robota se v případě reálného systému lépe hodí redukovaný pozorovatel, který měří polohy q_1, q_2 v kloubech a odhaduje úhlové rychlosti $\dot{q}_1$, $\dot{q}_2$. Tento pozorovatel je také více robustní vůči šumu a nepřesnostem v matematickém modelu odhadovaného systému. High gain pozorovatel, který podle stavů $\dot{q}_1$ a q_2 odhaduje stavy q_1 a $\dot{q}_2$, je velmi citlivý jak na šum měření, tak na odchylky mezi matematickým modelem (podle kterého pozorovatel predikuje odhadované stavy systému) a reálným systémem.

Řízení chůze robota a odhadování neměřených stavů bylo testováno pouze na jednom kroku robota z několika důvodů. Prvním hlavním důvodem bylo to, že jsem měl k dispozici pouze jeden laserový měřič vzdálenosti. Druhým hlavním důvodem byla schopnost robota stabilizovat pouze pohyb ve 2-D rovině, která je daná konstrukcí robota. Z tohoto důvodu by bylo třeba sestrojit nějaký typ základny, který by kráčejícímu robotovi umožnil volný pohyb ve 2-D rovině a stabilizoval ho ve zbývající ose. Posledním důvodem je chybějící model impaktu. U reálného systému dochází k tomu, že při dopadu nohy robota na zem robot ztratí část kinetické energie a dojde ke skokovému snížení úhlových rychlostí. Tento jev, který se nazývá *impakt*, je nutné pro realizaci více kroků robota namodelovat.

Během vývoje a testování softwaru robota pro řízení jeho chůze jsem narazil na několik problémů, které souvisí s tím, že se jedná o reálný systém a které v případě simulace nenastanou. Prvním problémem je omezená frekvence vzorkování a s tím související aktualizace měřených dat. V případě kráčejícího robota se jedná o velmi rychlý systém (jeden krok robota trvá cca 0.7s). Během kroku se hodnoty jednotlivých stavů poměrně rychle mění a při pomalejším vzorkování se potom hodnoty měřených stavů robota skokově mění. Tyto skokové změny činí velké potíže použitým typům pozorovatelů, které jsou citlivé na šum a na rychlé změny měřených stavů. Proto je pro správnou činnost nezbytně nutné použít co největší vzorkování a k tomu všechny měřené veličiny filtrovat. Vzorkování je omezeno rychlostí CAN sběrnice a také výpočetním výkonem PC. Hlavní regulátor a pozorovatel totiž obsahují množství nelineárních rovnic a jsou náročné na výpočet. Druhým důvodem, který souvisí s reálným systémem je odchylka mezi matematickým modelem a reálným systémem. Matematický model, který je použit v regulátoru a v pozorovateli, je zjednodušením reálného systému.

Zůstává zde prostor pro budoucí práci na modelu kráčejícího podaktuovaného robota. Pro lepší činnost proudových regulátorů by bylo vhodné přidat na desky plošných robota analogové filtry pro filtrování měřených proudů v motorech. Pro možnost konečné realizace chůze robota je nutné sestavit model impaktu. Pro snadnější testování robota by bylo vhodné sestrojit základnu, umožňující volný pohyb robota ve 2-D rovině a stabilizaci ve třetí ose. Také by bylo lepší měřit všechny stavy robota a úplně se tak vyhnout použití pozorovatelů. Hlavní algoritmus určený pro řízení chůze by při měření všech stavů robota byl více robustní a méně náročný na výpočet a umožňoval by tak rychlejší vzorkování.

Literatura

Åström, K.J. a Wittenmark, B. (1984), *Computer controlled systems: theory and design*, Prentice-Hall information and system sciences series, Prentice-Hall. ISBN 9780131643192.

Anderle, M. and Čelikovský, S. (2010*a*), Comparison of nonlinear observers for underactuated mechanical systems., *in* 'The 9th International Conference the Process Control 2010'.

Anderle, M. and Čelikovský, S. (2010*b*), Position feedback tracking of the acrobot walking-like trajectory based on the reduced velocity observer., *in* 'Proceedings of the 9th International Conference on Process Control'.

Anderle, M. and Čelikovský, S. (2010*c*), Reduced observer for acrobot in aplication of acrobot walking., *in* 'Preprints of the 8th IFAC Symposium on Nonlinear Control Systems'.

Anderle, M. and Čelikovský, S. (2010*d*), Sustainable acrobot walking based on the swing phase exponentially stable tracking., *in* 'Proceedings of the ASME 2010 Dynamic Systems and Control Conference'.

Anderle, M. and Čelikovský, S. and Henrion, D. and Zikmund, J. (2009), LMI based design for the acrobot walking, *in* 'Preprints of the 9th IFAC Symposium on Robot Control', Gifu, Japan, pp. 595–600.

Fantoni, I. and Lozano, R. (2002), *Non-linear control of underactuated mechanical systems*, Heidelberg: Springer Verlag. ISBN 978-1-85233-423-9.

Herout, P. (2001*a*), *Učebnice jazyka C*, Kopp.

Herout, P. (2001*b*), *Učebnice jazyka C - 2.díl*, Kopp.

Ljung, L. (1999), *System Identification: Theory for the User*, Prentice Hall.

Valentine, R. (1998), *Motor control electronics handbook*, McGraw-Hill. ISBN 9780070668102.

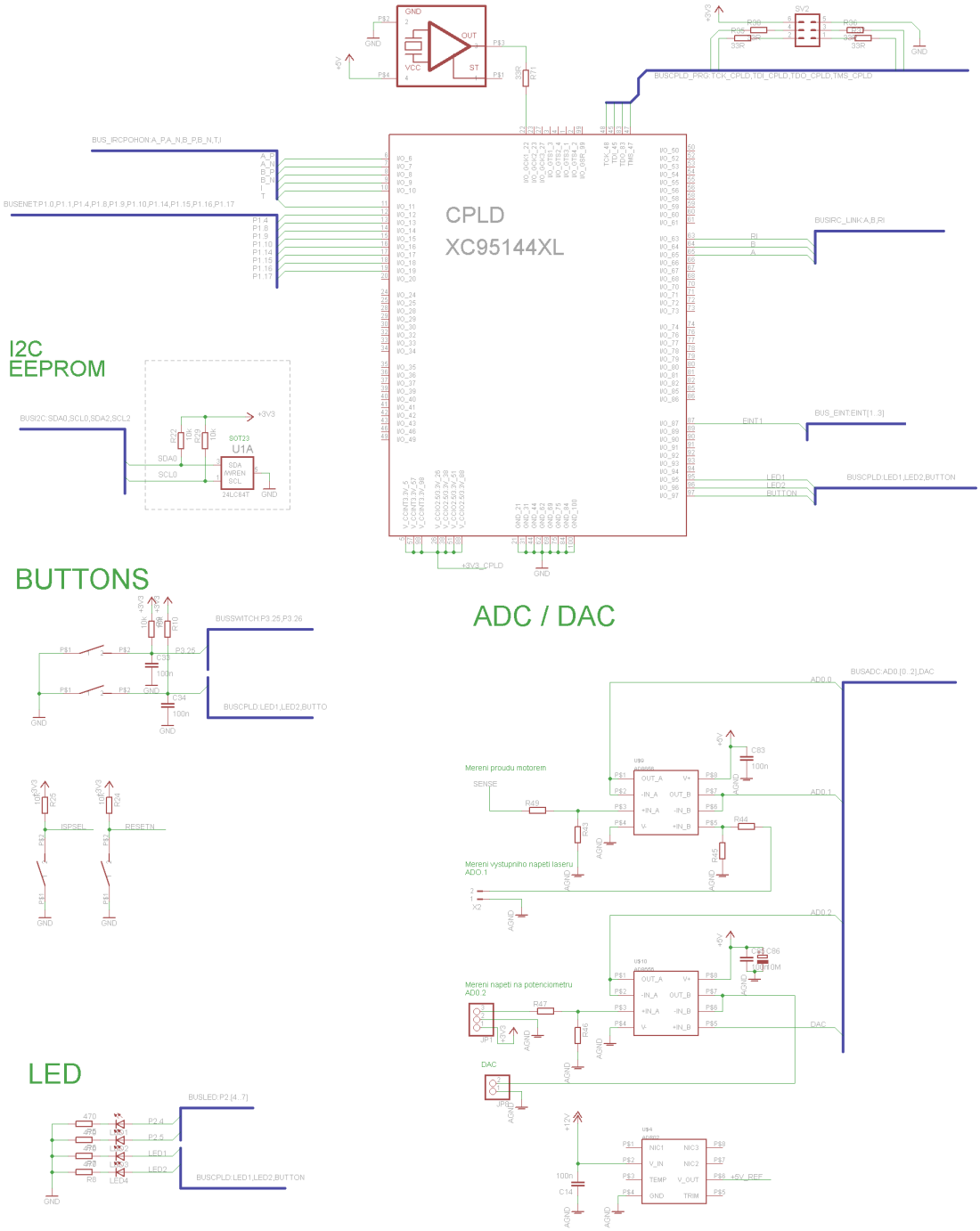
Čelikovský, S. and Zikmund, J. and Moog, C. (2008), Partial exact linearization design for the acrobot walking, *in* 'Preprints of the American Control Conference, 2008', Seattle, USA, pp. 874–879.

MATLAB dokumentace: www.mathworks.com

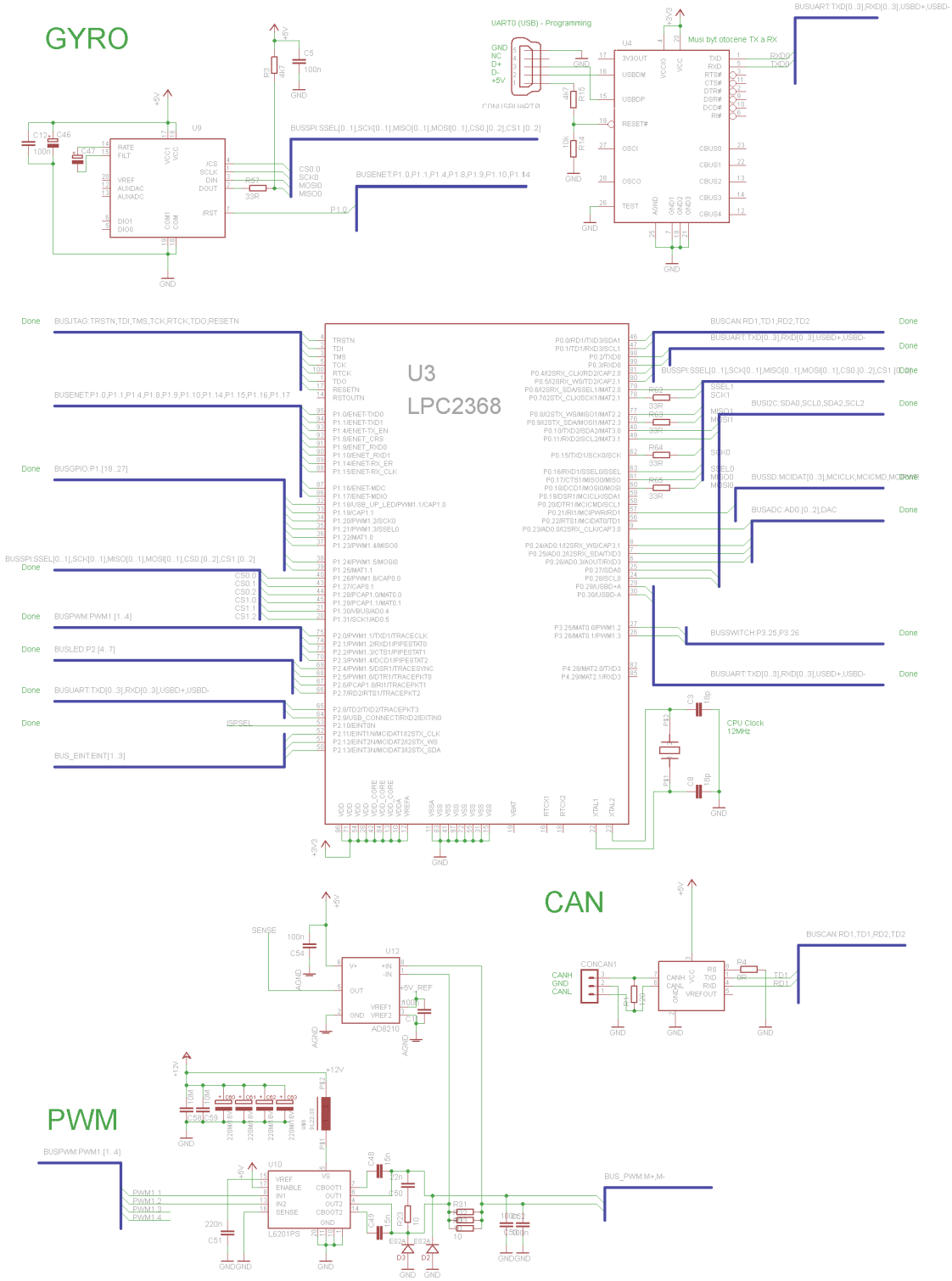
LPC23xx User manual

Příloha A

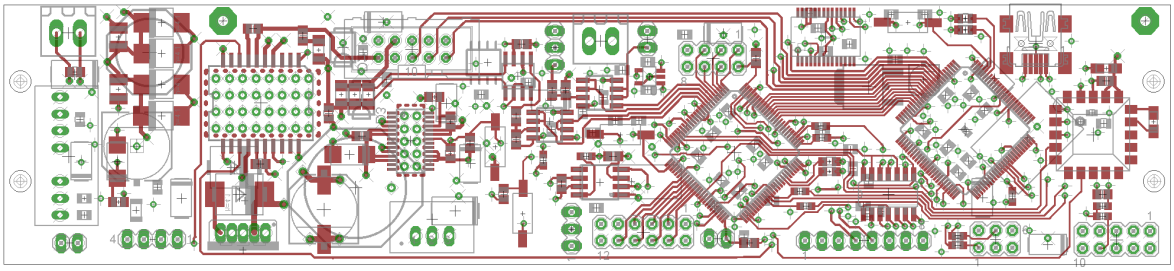
Schéma zapojení



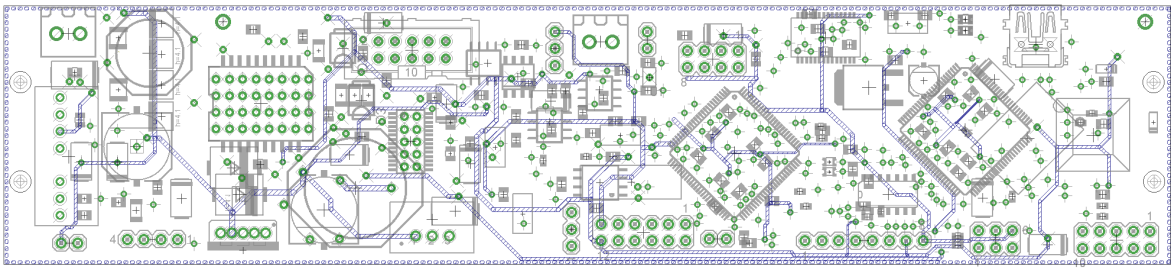
Obrázek A.1: Schéma zapojení



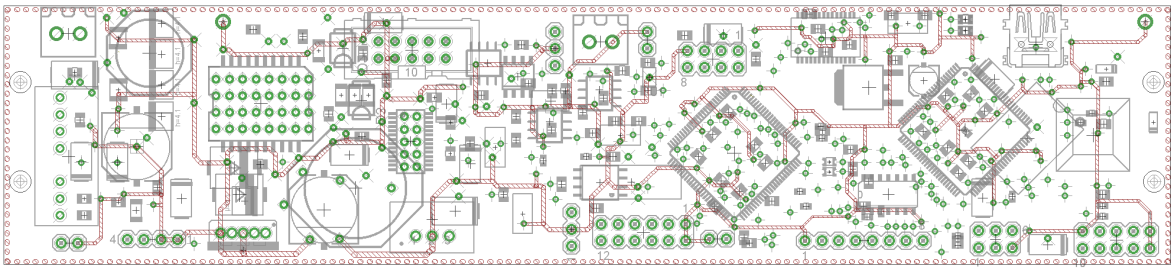
Obrázek A.2: Schéma zapojení



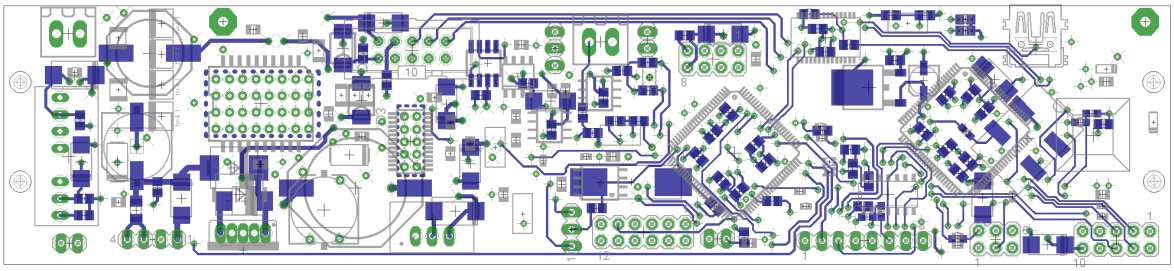
Obrázek A.3: Deska plošných spojů - první vrstva spojů



Obrázek A.4: Deska plošných spojů - druhá vrstva spojů



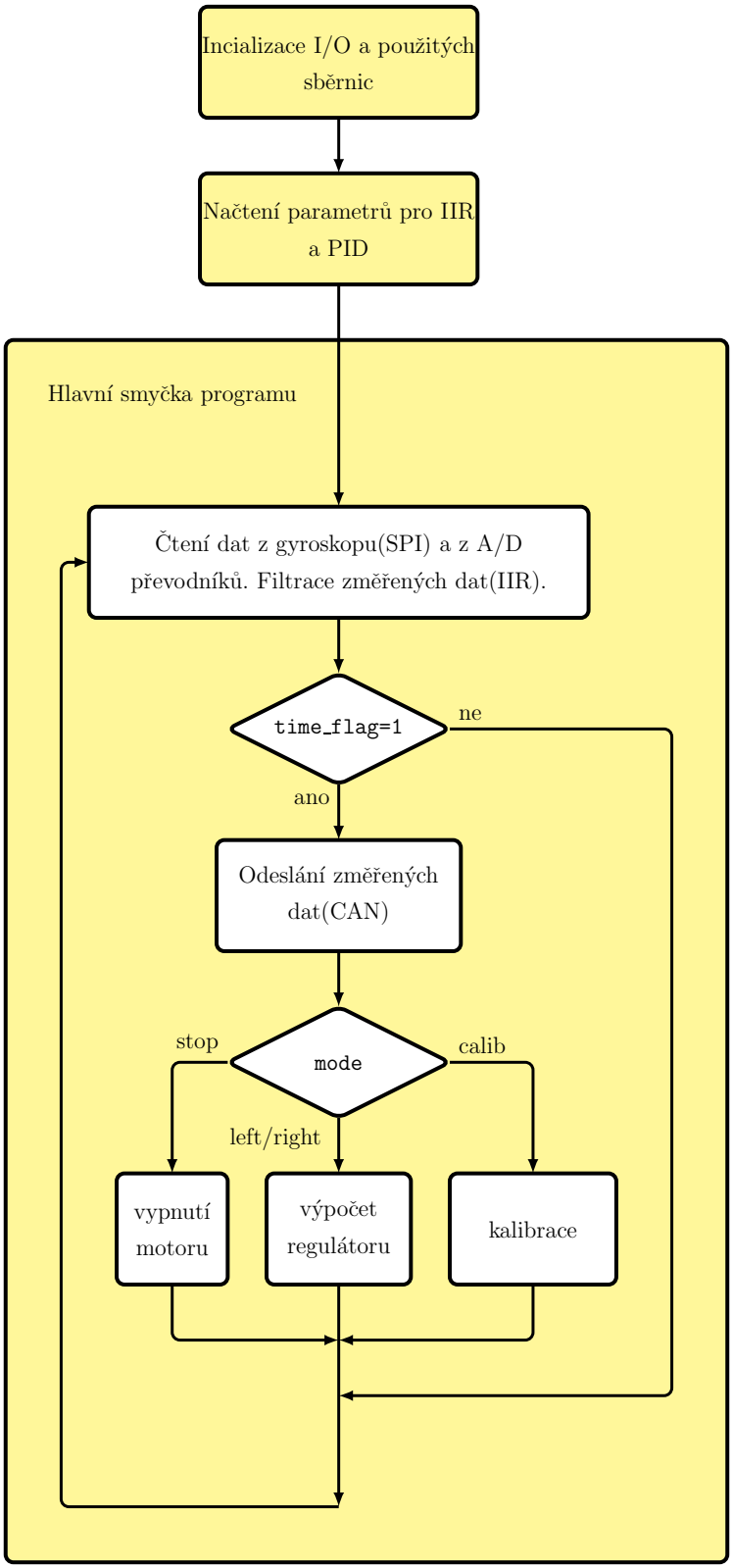
Obrázek A.5: Deska plošných spojů - třetí vrstva spojů



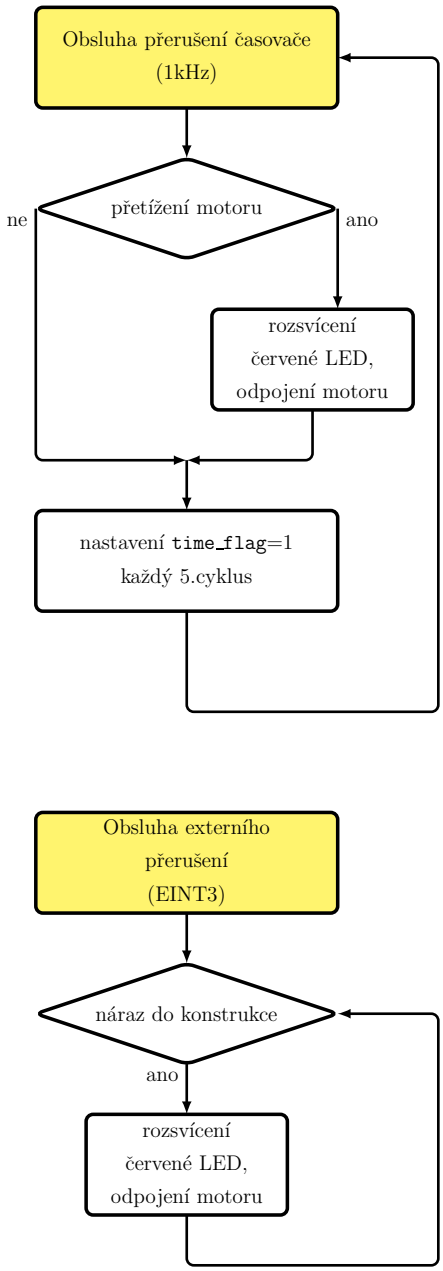
Obrázek A.6: Deska plošných spojů - čtvrtá vrstva spojů

Příloha B

Vývojové diagramy



Obrázek B.1: Program pro procesory robota - hlavní smyčka programu



Obrázek B.2: Program pro procesory robota - obsluha použitých přerušení

Příloha C

Obsah přiloženého CD

K této práci je přiloženo CD, které obsahuje:

- Adresář `Matlab`: Zdrojové kódy pro řízení a identifikaci z Matlabu.
  - Zdrojové kódy z Matlabu pro grey box identifikaci.
  - Rozhraní pro komunikaci mezi robotem a Matlabem vytvořené v Simulinku.
  - Simulinkový model pro řízení robota a odhadování jeho stavů.
- Adresář `Eclipse`: Zdrojové kódy pro procesory robota v jazyce C.
- Adresář `GUI`: Grafické uživatelské rozhraní použité pro komunikaci mezi robotem a PC. Adresář obsahuje projekt s uživatelským rozhraním, vytvořený v jazyce C# v programu Microsoft Visual Strudio 2008.
- Adresář `Eagle`: Podklady pro výrobu desek plošných spojů z programu Eagle.
- Adresář `Literatura`: Dokumentaci k použitým součástkám a odborné články, ze kterých jsem čerpal.
- Adresář `DP`: Text diplomové práce ve formátech PDF a PS.