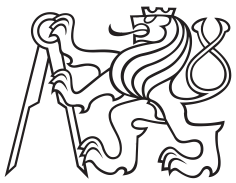
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Experimentální platforma pro rychlé prototypování řídicích algoritmů pro stejnosměrný motor s permanentním magnetem

Jiří Záhora

2016
Vedoucí práce: Ing. Zdeněk Hurák, Ph.D.

Poděkování / Prohlášení

Rád bych poděkoval vedoucímu práce Zdeňku Hurákovi a jeho kolegům za užitečné rady a podněty, dále pak projektu MacGyver ze strahovských kolejí za poskynutí prostoru a vybavení pro vypracování práce.

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

V Praze dne

. . . . . .

Abstrakt / Abstract

Tato práce se zabývá návrhem a testováním výukového přípravku k použití ve výuce automatického řízení. V první části práce jsou vysvětleny základní pojmy a metody používané při řízení stejnosměrného motoru, jako jsou matematický model motoru, kaskádní regulace, pulzně šířková modulace, měření proudu, polohy a odhadování rychlosti. Druhá část obsahuje popis přípravku s motorem a dalším příslušenstvím, postaveného na platformě Arduino s možností generování kódu v prostředí Simulink. Závěr práce je věnován experimentům na hotovém přípravku.

This thesis deals with the design and the testing of an educational device prepared for the teaching of automatic control methods. In the first part of the thesis, basic terms are explained and the explanation of the methods used in DC motor controlling is provided, for instance the mathematical motor model, the cascade control, the pulse-width modulation, the electric current and position measurement, and the velocity estimation. The second part is focused on the device and accessories description, built on the Arduino platform with the possibility of source code generation in Simulink software. The last part of the thesis examines the experiments performed on the completed device.

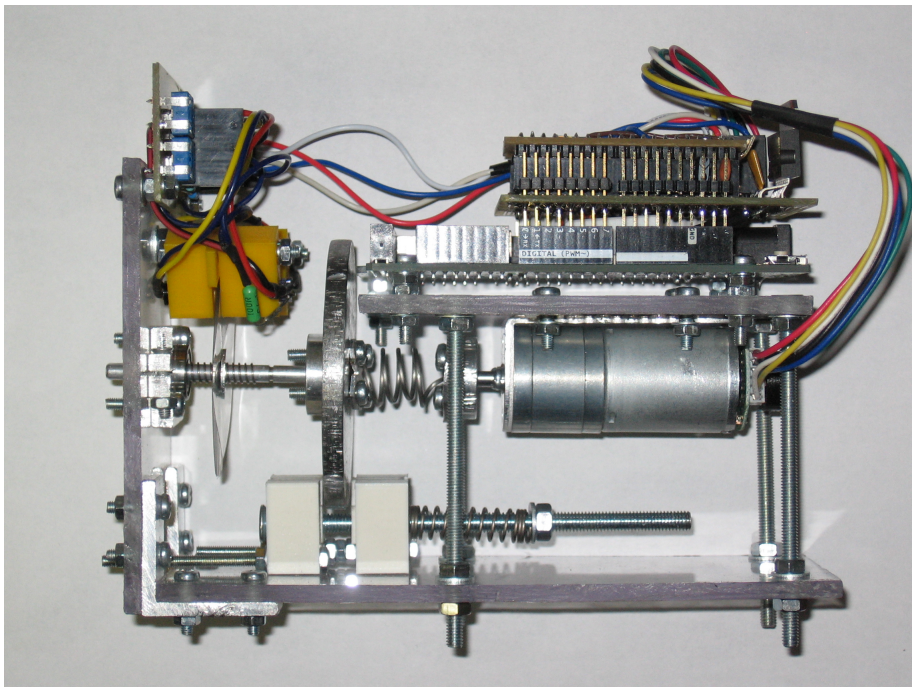
Obsah

Kapitola 1
Motivace a cíle

Pochopení základních principů automatického řízení a regulace je důležité pro všechny studenty kybernetiky a robotiky. Nejlépe se tyto dovednosti rozvíjejí prakticky na reálných demonstračních přípravcích. Pro výuku automatického řízení na FEL je k dispozici několik přípravků v laboratořích, ke kterým však mají studenti přístup pouze v době výuky. Proto se zrodila myšlenka zhotovit přenosný výukový přípravek malých rozměrů, sestavený ze snadno dostupných, levných dílů, vhodný pro zhotovení ve větším množství kusů, aby každý student mohl experimentovat libovolně dlouho, z pohodlí domova. Přípravek by měl také umožňovat programování a zobrazování hodnot v prostředí Matlab/Simulink. Cílem této práce je takový přípravek navrhnout, zhotovit a otestovat. Bude obsahovat stejnosměrný kartáčový motor s převodovkou, zátěží připojenou přes pružnou hřídel a brzdu. Dále snímač proudu tekoucího motorem a snímače natočení rotoru a zátěže. K řízení se použije platforma Arduino Due a H-můstek s možností nastavení hodnoty napětí pomocí pulzně šířkové modulace (PWM). Původní myšlenka byla v průběhu práce upravena na základě inspirace kurzem z MIT [11], tak že práce dává pouze návod na výrobu přípravku, aby si ho každý mohl zhotovit sám.

Kapitola 2
Teoretický úvod

Na úvod představím hotový cíl celé práce, poté bude následovat podrobnější rozbor zařízení a postupu při jeho návrhu.



Obrázek 2.1. Celý přípravek

Kompletní přípravek je na obrázku 2.1. Samotný systém určený k řízení sestává ze stejnosměrného motoru s převodovkou a připojenou zátěží. Zátěž představuje otáčející se železný disk, který je k motoru připojen pružinou, jenž simuluje chování mnohem větších obdobných systémů, kde vliv pružnosti hřídele a převodové mechaniky hraje nezanedbatelnou roli. Na zátěži také vidíme brzdu, co zanáší rušení do systému. Matematický popis nalezneme v následující kapitole 2.1.

Dále je systém osazen snímači různých veličin, též stavů systému. Jejich použití je zřejmé, abychom mohli zavést zpětnou vazbu. Na obrázku dobře vidíme připojení optického inkrementálního snímače polohy zátěže, jehož zhotovení je také součástí práce, kapitola 3.2.3. Další použité senzory na přípravku jsou magnetický inkrementální snímač polohy rotoru motoru, je obsažen přímo na použitém motoru, a snímač proudu motorem. Princip inkrementálních snímačů polohy, neboli enkodérů, bude vysvětlen v kapitole 2.4.

Nakonec část zajišťující výpočty a realizaci vstupu do systému a zpracování údajů ze snímačů vidíme na obrázku jako 3 plošné spoje zapojené do sebe umístěné nad motorem. Hlavní, spodní deska je koupené arduino Due, jenž zajišťuje veškeré výpočty. Prostřední deska je shield pro řízení motoru pomocí napětí s pulzně šířkovou modulací, vysvětleno v kapitole 2.3. Horní deska je modul s programovatelným logickým polem,

umožňující zjednodušení a urychlení zpracování některých dat. Návrh a popis funkce shieldu pro řízení motoru a modulu s logickým polem nalezneme v kapitolách 3.2.2 a 3.2.1 i se seznamem součástek použitých při výrobě.

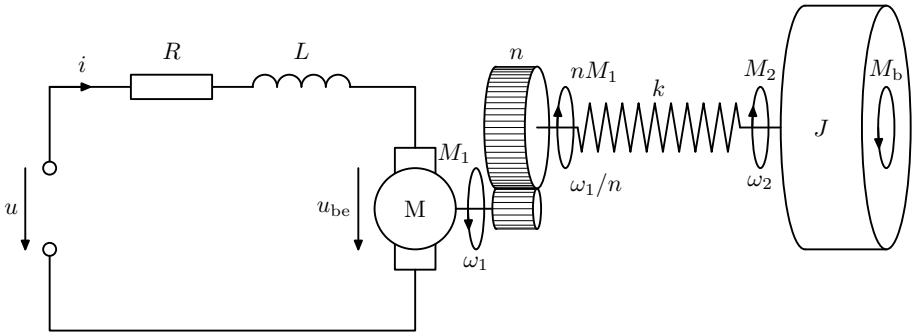
Fungování celého přípravku je následující. Motor a zátěž se otáčejí a generují výstupy systému, procesor arduina zpracovává výstupy, z nichž vypočítá vstup do systému, který je realizován shieldem. Samotné výpočty jsou implementace regulátorů, například řízení rychlosti a polohy zátěže, s kontrolou proudu motorem. Více o implementaci řídicích algoritmů bude uvedeno níže.

Porovnáme-li tento přípravek s jinými, komerčně dostupnými, například od výrobců Quanser [12], nebo INTECO [13], rozhodně nemůžeme očekávat takovou kvalitu provedení, ani přesnost možné regulace. Ale oproti takovýmto produktům zde nahlédneme blíže jádru věci a lépe poznáme různá úskalí spojená s automatickým řízením, která nejsou v precizních komerčních produktech dobře pozorovatelná. K dispozici se nabízí i porovnání s nějakou modulární stavebnicí, například open-source projekt MLAB [14], z té by podobný přípravek také šel sestrojit se srovnatelným výsledkem. Nicméně naše řešení zasahuje ještě o něco dále a umožňuje i poznání a ohmatání základních elektronických komponent, jež jsou ve větší či menší míře součástí každého řídicího systému. Přípravek se v zásadě velmi podobá projektu uvedenému v [11], pouze samotný systém byl zvolen tak, aby pokud možno simuloval zařízení, která se v běžné praxi hojně vyskytují. Motory automaticky pohybující něčím můžeme nalézt od stolních tiskáren po průmyslové linky.

2.1 Matematický model

2.1.1 Elektromechanické schéma

Pro naše účely budeme modelovat stejnosměrný motor s kartáčovými sběrači, převodovku, pružnou hřídel a brzděnou zátěž. Schéma celé soustavy je na obrázku 2.2. Vinutí motoru můžeme namodelovat sériovým spojením cívky s indukčností L, odporu R a zdroje zpětného napětí u_{be}. Motor působí momentem M_1 na rotor s momentem setrvačnosti J_{r}, otáčející se rychlostí ω_1 v prostředí s viskózním třením b. Je nutné poznamenat, že použitý lineární model tření je velmi nepřesný, zejména při malých rychlostech, ale problematika modelování tření není náplní práce, proto toto zjednodušení použijeme. Převodovka má převodový poměr n, moment setrvačnosti a tření zanedbáme, nebo zahrneme do parametrů rotoru. Hřídel mezi převodovkou a zátěží modelujeme torzní pružinou s tuhostí k. Zátěž s momentem setrvačnosti J se otáčí rychlostí ω_2 a působí na ní hřídel momentem M_2 a brzdný moment M_{b}. Velikost zpětného napětí je určena konstantou motoru k_{m} a vztahem $u_{\mathrm{be}} = k_{\mathrm{m}}\omega_1$, obdobně proud motorem i určuje velikost momentu $M_1 = k_m i$.



Obrázek 2.2. Schéma motoru s převodovkou, pružnou hřídelí a zátěží

2.1.2 Stavový popis

K řízení potřebujeme znát stavové rovnice systému. Ty získáme z pohybových rovnic, které zde sestavíme pomocí Lagrangeova přístupu. Vyjdeme ze známého tvaru Lagrangeovy rovnice

$$\frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{s}_i}\right) - \frac{\partial L}{\partial s_i} = -\frac{\partial D}{\partial \dot{s}_i} + Q_i \ , \tag{1}$$

$$L = T - V \ ,$$

kde L je Lagrangeova funkce, T kinetická, V potenciální energie, s_i, kde $i \in \{1, \ldots, n\}$, zobecněné souřadnice pro n stupňů volnosti, D příspěvky ztrát na odporech, neboli disipace, Q_i vnější síly působící na i-tou souřadnici. Systém na obrázku 2.2 má $n = 3$ stupně volnosti. Je to elektrický náboj v obvodu $s_1 = q$, úhel natočení rotoru motoru $s_2 = \varphi_1$ a úhel natočení zátěže $s_3 = \varphi_2$. Pomocí těchto souřadnic napíšeme potřebné vztahy:

$$T = \frac{1}{2}L\dot{q}^2 + \frac{1}{2}J_\mathrm{r}\dot{\varphi}_1^2 + \frac{1}{2}J\dot{\varphi}_2^2 \ ,$$

$$V = \frac{1}{2}k(\varphi_2 - \varphi_1)^2 \ ,$$

$$D = \frac{1}{2}R\dot{q}^2 + \frac{1}{2}b\dot{\varphi}_1^2 \ ,$$

$$Q_1 = u_\mathrm{be} + u = k_\mathrm{m}\dot{\varphi}_1 + u \ ,$$

$$Q_2 = k_\mathrm{m}\dot{q} \ ,$$

$$Q_3 = M_\mathrm{b} \ .$$

Dosazením do (1) získáme pohybové rovnice

$$L\ddot{q} = -R\dot{q} - k_\mathrm{m}\dot{\varphi}_1 + u \ ,$$

$$J_\mathrm{r}\ddot{\varphi}_1 - \frac{k}{n}\left(\varphi_2 - \frac{\varphi_1}{n}\right) = -b\dot{\varphi}_1 + k_\mathrm{m}\dot{q} \ ,$$

$$J\ddot{\varphi}_2 + k\left(\varphi_2 - \frac{\varphi_1}{n}\right) = M_\mathrm{b} \ ,$$

které pomocí substitucí

$$\dot{q} = i \ ,$$

$$\dot{\varphi}_1 = \omega_1 \ ,$$

$$\dot{\varphi}_2 = \omega_2 \ ,$$

přepíšeme do stavového popisu ve tvaru $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} + \mathbf{Eh}$, kde $\mathbf{x}$ jsou stavy systému, $\mathbf{u}$ vstupy a $\mathbf{h}$ rušení jednotlivých stavů. V tomto případě můžeme nastavovat pouze vstup u, zásah M_b lze považovat za rušení.

$$\begin{pmatrix} \dot{i} \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{pmatrix} = \begin{pmatrix} -\frac{R}{L} & -\frac{k_\mathrm{m}}{L} & 0 & 0 & 0 \\ \frac{k_\mathrm{m}}{J_\mathrm{r}} & -\frac{b}{J_\mathrm{r}} & 0 & -\frac{k}{J_\mathrm{r}n^2} & \frac{k}{J_\mathrm{r}} \\ 0 & 0 & 0 & \frac{k}{Jn} & -\frac{k}{J} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} i \\ \omega_1 \\ \omega_2 \\ \varphi_1 \\ \varphi_2 \end{pmatrix} + \begin{pmatrix} \frac{u}{L} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{M_\mathrm{b}}{J} \\ 0 \\ 0 \end{pmatrix} \tag{2}$$

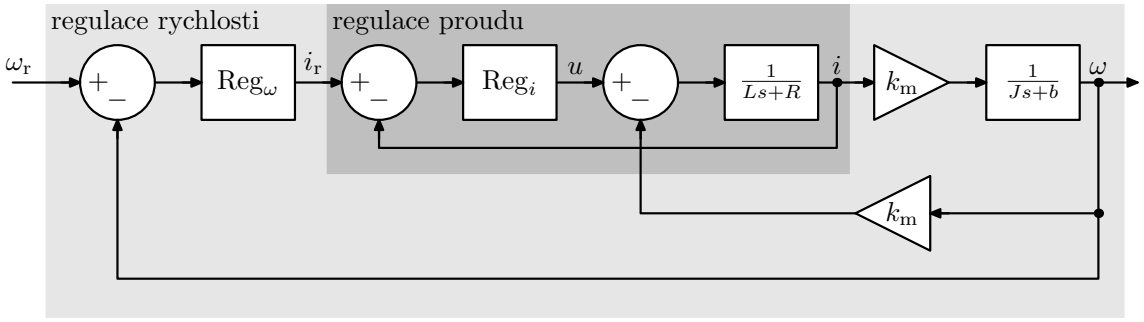
2.2 Kaskádní regulace

Pokud chceme řídit více stavů systému, jedním ze způsobů vícestavové regulace je kaskádní regulátor. Jeho princip popíšeme na příkladu samostatného stejnosměrném motoru, kde běžně chceme řídit velikost proudu motorem, abychom předešli poškození vinutí, nebo řídicí elektroniky, a rychlost otáčení motoru. Rovnice popisující pouze motor mají tvar

$$L\dot{i} = -Ri - k_m\omega + u$$
$$J\dot{\omega} = k_\mathrm{m} i - b\omega \ , \tag{3}$$

Při návrhu nejprve setřídíme stavy podle rychlosti dynamiky a postupně od nejrychlejšího navrhneme základními metodami regulátor pro dosažení přijatelné odezvy stavu, přičemž vliv ostatních stavů budeme považovat za rušení vstupující do systému. V případě motoru máme 2 stavy, rychlejší změna proudu vinutím a pomalejší změna rychlosti otáčení motoru. Budeme tedy navrhovat 2 regulační smyčky, proudovou a rychlostí. K regulaci proudu stačí jednoduchý PI regulátor. Tím uzavřeme proudovou smyčku regulátoru, kterou již při dalším návrhu nemusíme uvažovat, protože navenek se chová jako blok s přenosem 1. Toto tvrzení v praxi platí pouze pokud regulační perioda proudové smyčky je alespoň 10krát kratší, než perioda rychlostní smyčky, aby při každém akčním zásahu rychlostního regulátoru byl proud vinutím v ustáleném stavu. Poté opět můžeme návrh regulátoru rychlosti provést základními metodami, s jednoduchým výsledkem. Strukturu celého kaskádního regulátoru znázorňuje obrázek 2.3, na kterém je graf signálových toků. Při sestavování grafu použijeme rovnice (3), které pomocí Laplaceovy transformace přepíšeme do tvaru: (proměnné značené velkými písmeny jsou Laplaceovy obrazy odpovídajících veličin značených malými písmeny v časové oblasti)

$$I = \frac{-k_\mathrm{m}\Omega + U}{Ls + R} \ ,$$
$$\Omega = \frac{k_\mathrm{m} I}{Js + b} \ .$$



Obrázek 2.3. Graf kaskádního regulátoru stejnosměrného motoru s proudovou a rychlostní smyčkou

Hlavní důvod používání tohoto regulátoru v praxi je jeho snadný návrh a ladění, ačkoliv je zřejmé, že podmínka na zužování pásma omezuje maximální rychlost odezvy, především při zavedení více smyček regulace. Důkladnější popis kaskádní regulace lze najít například v [1].

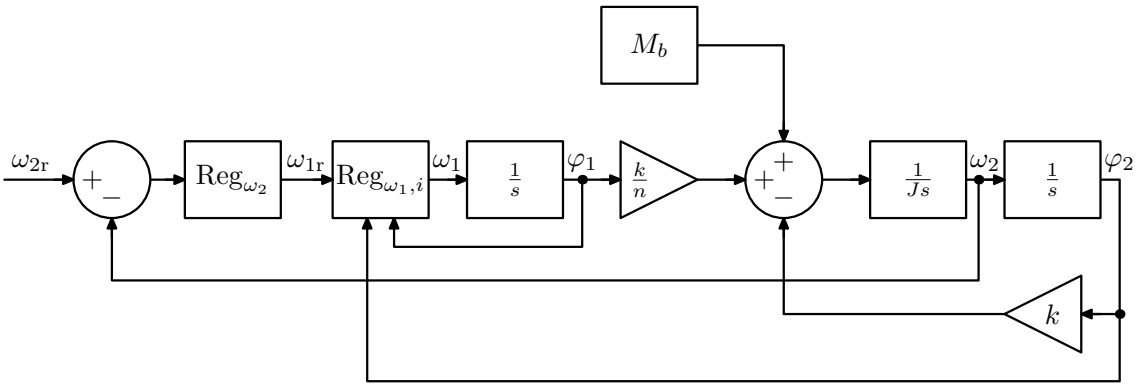
Rozšíření regulátoru pro řízení našeho přípravku může mít například následující strukturu zobrazenou na obrázku 2.4. Zde chceme řídit rychlost otáčení zátěže. Při návrhu použijeme stejný postup, jako v předchozím případě, ale vyjdeme ze stavové

rovnice celého systému (2). Pod blokem $\mathrm{Reg}_{\omega_1,i}$ se skrývá struktura podle obrázku 2.3, pouze se zavedením dalšího rušení od stavů φ_1, φ_2. Blok Reg_{ω_2} může opět reprezentovat nějaký základní regulátor, například PID. Zbytek struktury je znázorněním rovnic

$$\Omega_2 = \frac{\frac{k}{n}\Phi_1 - k\Phi_2 + M_b}{Js} ,$$

$$\Phi_1 = \frac{\Omega_1}{s} , \quad \Phi_2 = \frac{\Omega_2}{s} ,$$

které vyplývají z Laplaceovy transformace rovnic (2).

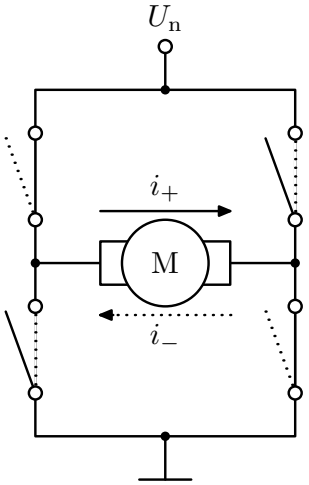


Obrázek 2.4. Graf kaskádního regulátoru rychlosti otáčení zátěže přípravku

Oproti řízení rychlosti samotného motoru bez zátěže, je tento regulátor doplněn o zpracování informace o rychlosti otáčení zátěže, která je díky pružné hřídeli jiná, než rychlost motoru jednoduše přenásobená převodovým poměrem převodovky. Samozřejmě by šly navrhnou vhodnější regulátory, ale na tomto typu je možné nejsnáze demonstrovat smysl a správnou funkci všech částí přípravku.

2.3 H-můstek a pulzně šířková modulace

Řízení napětí na motoru zajišťuje elektrický obvod H-můstek, základní princip jeho funkce je znázorněn na obrázku 2.5. Jedná se o 4 spínače, umožňující připojení napájecího napětí U_n na motor s libovolnou polaritou. Tím se docílí nastavení směru otáčení motoru. Spínače jsou zpravidla realizovány spínacími FET-transistory, pro malé výkony společně s další nutnou elektronikou v integrovaném obvodu.



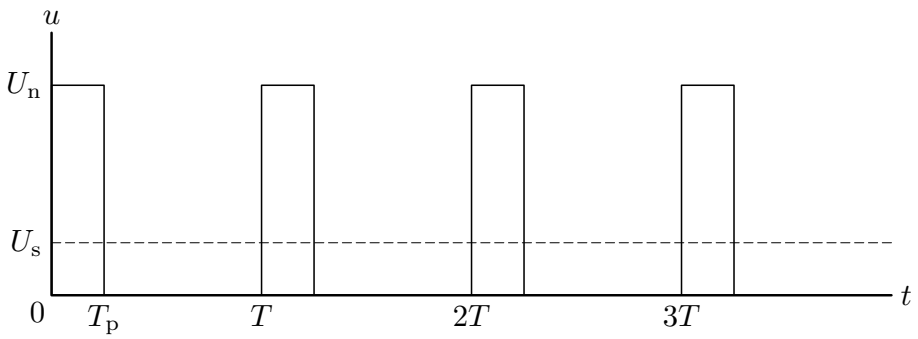
Obrázek 2.5. Schéma řízení směru otáčení motoru H-můstkem

Abychom mohli řídit i velikost napětí na motoru, využívá se pulsně šířkové modulace (PWM) napětí. Napájecí napětí U_n je spínáno v pulsech o trvání T_p, s periodou spínání T. Poměr T_p/T nazýváme činitel plnění. Za předpokladu, že zátěž řízená PWM napětím má charakter dolnopropustního filtru a modulační frekvence $f = 1/T$ je dostatečně vysoká, má výsledné napětí na zátěži hodnotu rovnou střední hodnotě PWM signálu

$$U_\mathrm{s} = \frac{T_\mathrm{p}}{T} U_\mathrm{n} \ .$$

Bohužel tento předpoklad není vždy splněn. Při experimentech na našem přípravku uvidíme například značný nežádoucí vliv modulace při měření proudu procházejícího motorem.

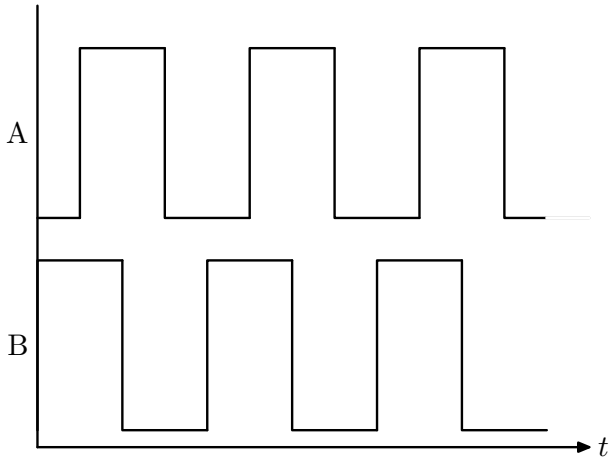
Průběh PWM napětí je znázorněn na obrázku 2.6. V případě, že zátěž je motor, musíme počítat se zpětným indukovaným napětím na vinutí, které by mohlo poškodit elektroniku H-můstku. Popis různých módů řízení, které tento problém řeší lze nalézt například v [4].



Obrázek 2.6. Pulsně šířková modulace

2.4 Snímání polohy, kvadraturní enkodér

Polohu budeme měřit inkrementálním snímačem. Ten vysílá impulsy v závislosti na změně polohy. Obecně snímač obsahuje měřítko (lineární, nebo rotační) se značkami rozmístěnými v konstantních intervalech d a snímač (magnet, fotodioda), který detekuje značku při průchodu kolem snímače. Přechod přes značku indikuje logickou 1, mimo značku logickou 0. Pro kódování informace o směru změny polohy použijeme kvadraturní enkodér. Jedná se o 2 snímače umístěné tak, aby z nich signály detekce značky přicházely s rozdílem $d/2$. Šířka značky je stejná jako šířka úseku bez značky. Signály ze snímačů A a B při pohybu konstantní rychlostí znázorňuje obrázek 2.7. Vidíme, že důsledkem posunu o $d/2$ dostaneme 2 obdélníkové signály posunuté o půl periody. Díky tomu můžeme určovat i směr změny polohy. Pro rotační enkodér s počtem značek N_p má nejmenší rozeznatelná změna úhlu natočení velikost $\theta_\mathrm{m} = \frac{2\pi}{4N_\mathrm{p}}$ [rad]. Kódování směru je zapsáno v tabulce 2.1. Směr lze určit z posledních 2 stavů signálů obou snímačů.



Obrázek 2.7. Průběh signálů kvadraturního enkodéru při pohybu konstantní rychlostí

AB_0	AB_1	směr
00	01	+
01	11	+
10	00	+
11	10	+
00	10	-
01	00	-
10	11	-
11	01	-

Tabulka 2.1. Kódování směru kvadraturního enkodéru, AB_0 logické hodnoty signálů snímačů A a B před změnou, AB_1 po změně

2.5 Odhadování rychlosti

Úhlovou rychlost nebudeme měřit přímo, ale pro její výpočet použijeme informaci o změně polohy, získanou inkrementálním snímačem s rozlišením θ_m, za čas. Základní i pokročilé metody výpočtu rychlosti i s určením chyby výpočtu jsou shrnuty v [3]. Mezi nejjednodušší způsoby výpočtu patří metoda měření frekvence, metoda měření periody a metoda CSDT (Constant Sample-time Digital Tachometer), která je spojením obou předchozích.

Měření frekvence spočívá v počítání impulsů ΔN za konstantní periodu T_s. Úhlovou rychlost určíme podle vztahu

$$\omega = \frac{\Delta N \theta_\mathrm{m}}{T_\mathrm{s}} \ [\mathrm{rad\ s^{-1}}] \ .$$

Metoda je tím přesnější, čím vyšší je úhlová rychlost.

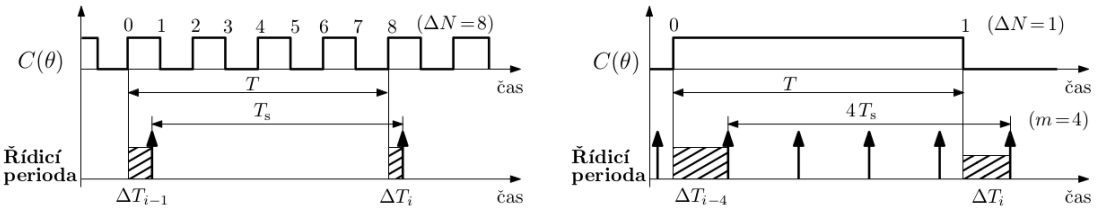
Výpočet rychlosti z periody T provádíme pomocí měření časového úseku mezi dvěma impulsy z enkodéru

$$\omega = \frac{\theta_\mathrm{m}}{T} \ [\mathrm{rad\ s^{-1}}] \ .$$

Metoda je přesnější pro menší rychlosti, ale měření probíhá asynchronně. V regulační smyčce se použije poslední naměřená hodnota, jejíž aktualizace pro velmi pomalé rychlosti nemusí proběhnout několik cyklů smyčky, z čehož například vyplývá obtížná regulace na nulovou rychlost.

Sjednocením obou způsobů je metoda *CSDT*. Její princip znázorňuje obrázek 2.8. Zde se v každém cyklu řídicí smyčky měří doba T_i od posledního impulsu z enkodéru do ukončení smyčky, dále je pro malé rychlosti počítán počet cyklů m řídicí smyčky T_s mezi dvěma impulsy enkodéru, pro velké rychlosti počet impulsů enkodéru ΔN během řídicí periody T_s. Z těchto údajů vypočteme úhlovou rychlost podle vztahu

$$\omega = \frac{\Delta N \theta_\mathrm{m}}{m T_\mathrm{s} + T_{i-m} - T_i} \ [\mathrm{rad\ s^{-1}}] \ . \tag{4}$$



a) Velké rychlosti **b)** Malé rychlosti

Obrázek 2.8. Časové diagramy metody CSDT, převzato z [3]

Chyba výpočtu záleží na přesnosti měření času, které je zpravidla realizováno vysokofrekvenčním čítačem. Při měření velmi pomalých rychlostí je zde stejný problém, jako při výpočtu rychlosti z periody. Částečné lze metodu vylepšit pomocí LVC (Low Velocity Compensation). To spočívá v omezení maximální možné rychlosti na základě informace, že během řídicí periody nepřišel žádný impuls ze snímače. Pokud nepřišel žádný impuls po m řídicích period, velikost úhlové rychlosti je shora omezená hodnotou

$$\omega_m = \frac{\theta_\mathrm{m}}{T_{i-m} + m T_\mathrm{s}} \ [\mathrm{rad\ s^{-1}}] \ .$$

Pro experimenty na přípravku bude použita metoda CSDT, ovšem nebude implementována přímo realizace rovnice (4), jelikož to s sebou nese jistá úskalí. Více o konkrétní realizaci bude uvedeno v kapitole 3.4.2.

Kapitola 3
Návrh přípravku

Tato část práce je věnována popisu konkrétní podoby navrženého přípravku včetně postupu při návrhu.

3.1 Hardware a software použitý při návrhu

Pro prvotní pokusy a vytvoření myšlenky o podobě přípravku byly využity tyto vývojové desky a díly:

- Arduino Uno - vývojová deska s avr procesorem ATMEGA328p
- Arduino Due - vývojová deska s arm procesorem ATSAM3X8E
- Arduino Motor shield - dvojitý H-můstek L298 umožňující řídit pomocí arduina 2 stejnosměrné motory, nebo jeden krokový a měřit procházející proud
- EP2C5T144 Altera CycloneII FPGA vývojová deska
- DC motor 6 V, 2,2 A s převodovkou 1:34 a magnetickým enkodérem 48 CPR, Pololu
- HEDS-9000 optický inkrementální enkodér

Podrobnější informace jsou v dokumentaci na přiloženém CD.
Použitý software:

- Matlab a Simulink s podpůrným balíkem pro práci s arduinem
- Arduino IDE
- Quartus

Seznamy elektronických součástek použitých při konečné realizaci jsou uvedeny v následující sekci 3.2 u popisů konkrétních zhotovených dílů.

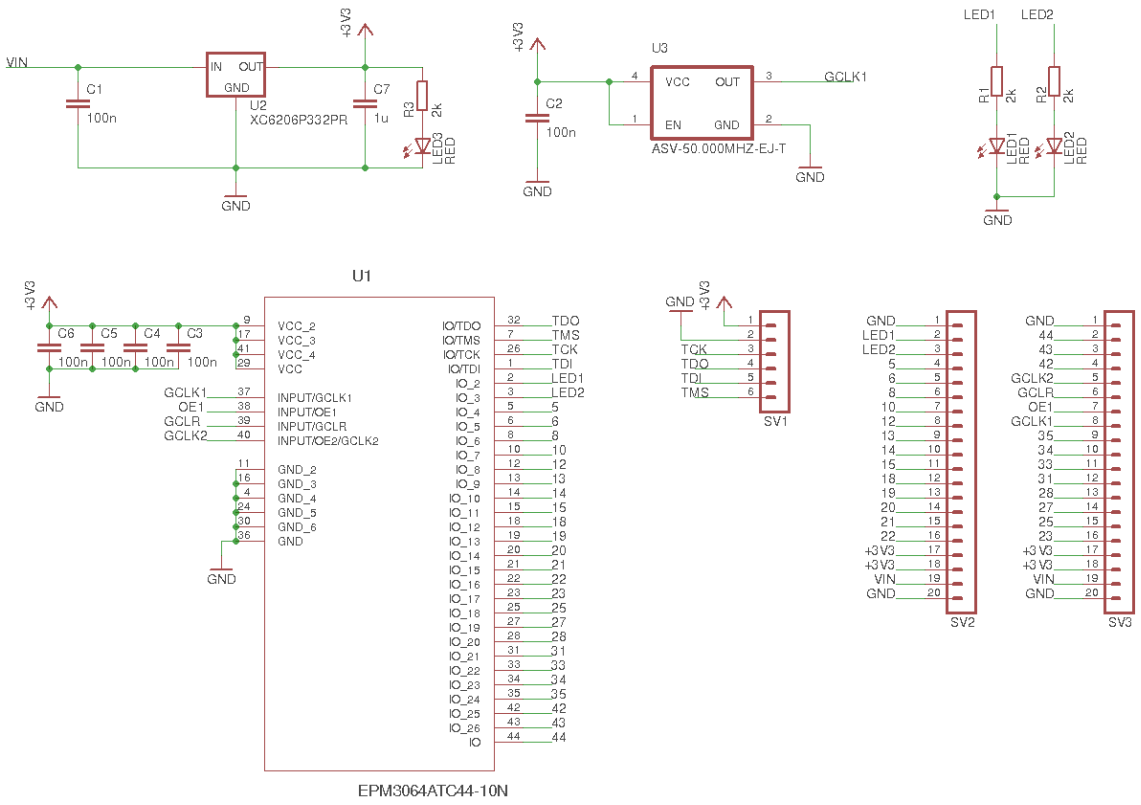
3.2 Návrh elektronické části

Při návrhu vyjdeme ze schématu řízené části přípravku 2.2 a jeho stavového popisu (2). Potřebujeme zajistit vstup u napětí na motoru a měření stavů i, φ_1, φ_2. Inspirujeme se Arduino Motor Shieldem, kde je základem integrovaný dvojitý H-můstek L298, doplněný rezistory pro měření procházejícího proudu principem úbytku napětí na rezistoru. Protože napětí na měřicích rezistorech je malé, je v obvodu použit operační zesilovač pro zesílení. Tím je zajištěn vstup u a výstup i. Pro měření stavů φ_1, φ_2 použijeme kvadraturní enkodéry. Použitý motor již je vybaven enkodérem s magnetickým kolečkem a hallovými sondami. Pro měření polohy zátěže použijeme optický inkrementální senzor. Signály z enkodérů je možné zpracovávat procesorem arduina, ale výpočetní výkon procesoru chceme použít především pro řídicí algoritmy, proto je výhodnější výstupy z enkodérů předzpracovat. Jednoduchá logika zpracování lze realizovat několika logickými hradly dostupnými v integrovaných obvodech, jenže toto řešení neumožňuje žádnou flexibilitu. Pro realizaci logiky výhodněji použijeme programovatelné logické pole CPLD, kde můžeme implementovat libovolnou logiku. Nakonec kvůli snazšímu

návrhu a ověřování funkčnosti rozdělíme obvod do 2 částí. První obvod bude shield, kompatibilní s deskou arduina, obsahující veškeré komponenty, kromě CPLD čipu. Ten bude na druhé desce, samostatně použitelné pro testování všech jeho funkcí. Navržený shield bude obsahovat konektory pro zasunutí modulu s CPLD.

3.2.1 CPLD modul

Na CPLD modul jsou dva požadavky. Použitý čip musí být použitelný s 3,3 V i 5 V logikou, protože signály z použitých enkodérů mají úroveň 5 V, zatímco arduino Due pracuje s 3,3 V logikou. Další požadavek je, aby čip bylo možné konfigurovat pomocí prostředí Quartus, se kterým by studenti kybernetiky a robotiky měli být seznámeni. Těmto požadavkům vyhovují například čipy série MAX3000A od výrobce ALTERA. Čipy normálně pracují s 3,3 V logikou, ale tolerují napětí 5 V na vstupu. Jejich výstup s úrovní 3,3 V odpovídá logické 1 i při použití standardní 5 V logiky. Návrh schématu je vytvořen podle [6], pouze s lehkými úpravami. Obvod obsahuje CPLD čip, stabilizátor napětí, krystalový oscilátor pro generování 50 MHz hodinového signálu, indikační LED napájení, 2 LED připojené na IO piny čipu, programovací konektor, konektory s ostatními vývody čipu a filtrační kondenzátory. Schéma obvodu je na obrázku 3.1.



Obrázek 3.1. Schéma CPLD modulu

Název	Množství	Hodnota	Pouzdro
U1	1	EPM3064ATC44-10N	TQFP44
U2	1	XC6206P332PR	SOT89
U3	1	ASV-50.000MHZ-EJ-T	7X5_SMD
C1-6	6	100 nF	C0805
C7	1	1 μF	C0805
R1-3	3	2 kΩ	R0805
LED1-3	3	red	LED0805

Tabulka 3.1. Seznam použitých součástek na CPLD modul

Obvod MAX3064A obsahuje 4 bloky logických polí, každý blok obsahuje vlastní napájení, proto je zařazen filtrační kondenzátor ke každému napájecímu pinu. Piny GND nejsou uvnitř obvodu propojeny, je nutné je všechny propojit na desce plošných spojů.

Pro spuštění CPLD modulu je nutné připojit napětí maximálně 6 V mezi piny VIN a GND. K programování obvodu je použit standard JTAG (Joint Test Action Group). Z něj jsou použity signály TDI (Test Data Input), TDO (Test Data Output), TCK (Test ClocK) a TMS (Test Mode Select). Stručné přiblížení, co je JTAG lze nalézt v [7]. Dále je potřeba zařízení USB blaster, sloužící k programování přes USB port počítače.

3.2.2 Shield pro řízení motorů

Hlavní část shieldu je dvojitý H-můstek L298 s výstupy pro měření proudu. Tento obvod umožňuje řízení 2 stejnosměrných motorů do 2 A. My potřebujeme řídit pouze jeden motor, takže stačí použít jeden H-můstek, nebo můžeme můstky propojit paralelně pro větší spolehlivost. K řízení H-můstku použijeme 3 piny arduina. DIR1, DIR2 pro nastavení směru otáčení, nebo rychlé brzdění motoru, v případě, že je na obou pinech stejná logická úroveň. Dále pin PWM k nastavení velikosti napětí na motoru. Výstupy můstku na motor jsou opatřeny diodami kvůli ochraně obvodu před případným zpětným indukovaným napětím z motoru a vratnou pojistkou PPTC pro ochranu před zkratem výstupů. Výstup měření proudu propojíme se zemí rezistorem. Napětí na rezistoru zesílíme operačním zesilovačem v neinvertujícím zapojení na potřebnou úroveň. Vztah pro výpočet napěťového zesílení A_u neinvertujícího zesilovače v zapojení podle 3.2 je

$$A_u = 1 + \frac{R_3}{R_1} \ .$$

Odpory R_1, R_3, R_4 potřebujeme navrhnou tak, aby maximálnímu povolenému proudu 2 A odpovídalo maximální napětí přípustné na analogovém vstupu arduina, v případě arduina Due 3,3 V, neboli chceme citlivost měření proud $1,65\ \frac{\mathrm{V}}{\mathrm{A}}$. Odpor R_4 zároveň musí být malý v porovnání s odporem motoru. Hodnoty odporů mohou být například

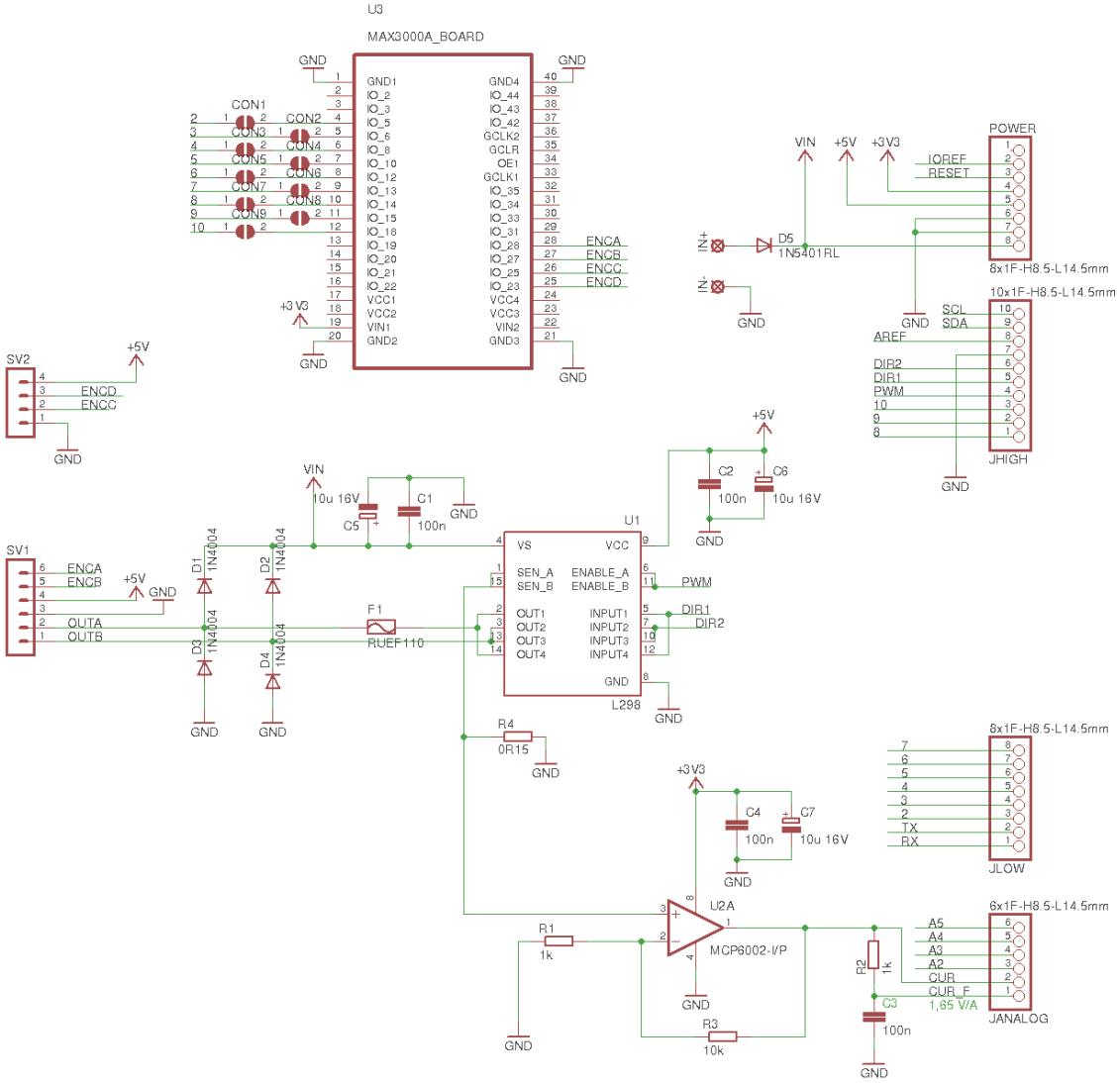
$$R_1 = 1\ \mathrm{k\Omega},\ R_3 = 10\ \mathrm{k\Omega},\ R_4 = 150\ \mathrm{m\Omega}\ .$$

Přesnou hodnotu citlivosti měření je nutno přeměřit, protože odpory rezistorů jsou nepřesné.

Operační zesilovač musí být typu rail-to-rail, aby jím bylo možné měřit v plném rozsahu hodnot. Ochranu analogového vstupu arduina proti příliš vysokému napětí, při proudu vyšším než 2 A, zajistíme připojením napájecího napětí 3,3 V na operační zesilovač, čímž vytvoříme saturaci. Protože napětí na motoru je řízeno PWM modulací, je modulován i proud motorem. Je více možností odstranění modulační frekvence z měřeného proudu. Můžeme zvýšit indukčnost motoru zapojením cívky do série tak, aby se

signál vyfiltroval již při průchodu motorem, nebo v tomto návrhu je zařazen RC článek typu dolní propust na výstup operačního zesilovače. Při použití RC článku musíme v návrhu regulátoru počítat z fázovým zpožděním, které tím vzniká. Na analogové vstupy arduina je pro porovnání přiveden filtrovaný i nefiltrovaný signál.

Shield dále obsahuje konektor k připojení napájecího napětí společného pro motor i arduino a ochranou diodu proti přepólování. Dále konektory k připojení motoru a enkodérů. Nakonec je zde CPLD modul, na který jsou přivedeny výstupy enkodérů a několik pinů modulu je vyvedeno na digitální piny arduina s možností je buďto připojit cínovou propojkou, nebo ne. Celé schéma je na obrázku 3.2.



Obrázek 3.2. Schéma shieldu pro arduino

Název	Arduino pin	Funkce
DIR1	12	nastavení H-můstku
DIR2	13	nastavení H-můstku
PWM	11	řízení napětí na motoru
CUR_F	A0	filtrované měření proudu
CUR	A1	měření proudu
2-10	2-10	propojení arduina a CPLD modulu
ENCA-D		výstupy enkodérů
OUTA,B		výstupy H-můstku

Tabulka 3.2. Rozložení pinů shieldu

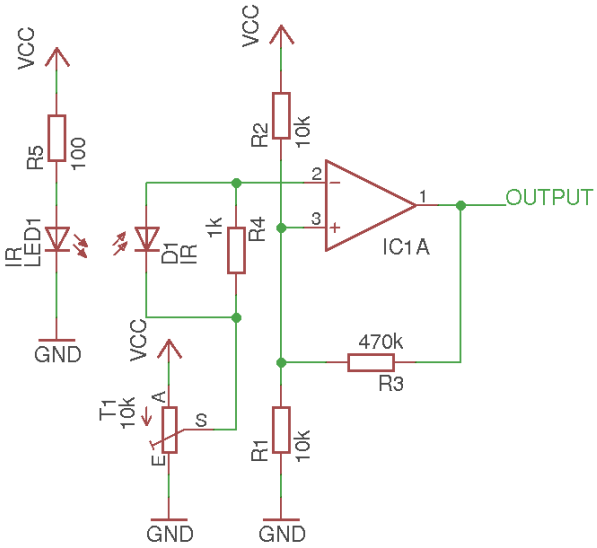
Název	Množství	Hodnota	Pouzdro
U1	1	L298	MULTIWATT-15
U2	1	MCP6002-I/P	DIL08
U3	1	CPLD modul	
D1-4	4	1N4004	
D5	1	1N5401RL	
R1, R2	2	1 kΩ	
R3	1	10 kΩ	
R4	1	150 mΩ	
C1-4	4	100 nF	
C5-7	3	10 μF	
F1	1	RUEF110	

Tabulka 3.3. Seznam použitých součástek na shield, neuvedená pouzdra součástek jsou standardní pro klasickou zástavbu skrz desku

3.2.3 Enkodér polohy zátěže

Původní záměr byl měřit polohu zátěže pomocí optického inkrementálního senzoru HEDS-9000 s kódovacím kolečkem s 500 značkami, vytisknutým na laserové tiskárně na průhlednou fólii. Ukázalo se, že tisk s takovým rozlišením je problematický a s menším rozlišením senzor díky zabudované logice zpracování signálů nefunguje. Další problém by byl upevnění kolečka a senzoru s dostatečnou přesností, proto bylo zvoleno řešení zhotovit celý senzor ze základních komponent. To sice dosahuje menšího rozlišení, ale umožňuje větší volnost nastavení správné funkce.

Senzor pracuje na principu snímání záření emitovaného LED diodou pomocí fotodiody. Abychom odstranili vliv okolního světla, použijeme diody pracující v infračerveném spektru. Mezi diodami je otáčející se kódovací kolečko. Fotodioda funguje jako zdroj proudu, jehož velikost odpovídá momentálnímu osvětlení. Experimentálně bylo vyzkoušeno, že při zatížení použité fotodiody rezistorem 1 kΩ získáme na rezistoru napětí přibližně přímo úměrné osvětlení fotodiody, v našem případě v rozmezí hodnot $(0 - 0,5)$ V. Analogový signál z fotodiody poté digitalizueme pomocí Schmittova klopného obvodu, který sestavíme z operačního zesilovače a několika odporů. Celé schéma jednoho kanálu senzoru je na obrázku 3.3. Tyto kanály jsou dva, nastavené tak, aby z nich signály přicházely podle obrázku 2.7.



Obrázek 3.3. Schéma jednoho kanálu enkodéru

Odpory R1, R2, R3 nastavují pásmo necitlivosti klopného odvodu na hodnotu zhruba $V_{cc}/2 \pm 0,005V_{cc}$, napájecí napětí je $V_{cc} = 5$ V. Trimr T1 nastavíme tak, aby napětí generované fotodiodou na vstupu operačního zesilovače nabývalo hodnot pokud možno symetricky kolem úrovně $V_{cc}/2$. Odpor R5 pouze omezuje velikost proudu LED diodou.

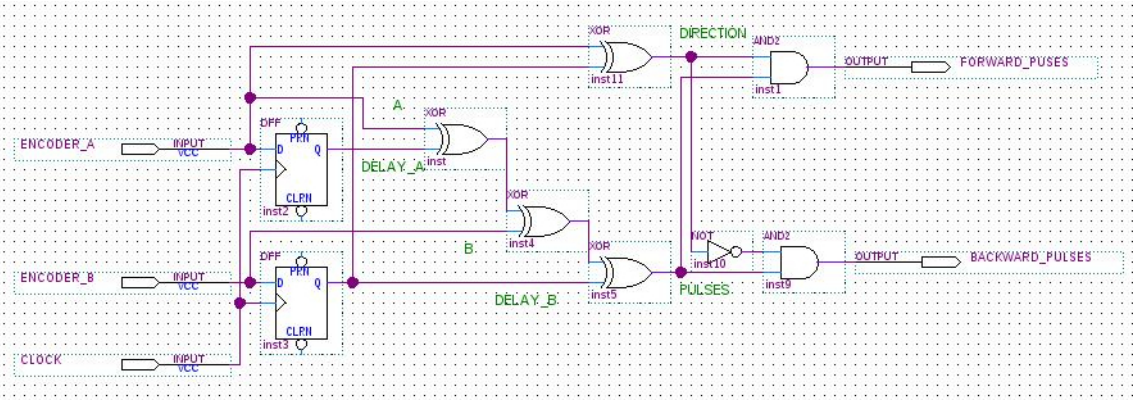
3.2.4 Kvadraturní dekodér

Kvadraturní dekodér realizujeme logickým obvodem uvnitř CPLD čipu. Od implementované logiky požadujeme čítání impulzů s rozlišením směru otáčení. Vstupem tedy jsou 2 signály kvadraturního enkodéru, výstup dekodéru můžeme navrhnout více způsoby v závislosti na možnostech arduina, ale vždy budou také 2. Protože impulsy enkodéru přicházejí nepravidelně a potřebujeme je zaznamenat všechny, použijeme možnost procesoru přerušení hlavního programu na základě změny vstupního signálu, tzv. *interrupt*. Uvnitř procesoru je interrupt realizován analogovým komparátorem, kde porovnáváme vstupní hodnotu s jinou definovanou a při změně výstupu komparátoru dojde k vykonání části programu zvané *interrupt service routine* (ISR). Bližší informace lze nalézt v dokumentaci k danému procesoru, vysvětlení použití interruptu na arduinu je popsáno v [8], společně s výčtem pinů, které funkci interrupt umožňují. Zpravidla požadujeme aby ISR proběhla co nejrychleji a příliš neomezovala hlavní program procesoru.

Výstupy dekodéru pro zpracování arduinem Due je nejlepší navrhnout tak, aby na jednom pinu přicházely impulsy při otáčení v kladném směru, na druhém v záporném. Ve dvou ISR poté stačí inkrementovat, nebo dekrementovat čítač pulsů. Chceme-li použít arduino Uno ke zpracování dvou enkodérů, musíme výstupy logiky navrhnou jinak, protože arduino Uno má pouze 2 interrupt piny, zatímco předešlý návrh vyžaduje 4 pro dva enkodéry. V tomto případě přivedeme na interrupt pin všechny impulsy enkodéru a na druhý standardní pin přivedeme buď logickou 0, nebo logickou 1 v závislosti na směru otáčení. V ISR pak musíme vyčíst směr otáčení a až podle toho inkrementovat, nebo dekrementovat čítač.

Podrobně popsaný návrh dekodéru je v [9] i s příkladem VHDL kódu. Schéma logiky je na obrázku 3.4. Jedná se o implementaci tabulky 2.1. K vyhodnocení potřebujeme 2 poslední hodnoty enkodéru, proto jsou v obvodu zařazeny jednobitové paměti hradla D. Detekci směru otáčení realizujeme hradlem XOR, generování pulsů realizujeme také několika hradly XOR, v tomto případě zapojenými jako funkce modulo 2. To zajistí

hodnotu log 1 na výstupu po dobu 1 hodinového pulsu při změně libovolného vstupu. Hradla AND slouží pro směrování pulsů do různých výstupů v závislosti na směru otáčení. Protože signály z enkodéru přicházejí asynchronně, je vhodné ještě zařadit další hradla D na vstupy, pro synchronizaci s hodinovým signálem, jak je také popsáno v [9].

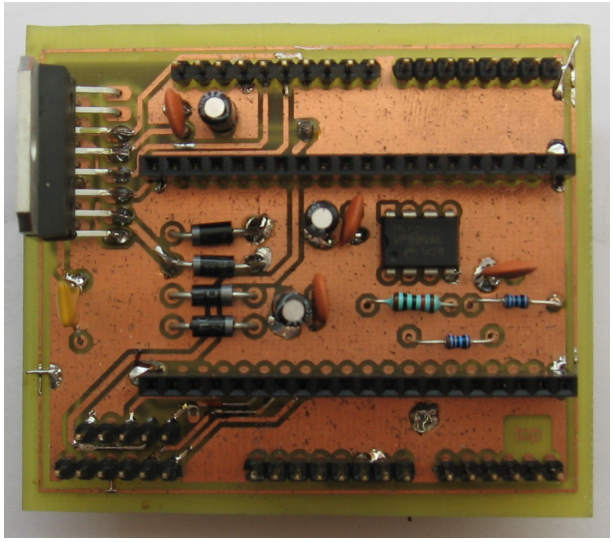


Obrázek 3.4. Schéma logiky zpracování enkodéru

3.3 Realizace návrhu

Konstrukci přípravku můžeme navrhnout buď tak, aby celé zařízení bylo co nejmenší a kompaktní, ale jelikož se jedná o experimentální zařízení, je vhodnější mít dobrý přístup ke všem částem přípravku, především k pinům arduina, kde mnohdy potřebujeme měřícím přístrojem ověřit průběhy napětí.

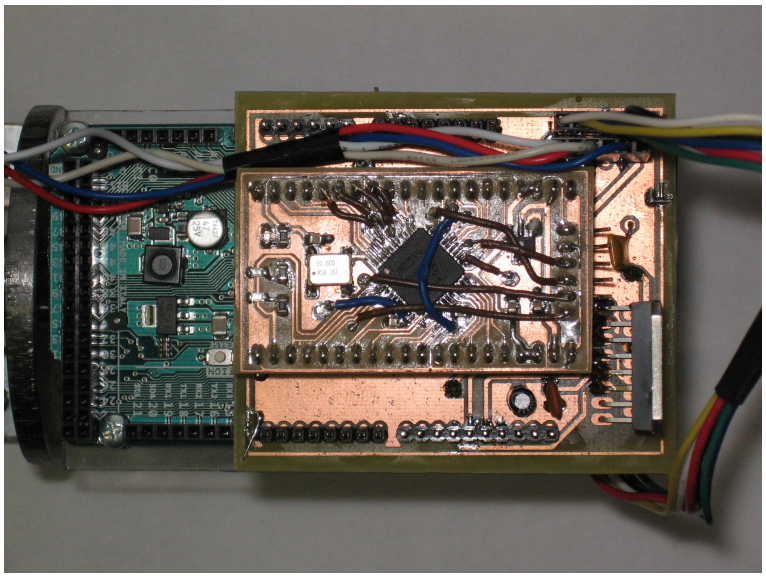
Na obrázku 3.5 je deska shieldu pro řízení motorů. Jedná se pouze o prvotní návrh, na kterém byly odlaďovány chyby. Oproti konečnému návrhu 3.2 zde například chybí konektor pro napájení motoru s ochranou diodou proti přepólování, protože bylo zamýšleno napájet motor přes konektor na arduinu, jenže ochranná dioda arduina není dimenzovaná na proudy potřebné k napájení motoru.



Obrázek 3.5. Shield dro arduino

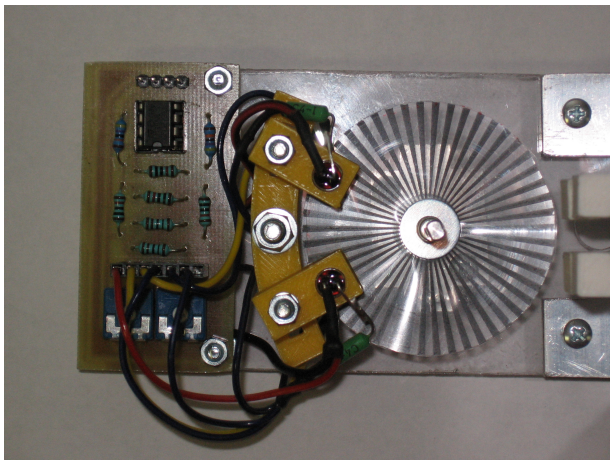
Obrázek 3.6 ukazuje arduino osazené shieldem i CPLD modulem. CPLD modul byl opět vytvořen provizorně za účelem odladění chyb návrhu, které zde nebyly objeveny

žádné. Plošný spoj modulu byl navržen oboustranný, ale zhotoven jako jednostranný s propojkami. Chladič H-můstku není potřeba, protože proudy obvodem v této aplikaci nejsou příliš vysoké.



Obrázek 3.6. Arduino se shield-em a CPLD modulem

Další obrázek 3.7 znázorňuje provedení zhotoveného enkodéru. Na hřídeli je připevněno kódovací kolečko s 50 značkami. Po obvodu kolečka jsou pomocí držáků vytištěných na 3D tiskárně připevněny 2 páry LED dioda, fotodioda. Držáky vyzařujících diod obsahují štěrbiny široké 1 mm, pro zúžení paprsku na rozměry srovnatelné s šířkou značek na kódovacím kolečku. Odtud vedou kabely zakončené konektorem k připojení diod na desku se Schmittovými klopnými obvody. Napájení a výstupy obou kanálů enkodéru jsou přístupné přes další konektor na desce.



Obrázek 3.7. Snímač polohy zátěže

Kompletní přípravek již byl představen v úvodu práce, obrázek 2.1, pouze zde doplníme několik konstrukčních detailů. Nosná konstrukce je řešena plexisklovými deskami a šrouby, pokud možno co nejjednodušeji a se snadnou montáží. Jako zátěž je použit železný disk o průměru zhruba 6 cm a tloušťce 5 mm. Držák k připevnění motoru a nástavec na hřídel byly pořízeny od Pololu společně s motorem. Brzda zátěže je opět vytištěná na 3D tiskárně. Jemnost regulace síly stisku čelistí brzdy záleží na tuhosti pružiny vložené mezi čelist a dotahovací matku.

3.4 Programování Arduina

Na závěr této kapitoly uvedu poznatky o programování arduina a popis vytvořených podpůrných funkcí.

Nejsnáze se arduino programuje pomocí ArduinoIDE v C/C++ s využitím knihovny *Arduino.h*. Pro programování řídicích algoritmů a zobrazování průběhů je pohodlnější použít Simulink. Zde můžeme programovat pomocí propojování simulinkových bloků bez nutnosti psaní textového kódu. Kód v jazyce C vytvoří Simulink automaticky za nás. Simulink také používá knihovnu *Arduino.h*. Velkou výhodou je možnost sledovat průběhy signálů a měnit parametry programu za běhu na platformě arduino Due v takzvaném externím módu. To je hlavní důvod, proč je preferována právě tato platforma. Arduino Uno toto neumožňuje. V rámci práce byly vytvořeny funkce doplňující možnost vykreslování průběhů veličin v Matlabu za běhu programu na Unu. Popis jejich implementace a použití je v následující kapitole 3.4.1. Základy programování arduina v Simulinku jsou popsány v [5].

Používání externího módu má určité limity, se kterými musíme počítat především při zobrazování průběhů veličin. Maximální nastavitelná rychlost přenosu dat sériovou linkou mezi arduinem a počítačem je 115200 baud/s. Dal by se tedy například očekávat spolehlivý přenos jedné hodnoty typu double za jednu milisekundu. Nicméně experimenty ukázaly, že přenos dat přímo do Matlabu je silně nespolehlivý, takže po přenosu několika hodnot ve správných intervalech dochází k náhodně dlouhým pauzám, ve kterých se data nepřenesou žádná. Tento jev nastává při libovolné frekvenci přenosu hodnot. Výraznější tento jev je, pokud chceme zobrazovat více veličin zároveň, z nichž každá se aktualizuje s jinou frekvencí. V takovém případě se informace o pomalejším průběhu nepřenese téměř žádná. Přenos dat do Matlabu při použití externího módu lze realizovat pomocí simulinkových bloků *Scope*, nebo *To Workspace*. Spolehlivější přenos dat do Matlabu, vhodný k jejich dalšímu zpracování, lze realizovat funkcemi vytvořenými prvotně pro arduino Uno popsanými v kapitole 3.4.1.

Budeme potřebovat měnit frekvenci PWM. Přednastavená frekvence 500 Hz, nebo 1000 Hz knihovnou Arduino.h je pro naše účely příliš nízká. Chceme-li nastavit PWM frekvenci arduina Uno, musíme nahlédnout do dokumentace k procesoru ATmega328. Zde nás zajímá především Timer/Counter Control Register (TCCR2A, TCCR2B), který nastavuje parametry časovače. Například příkaz `TCCR2B = _BV(CS20);` nastaví PWM frekvenci pinů 3 a 11 na maximální hodnotu 32 kHz.

Změna PWM frekvence arduina Due je snazší. Stačí provést změnu v konfiguračním souboru *variant.h*. Zde najdeme řádek `#define PWM_FREQUENCY 1000` definující frekvenci v jednotkách Hz na pinech 6–9 a řádek `#define TC_FREQUENCY 1000` definující frekvenci na ostatních PWM pinech a přepíšeme definice na vyhovující hodnoty. Na systému Linux může být umístění souboru *variant.h* například */.arduino15/packages/arduino/hardware/sam/1.6.4/variants/arduino_due_x/variant.h* pro práci s Arduino IDE, nebo */Documents/MATLAB/SupportPackages/R2015a/arduino-1.5.6-r2/hardware/arduino/sam/variants/arduino_due_x/variant.h* pro práci s Matlabem a Simulinkem.

3.4.1 Doplňující funkce pro přenos dat z arduina

Jak bylo zmíněno výše, balíky pro programování arduina pomocí Simulinku neobsahují možnost přenosu dat z arduina Una do Matlabu a použití externího módu arduina Due je nespolehlivé. Proto tato funkce byla doplněna.

V Simuliknu máme možnost použít blok *Serial Transmit*, kterým lze zasílat jednotlivé byty dat na USB port arduina. Můžeme tedy všechna data, která chceme sledovat,

přetypovat matlabovskou funkcí `typecast` na typ `uint8`, seřadit do vektrou a odeslat na USB port, kde je z Matlabu budeme opět načítat. Abychom při čtení poznali začátek datové sekvence, na začátek vektoru dat zařadíme neměnnou sekvenci několika bytů.

Z Matlabu nemůžeme přímo přistupovat k USB portu, proto využijeme možnost volat funkce napsané v jazyce C a vytvoříme takzvaný MEX soubor. Na stránkách MathWorks [10] jsou k dispozici rozsáhlé návody pro tvorbu MEX souborů. Ve své podstatě se jedná pouze o definici vstupů a výstupů vlastní funkce ve formátu, se kterým pracuje Matlab. Vytvořený MEX soubor *readFromArduino.c* implementuje funkci matlabu `readFromArduino(_)`, která zajišťuje otevření USB portu, čtení dat a uzavření portu. Jednotlivé úkony funkce jsou vybírány podle použitých parametrů. Sobuor *readFromArduino.c* je na přiloženém CD a obsahuje i popis použití. Funkci musíme nejprve v Matlabu zkompilovat příkazem `mex readFromArduino.c`.

S možností číst data přímo z USB portu již jednoduše vytvoříme funkci, nebo skript Matlabu, který uloží data do matic, nebo rovnou vytvoří grafy. Příklad implementace funkce pro vykreslování dat v reálném čase za běhu arduina je na přiloženém CD pod názvem *arduinoGraf.m*. Musíme pouze dát pozor, abychom data četli ve správném pořadí, v jakém jsou posílána a je nezbytné ošetřit uzavření portu po ukončení skriptu i v případě přerušení chybou. Toho docílíme použitím funkce Matlabu `onCleanup`.

3.4.2 Implementace měření rychlosti

Pro výpočet rychlosti na základě údajů z enkodérů vytvoříme nový simulinkový blok. Postup tvorby bloku je popsán v [2]. Podobně jako se používá MEX soubor k volání C kódu z Matlabu, zde použijeme S-funkci k volání C kódu ze Simulinku. Simulinkový blok *S-Function Builder* poskytuje rozhraní pro relativně snadnou implementaci S-funkcí. Potřebujeme nadefinovat vstupy, výstupy a počáteční inicializaci. Deklarace funkcí načteme jako standardní C knihovny.

Prvotní návrh počítal s implementací odhadu rychlosti algoritmem CSDT (kapitola 2.5). Ukázalo se ale, že tento algoritmus má jeden nedostatek. V rovnici (4) pro výpočet rychlosti vystupují 3 časové údaje, z nichž každý je určen s nějakou přesností, zbytečně tím tedy roste výsledná chyba výpočtu. Nejzásadněji chybu zvyšuje nepřesnost periody regulační smyčky. Nemáme zaručeno, že smyčka bude vždy trvat stejnou dobu, zvláště pokud implementujeme složitější výpočet, a zároveň zasíláme data k zobrazení. Trvání smyček s periodou 1 ms a kratších se často zpožďují o těžko předvídatelné intervaly.

Lepších výsledků bylo dosaženo přímým měřením doby mezi impulsy. S každým impulsem se zároveň uloží čas příchodu s přesností na mikrosekundy. Výpočet rychlosti se poté vykonává podle potřeby regulátoru a použije se k němu aktuální údaj polohy a údaj použitý k předchozímu výpočtu. Výsledný princip je stejný jako u metody CSDT, ale odstraníme vliv nedodržování trvání regulačních smyček. Příklad implementace čtení polohy a výpočet rychlosti je na přiloženém CD v souboru `encoder_compute.h`. Vzhledem k jednoduchosti a malému rozsahu funkcí je implementace zapsána přímo v hlavičkovém souboru a je možné tento souboru použít jako knihovnu vlastního simulinkového bloku.

Kapitola 4
Experimenty na přípravku

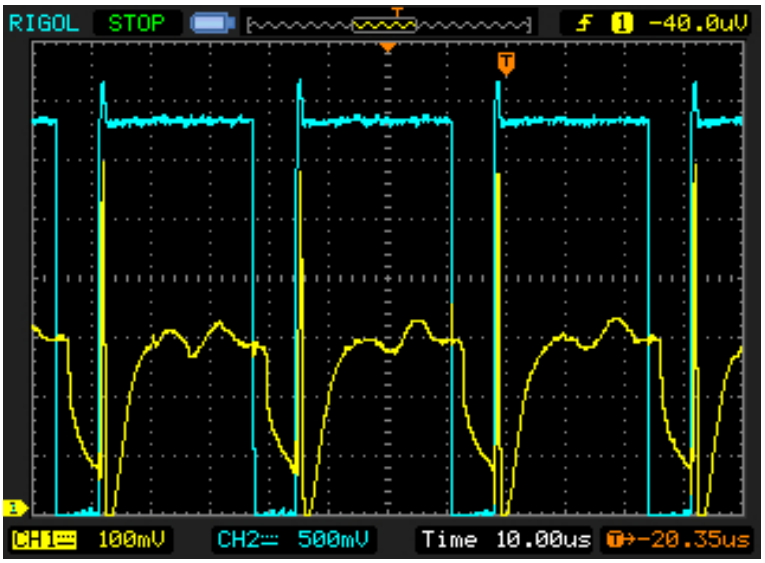
Tato část práce se zabývá ověřením správné funkce navrženého přípravku a demonstrace jeho použití. Většina popsaných experimentů byla provedena dvakrát, poprvé ve fázi návrhu s použitím výše uvedených vývojových desek, podruhé na hotovém přípravku. Uvedena zde jsou pouze měření na přípravku.

4.1 Test vstupů a výstupů

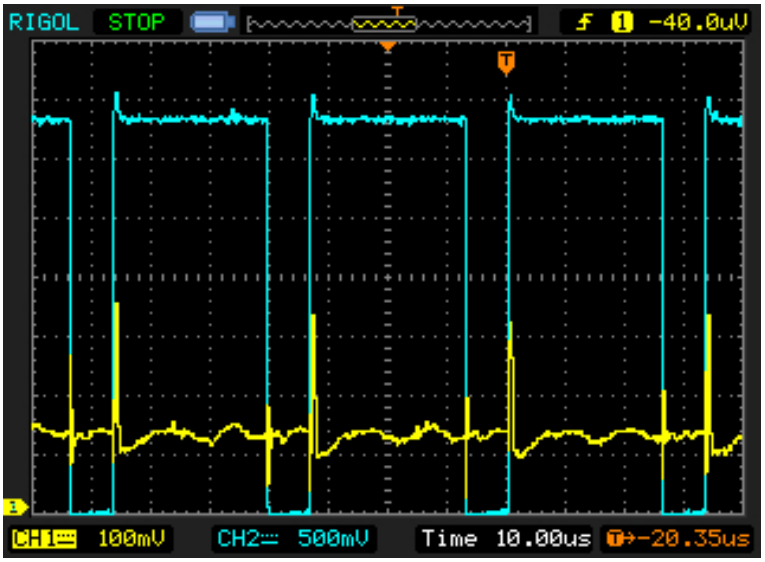
Prvním experimentem chceme ověřit správnou funkci vytvořené elektroniky. Pouze nastavíme napájecí napětí, piny řízení H-můstku a frekvenci PWM modulace na tyto úrovně:

$$\text{DIR1} = 1,\ \text{DIR2} = 0,\ \text{PWM} = 200,\ \text{VIN} = 6\ \text{V}, f_{\text{PWM}} = 30\ \text{kHz}\ .$$

Osciloskopem a voltmetrem změříme všechny výstupy. Výsledky měření osciloskopem jsou na následujících obrázcích.



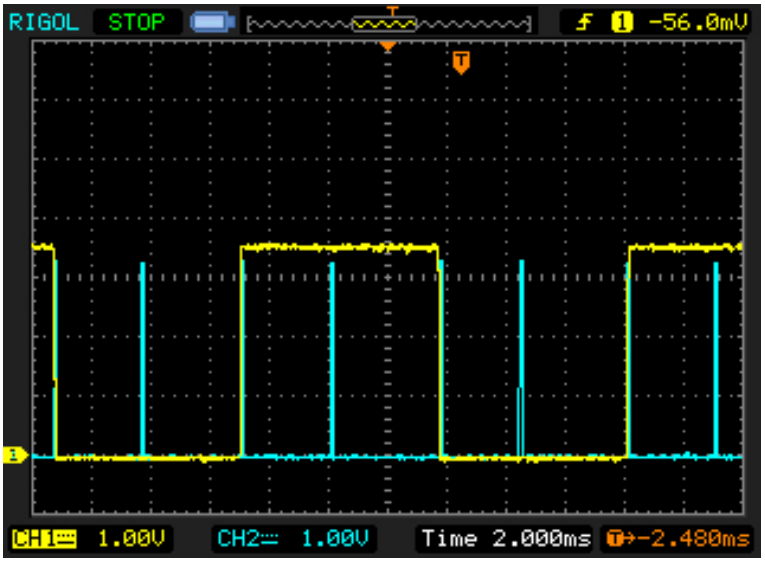
Obrázek 4.1. CH1 výstup měření proudu (pin CUR), CH2 PWM signál (pin PWM)



Obrázek 4.2. CH1 výstup měření proudu s použitím filtru (pin CUR_F), CH2 PWM signál (pin PWM)

Z obrázku 4.1 vidíme, že vliv PWM modulace na měření proudu je značný. Filtrací RC článkem s mezní frekvencí zhruba 10 kHz se signál vylepší 4.2. Pro lepší filtraci by bylo vhodné mezní frekvenci snížit, ale před tím musíme proměřit dynamiku motoru, abychom ze signálu neodfiltrovali i pásmo důležité pro řízení. Dále by bylo zajímavé proměřit fázový posun vznikající na RC článku, ale vidíme, že signál je i nadále zarušený vlivem PWM natolik, že si dovolím vliv fázového posunu dále neuvažovat.

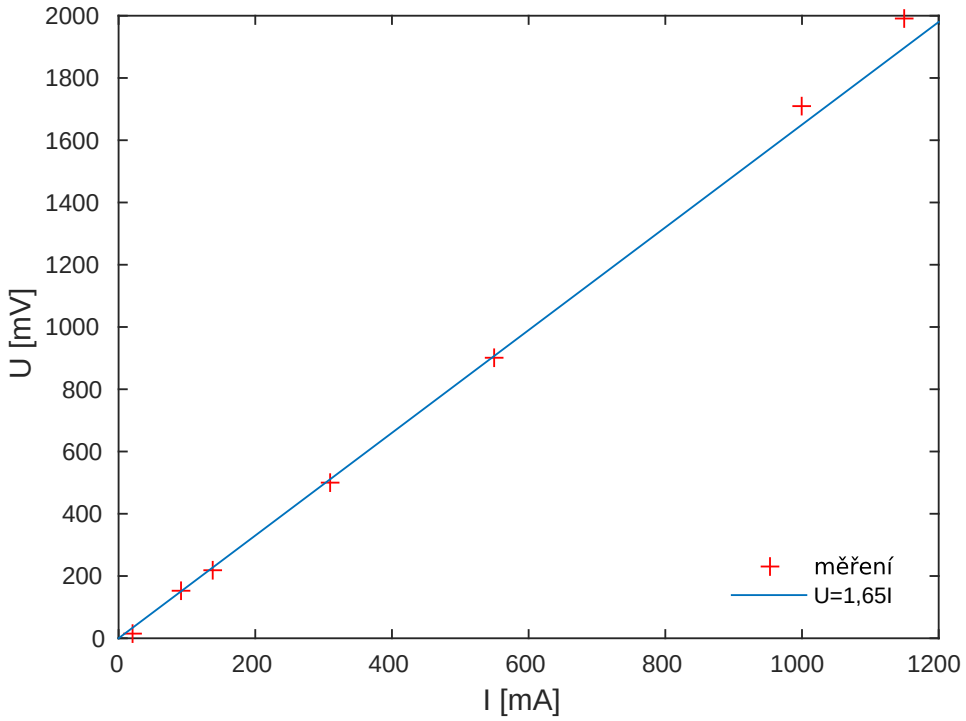
Následující obrázek 4.3 znázorňuje správnou funkci zhotoveného snímače polohy zátěže a dekodéru. Vidíme zde 4 impulzy dekodéru za jednu periodu signálu enkodéru. Druhý kanál nemohl být použitým osciloskopem zároveň zobrazen, ale můžeme si ho představit jako první, posunutý hranami obdélníku do pozice impulzů mimo hrany prvního kanálu. Při měření se zátěž otáčela konstantní rychlostí.



Obrázek 4.3. CH1 1 kanál enkodéru zátěže, CH2 výstup dekodéru

Nakonec proběhlo ověření citlivosti měření proudu. Pomocí ampérmetru a voltmetru byly změřeny střední hodnoty proudu motorem a napětí na výstupu operačního zesilo-

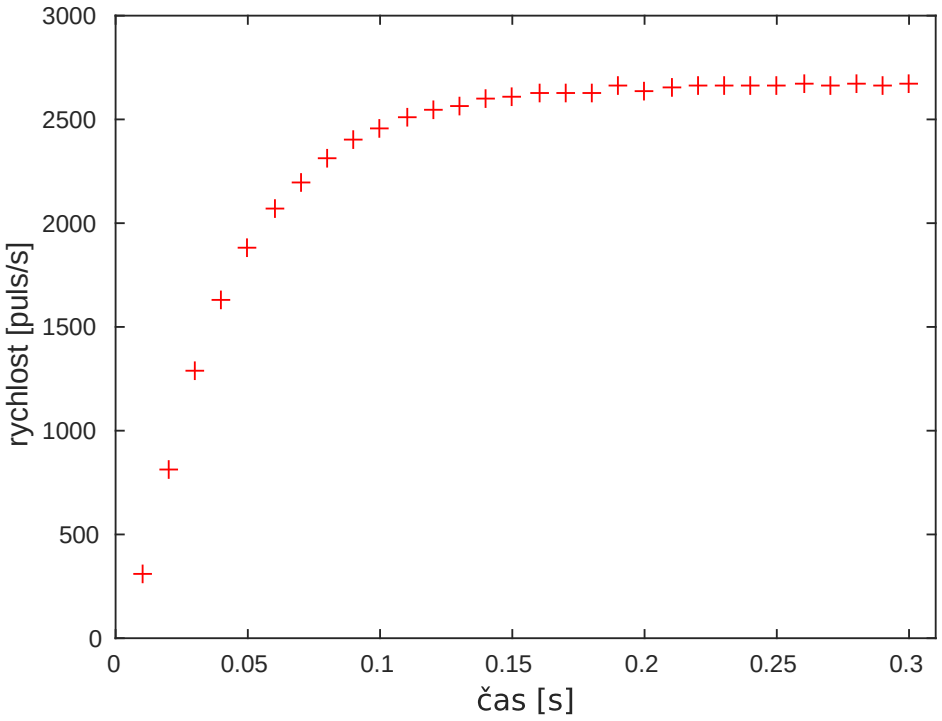
vače. Naměřené hodnoty jsou zaneseny v grafu 4.4, společně s předpokládanou závislostí při citlivosti $1,65\ \frac{\mathrm{V}}{\mathrm{A}}$. Vidíme, že měření alespoň přibližně odpovídá předpokladu.



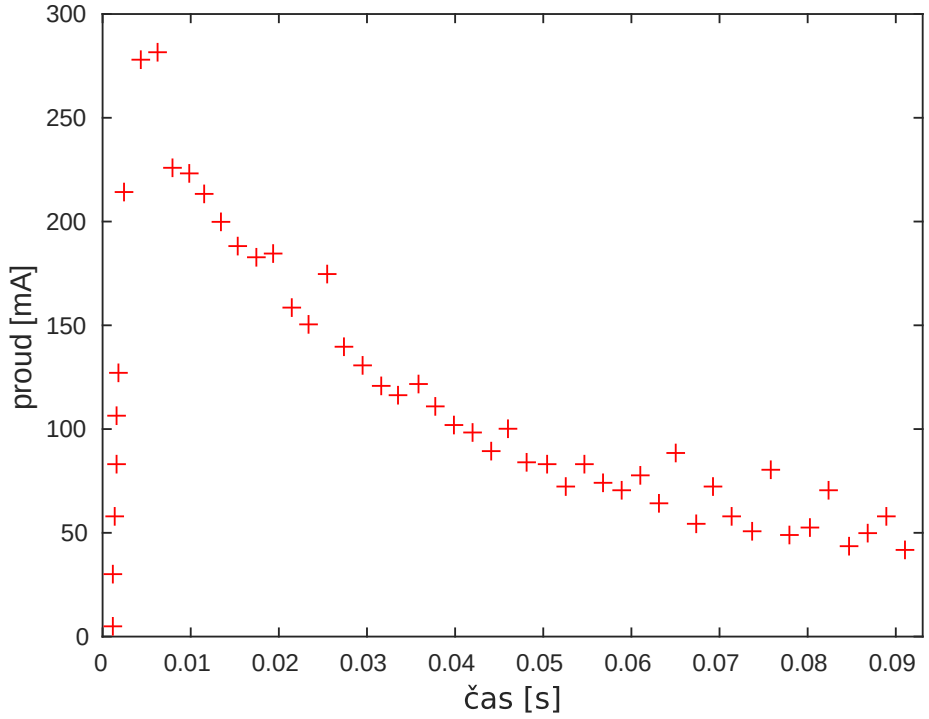
Obrázek 4.4. Ověření správného nastavení citlivosti měření proudu

4.2 Identifikace motoru

Dalším pokusem prozkoumáme chování motoru, za účelem odhadu vhodných hodnot parametrů PWM frekvence, period regulačních smyček řízení a frekvence hodinového signálu dekodéru. Budeme měřit odezvu systému na skok. Systém je v tomto případě nezatížený motor, skok je realizován změnou napětí na motoru z 0 na 5 V a měřené stavy jsou proud motorem a rychlost otáčení hřídele. Naměřené odezvy jsou na grafech 4.5 a 4.6. Z grafu 4.5 vidíme, že k ustálení rychlosti došlo zhruba mezi 0,1 s a 0,2 s. Periodu rychlostní regulační smyčky T_v volíme zhruba setinu až desetinu této doby. Změny proudu motorem jsou rychlejší. Na grafu 4.6 vidíme rychlý náběh proudu trvající řádově milisekundy, periodu proudové smyčky T_i opět volíme alespoň desetkrát nižší, než trvání nejrychlejší změny stavu systému.



Obrázek 4.5. Odezva rychlosti otáčení nezatíženého motoru na skok 5 V



Obrázek 4.6. Odezva proudu nezatíženým motorem na skok 5 V

Další parametr, který potřebujeme nastavit je frekvence PWM signálu f_{PWM}. Potřebujeme, aby byla vyšší, než frekvence změn nejrychlejších stavů systému, aby bylo možné modulaci odfiltrovat v místech, kde je nežádoucí. Zároveň musí být nižší, než

mezní frekvence použitého H-můstku. Nejrychlejším stavem systému je proud motorem, pro který potřebujeme pásmo alespoň do 10 kHz, odhadnuto z grafu 4.6. Z dokumentace k H-můstku L298 vyčteme maximální frekvenci změny napětí na výstupu, při maximálním proudu, 40 kHz, doporučenou 25 kHz. Maximálních proudů zpravidla nebude dosahováno, proto můžeme frekvenci nastavit lehce vyšší, než doporučenou, například $f_{\mathrm{PWM}} = 30$ kHz. Mezní frekvenci filtru měření proudu nastavíme na 10 kHz.

Poslední důležitý parametr je frekvence hodinového signálu dekodéru f_{CLK}. Impulsy z dekodéru mají dobu trvání jednu periodu hodinového signálu. Pokud by perioda byla příliš krátká, mohly by procesoru některé impulsy uniknout, pokud příliš dlouhá, docházelo by k chybnému zpracování dat z enkodérů. f_{CLK} volíme alespoň 10 krát vyšší, než maximální frekvence impulsů dekodéru. Navržený CPLD modul obsahuje oscilátor 50 MHz, nižší frekvenci vytvoříme nejjednodušeji dělením mocninami 2. Z grafu 4.5 vidíme, že frekvence impulsů dekodéru dosahuje maximálně jednotek kHz, f_{CLK} tedy volíme desítky kHz. Přijatelná frekvence hodin může být například $50/2^{11}$ MHz $= 24,4$ kHz. Všechny navržené hodnoty parametrů jsou shrnuty v následující tabulce 4.1.

Parametr	Hodnota	Význam
T_i	$(0,1 - 1)$ ms	perioda proudové regulační smyčky
T_v	$(1 - 10)$ ms	perioda rychlostní regulační smyčky
f_{PWM}	30 kHz	frekvence PWM modulace
f_{CLK}	24,4 kHz	frekvence hodinového signálu dekodéru

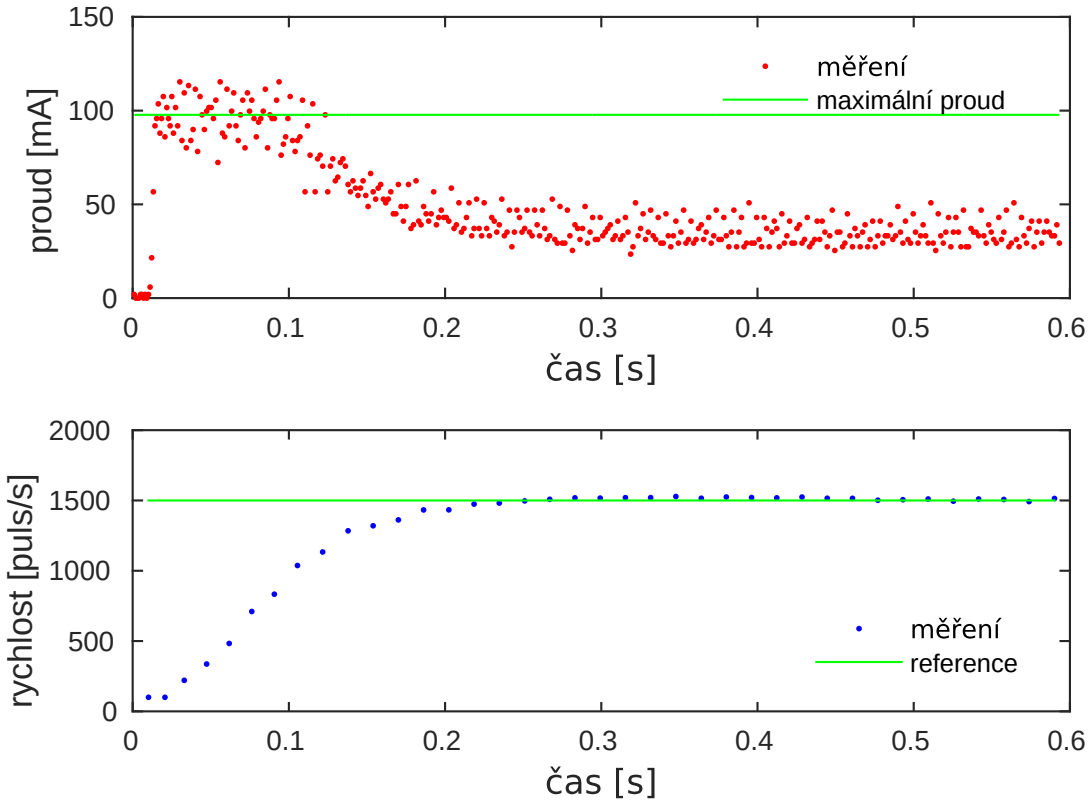
Tabulka 4.1. Odhadnuté parametry pro řízení

4.3 Řízení motoru

Poslední experiment demonstruje použitelnost přípravku k zamýšlenému účelu výuky automatického řízení. Původní záměr demonstrovat funkci kaskádní regulací rychlosti otáčení zátěže, popsané v 2.2, nemohl být realizován z důvodu nesprávné funkce snímače polohy zátěže. Rozbor tohoto problému je v následující kapitole. Omezíme se tedy pouze na řízení proudu motorem a rychlosti otáčení nezatíženého motoru.

Struktura testovaného regulátoru odpovídá obrázku 2.3. Pro regulaci rychlostní smyčky byl zvolen PI regulátor, s omezením maximálního výstupu, který udává referenci proudové smyčky. Ta je také řízena PI regulátorem. Implementace proběhla v Simulinku i v ArduinoIDE se srovnatelnými výsledky. Implementace v Simulinku je sice rychlá a pohodlná, nicméně je doprovázena určitými nepříjemnostmi, které jsou rozebrány v následující kapitole. Ladění probíhalo pomocí experimentální Ziegler-Nicholsovy metody. Na grafech 4.7 vidíme naměřené odezvy proudu a rychlosti na jednotkový skok vstupu. V tomto případě byla perioda proudové smyčky 1 ms, rychlostní 10 ms. Hodnoty referenční rychlosti a maximálního povoleného proudu jsou zaneseny v grafech. Vidíme správnou funkci omezení maximálního proudu v rámci přesnosti měření. K ustálení rychlosti došlo zhruba za 0,2 s bez překmitu.

Z tabulky 4.1 plyne, že regulační periody mohli být navrženy zhruba 10krát kratší, jenže při tak rychlých smyčkách již dochází k zahlcení sériové linky komunikace, nemůžeme tedy správně naměřit odezvy. Správnou funkci regulace proudu s použitím rychlejší smyčky lze pozorovat osciloskopem, regulaci rychlosti můžeme kontrolovat přímo okem.



Obrázek 4.7. Průběhy rychlosti a proudu při kaskádní regulaci na rychlost 1500 puls/s, s omezením maximálního proudu motorem

4.4 Nedořešené problémy

Experimenty odhalily také několik problémů, které by ještě bylo třeba vyřešit.

První se týká zhotoveného senzoru polohy zátěže. Ačkoliv samotný test výstupu enkodéru a dekodéru (obrázek 4.3) ukazuje na správnou funkci, při čítání arduinem dochází k detekci několika impulzů namísto jednoho. Příčina tohoto jevu nejspíše spočívá v použití nevhodného operačního zesilovače, který není typu rail-to-rail. Při přechodu stavů výstupu klopného obvodu dochází k vysokofrekvenčním zákmitům. Ty se skrz CPLD modul přenášejí až na interrupt vstup arduina, kde způsobují chybné čítání. Problém by mělo odstranit použití vhodného operačního zesilovače, který momentálně nebyl k dispozici.

Druhá nepříjemnost souvisí s programováním regulátorů v Simulinku. Pokud implementujeme jednoduchý PID regulátor jednoho stavu pracující s nějakou periodou smyčky, vše funguje dobře, ale pokud v programu vytvoříme více regulačních smyček vzájemně propojených, s různými periodami, může docházet k nepředvídatelným výpadkům funkcí různých bloků. Přizpůsobení částí programu s různými periodami by měl řešit simulinkový blok *Rate Transition*, jenže ani s jeho použitím není vždy zaručena správná funkce. Tento problém nejspíše lze vyřešit vhodným nastavením parametrů bloků Simulinku. Bohužel se mi zatím nepovedlo zjistit jakým, ani při hledání odpovědi na fórech [10].

Kapitola 5
Zhodnocení

Cílem práce bylo navrhnout levný, snadno zhotovitelný, experimentální přípravek pro účely testování řídicích algoritmů implementovaných v prostředí Simulink. Vytvořený návrh je jednou z možností provedení. Hlavní výhoda tohoto řešení je jeho vysoká variabilita. Jednak ve volnosti výběru, zda jako hlavní platformu použijeme arduino Due, nebo levnější, ale také dostatečně výkonné arduino Uno. Dále v možnosti konfigurovat vlastní implementace logických polí na CPLD modulu i pro komplikovanější úkony, než je předvedené zpracování signálů enkodéru. Poslední výhoda je případné snadné zjednodušení celého návrhu pro ty, kterým se nelíbí takové množství elektroniky. Můžeme například odstranit CPLD modul a jeho funkci nahradit výpočty na procesoru. Snímač polohy zátěže lze také zjednodušit odstraněním klopných obvodů a analogové signály z fotodiod zpracovávat procesorem. Nicméně dle mého názoru alespoň základní znalost návrhu, výroby a zprovoznění elektrických obvodů patří ke studiu na FEL, proto na přiloženém CD jsou dostupné pouze kompletní podklady pro výrobu CPLD modulu, vytvořené v prostředí Eagle. Návrh shieldu je přenechán každému zájemci o jeho výrobu. Pravděpodobně každého napadne mnoho úprav a vylepšení představeného návrhu, což lze také považovat za jednu z jeho výhod.

Literatura

[1] Michael A. Johnson, Mohammad H. Moradi. *PID Control, New Identification and Design Methods.* Springer-Verlag London, 2005.

[2] *Writing a Simulink Device Driver block: a step by step guide.* The MathWorks, Inc., 2015.

[3] Martin Gurtner. *Pokročilé metody návrhu velmi přesného řízení pohybu.* ČVUT FEL Katedra řídicí techniky, 2013.

[4] Slow, Mixed and Fast Decay Modes `http://ebldc.com/?p=86`

[5] Getting Started with Arduino Hardware `http://www.mathworks.com/help/supportpkg/arduino/examples/getting-started-with-arduino-hardware.html`

[6] Altera MAX3000A CPLD breakout v1 `http://dangerousprototypes.com/docs/Jose:_Altera_MAX3000A_CPLD_breakout_v1`

[7] JTAG – A technical overview `http://www.xjtag.com/support-jtag/jtag-technical-guide.php`

[8] Arduino attachInterrupt() `https://www.arduino.cc/en/Reference/AttachInterrupt`

[9] Quadrature Decoder `http://www.fpga4fun.com/QuadratureDecoder.html`

[10] MathWorks `http://www.mathworks.com/`

[11] Introduction to Feedback Control Theory, MIT `https://www.edx.org/course/introduction-feedback-control-theory-mitx-6-302-0x`

[12] Quanser `http://www.quanser.com`

[13] INTECO `http://www.inteco.com.pl`

[14] MLAB `http://www.mlab.cz/index.cs.html`

Příloha A
Zkratky

CSDT	Constant sample-time digital tachometer
CPLD	Complex programmable logic device
FEL	Fakulta elektrotechnická
ISR	Interrupt Service Routine
JTAG	Joint test action group
LVC	Low velocity compensation
PID	Proporcionální, integrační a derivační
PWM	Pulse-width modulation

Příloha B
Obsah přiloženého CD

text/	Tato práce ve formátu pdf
simulink/	Příklady simulinkových schémat a podpůrné funkce
eagle/	Schéma a deska plošných spojů pro CPLD modul
dokumentace/	Dokumentace důležitých součástek

Příloha C
Zadání této práce

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jiří Z á h o r a

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Experimentální platforma pro rychlé prototypování řídicích algoritmů pro stejnosměrný motor s permanentním magnetem

Pokyny pro vypracování:

Navrhněte a postavte experimentální platformou pro rychlé prototypování řídicích algoritmů pro běžný stejnosměrný motor s permanentním magnetem.

1. Platforma nechť obsahuje nějaký malý dobře dostupný modelářský motor s převodovkou a vyměnitelnou rotační zátěží (diskem), případně i s nastavitelným mechanickým třením (brzdou) či dokonce pružným prvkem (torzní pružina, pružný převodový řemen).
2. Měřen nechť je elektrický proud i úhel natočení hřídele i zátěže (pomocí inkrementálních snímačů).
3. Generované řídicí napětí modulováno pomocí PWM.
4. Platforma nechť je malá, levná, přenosná, a fungující i bez připojení k PC (PC pouze pro generování kódu a zpracování naměřených dat).
5. Platforma nechť je postavena na některém z dnes rozšířených jednodeskových počítačů jako je Arduino, Raspberry Pi, mBed či BeagleBone, pro které je možné generovat kód v prostředí Matlab/Simulink.

Seznam odborné literatury:

[1] J. Zemánek, Z. Hurák: Experimental modular platform for distributed planar manipulation by shaping magnetic field. Sent for publication. 2015.
[2] Z. Hurák, J. Zemánek: Feedback linearization approach to distributed feedback manipulation, In proc. American Control Conference (ACC), Montréal, Canada, pp. 991-996, June 2012. DOI: 10.1109/ACC.2012.6315262.
[3] D. Martinec, I. Herman, Z. Hurák, and M. Šebek: "Wave-absorbing vehicular platoon controller", European Journal of Control, vol. 20, no. 5, pp. 237-248, Sep. 2014.

Vedoucí bakalářské práce: Ing. Zdeněk Hurák, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 16. 12. 2015