VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELIGENT SYSTEMS

ŘÍZENÍ CHAPADLA ROBOTICKÉHO RAMENE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADIM LUŽA

BRNO 2011

Sem vložte zadání...

Sem vložte zadání...

Sem vložte zadání...

Sem vložte zadání...

Sem vložte zadání...

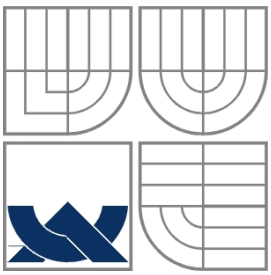
Sem vložte zadání...

Sem vložte zadání...

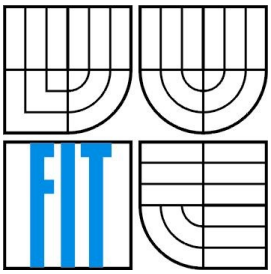
Sem vložte zadání...

Sem vložte zadání...

Sem vložte zadání...



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELIGENT SYSTEMS

ŘÍZENÍ CHAPADLA ROBOTICKÉHO RAMENE

CONTROL OF A ROBOTIC ARM GRIPPER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADIM LUŽA

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. FRANTIŠEK ZBOŘIL CSc.

BRNO 2011

Abstrakt

Práce se zabývá návrhem regulátoru síly pro robotické chapadlo Schunk PG70. V první části popisuje vlastnosti, elektrické zapojení, propojení s řídícím systémem a testování chapadla. V druhé části práce popisuje návrh softwarové regulace síly stisku, implementaci regulátoru a průběh a výsledky experimentů s regulací. Součástí práce je i návrh modifikací pro zpřesnění regulace a návrhy rozšíření demonstrující možnosti dalšího využití regulátoru. Regulace byla v rámci práce vyřešena, ale výsledky experimentů nejsou ideální.

Abstract

This document describes design of regulator of gripping force for robotic gripper Schunk PG70. In the first part of this document there is described electrical connection, interconnection with control system and testing of robotic gripper. The second part document desribes design of software regulator of gripping force, implementation of regulator, process of experimenting with it and results of experiments. Document also contains suggestions for improvements and extentions of regulator in order to gain more precise regulation and advanced functionality. Problem of regulation of gripping force was solved but results of experiments are not perfect.

Klíčová slova

Robotika, regulace síly, řízení, Schunk PG70, Melfa RV-6S

Keywords

Robotics, regulation of force, control, Schunk PG70, Melfa RV-6S

Citace

Luža R.: Řízení chapadla robotického ramene, diplomová práce, Brno, FIT VUT v Brně, 2011.

Řízení chapadla robotického ramene

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Ing. Františka Zbořila CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radim Luža
2011

Poděkování

Rád bych poděkoval vedoucímu práce doc. Ing. Františkovi Zbořilovi a mému kolegovi bc. Michalovi Dudkovi.

Obsah

1 Úvod

Tato práce se zabývá instalací, konfigurací a regulací průmyslového robotického chapadla. Stěžejním bodem práce je regulace síly stisku robotického chapadla, které samo o sobě nemá pro přímou regulaci síly stisku prostředky.

1.1 Náplň práce

Ačkoliv se práce zabývá především regulací síly stisku robotického chapadla, její záběr je poněkud širší. V úvodu je popsán systém, pro který je práce řešena, a účel, kterému má robotické chapadlo sloužit. Z toho potom vyplývají cílové podmínky, které musí navrhovaný regulátor splňovat. Návrh je limitován dostupným hardwarem, nicméně v úvodních podkapitolách jsou popsány i možnosti hardwarových rozšíření, které by mohly vést ke zlepšení vlastností regulace.

Druhá kapitola se zabývá realizací propojení chapadla s ramenem, na kterém je umístěno, a řídicím počítačem. Tato kapitola je rozdělena do čtyř podkapitol. První popisuje způsob připojení napájení chapadla. Druhá popisuje propojení datové cesty a použité rozhraní pro přenos dat. V druhé podkapitole je také zmíněno propojení řídicího počítače s ramenem a popsána souvislost ovládání ramene s ovládáním ruky. Třetí podkapitola se potom zabývá softwarovými nástroji pro řízení chapadla. Popisuje nástroje a dokumenty poskytnuté výrobcem a podrobněji se věnuje knihovně m5api, která je v této práci využívána pro komunikaci s chapadlem. Čtvrtá z podkapitol se zabývá testováním komunikace s chapadlem a rozborem dynamického chování chapadla.

Třetí kapitola se věnuje návrhu řídicího softwaru – tedy samotného regulátoru. Návrh postupuje shora dolů. Nejdříve jsou definovány parametry, které by měla regulace splňovat. Následně je specifikováno rozhraní, kterým bude regulátor komunikovat s okolními systémy. Dále je popsán obecný princip regulace a navrženo blokové schéma. Obecný princip je postupně konkretizován a doplňován o matematické vztahy popisující principy jednotlivých funkčních bloků. Struktura a chování regulátoru jsou popsány formalismem DEVS. Na závěr této kapitoly je popsána struktura souborů zdrojového kódu způsob rozdělení funkcionality jednotlivých bloků regulátoru do těchto souborů.

Čtvrtá kapitola se zabývá implementací. Popisuje použité technologie, programovací jazyky, nástroje a platformy, na kterých je navrhovaný regulátor schopen fungovat, a také závislosti regulátoru – tedy na jakých knihovnách a komponentách implementace regulátoru závisí.

Kapitola pátá se zabývá experimentováním s regulací. U každého z prováděných experimentů popisuje účel experimentu a způsob jeho realizace. Ke každému experimentu jsou dále prezentovány výsledky formou grafu nebo tabulky. Popis každého experimentu je doplněn o diskuzi získaných výsledků. Úkolem diskuze je porovnat reálné výsledky s předpoklady a v případě odlišností určit pravděpodobnou příčinu.

Předposlední kapitola práce se zabývá možnostmi rozšíření regulátoru za účelem zdokonalení regulace mimo možnosti stávajícího řešení a také dalším využitím dat získávaných během regulace. U jednotlivých rozšíření je popsán jejich princip a přínos, který by měla mít. Velká část kapitoly je věnována rozpoznávači materiálu, který byl v rámci návrhu rozšíření implementován a testován.

Poslední kapitolou je závěr. Je to krátká kapitola shrnující průběh práce. Cílem závěru je především zhodnotit navržený a implementovaný regulátor. Dále závěr popisuje překážky, které při vývoji vznikly a hodnotí způsob jejich překonání. V neposlední řadě jsou v závěru popsány i plány budoucího využití a rozvoje projektu.

1.2 Motivace

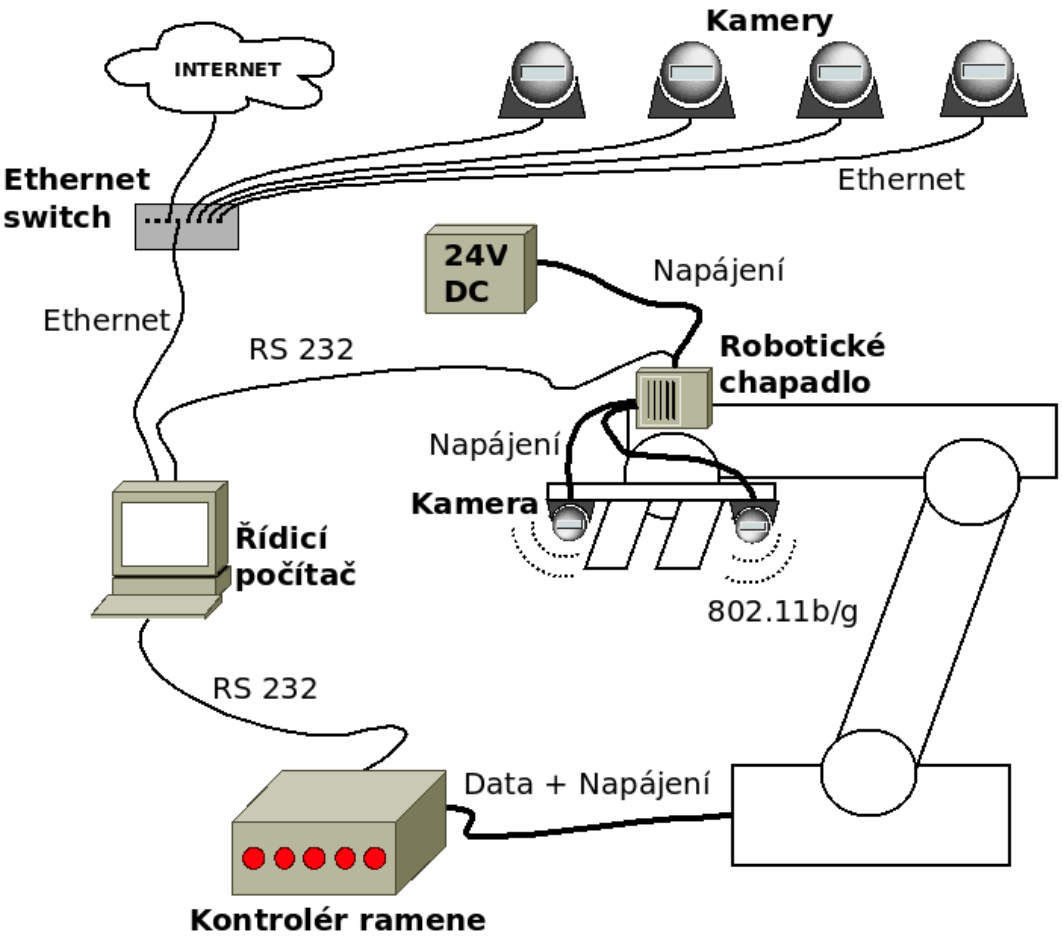
Motivací pro vznik tohoto projektu byla především potřeba regulovat sílu stisku chapadla bez dalších podpůrných snímačů a přídavného hardwaru. Regulace síly stisku je nezbytná právě u akademických a experimentálních úloh, pro které má rameno v budoucnu sloužit, a u kterých je potřeba počítat s potřebou uchopovat neznámé objekty tak, aby objekty nebyly chapadlem nevratně deformovány a po-

škozovány. Další motivací je ochrana samotného chapadla. Přestože má chapadlo možnost nastavit pevnou hranici proudu, kterou ho lze chránit před zničením, je nutné tuto hranici držet poměrně vysoko, aby neomezovala rozběhy chapadla, kdy jeho odběr vzrůstá. Navíc odběr chapadla může být při udržování rychlosti pohybu prstů velmi proměnlivý. Proto by byla sofistikovanější metoda omezení síly stisku chapadla užitečná.

V neposlední řadě je motivací také samotné experimentální řešení regulátoru. Regulace síly stroje pomocí měření proudu do motoru je řešení poměrně nestandardní. Komplikací navíc je v tomto případě vliv vlastního regulátoru chapadla, který zajišťuje funkce jako konstantní rychlost pohybu prstů, nebo plynulý rozběh, a mění okamžitý odběr proudu. Překonat tyto překážky nemusí být snadné. Může se tedy stát, že práce nepovede k uspokojivému řešení. Na druhou stranu v případě úspěchu usnadní použití chapadla pro další úlohy. Užitečné také mohou být výstupy ze samotného měření proudu, které mohou být případně využity i v jiných úlohách, než je regulace síly stisku prstů.

1.3 Popis situace

Současná podoba celého systému robotického ramene je znázorněna na obrázku č.1. Základem je robotické rameno Mitsubishi typu RV-S6 s řídicím kontrolérem CR2B-574. Na pozici komponenty je na tomto rameni umístěno robotické chapadlo Schunk PG70, kterého se tato práce týká především. Rameno svými pohyby zajišťuje přesun chapadla na vhodnou pozici vůči uchopovanému objektu, chapadlo samotné pak zajišťuje uchopení objektu. Celý tento systém je řízen počítačem typu PC se standardním operačním systémem.



Obrázek 1: Náčrt celkového systému robotického ramene.

Do značné míry nezávislý na robotickém rameni a jeho řízení je systém kamer. Těchto kamer je celkem šest – čtyři jsou umístěny nad ramenem a poskytují celkový přehled o poloze ramene vůči ostatním objektům a stěnám operační místnosti. Další dvě kamery jsou potom umístěny přímo na robotickém chapadle. Jsou napájeny ze zdroje chapadla. Tyto dvě kamery představují „oči“ robotického ramene. Míří přímo na prsty chapadla a zobrazují detail uchopovaného objektu a prstů. Účelem kamer je především optický kontakt s ramenem pro vzdáleného operátora. V druhé řadě je tu potom možnost využití kamer k navigaci a ochraně ramene a chapadla před kolizí, případně k detekci nestandardního nebo chybného chování robotických zařízení.

Jak už bylo naznačeno výše, chapadlo Schunk PG70 nemá přímou regulaci síly a cílem této práce je tuto funkci bez nutnosti zásahu do hardwaru doplnit. K regulaci síly je v rámci této práce využíváno měření okamžitého odběru proudu chapadla. Bohužel chapadlo neposkytuje rozhraní pro kontinuální měření proudu. Okamžitý proud chapadlo poskytuje pouze jako odpověď na adekvátní dotaz. Z pohledu chapadla je navíc poskytování informací o aktuálním stavu druhořadé – především se procesor chapadla zabývá řízením prstů, proto není možné žádat okamžitý proud příliš často, jinak chapadlo vyhlásí stav „busy“ a přestane na výzvy reagovat. Navíc komunikace s chapadlem probíhá po sériové lince, takže mezi odesláním dotazu a získáním odpovědi uběhne z hlediska regulace nezanedbatelný časový okamžik. S těmito omezeními je třeba při návrhu regulátoru počítat.

Ačkoliv by bylo efektivnější měřit sílu tenzometry umístěnými v prstech chapadla, byla snaha se tomuto řešení vyhnout. Jedním z důvodů, proč nepoužít tenzometry, je potřeba dalšího ne zcela levného hardwaru – tenzometry odpovídajících tvarů a rozměrů jsou poměrně drahé a pro jejich činnost by byla potřeba další vyhodnocovací jednotka s přesným AD převodníkem. Tato jednotka s vhodným rozhraním by pravděpodobně musela být zakázkově vyrobena a její cena by tedy rovněž nebyla malá. Dalším nepříjemným důsledkem by byly přibývající kabely, které musí být vedeny celým tělem ramene až k chapadlu a omezují pak pohyblivost a spolehlivost celého ramene. S podporou těchto argumentů je pak hlavním důvodem nasazení regulace měřením proudového odběru fakt, že přesnost regulace síly pro výše popsané účely nemusí být vysoká. Zajistit ochranu uchopovaných předmětů a chapadla samotného lze i s chybou regulace síly v řádu desítek procent.

1.4 Popis hardwaru chapadla

Tato podkapitola se věnuje podrobnému popisu robotického chapadla Schunk PG70. Veškeré detaily, které nejsou v této kapitole uvedeny, lze najít v [1].

Schunk PG70 je dvouprsté robotické chapadlo. Je poháněno elektromotorem. Prsty se pohybují synchronně proti sobě. Chapadlo je řízeno polohou – tedy pro rozevření nebo sevření prstů je nutno zadat novou polohu prstů – buďto absolutně, nebo relativně vůči aktuální poloze. Vlastnosti pohybu prstů robotického chapadla lze ovlivnit nastavením rychlosti pohybu prstů a zrychlením, jakým mají prsty na zadanou rychlost zrychlit. Pro případ, kdy prsty nemohou dosáhnout cílové pozice vlivem překážky nebo mechanické závady lze nastavit proudové omezení, při jehož překročení se pohyb prstů zastaví. Pro potřeby zpomalení nebo zastavení pohybu prstů je chapadlo vybaveno elektrickou brzdou.

Z hlediska elektrických parametrů vyžaduje chapadlo napájení stejnosměrným napětím 24V. Toto napětí je vyžadováno jak pro motor chapadla, tak pro řídicí logiku, avšak návod chapadla striktně vyžaduje, aby kvůli proudovým nárazům byly tyto okruhy odděleny. Řídicí jednotka chapadla má sice oproti motoru jen minimální odběr, ale je náchylná na kolísání napětí. Poklesy napětí způsobené proudovými rázy při rozběhu motoru nebo při aktivaci brzdy by mohly způsobit její selhání. Mechanické a elektrické vlastnosti chapadla jsou shrnuty v tabulce č.1.

Chapadlo nabízí tři rozhraní pro propojení s nadřazeným systémem, kde je možné pro propojení použít vždy právě jedno z nich (komunikace více rozhraními současně není podporována). Rozhraní jsou Profibus DP, CAN Bus a RS232. Jedná se o standardní průmyslová rozhraní.

Parametr [jednotka]	Hodnota
Maximální dráha jednoho prstu [mm]	35
Maximální síla stisku (při 2.3A) [N]	200
Maximální rychlost pohybu prstů [mm/s]	82
Maximální zrychlení [mm/s^2]	328
Maximální chyba opakování pozice [mm]	0,05
Napájecí napětí motoru – DC [V]	24
Napájecí napětí vnitřní logiky – DC [V]	24
Nominální odběr proudu [A]	2,3
Maximální odběr proudu [A]	7,5
Převodový poměr	1:2
Rozlišení inkrementu pozice	2000

Tabulka 1: Parametry robotického chapadla Schunk PG70

Profibus je průmyslová sběrnice standardizovaná normou EN50170. Jedná se o sériovou sběrnici využívající jako přenosové médium buďto kroucenou dvojlinku (včetně nízkoúrovňového protokolu RS485), optické vlákno, nebo proudovou smyčku IEC 1158-2 pro prostředí s nebezpečím výbuchu. Přenosová rychlost sběrnice je nepřímo úměrná délce sběrnice – pro délku 1,2km a více je přenosová rychlost 9kb/s, pro délku do 100m až 12Mb/s. Profibus využívá vrstvového modelu ISO/OSI, implementuje však jen tři vrstvy – fyzickou, linkovou a aplikační. Fyzická vrstva zahrnuje komunikační média – optický kabel, RS485 a IEC 1158-2. Linková vrstva určuje způsob komunikace a sdílení sběrnice mezi zařízeními. Zařízení na sběrnici komunikují v režimu master-slave, kdy master vyzývá slave a slave odpovídá (tzv. polling). Na sběrnici může být více zařízení v režimu master – ta si pak předávají sběrnici algoritmem token passing, kdy si zařízení předávají v kruhu „právo mluvit“ – token s tím, že ten kdo mluvit nepotřebuje, jen předá token dál. Profibus nabízí dvě varianty sběrnice – Profibus DP a Profibus PA. Profibus DP je základní varianta nabízející různá přenosová média a různé rychlosti komunikace. Používá se pro automatizaci v budovách a běžném průmyslovém prostředí. Profibus PA je naopak varianta nabízející jediné komunikační médium IEC 1158-2 a pevnou přenosovou rychlost 31,25kb/s. Tato varianta je navržena pro nebezpečná a výbušná prostředí. Sběrnice umožňuje stavět strukturovanou infrastrukturu, kdy jsou například segmenty Profibus PA propojeny sběrnicí Profibus DP. Podrobnější informace o Profibus lze najít v [4].

CAN Bus je sběrnice původně navržená pro propojení jednotek v automobilech. Rozšířila se však i do průmyslu. Sběrnice je propojena symetrickým nebo nesymetrickým párem vodičů dle normy ISO11989. Přenosová rychlost na sběrnici CAN Bus je 1Mb/s při dodržení maximální délky sběrnice 25m. Sběrnice CAN Bus nepoužívá přímo adresování zařízení. Namísto toho má každý typ zprávy jedinečný identifikátor, který určuje jak příjemce a účel zprávy, tak prioritu zprávy. Tímto způsobem je možné posílat zprávy i více než jednomu zařízení. Řízení přístupu ke sběrnici je zajištěnou prioritou zprávy. Určení priority je realizováno bitovou arbitráží, kdy jednotlivá zařízení při logické úrovni bitu 0 spínají signálový vodič do země. Každé zařízení vysílá a zároveň poslouchá, co přijalo. Zprávy s nižším identifikátorem mají v identifikátoru dříve bity s hodnotou 0. Jakmile zařízení zjistí, že vysílá v identifikátoru 1 a slyší 0, přestane vysílat a tím uvolní sběrnici prioritní zprávě. Tento mechanizmus zařízením umožňuje arbitráž bez nutnosti znát ostatní zařízení na sběrnici a zároveň při něm nevznikají kolize znehodnocující přenášená data. Pro spolehlivou komunikaci využívá sběrnice kontrolní součty rámců a kontrolu struktury rámců. Z hlediska role při komunikaci na sběrnici jsou si zařízení v principu rovna. Postavení zařízení vůči ostatním je určeno nepřímo hodnotou identifikátoru zpráv, které zařízení vysílá. Další informace a popis protokolu lze najít v [5].

Posledním z dostupných rozhraní chapadla je RS232 [6]. Na rozdíl od předchozích rozhraní RS232 není sběrnice, ale sériová linka propojující vždy jen dvě zařízení. Z výše popsaných rozhraní se také jedná o rozhraní nejstarší. RS232 bylo původně navrženo jako univerzální rozhraní pro komunikaci dvou zařízení do vzdálenosti 20m. Výhodou RS232 je možnost toto rozhraní poměrně snadno konvertovat na sériové linky používané v mikrokontrolérech. Jako přenosové médium používá RS232 ideálně stíněný metalický kabel s 25 vodiči, alternativou je přenos po 9 vodičích, dnes používaný ve většině zařízení. V praxi se však na menší vzdálenosti používají kabely bez stínění a často i s menším počtem vodičů, protože při některých režimech komunikace nejsou všechny vodiče využity. Minimální implementace využívá pouze 3 vodiče – vysílací, přijímací a společnou zem. Takto minimalisticky je RS232 implementováno i u chapadla Schunk PG70. Používané úrovně napětí jsou z hlediska polovodičové logiky nestandardní: 5 až 15V pro úroveň logické 1 a -5 až -15V pro úroveň logické 0 na straně vysílače. Na straně přijímače jsou pak úrovně detekovány v rozsahu 3 až 25V a -3 až -25V. Rozhraní standardně podporuje komunikační rychlosti od 110b/s do 115200b/s. V praxi je běžné, že zařízení podporuje jen podmnožinu těchto rychlostí. RS232 podporuje dva režimy komunikace – synchronní, kdy jsou data synchronizována hodinovým signálem. Tento režim je vhodný pro velké objemy dat. Implementace rozhraní s 9 vodiči však synchronní režim nepodporuje. Druhým a častějším režimem je režim asynchronní komunikace, kdy je synchronizace odvozena z úvodní sekvence přenášeného rámce. Komunikační protokol standardu RS232 poskytuje jen základní detekci chyb v datech pomocí paritního bitu. Pro spolehlivější detekci a případně opravu chyb je potřeba použít sofistikovanější kódování na vyšších vrstvách komunikace.

Nejen z hlediska rozhraní, ale i chapadlo samotné je z hlediska mechanické odolnosti a elektrických izolací navrženo pro provoz v průmyslových aplikacích.

1.5 Popis robotického ramene

Robotické chapadlo je úzce spjato s robotických ramenem, na kterém je umístěno, proto bude rameni věnována jedna podkapitola. Robotické rameno jako celek je složeno ze dvou částí – samotného ramene a kontroléru, který rameno řídí. Jak už bylo popsáno výše, rameno je výroby Mitsubishi MELFA typu RV-6S a řízeno je kontrolérem CR2B-574. Detaily, které tato podkapitola nezmiňuje, jsou dostupné v manuálech [2] a [3]. Rameno je poháněno elektrickými servomotory. Napájení poskytuje kontrolér ramene. Každý kloub ramene je vybaven elektrickou brzdou. Kromě elektrických rozvodů obsahuje rameno také rozvody pneumatické. Rameno ani kontrolér neobsahují pneumatický kompresor – pneumatické rozvody jsou určeny výhradně k připojení periferií. Rameno má i obvody pro ovládání pneumatických ventilů a poskytuje pro ně rozhraní v řídicím softwaru ramene.

Pohyblivost ramene zajišťuje šest na sobě nezávislých rotačních kloubů. Natočením těchto kloubů je řízena pozice a orientace chapadla v půlkulovém prostoru okolo základny ramene. Vzhledem k omezenému rozsahu kloubu v základně ramene není tento prostor pokryt celý. Navigace ramene je relativní vzhledem ke stanoveným nulovým polohám kloubů. Kromě poloh kloubů nemá rameno žádnou jinou zpětnou vazbu o své poloze.

Samotné rameno je jen „nástroj“, k jehož ovládání slouží kontrolér. Rameno může řídit více typů kontroléru – v tomto případě je použit CR2B-574. Úkolem kontroléru je jednak ovládání ramene a jednak napájení pohonné soustavy ramene. Kontrolér obsahuje napájecí zdroje generující napětí potřebná pro servopohony ramene – stejnosměrná nízká napětí.

Z pohledu této práce jsou významné především možnosti ovládání a propojení s okolními systémy. Kontrolér nabízí dvě možnosti ovládání: přímé ruční ovládání pomocí teaching padu a ovládání z nadřazeného systému. Pro propojení s nadřazeným systémem je kontrolér vybaven rozhraními Extended serial, Ethernet a RS232C. Z těchto rozhraní bylo využito RS232C k propojení s řídicím počítačem. K popisu chování ramene je použit jednoduchý procedurální jazyk MELFA BASIC IV. V tomto jazyce je potom zapsána posloupnost dílčích akcí, kterými je realizován komplexní pohyb ramene – například přenesení předmětu. Kontrolér poskytuje i určitou abstrakci nad ovládáním ramene, takže není třeba ovládat vždy jen natočení konkrétních kloubů, ale je možné zadat například lineární pohyb do cílových souřadnic. Toto zjednodušuje zápis řídicích programů ramene.

Parametr [jednotka]		**Hodnota**
Počet stupňů volnosti		6
Pohon		AC servomotory s brzdou na všech osách
Detekce pozice		Absolutní
Operační rozsah [°]	Základna	+170 .. -170
	Rameno	+135 .. -92
	Loket	+166.. -107
	Otočení zápěstí	+160 .. -160
	Naklopení zápěstí	+120 .. -120
	„Šroubování“ zápěstí	+360 .. -360
Úhlové rychlosti [°/s]	Základna	401
	Rameno	321
	Loket	401
	Otočení zápěstí	352
	Naklopení zápěstí	450
	„Šroubování“ zápěstí	660
Maximální zátěž (zápěstí kolmo k zemi) [kg]		6
Hmotnost		cca. 60 kg
Maximální chyba opakování pozice [mm]		0,02

Tabulka 2: Parametry robotického ramene Mitsubishi MELFA RV-6SL-S12

2 Realizace připojení chapadla

Tato kapitola popisuje propojení a komunikaci s chapadlem. Je zde rozebráno elektrické zapojení robotického chapadla a jeho propojení s řídicím počítačem. Popis se podrobněji zabývá rozhraním RS232 použitým ke komunikaci s chapadlem. Významnou část kapitoly pak tvoří popis softwaru určeného pro práci s chapadlem – především pak knihovny *m5api* .

2.1 Napájení

Chapadlo Schunk PG70 vyžaduje napájení stejnosměrným napětím 24V. Pro spolehlivou funkčnost je doporučeno oddělit napájecí větev pohonu a logiky, ačkoliv obě větve vyžadují stejné napětí. Každá větev by měla být napájena ze samostatného zdroje a vedena samostatnými vodiči. Toto opatření má zabránit výpadkům řídicích obvodů chapadla vlivem poklesu napětí při proudových rázech vznikajících při rozběhu a brzdění prstů. Toto řešení tedy vyžaduje dva nezávislé napájecí obvody – na napájecí obvod logiky jsou kladeny požadavky zejména z hlediska stability napětí zdroje(-4%/+10%), na napájecí obvod pohonu jsou potom kladeny nároky především výkonové – vymezené maximálním proudovým odběrem chapadla 7,5A s doporučenými 20% rezervy.

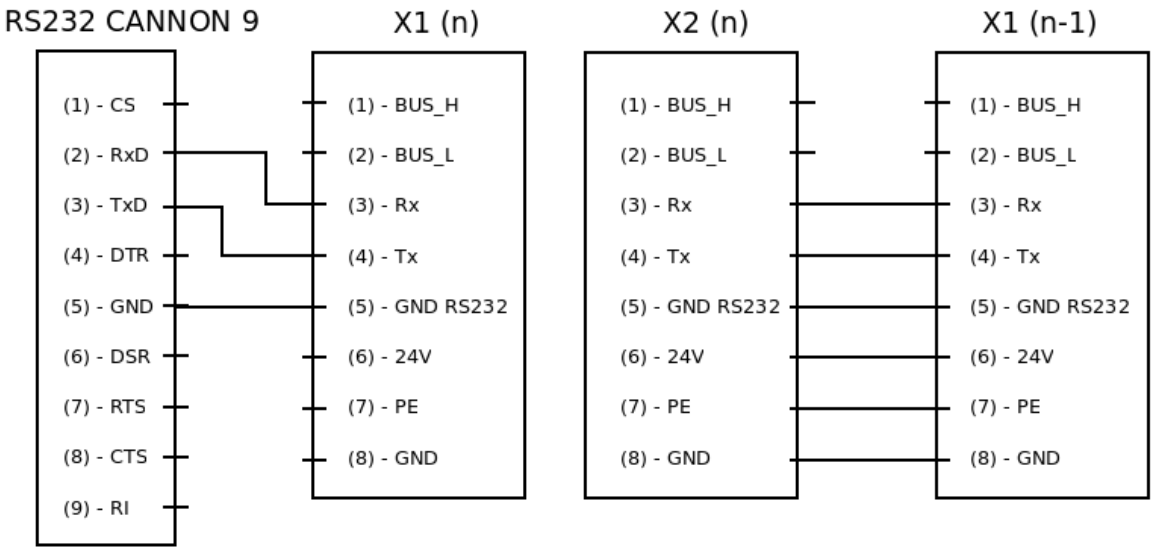
Robotické rameno MELFA RV-6SL je sice přizpůsobeno k připojení a ovládání robotických chapadel, ovšem pouze chapadel pneumatických. Je jím protažen pneumatický rozvod a v zápěstí ramene jsou vyvedeny vstupy a výstupy pro připojení snímačů a ventilů pneumatického chapadla. Ačkoliv jsou ventily spínány stejnosměrným napětím 24V, což odpovídá napájecímu napětí chapadla Schunk PG70, neposkytují výstupy dostatečný proud a navíc jsou spínané kontrolérem ramene, což by mohlo při nevhodně zadaném programu způsobovat výpadky chapadla. Permanentní napětí je vyvedeno pouze pro napájení snímačů – pin A1 na konektoru HC1. Napájení snímačů sice neposkytuje dostatečný proud pro pohon robotického chapadla, nicméně je poměrně stabilní a tak ho bylo možné použít pro napájení řídicích odvodů chapadla. Zem pro toto napětí je vyvedena na více konektorech – byl použit pin A1 na konektoru GR2. Toto řešení je výhodné v tom, že ušetřilo jeden kabel tažený celým tělem ramene. U kabelů tažených tělem ramene k ruce není hlavní překážkou cena kabelu, která by navíc v případě tohoto kabelu nebyla vysoká, ale spíše zaplněná kabelová cesta uvnitř ramene a také spolehlivost, kdy je každý kabel vedený klouby ramene při pohybu ramene mechanicky namáhán a může tedy být zdrojem poruch.

Pro napájení pohonu chapadla bylo třeba použít externí elektrický zdroj. Podle parametrů chapadla snadno spočítáme, že zdroj musí být schopen dodávat proud alespoň 9A (7,5*120/100) při 24V stejnosměrného napětí. Podle těchto parametrů byl vybrán zdroj Siemens 6EP1334-2AA01. Od zdroje bylo nutné protáhnout tělem ramene napájecí kabel s dostatečným průřezem. Jako napájecí kabel byla vybrána splétaná dvojlinka – splétané vodiče proto, že jsou dostatečně mechanicky odolné a dvojlinka z toho důvodu, že vícevrstvá izolace není vzhledem k okolnímu prostředí potřeba. Díky tomuto řešení je napájení pohonu chapadla zcela nezávislé na rameni a kontroléru – jeho výpadek nelze na rozdíl od napájecího napětí řídicích obvodů chapadla z kontroléru ramene nijak detekovat. To sice nemá vliv na funkčnost chapadla, ale je potřeba tento fakt brát v potaz při ošetření chybových stavů.

2.2 Datová cesta

V případně propojení komunikační linky chapadla s nadřazeným systémem existuje více možností. Chapadlo Schunk PG70 nabízí více rozhraní a protokolů, kterými je možné ho připojit. Každé rozhraní má jiné vlastnosti a je vhodné do jiných podmínek. Současně však dokáže chapadlo komuni-

kovat pouze jedním rozhraním a ostatní musí být zakázána. Možné alternativy jsou Profibus DP, CAN Bus a RS232.

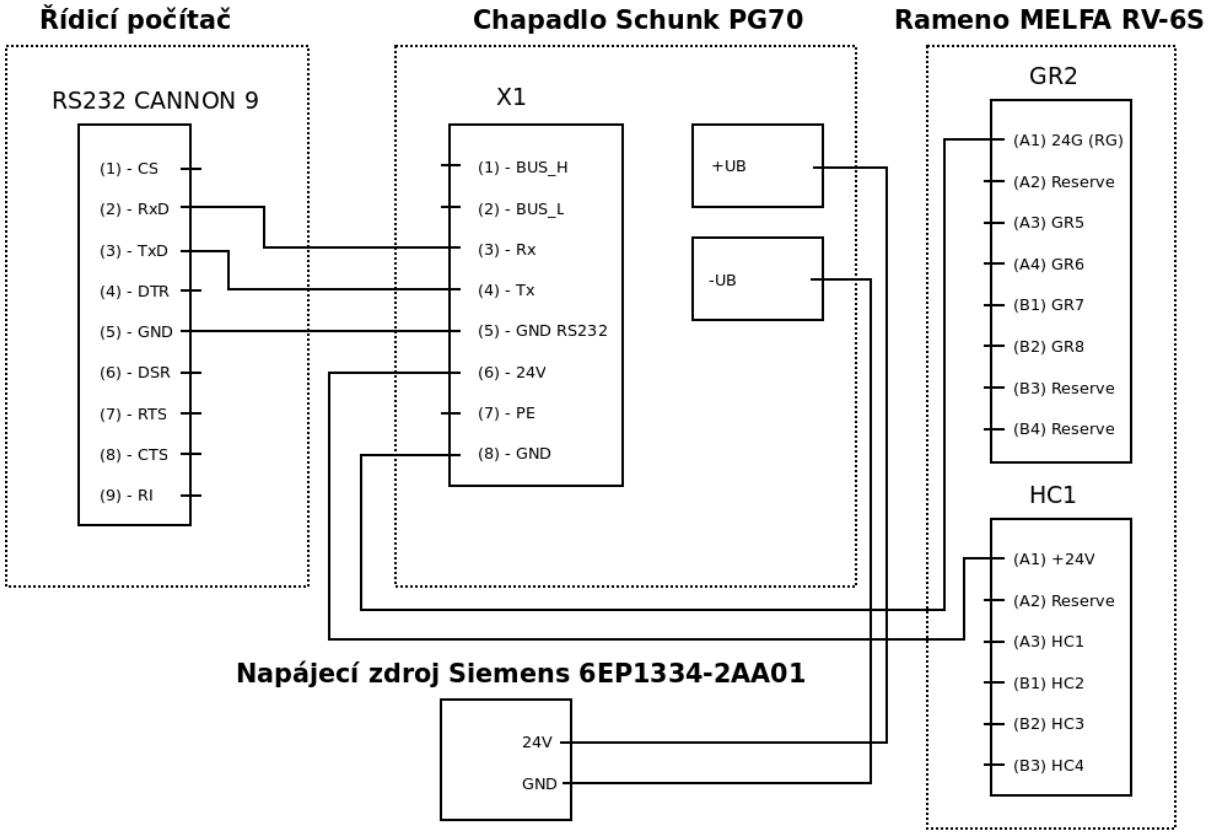


Obrázek 2: Propojení více modulů Schunk na jedné lince RS232.

Z možných rozhraní bylo pro propojení chapadla s řídicím počítačem vybráno rozhraní RS232. Hlavním důvodem pro toto rozhodnutí byla dostupnost rozhraní RS232 na řídicím počítači – nebylo tedy třeba dokupovat další rozšiřující karty. Dalším důvodem byla jednoduchost tohoto rozhraní. U RS232 odpadají komplikace s adresováním zařízení na sběrnici a datový rámec je poměrně snadno čitelný pro případ ladění. Navíc chapadlo Schunk PG70 nabízí možnost připojit na jednu linkou RS232 vedoucí k němu další zařízení komunikující kompatibilním protokolem. Propojení je realizováno tak, že chapadlo komunikaci zprostředkovává. Zprostředkovatelů linky může být více za sebou. Takto lze na jednu linku RS232 připojit obecně libovolný počet zařízení omezený teoreticky jen adresovacím rozsahem komunikačního protokolu. Propojení více zařízení na jedné lince RS232 je znázorněno na obrázku č.2. Hodnoty v závorce představují pořadové číslo zařízení. Koncové zařízení s číslem 1 nemá připojené výstupy ze svorkovnice X2 a má propojenou propojku J3 – tou je zařízení přepnuto do režimu, kdy se nepokouší zprostředkovávat komunikaci dále.

Schéma na obrázku 2 popisuje zapojení obecného počtu modulů. V našem případě je použit jen jeden modul, který je připojen přímo k RS232 výstupu na řídicím počítači a má propojenou konfigurační propojku JP3. Konkrétní celkové zapojení chapadla včetně napájeních okruhů je zobrazeno na schématu na obrázku 3.

Chapadlo je připojeno samostatnou linkou RS232. Další samostatnou linkou RS232 je připojen kontrolér ramene. Mezi chapadlem a ramenem není přímé datové spojení a tedy mezi nimi není žádná vazba na hardwarové úrovni. Obě zařízení jsou připojena k řídicímu počítači a ten řeší veškerou koordinaci jejich chování softwarově. Software pro koordinaci robotického chapadla a ramene nebyl výrobci dodán a v době psaní práce je ve fázi vývoje zde na FIT.



Obrázek 3: Celkové schéma připojení chapadla Schunk PG70.

Pro propojení vodičů rozhraní RS232 byl vybrán tenký nestíněný telefonní kabel. Kabel má čtyři samostatně izolované žíly a vnější izolaci z měkké gumy. Je určen výhradně pro provoz ve vnitřních prostorách. Tento kabel sice není z hlediska parametrů pro rozhraní RS232 ideální, ale vzhledem k malé délce kabelu (do 10m) by neměl způsobovat výraznější potíže při komunikaci. Reálné experimenty ukazují, že komunikace funguje bez problémů. Důvodem, proč byl vybrán právě tento kabel je jeho mechanické provedení. Kabel je velmi tenký a odolný vůči mechanickému namáhání, protože vnitřní žíly jsou splétané. Přebývající vodič byl v zapojení použit pro posílení signálové země GND RS232.

2.3 Software pro komunikaci s chapadlem

Pro zjednodušení práce s chapadlem byly výrobcem dodány softwarové nástroje. Jejich použití není nezbytné, ale výrazně zjednodušují práci. Především při zprovozňování chapadla a diagnostice chyb je lepší spoléhat se na odzkoušený software. Výhodou použití softwarových komponent od výrobce je také abstrakce nad komunikačním protokolem, který chapadlo používá, a nad inicializací a řízením jednotlivých komunikačních rozhraní. Ke všem softwarovým nástrojům je k dispozici podrobná dokumentace [1] a [7]. V této podkapitole se zaměříme na popis jednotlivých nástrojů a vazeb mezi nimi.

Jádrem softwarové podpory od firmy Schunk je knihovna *m5api*. Tato knihovna poskytuje rozhraní v programovacích jazycích C/C++ a VisualBasic pro práci s chapadlem. Umožňuje vývoj aplikací odstíněný od komunikačního protokolu a detailů jednotlivých rozhraní. Nevýhodou použití knihovny *m5api* je ale její zastaralost z hlediska podporovaných verzí operačních systémů a kompilátorů. Navíc byla s chapadlem dodána jen ve verzi pro operační systém MS Widnows. Tyto nevýhody

jsou zásadní především proto, že knihovna je proprietární a není k ní dostupný zdrojový kód, takže není možné knihovnu pro novější operační systémy přeložit a případně upravit.

Knihovna *m5api* poskytuje v rámci svého rozhraní sadu funkcí plně pokrývající možnosti konfigurace a ovládání robotického chapadla. Rozhraní knihovny je univerzální pro různé typy chapadel – chapadla se liší jen v limitech zadávaných údajů. Tyto limity si zařízení uchovávají v paměti a lze je z nich za chodu získat. Funkce obecně lze rozdělit na dvě kategorie: funkce vstupní, kterými je čtena konfigurace a získáván stav zařízení a funkce výstupní sloužící k zasílání příkazů od nadřazeného systému k zařízení. Obě tyto kategorie mají několik podkategorií podle účelu, kterému funkce slouží. Chapadlu lze některé parametry zadávat nejen absolutně, ale i relativně vzhledem k aktuální hodnotě a také v inkrementech. Inkrement je interval spojité hodnoty daný rozsahem spojitého parametru a počtem dílků na rozsah. Inkrement je nastavitelný a lze ho využít ke zjednodušenému zadávání spojitého parametru diskrétní hodnotou malého rozsahu, která navíc na rozdíl od reálné spojité hodnoty parametru může být stejná pro různé typy chapadel.

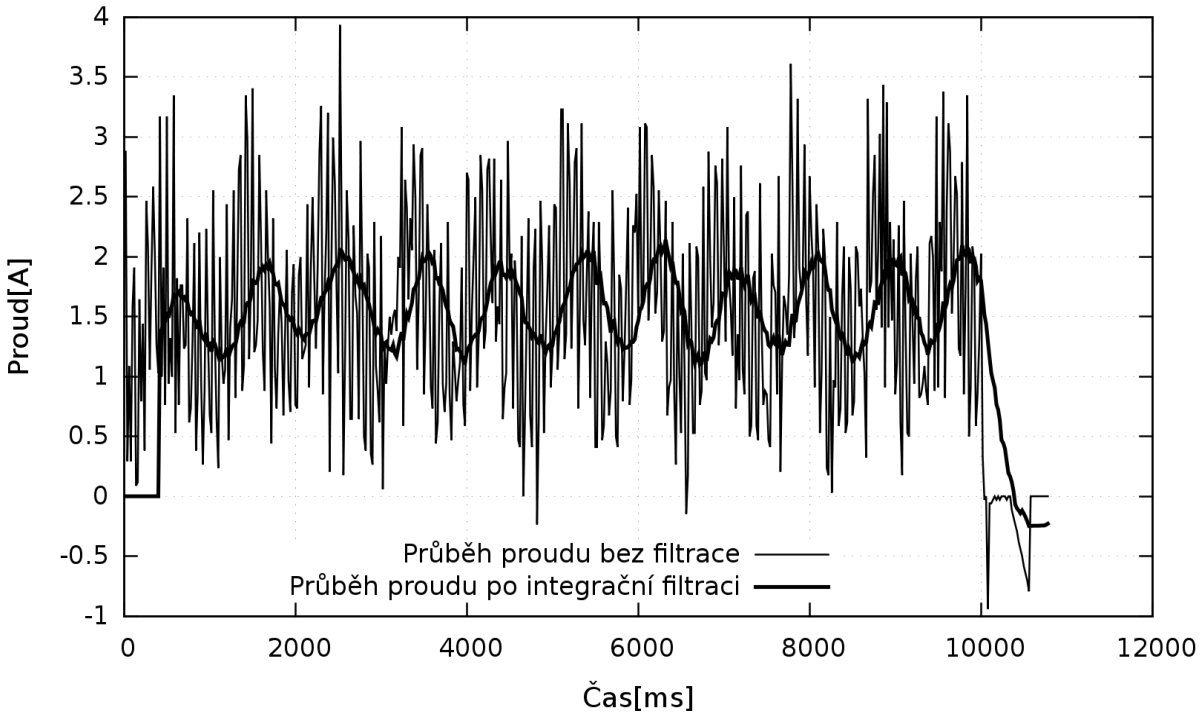
Práce s knihovnou je poměrně intuitivní. Nejdříve knihovnu zinicializujeme. K tomu je použita funkce *Pcube_openDevice*. Inicializací určíme rozhraní, kterým budeme s připojenými zařízeními komunikovat. V našem případě je připojené zařízení jen jedno na rozhraní RS232. Potom je vhodné zařízení zkonfigurovat a uvést do výchozí pozice pro případ, že by ho nějaký jiný použitý program přenastavil. Dále už můžeme zařízení ovládat dle potřeb – typicky zadáme příkaz k pohybu a potom iterativně čteme stav, dokud není pohyb dokončen. Při ukončení práce se zařízením je korektním postupem knihovnu deinicializovat a tím uvolnit zdroje, i když to moderní operační systémy udělají samy při ukončení hlavního procesu aplikace.

Nad touto knihovnou je postavený nástroj PowerCube Demo. PowerCube Demo je nástroj pro diagnostiku a ruční ovládání chapadla. Zároveň slouží jako demonstrace použití knihovny *m5api*. Při prvním spuštění je nutné program nastavit inicializačním řetězcem (viz [7]). Po správném nastavení je možné provést scan sběrnice pro nalezení kompatibilních zařízení. Pokud vše funguje správně, jsou dostupná zařízení nalezena a je možné s nimi dál pracovat. Výchozí dialog pro práci se zařízeními umožňuje provést zastavení všech zařízení, resetování zařízení a uvedení zařízení do výchozích pozic. Při výběru konkrétního zařízení jsou pak zobrazeny širší možnosti pohybu – cílová poloha, zrychlení a rychlost při pohybu. V okně jsou zobrazeny informace o aktuální poloze a okamžitém odběru zařízení a také všechny dostupné stavové příznaky včetně jejich hodnoty. Tyto informace jsou užitečné při oživování, kdy je možné sledovat, v jaké stavu se zařízení nachází.

2.4 Testování připojení chapadla

Po té co bylo chapadlo připojeno a byly nainstalovány potřebné softwarové nástroje, bylo třeba ho oživit a otestovat. Prvním krokem, bylo otestování komunikace s chapadlem programem PowerCube Demo. Tento program byl vyvinut přímo výrobcem, takže ho bylo možné považovat za spolehlivě funkční. Po spuštění a nastavení programu byl proveden scan sběrnice a chapadlo bylo úspěšně nalezeno. Byl překontrolován stav chapadla podle ohlášených stavových informací. Základní funkčnost chapadla byla prověřena na různých pohybech prstů.

Dále bylo potřeba otestovat dynamické vlastnosti chapadla – především odběr proudu při pohybu prstů. Za tímto účelem byl vytvořen malý testovací program, který komunikoval s chapadlem, cyklicky posílal výzvu k vyslání hodnoty aktuálně odebíraného proudu a zaznamenával výsledky. Při prvním spuštění tohoto programu vracelo chapadlo chybu *busy*. Z toho můžeme usoudit, že chapadlo není schopno zvládat odesílání informace plnou rychlostí rozhraní. Proto byly mezi dotazy vkládány čekací smyčky. Experimentováním byla zjištěna délka 40ms pro čekání mezi dotazy na aktuální proud – tedy vzorkovací frekvence 25Hz. S takto seřízeným testovacím programem bylo možné získat průběh odběru proudu v čase. Byl zaznamenán průběh odběru proudu na prázdno při rychlosti prstů 1mm/s – viz graf 1.



Graf 1: Průběh odběru proudu chapadla při pohybu prstů bez kolize v čase.

Z grafu je vidět, že průběh je značně proměnlivý. Odběr kmitá v rozmezí 0A až 3A okolo přibližné střední hodnoty 1,5A. Navíc se průběh při opakování stejného pohybu mění. Aby bylo možné z průběhu alespoň nějaké charakteristiky vyčíst, bylo třeba jej profiltrovat. Byl použit jednoduchý integrační filtr průměrující několik po sobě jdoucích vzorků. Z takto filtrovaného průběhu už můžeme vyčíst některé rysy chování chapadla při pohybu. Předně vidíme, že okamžitý odběr kolísá po křivce připomínající sinusoidu nebo trojúhelníkový průběh. Z toho je tedy patrné, že kolísání odběru má určitou pravidelnost. Dále vidíme, že filtrovaný odběr kolísá okolo střední hodnoty v přibližně konstantním rozsahu a to i přesto, že zachycené špičky dosahují náhodně vyšších hodnot (nad 3,5A). Nakonec si můžeme kolem vzorku v čase 10000ms povšimnout odklonění nefiltrované křivky, která klesá skokově do nuly a filtrované křivky, která klesá pod úhlem. Úhel mezi křivkami reprezentuje zpoždění vzniklé integrací. Toto zpoždění omezuje použití integračního filtrování při regulaci v reálném čase. Posledním zajímavým jevem v tomto grafu je nástup zelené křivky z 0 se zpožděním. Zpoždění vzniká inicializací integračního filtru, který do doby, než je inicializován, vrací hodnotu 0. Je to způsobeno implementací filtru, kde je tímto způsobem signalizováno naplnění bufferu filtru. Hodnoty na výstupu filtru jsou takto buďto integrované v zadaném rozsahu a nebo nulové.

Za účelem experimentování s regulátorem bez nutnosti neustálé komunikace s chapadlem byla vytvořena sada záznamů průběhů proudu tekoucího do motoru chapadla. Tyto záznamy obsahují jak průběhy proudu při pohybu prstů na prázdno při různých rychlostech, tak průběhy při pohybu prstů v zátěži při stiskání objektu. Data jsou uložena v textovém formátu CSV.

3 Návrh řídicího softwaru

Kapitola Návrh řídicího softwaru se zabývá návrhem řešení softwarového regulátoru. Návrh je postupný ve směru shora dolů. Z počátku jsou definovány parametry, které by měl regulátor splňovat. Těmto parametrům se podřizuje zbytek návrhu.

3.1 Parametry regulátoru

Byla snaha definovat parametry regulátoru tak, aby byl použitelný pro bezpečné uchopování objektů. Návrh si neklade za cíl vysoce přesnou regulaci síly. Pro takovou funkčnost není systém vhodně koncipován. Zásadní omezení představují také nepříliš povzbudivé výsledky testu dynamického chování chapadla. Vzorkovací frekvence výrazně omezuje rychlost pohybu, při které bude regulace přijatelně fungovat. V zašuměném signálu, což je prakticky každý reálný signál, je potřeba několik vzorků pro rozpoznání trendu. Minimem jsou tři vzorky – současný a dva předchozí, aby vůbec existovala naděje na rozpoznání, zda se jedná o výkyv vlivem šumu, nebo skutečnou změnu hodnoty. Tento minimální požadavek ale stačí pouze pro vyhlazení odchylek ovlivňujících nejvýše jeden vzorek ze vzorků za sebou jdoucích. Pro delší odchylky musí být vzorků více. V tomto případě je potřeba vzorků podstatně větší množství vzhledem k intenzitě šumu v datech. Integrační filtr, kterým byla data filtrována v grafu 1, průměroval 20 vzorků. Přesto je vidět, že výstup filtru je zvlněný a bude potřeba délka několika vzorků výstupu filtru ke správné interpretaci úseku průběhu – experimentálně stanoveno na 4 až 5 vzorků . Tedy chceme-li, aby regulátor zareagoval na dráze 1mm, je nejvyšší možná rychlost pohybu prstů 5mm/s (1000ms / (40ms * 5 vzorků)). K tomu je třeba ještě připočíst časovou rezervu pro zpoždění při výpočtu údaje a komunikaci s chapadlem. Takto odhadnuté zpoždění reakce regulátoru na kolizi s překážkou je ovšem nejoptimističtější varianta, které předpokládá, že regulátor zareaguje ihned, jak to bude teoreticky možné. To ale není prakticky použitelné, protože by takto regulátor při každém výkyvu průběhu proudu vyvolával „falešné poplachy“ a rozpoznával kolizi s překážkou i tam, kde není. V praxi bude nutné, aby kolize nějakou dobu trvala. Problémem pro regulaci je také zvýšený odběr při rozběhu chapadla, kdy je motor v zátěži. Prakticky to znamená, že po určitý čas od rozběhu prstů nebude regulace splňovat zadané parametry. Ještě významnějším zpožděním než rozběh hardwaru je „rozběh“ softwaru. Regulátor není schopen korektně regulovat, dokud se nenaplní buffery filtrů a dokud se stav filtrů neustálí po proudovém nárazu při rozběhu motoru chapadla. Skutečná doba zpoždění byla proto nastavena na výrazně vyšší, než jakou naznačoval optimistický odhad.

Vlivem filtrace a zpoždění reakce při regulaci vzniká značná chyba. K této chybě je nutné uvážit ještě odběr chapadla na prázdno, který není stejný po celé délce dráhy prstů, jak je při bližším zkoumání patrné z grafu 1. Proto byla tolerance měření síly regulátorem nastavena poměrně benevolentně na +-20% ze zadané hodnoty. Tyto parametry by měly být dosažitelné při navrhované rychlosti 1mm/s. Je pravděpodobné, že pro vyšší rychlosti pohybu prstů bude regulace méně přesná. Požadavky na regulátor byly shrnuty do tabulky.

Rozsah zadávané síly byl nastaven na 10N až 40N. Ruka sice umožňuje při stisku vyvinou větší sílu, ale pro nedestruktivní uchopování předmětů nejsou větší síly příliš užitečné. Naopak dolní hranice 10N by sice bylo ideální ještě snížit, ale při takto malých silách se už příliš silně projevuje vliv šumu a regulátor pak těžko může dodržet zadanou přesnost. Regulace pod minimum rozsahu je možná, ale chyba bude pravděpodobně výrazně větší.

Rychlost pohybu prstů byla nastavena na 1mm/s, což je kompromis mezi dostatkem času pro regulaci během stiskání objektu a rozumně rychlým uchopením objektu. Regulátor umožní zadávat i rychlosti nižší a vyšší, ale pro tyto rychlosti už nemusí zadané parametry platit. To se týká především rychlostí vyšších.

Parametry regulace:

Parametr	Hodnota
Chyba měření síly	+-20% z nastavené hodnoty
Rychlost odezvy	600ms
Rozsah zadávané síly	10-40N
Rychlost pohybu prstů	1mm/s
Doba rozběhu	1000ms
Úroveň detekce kolize	do 5N

Tabulka 3: Parametry regulace.

3.2 Návrh rozhraní

Vzhledem k tomu, že regulátor by měl být v budoucnu součástí komplexnějších systémů, je nutností vhodně a pevně definované rozhraní. Zde se setkávají dva protichůdné požadavky. Na jedné straně je potřeba navrhnout software tak, aby ho bylo možné do budoucna rozvíjet a upravovat, a na straně druhé je nutné, aby měl pevně definované rozhraní. Jakákoliv změna rozhraní, která způsobí nekompatibilitu s předchozí verzí, totiž vynucuje přepsání veškerého dalšího softwaru, který tento software využívá. Navrhovaný regulátor by sice měl komunikovat především jednosměrně – uživatel zadá parametry a regulátor autonomně zajistí jejich dodržení, ale data, se kterými regulátor pracuje, a výstupy regulace jsou užitečné i pro nadřazené systémy. Přenos těchto dat směrem k obklopujícím systémům byl proto při návrhu rozhraní zohledněn. V tabulkách 4 a 5 byly shrnuty parametry, které by měl regulátor přebírat a informace, které by měl vracet.

Počet vzorků vracený procesem regulace coby výstup poskytuje informaci o tom, jak dlouho pohyb trval – lze ho převést na čas násobením vzorkovací periodou. Ekvivalentní informaci poskytuje index posledního vzorku.

Aby bylo použití regulátoru co nejefektivnější, byla navržena rozhraní dvě – jedno v podobě hlavičkových souborů jazyka C++ pro integraci do programů a projektů a druhé v podobě spustitelného souboru pro integraci do skriptů. Všechny vstupy a výstupy tak jak jsou popsány v tabulkách 4 a 5 poskytuje pouze rozhraní pro jazyk C++. Rozhraní pro skripty v podobně binárního souboru nabízí pouze jejich podmnožinu.

Vstupy:

Parametr	Jednotka
Hodnota limitní síly	[N]
Rychlost pohybu prstů	[m/s] – kompatibilní s rozhraním *m5api*
Cílová pozice	[m] – relativní od výchozí pozice, komp. s *m5api*
Zapnutí výstupu okamžitých hodnot proudu	-
Zapnutí upozornění při kolizi prstů s objektem	-

Tabulka 4: Vstupy.

Výstupy:

Parametr	Jednotka
Stav ukončení pohybu – s kolizí/bez kolize	-
Okamžitý odběr proudu	[A]
Upozornění na kolizi – index vzorku	-
Pozice prstů při kolizi	[m] – relativní od výchozí pozice
Celkový počet vzorků při ukončení pohybu	-
Pozice ukončení pohybu	[m] – relativní od výchozí pozice

Tabulka 5: Výstupy.

Ačkoliv knihovna *m5api* má rozhraní pro jazyk C využitelné i v jazyce C++, rozhraní regulátoru je omezeno pouze na C++. Toto omezení bylo motivováno tím, že jazyk C++ umožňuje použití výchozích hodnot parametrů funkcí. Pokud programátor nepotřebuje změnit výchozí hodnoty parametrů, nemusí tyto hodnoty vůbec zadávat. Tím se rozhraní regulátoru velmi zjednodušuje, protože programátor-uživatel nemusí zadávat při každém volání funkcí rozhraní množství předdefinovaných výchozích hodnot pro málo používané parametry a přitom nepřichází o možnost jejich zadání.

Rozhraní pro knihovnu v jazyce C++:

Inicializace regulátoru – načtení dynamické knihovny:

```
void HAND_Init();
```

Inicializaci je potřeba provést vždy před prvním použitím regulátoru. Inicializace nesmí být volána více než jednou.

Reset modulu – nastavení výchozích hodnot stavových registrů:

```
int HAND_ResetModule(std::string pInitString, int *dev_ptr, int
moduleID);
```

Modul chapadla by měl být resetován před každým zahájením pohybu. Pokud totiž došlo při pohybu k zastavení modulu vlivem překročení hardwarového proudového limitu, je tato skutečnost modulem vnímána jako chyba a bez explicitního resetu modul nepřijme další požadavky. Hodnota získaná ukazatelem *dev_ptr* je potom použita ve funkcích realizujících pohyb prstů chapadla.

Význam parametrů:

- *pInitString* – inicializační řetězec ve formátu knihovny *m5api.*
- *dev_ptr* – ukazatel na proměnnou, kam se uloží ID zařízení.
- *moduleID* – ID modulu zařízení – pro komplexní zařízení (u PG70 vždy 1).

Reset modulu a uvedení prstů do výchozí pozice – neblokující:

```
int HAND_ResetAndHomeModule(std::string pInitString, int *dev_ptr, int
moduleID, void (*finalize_event) (int));
```

Tato funkce provede reset tak, jak je popsán výše a potom uvede modul chapadla do výchozí pozice. Bez tohoto úkonu neakceptuje modul žádné příkazy k pohybu. Uvedení do výchozí pozice stačí provést vždy jednou po zapnutí napájení modulu.

Význam parametrů:

- *pInitString* – inicializační řetězec ve formátu knihovny *m5api.*
- *dev_ptr* – ukazatel na proměnnou, kam se uloží ID zařízení.
- *moduleID* – ID modulu zařízení – pro komplexní zařízení (u PG70 vždy 1).
- *finalize_event* – funkce, která je zavolána při dokončení operace regulátoru.

Pohyb prstů bez regulace síly (pouze hardwarové omezení proudu) – neblokující:

```
int HAND_MoveNoRegulation(int dev, int  moduleID, float position, float
speed, void (*finalize_event) (int));
```

Význam parametrů:

- *dev* – ID zařízení na sběrnici
- *moduleID* – ID modulu zařízení – pro komplexní zařízení (u PG70 vždy 1).
- *position* – cílová pozice, na kterou se mají prsty pohnout [m].
- *speed* – rychlost, jakou se mají prsty pohybovat.
- *finalize_event* – funkce, která je zavolána při dokončení operace regulátoru.

Posunutí prstů na zadanou pozici zadanou rychlostí s omezením síly a výstupem hodnot – neblokující:

```
int HAND_MovePosForce(int dev, int moduleID, CalTab
*calibration_table,float position, float force, float speed, void (*fina-
lize_event)(int), int* dst_sample=NULL, float* dst_pos=NULL, void
(*sample_out_f)(struct sample)=NULL);
```

Z deklarace funkce je vidět, že parametry *dst_sample*, *dst_pos* a *sample_out_f* mají předdefinované výchozí hodnoty. Je tedy možné volat funkci bez zadání těchto parametrů. To zjednodušuje práci uživatelům, kteří nevyžadují informace o průběhu regulace.

Význam parametrů:

- *dev* – ID zařízení na sběrnici.
- *moduleID* – ID modulu zařízení – pro komplexní zařízení (u PG70 vždy 1).
- *calibration_table* – ukazatel na objekt kalibrační tabulky – tato tabulka slouží k převodu. zadané síly na hodnotu limitního proudu pro regulátor – pokud je ukazatel roven NULL, považuje funkce zadanou hodnotu síly za přímo hodnotu proudu.
- *force* – síla, při jejímž dosažení se pohyb prstů zastaví [N].
- *position* – cílová pozice, na kterou se mají prsty pohnout [m].
- *speed* – rychlost, jakou se mají prsty pohybovat.
- *finalize_event* – funkce, která je zavolána při dokončení operace regulátoru.
- *dst_sample* – vzorek, na kterém pohyb prstů skončil.
- *dst_pos* – relativní pozice vůči výchozí pozici prstů, na které se prsty zastavily.
- *sample_out_f* – ukazatel na funkci, která se zavolá při příchodu vzorku okamžitého proudu. – parametrem je struktura `struct sample` popsaná níže.

Poznámka „neblokující“ u některých těchto funkcí říká, že funkce předá parametry a skončí – nečeká na dokončení operace zařízením. Regulátor potom běží v odděleném vlákně. Na to je třeba brát zřetel především při použití funkce *sample_out_f*, protože volání této funkce bude probíhat asynchronně k běhu zbytku programu. Důležitou skutečností okolo neblokujících funkcí je také to, že tyto funkce nemohou vracet chyby v případě problému během regulace. Tedy to, že neblokující funkce vrátí bezchybný stav, ještě neznamená, že chyba nemohla později nastat.

```
struct sample_struct {
      int sample_index;
      float sample_val;
      bool collision;
};
```

Význam položek:

- *sample_index* – index vzorku.
- *sample_val* – hodnota vzorku.
- *collision* – informace o tom, zda při tomto vzorku byla detekována kolize.

Návratová hodnota funkcí je celé číslo. Toto číslo je chápáno jako posloupnost jednobitových příznaků. Význam jednotlivých bitů je popsán od nejméně významného (LSB) v tabulce č.6. Hodnota příznaku 0 tedy znamená, že všechny bity jsou nulové a tedy nenastala žádná chyba. Pokud nastala chyba, nejsou výstupní hodnoty definovány. Je tedy třeba vždy testovat návratovou hodnotu funkcí.

Bit	**Význam**
0	Kolize při pohybu – u neblokujících funkcí vždy 0
1	Chyba zařízení
2	Neplatné parametry

Tabulka 6: Význam bitů v návratové hodnotě funkce.

Stejné parametry mají i blokující funkce. Blokující funkce jsou alternativou k neblokujícím funkcím. Liší se v tom, že vrací hodnoty až v okamžiku, kdy zařízení skutečně operaci dokončí.

```
void HAND_ResetAndHomeModuleBlock(std::string pInitString, int *dev_ptr,
int  moduleID);

void HAND_MoveNoRegulationBlock(int dev, int  moduleID, float position);

int HAND_MovePosForceBlock(int dev, int moduleID, CalTab
*calibration_table,float position, float force, float speed, int*
dst_sample=NULL, float* dst_pos=NULL, void (*sample_out_f)(struct
sample)=NULL);
```

Druhým rozhraním regulátoru je potom spustitelný soubor, který prostřednictvím knihovny regulátoru ovládá chapadlo. Spustitelný soubor neposkytuje kompletní množinu podporovaných operací. Umožňuje jen měnit pozici prstů s omezením síly. Je určen pro ovládání ze skriptů a webových služeb. Rozhraní tedy nabízí následující parametry:

- *-f* ... limitní síla [N].
- *-vel* ... rychlost pohybu prstů [m/s] (nepovinné - výchozí hodnota je 0.001).
- *-pos* ... pozice, na kterou se mají prsty přesunout.
- *-initstr* ... vlastní inicializační řetězec – ovlivňuje mj. použitý sériový port pro komunikaci (nepovinné – výchozí hodnota „RS232:1,9600“).
- *-clt* ... cesta k datovému souboru kalibrační tabulky (nepovinné – výchozí hodnota „clt0.001.clt“).
- *-heatup* ... zahřívací procedura – provede několik pohybů prstů bez regulace síly, provádí také přesunutí prstů modulu do výchozí pozice.
- *-calibrate* ... spustí kalibraci.

Hodnoty parametrů *-pos, -vel* a *-f* jsou čísla s plovoucí řádovou čárkou, parametr *-initstr* je řetězec a parametr *-clt* je cesta k souboru standardní v daném typu OS. Parametry *-f* a *-pos* jsou povinné. Parametry *-heatup* a *-calibrate* mají vyšší váhu, než příkazy k pohybu – pokud je některý z nich zadán, ostatní parametry jsou ignorovány. Výstupem je návratový kód procesu odpovídající popsaným návratovým kódům funkcí regulátoru viz tabulka 6. Volání spustitelného souboru může vypadat takto:

```
$ ./HAND_cmd.exe -pos -0.01 -f 15.5
```

Znak $ na začátku příkazu představuje shell prompt. Program čeká na dokončení operace na zařízení, potom skončí.

Parametr *-calibrate* spouští kalibraci prstů, kdy program postupně zvyšuje limit proudu od interně definovaného minima 3,5A po maximum 7,5A a zaznamenává pomocí externího spustitelného souboru nebo skriptu *measure* průběh stisku. Pro každou hodnotu limitního proudu provede 10 měření. Pro každé měření vytváří CSV soubor. Z takto vzniklých souborů je potom dalšími nástroji generována kalibrační tabulka. Externí spustitelný soubor *measure* zajišťuje komunikaci se siloměrem. Program *HAND_cmd* ho spouští při každém jednotlivém měření a na konci každého měření ho ukončuje.

3.3 Princip regulace

Další fází návrhu, když známe parametry a rozhraní regulátoru, je návrh principu regulace. Na základě tohoto návrhu jsou pak navrhovány a implementovány jednotlivé funkční bloky. Regulátor musí fungovat v reálném čase. To je nutné při návrhu principu zohlednit.

Robotické chapadlo umožňuje nastavit hardwarově limitní proud. Toto nastavení sice nestačí pro přesnou regulaci síly, ale je výhodné ho použít jako omezovač pro případ, že by softwarová regulace selhala. Hardwarový limit proudu je citlivý i na ostré špičky odběru a tak je nutné jej nastavovat s dostatečnou rezervou, aby nezpůsoboval předčasná zastavení pohybu. Regulaci lze tedy rozdělit na dvě úrovně – nepřesnou ale vysoce spolehlivou hardwarovou regulaci a přesnější softwarovou regulaci, která ale může být vlivem výpadků komunikace méně spolehlivá. Softwarová regulace pracuje vždy v rozsahu hardwarové regulace. Softwarová regulace může zareagovat dříve nebo nejvýše ve stejném okamžiku, jako regulace hardwarová. V podstatě hardwarovou regulaci zpřesňuje.

Nyní se zaměříme na průběh regulace v okamžiku zadání povelu k pohybu prstů s omezením síly. První akcí je nastavení hardwarového proudového omezení. To je nastaveno na konstantní hodnotu 8A. Omezení na 8A umožní dosáhnout sil výrazně překračujících rozsah softwarové regulace (okolo 100N). Takto široké omezení je ovšem nutné kvůli proudovým nárazům při rozběhu prstů. Nižší hodnoty při experimentování způsobovaly náhodné výpadky. Síla 100N odpovídá zhruba zatížení uchopovaného objektu desíti kilogramy.

V okamžiku, kdy regulátoru přijde příkaz k pohybu prstů s omezením síly, regulátor resetuje chybové příznaky chapadla a nastaví hardwarové omezení maximálního odebíraného proudu. Potom zahájí pohyb prstů se softwarovou regulací. V průběhu softwarové regulace se řídí algoritmem č.1.

Algoritmus pracuje tak, že nejdříve zinicializuje zpracování dat, jako je naplnění bufferů výchozími hodnotami aj. a nastaví příznaky kolize a STOP do *false*. Potom cyklicky čte vzorky okamžitého proudu, zpracovává je filtrací a zkoumá hodnotu okamžitého proudu. Pokud dosud nebyla kolize detekována (*COLLISION_FLAG=false*), potom zjišťuje, zda nebyla překročena hodnota proudu na prázdno. Proud na prázdno je proud odebíraný pohybujícími se nezatíženými prsty. V případě, že hranice překročena je, nastaví příznak kolize (*COLLISION_FLAG=false*) a ověří, zda nebyl překročen i proud pro omezení síly. V případě, že tento proud překročen byl, nastaví příznak STOP a tím vynutí ukončení cyklu a zastavení prstů. V opačném případě pokračuje další iterací. Pokud probíhá iterace cyklu v situaci, kdy už kolize detekována byla, hlídá se jen překročení proudu pro omezení

síly a při překročení se opět nastavuje příznak STOP. Alternativně také může regulační cyklus skončit dosažením cílové pozice prstů bez překročení síly.

```
inicializace zpracování dat
COLLISION_FLAG = false
STOP = false
do while ((¬STOP) ∨ (není dosažena cílová pozice))
  Získání vzorku okamžitého odběru proudu
  Předzpracování vzorku filtrací
  if (¬COLLISION_FLAG)                    /* pokud už dříve nebyla detekována kolize */
    if (detekována kolize z aktuálního vzorku)
      COLLISION_FLAG = true
      if (odebíraný proud překračuje limit)
        STOP = true
      end if
    end if
  else
    if (odebíraný proud překračuje limit)
      STOP = true
    end if
  end if
loop
zastavení pohybu prstů
```

Algoritmus 1: Algoritmus softwarové regulace.

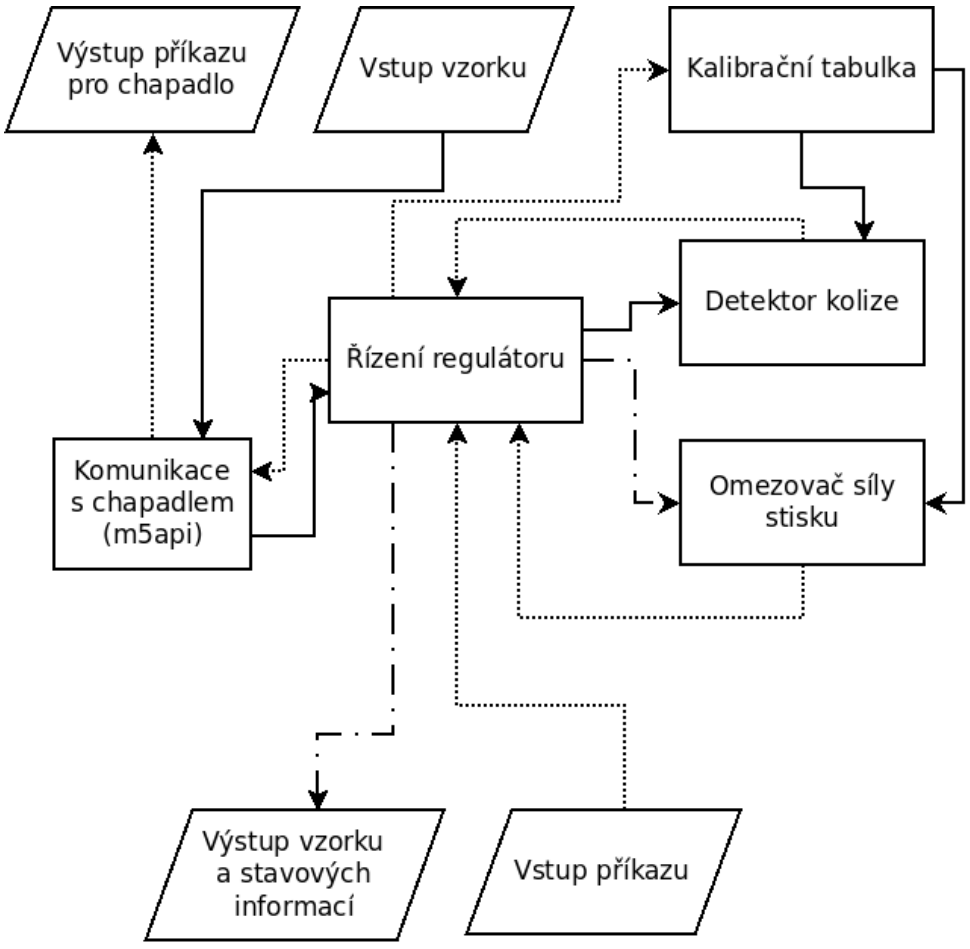
3.4 Fyzikální stránka regulace

Z fyzikálního hlediska je regulace síly stisku založena na zkoumání odchylek od ustáleného stavu fyzikální soustavy. Tímto ustáleným stavem je zde pohyb prstů konstantní rychlostí po přímočaré trajektorii. V takovémto stavu by pohon prstů teoreticky neměl být vůbec zatížený a proudový odběr by měl být nulový. To se ovšem v praxi neděje. V praxi je pohyb prstů díky tření převodové soustavy pohybem rovnoměrně zpomaleným. Toto platí za předpokladu, že je tření po celé dráze prstů konstantní. Tento předpoklad sice nemusí být nutně splněn, nicméně pro účely regulace stačí i slabší předpoklad říkající, že součinitel smykového tření se nesmí měnit příliš prudce. Z grafu 1 lze odvodit, že tento předpoklad je splněn – tmavší filtrovaný průběh má po celé dráze přibližně stejný charakter bez prudkých změn. Z průběhu v grafu je také patrné, že proud není rozhodně roven nule. To je způsobeno tím, že řídicí elektronika chapadla se snaží udržet konstantní rychlost pohybu prstů a regulací proudu do motoru vyrovnává vliv tření. Z velmi zašuměného průběhu nefiltrovaného proudu (světlejší průběh) je možné odhadovat, že motor je regulován pulsní šířkovou modulací, nebo obdobným pulsním principem. Jedná se pouze o odhad, protože získaný průběh je velmi pravděpodobně podvzorkovaný. Tento fakt ovšem zásadním způsobem komplikuje návrh regulátoru, protože namísto ustáleného klidového odběru proudu při konstantní rychlosti – byť třeba i nenulového je vstupem pro regulátor série ostrých špiček střídaných hodnotami blízko nuly. Úkolem regulátoru tedy bude v tomto silně zašuměném průběhu klidový proud při ustáleném stavu fyzikální soustavy chapadla rozpoznat a správě detekovat a interpretovat odklony od tohoto klidového proudu.

Regulátor pracující v ideálních podmínkách by mohl jednoduše fungovat tak, že by našel hladinu klidového proudu a potom měřil velikost odchylky od této hladiny. Pokud by velikost odchylky překročila stanovenou mez, vyslal by chapadlu příkaz k zastavení prstů. Okamžik kolize prstů s překážkou by také nebylo obtížné detekovat. Nastal by přesně ve chvíli, kdy by se odebíraný proud měřitelně zvýšil nad klidovou hodnotu. Při návrhu regulátoru jsem zvolil přístup držet se pokud možno principu regulátoru pracujícího v ideálním prostředí s tím, že klíčovou roli bude hrát předzpracování signálu. Úkolem předzpracování tedy bude přiblížit se co nejvíce ideálnímu prostředí. Je vysoce pravděpodobné, že ani sofistikovaným předzpracováním nebude vstup ideální. To zkomplikuje hledání hladiny klidového proudu. Vhodné předzpracování a rozumně spolehlivé určení hladiny klidového proudu se tedy pravděpodobně stanou klíčovými úkoly návrhu.

3.5 Struktura regulátoru

Podle principu regulace popsaného algoritmem 1 byla navržena vysokoúrovňová struktura regulátoru. Tato vysokoúrovňová struktura (obrázek 4) popisuje propojení abstraktních bloků regulátoru. V blokovém diagramu jsou také zobrazena rozhraní regulátoru, kterými komunikuje regulátor s chapadlem a s nadřazeným systémem.



Obrázek 4: Blokové schéma regulátoru.

Blok je zde chápán jako logicky oddělená část celku. Šipky propojující bloky znázorňují tok informací – směr toku je reprezentován šipkou. Plné čáry představují přenos dat – tedy vzorků okamžitého proudu. Tečkovaná čára naopak reprezentuje přenos pouze řídicích informací – například

příkazů, jak se mají prsty pohnout, s jakou sílou apod. V diagramu se také vyskytuje čára přerušovaná tečkou. Takováto čára představuje přenos jak dat, tak řídicích a zpětnovazebních informací.

Centrem regulátoru je blok *Řízení regulátoru*. Tento blok ovládá a synchronizuje ostatní bloky a zajišťuje interní logiku regulátoru. Zajišťuje správnou inicializaci vnitřních stavových proměnných regulátoru a ostatních bloků. Algoritmus regulace 1 popsaný výše je vykonáván právě tímto blokem. V průběhu každé iterace komunikuje s ostatními bloky. Nejprve se dotáže na vzorek dat *Komunikace s chapadlem* – tedy vykoná řádek *Získání vzorku okamžitého odběru proudu*. Získaný vzorek potom odešle, jak naznačuje plná čára, *Detektoru kolize* a *Omezovači síly*. Potom provádí v závislosti na aktuálním vnitřním stavu regulátoru dotaz na výskyt kolize, kdy volá *Detektor kolize* a nebo dotaz na překročení limitní síly, kdy komunikuje s *Omezovačem síly*. V případě, že vyhodnotí překročení limitní síly a interní stav regulátoru *STOP* je nastaven na hodnotu *true*, ukončí *Řízení regulátoru* hlavní smyčku a zavolá opět *Komunikaci s chapadlem* a předá jí příkaz k okamžitému zastavení prstů – tedy provede řádek *zastavení pohybu* prstů algoritmu 1. Kromě komunikace s ostatními bloky zajišťuje *Řízení regulátoru* také komunikaci s nadřazeným systémem – tedy obsahuje přímo rozhraní knihovny regulátoru tak, jak je definováno v kapitole 3.2. Směrem od nadřazeného systému tedy přichází požadavky na pohyb prstů s regulací síly a zpět odesílá regulátor jednak hlášení o stavu regulace a jednak umožňuje v případě, že je to nadřazeným systémem požadováno, odesílání každého zachyceného vzorku nadřazenému systému. Zachycené vzorky jsou odesílány bez jakýchkoliv úprav a předzpracování.

Blok *Komunikace s chapadlem* má za úkol, jak už bylo popsáno výše, komunikovat se samotným hardwarem chapadla. Klíčovou částí tohoto bloku je knihovna *m5api*, která provádí generování požadavků po sériové lince a interpretaci odpovědí z řídicí jednotky chapadla. Veškeré přenosy příkazů a dat z nebo do bloku *Řízení regulátoru* jsou tedy blokem *Komunikace s chapadlem* zprostředkovány hardwaru. Uvnitř bloku jsou veškeré přenosy převáděny mezi protokolem chapadla a rozhraním funkcí v jazyce C. Blok samotný zaobaluje knihovnu *m5api* tak, aby odstínil zbytek regulátoru od rutinních operací, jakými je načtení a inicializace knihovny nebo uvolnění alokovaných prostředků.

Další blokem regulátoru je *Detektor kolize*. Tento funkční blok se stará o rozpoznání kolize prstů s překážkou. Ideálně by měla detekce kolize reagovat ihned při styku prstů s překážkou. *Dektektor kolize* není autonomní – spoléhá plně na externí řízení. Po každém vloženém vzorku vrací informaci o tom, zda při tomto vzorku nastala kolize, nebo nikoliv. Lze si ho představit jako Mealyho stavový automat, který přijímá vstupní řetězec vzorků okamžitého průběhu odběru proudu a při každém přechodu mezi vzorky generuje výstup. Vnitřní stav *Detektoru kolize* je pak reprezentován množinou stavů automatu.

Blok *Omezovač síly stisku* zpracovává vzorky průběhu okamžitého odebíraného proudu a vyhodnocuje, zda byla překročena zadaná limitní síla. Před začátkem každého pohybu prstů s regulací síly je komponentou *Řízení regulátoru* její vnitřní stav resetován a je jí nastavena limitní síla, jejíž překročení má střežit. Potom už se *Omezovač síly stisku* chová opět jako Mealyho stavový automat. Při vstupu mění interní stav a generuje výstup. Jakmile na základě současného vzorku a historického průběhu vyhodnotí překročení zadané limitní síly, vystaví tento fakt na výstup. *Řízení regulátoru* na něj adekvátně zareaguje.

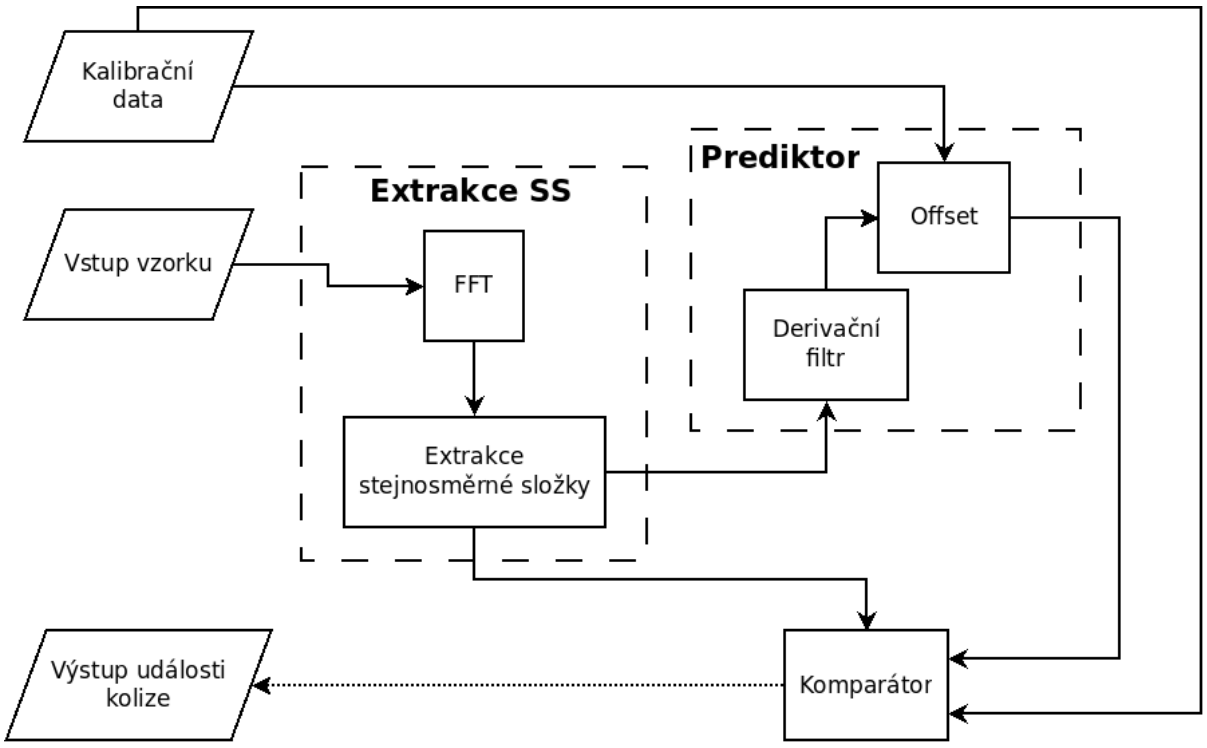
Poslední součástí regulátoru je blok *Kalibrační tabulka* reprezentující tabulku parametrů regulátoru. Hodnoty z této tabulky parametrizují bloky *Detektor kolize* a *Omezovač síly stisku*. Kalibrace je nutná proto, že u reálných systémů dochází ke změnám vlastností vlivem stárnutí, opotřebení a změnám okolního prostředí. V tomto konkrétním případě se vlivem změny teploty a opotřebením může měnit tření mechanické části chapadla, což se promítne i do proudového odběru pohonu. Proto je čas od času nutné skutečné parametry reálného systému změřit a uložit je pro pozdější použití coby parametry řídicích programů pracujících se vstupy z tohoto reálného systému. V případě regulátoru síly stisku jsou v rámci kalibrace ukládány tyto parametry: stejnosměrný posun klidového průběhu okamžitého proudu, limitní hranice proudu pro jednoznačné rozpoznání kolize a převodní tabulka hodnot limitní síly na hodnotu maximálního odebíraného proudu.

Ačkoliv byl popis chování bloků z pohledu zvenčí demonstrován Mealyho stavovým automatem, není tento formální model pro popis vnitřního principu bloků vhodný. Problém je především

v tom, že konečný stavový automat modeluje konečnou množinu diskrétních stavů a přechodů mezi nimi. Mealyho automat je potom pouhým rozšířením stavového automatu o generování výstupu při každém přechodu mezi stavy. Bloky regulátoru ale mají interní stav spojitý. Stav je u nich reprezentován nějakým spojitým „skóre“, které je porovnáváno s limitní hranicí, při jejímž překročení je změněn výstup z výchozí hodnoty *false* na hodnotu *true*. Spojitý stav implikuje stavový automat s nekonečným počtem stavů, což ale neodpovídá definici konečného stavového automatu. Proto byl pro formální popis struktury regulátoru použit formalismus DEVS.

3.6 Detektor kolize

Tato podkapitola se věnuje podrobněji návrhu bloku *Detektor kolize*. Popis byl oddělen do podkapitoly proto, že *Detektor kolize* je netriviálním blokem a její popis je tudíž obsáhlejší. Interní struktura detektoru je popsána blokovým diagramem na obrázku č.5. Pro diagram platí stejné konvence, jako pro diagram celkové struktury regulátoru – tedy plná čára reprezentuje přenos dat a tečkovaná čára představuje napojení události. Oproti diagramu výše jsou zde objekty s přerušovaným obvodem. Jedná se o složené bloky. Tyto bloky jsou složeny z podbloků a jejich interní struktura je v diagramu zakreslena. Jejich celková funkce má však z hlediska popisu význam a proto jsou v diagramu znázorněny. Obdélníky jsou vyznačeny jednotlivé bloky a kosodélníky vstupy a výstupy představující rozhraní se zbytkem regulátoru. *Vstup vzorku* je vstupem od *Řízení regulátoru*, kterým přichází načtený vzorek okamžitého odběru proudu. *Výstup události kolize* je propojen na vstup *Řízení regulátoru* a předává řídicímu bloku regulátoru informaci o tom, zda kolize při daném vzorku nastala, nebo ne. Vstup *Kalibrační data* není v celkovém diagramu nijak znázorněn. Je to vstup parametrů ovlivňujících výpočet kolize – konkrétně se jedná o stejnosměrný posuv prediktoru a limitní proud, při jehož překročení je kolize vyhlášena okamžitě.



Obrázek 5: Vnitřní struktura složeného bloku Detektor kolize.

Detektor kolize se skládá ze tří základních bloků. Jsou jimi *Extrakce stejnosměrné složky*, *Prediktor* a *Komparátor*. *Extrakce SS* a *Prediktor* jsou složené bloky – jsou znázorněny přerušovanou čarou okolo interních bloků, které zahrnují.

Ze signálu přicházejícího z chapadla je nejdříve blokem *Extrakce SS* izolována pouze stejnosměrná složka a tím odstraněn šum. *Extrakce stejnosměrné složky* se skládá ze dvou podbloků: *FFT* a *Extrakce stejnosměrné složky*. *FFT* je blok Fourierovy transformace. Provádí takzvanou rychlou Fourierovu transformaci. Jeho úkolem je převod vstupního průběhu na spektrum. Aby bylo možné spektrum signálu určit, nestačí jeden vzorek, ale je potřeba úsek průběhu. V tomto konkrétním případě je úsek dlouhý 8 vzorků. Pro každý vstupní vzorek tedy použije pro výpočet spektra 7 vzorků předchozích. Získané spektrum zpracovává blok *Extrakce stejnosměrné složky*. Ten vynásobí spektrum signálu spektrem dolnopropustního filtru a převede modifikované spektrum opět na signál. Blok *Extrakce stejnosměrné složky* nepoužívá zpětnou Fourierovu transformaci, protože amplitudu stejnosměrné složky není potřeba převádět na průběh – stejnosměrná složka je konstantní. Dolní propusť potlačuje vliv vysokých a středních frekvencí a nechává pouze frekvence nízké. Tímto způsobem jsou odstraněny špičky, které vznikají skokovou změnou mezi jednotlivými vzorky. Špičky tedy mají vysokou frekvenci. Naopak zůstanou jen nízké frekvence. V tomto případě je filtr dolní propusť seřízena tak, aby propouštěla pouze stejnosměrnou složku. Stejnosměrná složka odpovídá původní konstantní hladině klidového proudu ideálního vstupu. Ve skutečnosti je výsledný filtrovaný průběh zvlněný, jak je vidět v grafu 2, protože filtrace probíhá vždy jen nad krátkými úseky signálu a v těch není možné rozlišit nízkofrekvenční šum, který se na krátkém úseku signálu chová jako konstantní hladina. Princip bloku *Extrakce SS* lze popsat následující funkcí:

$$OUT = (\mathrm{FFT}((IN, HISTORY[1..N-1])) * (1,0,0,0,...))[1] \qquad (1)$$

Kde:

IN	... vstupní vzorek nefiltrovaného průběhu okamžitého proudu [*A*].
OUT	... výstupní vzorek extrahované stejnosměrné složky [*A*].
N	... délka zpracovávaného úseku signálu ve vzorcích [-].
HISTORY	... vektor historických hodnot délky N-1 [*[A]*].
[x]	... konkrétní složka vektoru nebo záznamu.

Z takto předzpracovaného signálu se potom blok *Prediktor* snaží odhadnout průběh signálu tak, jak by vypadal v případě pohybu prstů bez kolize. Prediktor se skládá ze dvou podbloků – derivačního filtru a bloku přidávajícího signálu offset. Derivační filtr nejdříve odstraní stejnosměrnou složku vstupujícího signálu. Pracuje s výrazně větším počtem vzorků než Fourierova transformace v komponentě *Extrakce SS* a může tedy odstranit i velmi nízké frekvence. Délka úseku signálu, nad kterým derivační filtr pracuje, byla experimentálně stanovena na 17 vzorků. Filtr sice odstraní stejnosměrnou složku, ale zachová přitom původní tvar signálu včetně šumu na vyšších frekvencích. Stejnosměrná složka je pak blokem offset do signálu opět dodána, ale už jako ryze konstantní složka. Přidávaný offset je prvním z kalibračních parametrů detektoru. V případě, že se reálný signál vlivem kolize pomalu mění – tedy mění se jeho stejnosměrná složka, vznikne mezi reálným a predikovaným signálem rozdíl díky konstantní stejnosměrné složce u prediktoru rozdíl. Je vhodné podotknout, že měnící se stejnosměrná složka není ve skutečnosti stejnosměrná. Vlivem změny se jedná o složku s velmi nízkou frekvencí. Přesto je pro zjednodušení složka s velmi nízkou frekvencí nazývána stejnosměrnou. Vzhledem k tomu, že prediktor pracuje s historickým signálem omezené délky, není ani predikce dokonalá. Predikovaný průběh postupně následuje změnu stejnosměrné složky, ale výrazně pomaleji, než skutečný filtrovaný průběh. Princip prediktoru lze tedy popsat tímto vzorcem:

$$OUT = \Delta IN + OFFSET \qquad (2)$$

Kde:

OUT	... výstupní vzorek predikce.
IN	... vstup předzpracovaného okamžitého proudu.

OFFSET ... stejnosměrný posuv predikce tak, aby měla stejný stejnosměrný posuv, jaký odpovídá klidovému proudu pohonu chapadla.

Blok *Komparátor* vyhodnocuje rozdíl mezi průběhem skutečného signálu a signálu předpovězeného prediktorem. Odchylku průběhů hodnotí *Komparátor* interním skóre. Pokud skóre překročí stanovenou mez, vyhlásí *Komparátor* kolizi. Do skóre je započítána jak okamžitá odchylka signálů, tak historie odchylky. Kolize se projevuje zvýšením odběru proudu. Proto je kolize detekována jen na neklesajícím průběhu – tedy jen pokud je rozdíl aktuální a minulé hodnoty, hodnota *LastValDelta,* kladná. Vyhodnocení probíhá podle vzorce (3).

$$Score = Score + (Val - LastPrediction) \cdot (Val - LastPrediction + LastValDelta) \qquad (3)$$

Kde:

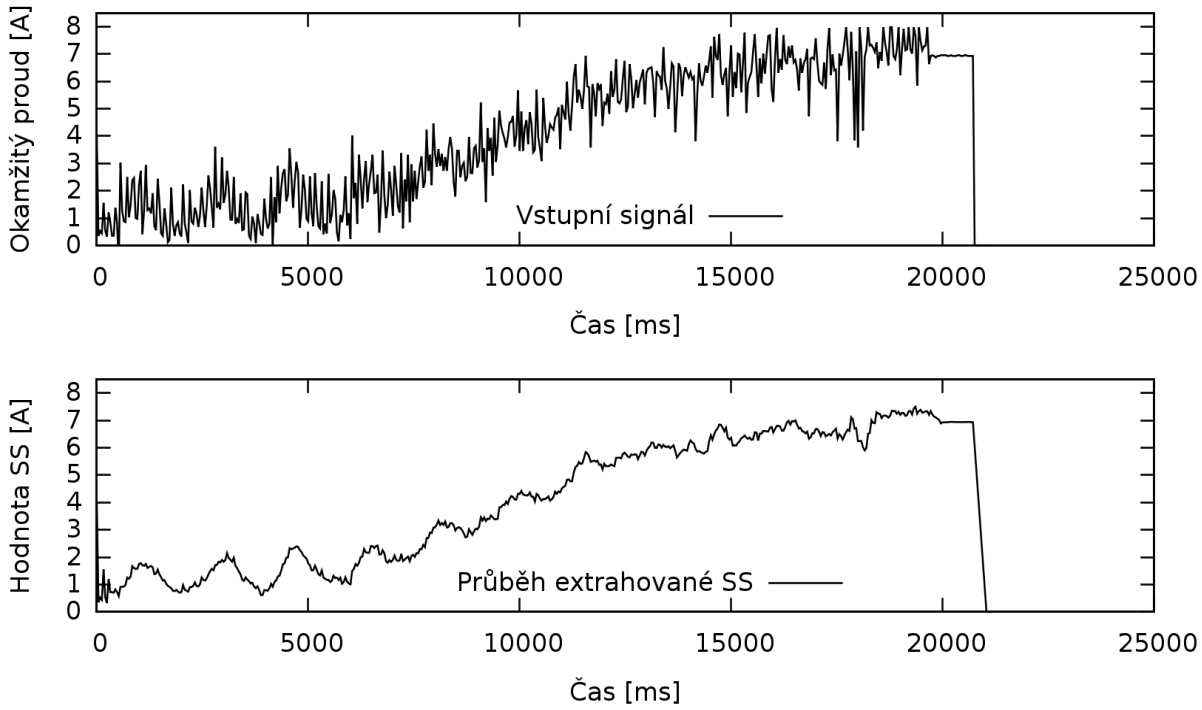
Score ... hodnota skóre .
Val ... aktuální hodnota SS signál.
LastPrediction ... aktuální hodnota signálu prediktoru.
LastValDelta ... rozdíl aktuální a minulé hodnoty.

Do skóre jsou zahrnuty jak rozdíl skutečného signálu od predikovaného, tak i strmost reálného signálu. Vzorec lze interpretovat jako součet druhé mocniny rozdílu skutečné hodnoty od predikované a součinu stejného rozdílu s derivací v posledním bodě průběhu. Výsledek je potom přičten k aktuálnímu skóre. Oproti tomuto popisu je vzorec zjednodušen. V případě, že aktuální hodnota okamžitého proudu bude nižší, než předpověď prediktoru, je skóre resetováno do výchozí hodnoty. Skutečná hodnota, která je nižší než predikovaná, totiž znamená, že kolize určitě nenastává. Princip resetování skóre umožňuje komparátoru překonat mírné odchylky skutečného a predikovaného průběhu a přitom neztratit citlivost v okamžiku kolize. Jediným problémem je rychlost rozpoznání. Je potřeba obvykle několik vzorků, aby jejich součet přesáhl limit. Druhá mocnina navíc snižuje váhu malých rozdílů. Díky tomu sice vyhodnocení založené na skóre obstojně zvládá šum, ale je příliš pomalé pro prudké nárůsty odběru – typicky při kolizi prstů s nepružným předmětem. Proto komparátor paralelně s výše popsaným skóre vyhodnocuje překročení pevné maximální hodnoty signálu. Tato maximální hodnota je druhým z kalibračních parametrů detektoru. *Komparátor* tedy sleduje oba limity a kolizi vyhlásí při překročení kteréhokoliv z nich.

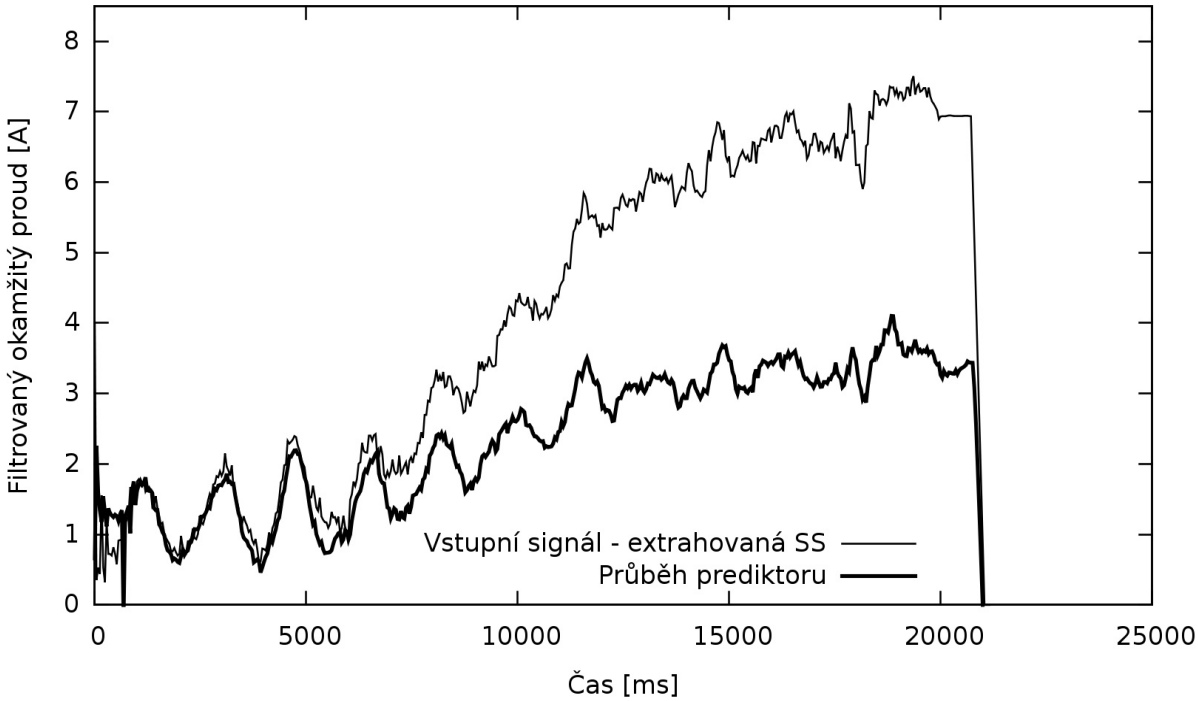
Činnost jednotlivých výše popsaných bloků nejlépe demonstrují grafy. Grafy 2, 3 a 4 vizualizují efekt činnosti jednotlivých bloků na vstupující signál. Z grafů je také patrné, jaký průběh signálu do kterého bloku vstupuje. Všechny grafy jsou generovány nad jedním konkrétním průběhem signálu. Je to průběh okamžitého odběru proudu během kolize prstů s překážkou z měkkého a pružného materiálu. Tento vstup byl zvolen pro svou názornost. Je na něm dobře patrný oakmžik kolize a lze na něm sledovat postupný nárůst odběru vlivem zvyšující se síly působící proti prstům. Pro účely demonstrace chování bloků není tento průběh omezený softwarovou regulací, ale až hardwarovým limitem 8A.

Graf 2 zobrazuje účinky extrakce stejnosměrné složky na vstupní zašuměný signál. Z grafu je patrné, že nízkofrekvenční zvlnění extrakce stejnosměrné složky neodstranila. V průběhu zůstávají i pozůstatky vysokofrekvenčního šumu, ale amplituda tohoto šumu byla výrazně snížena. V oblasti nižšího odběru proudu je do času přibližně 7000ms je zřetelně vidět také zvlnění způsobené nízkofrekvenčním šumem.

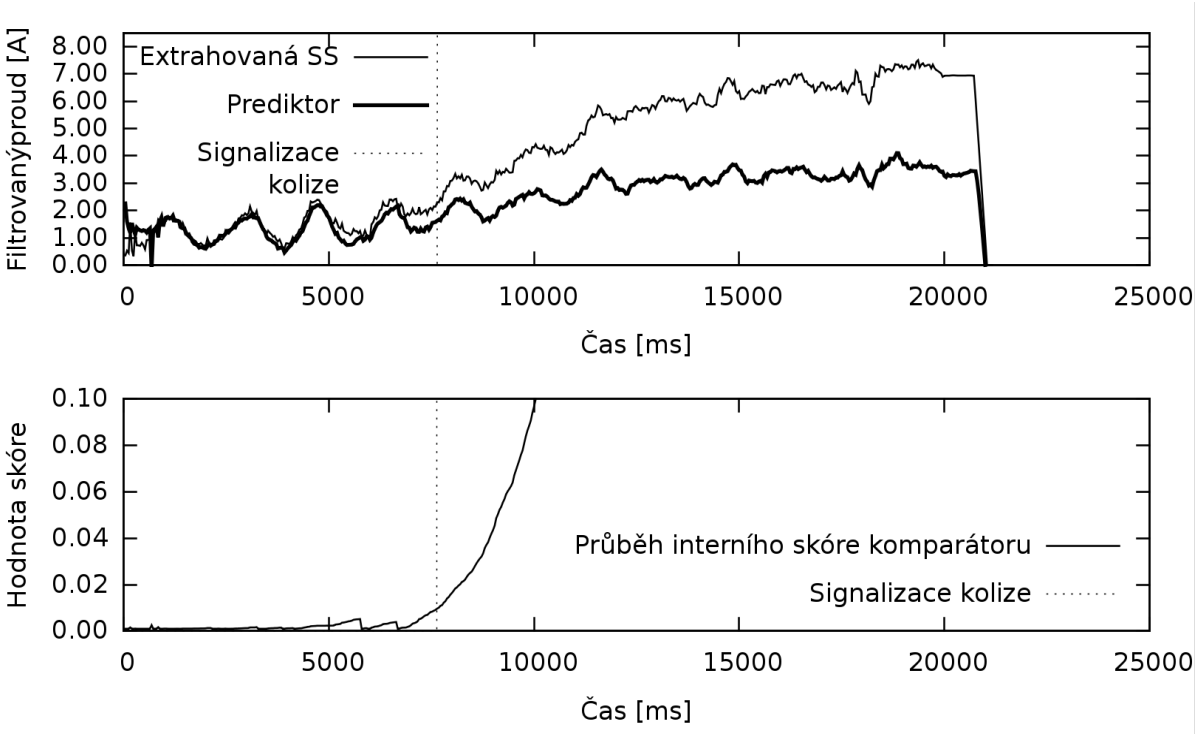
Graf 3 demonstruje rozdíl mezi filtrovaným vstupním průběhem a predikcí klidového průběhu. Z grafu je vidět, že průběhy jsou si v oblasti, kde se prsty pohybují bez odporu, velice podobné. Naopak v okamžiku, kdy začne vlivem interakce s překážkou stoupat proud odebíraný pohonem chapadla, začnou se průběhy rozcházet. S nárůstem odebíraného proudu rozdíl mezi průběhy roste. V grafu je také dobře patrná nedokonalost prediktoru. Predikovaný průběh následuje trend původního signálu a také mění svůj stejnosměrný posuv, ačkoliv by se to ideálně dít nemělo.



Graf 2: Extrakce stejnosměrné složky ze vstupního signálu.



Graf 3: Srovnání průběhu skutečného filtrovaného signálu a prediktoru.



Graf 4: Srovnání průběhu skóre komparátoru s rozdílem skutečného a predikovaného průběhu.

Graf 4 zobrazuje průběh skóre uvnitř komparátoru. Průběh skóre je v něm srovnáván s průběhy vstupů komparátoru – tedy s filtrovaným vstupem a průběhem generovaným prediktorem. Z průběhů je poměrně dobře vidět, že tam, kde skutečný průběh překračuje ten predikovaný, skóre roste. V čase mezi 6000ms a 7000ms vyhodnocuje komparátor kolizi. Okamžik kolize je vyznačený tečkovanou čarou. V oblasti okolo času 6000ms je možné si všimnout několika pozvolných nárůstů skóre následovaných prudkým skokem zpět na nulu. Ostré skoky do nuly jsou způsobeny tím, že průběh skutečného vstupu se dostal pod hodnotu predikovaného průběhu v daném čase. V takové situaci je skóre vynulováno.

3.7 Omezovač síly stisku

Blok *Omezovač síly stisku* je netriviální, proto mu byla věnována samostatná podkapitola. Tento blok hraje při regulaci klíčovou roli, protože rozhoduje o okamžiku, kdy došlo k překročení limitní síly. Základním úkolem *Omezovače síly stisku* je ohodnocení každého příchozího vzorku podobně jako u *Detektoru kolize* Vnitřní struktura omezovače je popsána blokovým diagramem na obrázku č.6. Diagram dodržuje stejné konvence, jako předchozí diagramy týkající se struktury regulátoru. Vstupy a výstupy jsou – s výjimkou vstupu *Kalibrační data* – propojeny s blokem *Řízení regulátoru*. Oproti *Detektoru kolize* má *Omezovač síly* více vstupů. Přibyl vstup *Omezení síly*. Tento vstup souvisí s inicializací regulace. Před zahájením každého pohybu prstů s regulací síly je tímto vstupem zaslána hodnota síly, jejíž překročení má omezovač hlídat.

Před zahájením samotné regulace zašle blok *Řízení regulátoru* hodnotu limitní síly, na kterou požaduje uživatel sílu stisku prstů omezit. Tuto hodnotu je potřeba nejprve převést na proud. To zajišťuje blok *Výpočet limitního proudu*. Vstupem pro tento blok je jednak hodnota limitní síly a také ka-

librační tabulka. Kalibrační tabulka je převodní tabulka síly na maximální proud a opačně. Obsahuje v jednom sloupci hodnoty proudu a v druhém odpovídající hodnoty síly získané měřením. Kalibrační tabulka plní úkol převodní funkce síly na proud. Vzhledem k tomu, že tabulka obsahuje konečnou množinu diskrétních záznamů, je vlastně interpolací převodní funkce polynomem nultého stupně. Aby mohl blok *Výpočet limitního proudu* určit přesněji hodnotu proudového omezení i pro hodnotu síly, pro kterou není v tabulce záznam, je nutné převodní funkci interpolovat polynomem vyššího stupně. Blok provádí nejjednodušší interpolaci polynomem prvního stupně – tedy propojí sousední body přímkou. Algoritmus převodu pracuje tak, že najde hodnotu síly nejbližší požadované síle a k ní dohledá druhou nejbližší hodnotu dle velikosti proudu. Prakticky to znamená, že najde krajní body úseku, do kterého spadá, v grafu, kde je na x-ové ose proud a na y-ové ose síla. V tomto úseku vypočítá odpovídající limitní proud pro zadanou sílu principem trojúhelníkové nerovnosti. Takto relativně komplikovaný výpočet se provádí proto, že průběh síly v závislosti na odebíraném proudu je silně zvlněný a ne vždy nutně platí, že s rostoucím limitním proudem síla roste. V průběhu závislosti síly na limitním proudu existují úseky, kde i přes rostoucí hodnotu proudu síla stisku klesá. Převod je popsán vztahem (4). Konkrétní průběh je pak zobrazen v grafu 5.

$$OUT = \frac{I_{closest} - I_{second}}{F_{closest} - F_{second}} \cdot (F_{\mathrm{IN}} - F_{second}) + I_{second} \tag{4}$$

Kde:

$$F_{closest} = \arg\min_{F_{\mathrm{CAL}}} |F_{\mathrm{IN}} - F_{\mathrm{CAL}}|$$

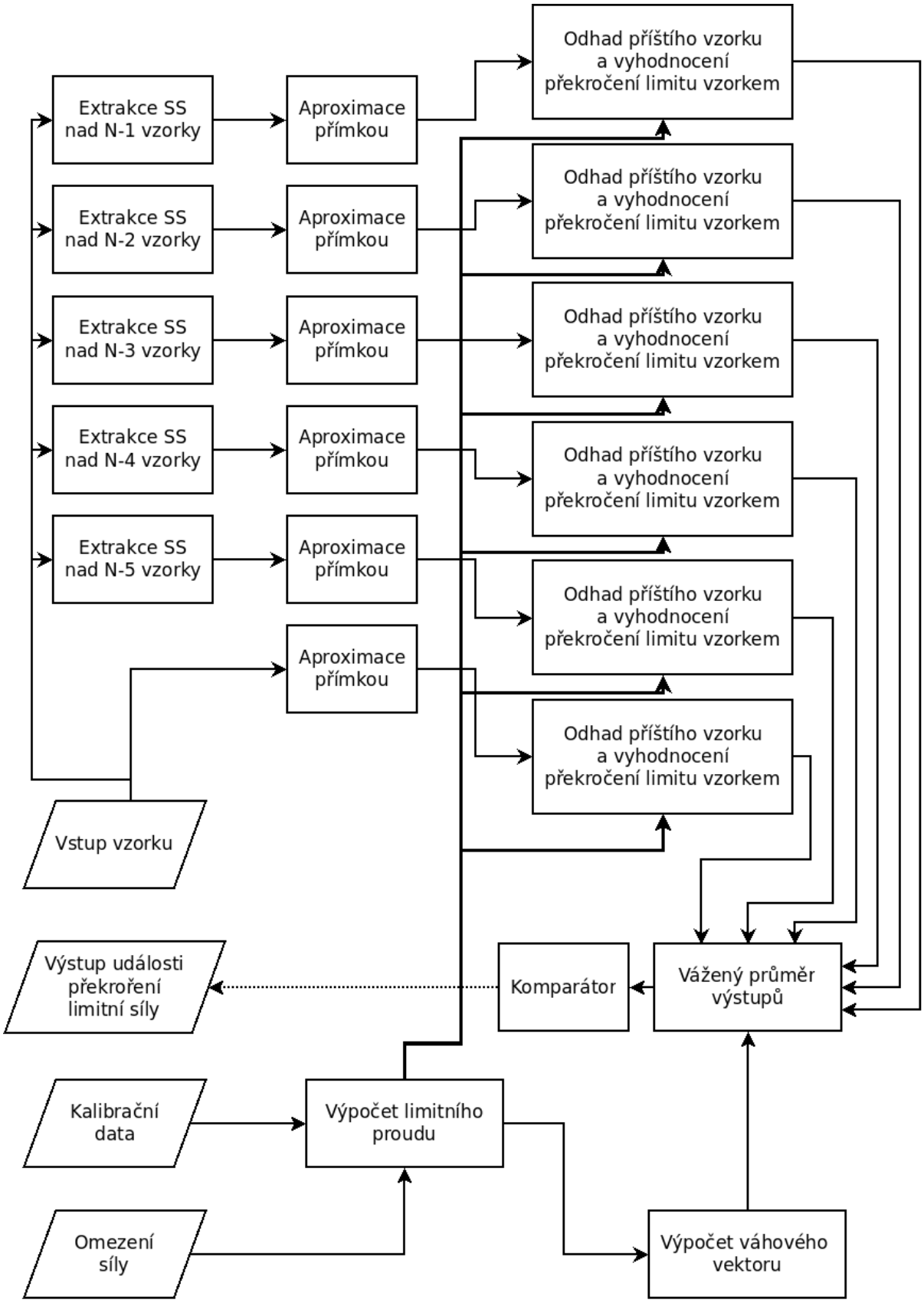
$$I_{closest} = \varphi(F_{closest})$$

$$I_{second} = \arg\min_{I_{\mathrm{CAL}}} (\sigma[(I_{\mathrm{CAL}} - I_{closest}) \cdot (F_{\mathrm{IN}} - F_{closest})])$$

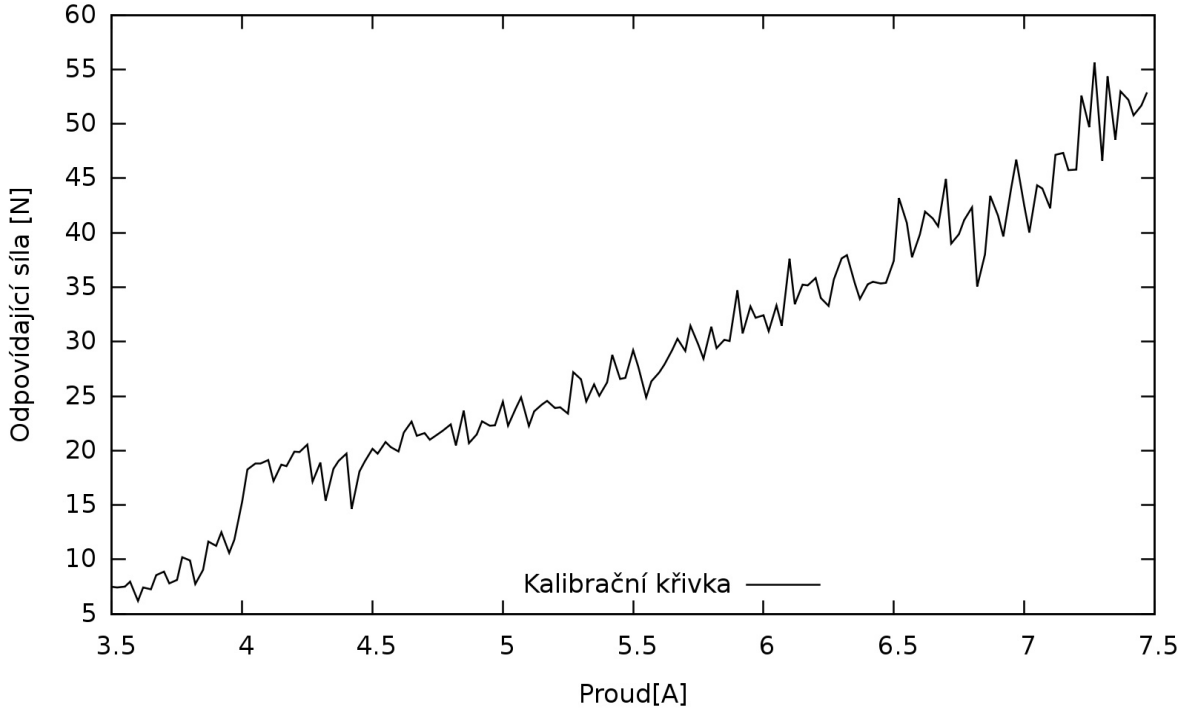
$$F_{closest} = \varphi^{-1}(I_{closest})$$

$F_{closest}$ ... hodnota síly z kalibrační tabulky nejblíže požadované síle na vstupu bloku.
F_{IN} ... hodnota síly zadaná na vstupu.
F_{CAL} ... obecně hodnota síly uložená v tabulce.
$I_{closest}$ ... hodnota proudu odpovídající síle $F_{closest}$ v tabulce.
φ ... převodní funkce $\mathbb{R} \to \mathbb{R}$ síly v tabulce na odpovídající proud.
I_{second} ... druhá hranice intervalu na proudové ose, do kterého spadá vstup.
I_{CAL} ... obecně hodnota proudu uložená v tabulce.
σ ... zobrazení $\mathbb{R} \to \mathbb{R}: \begin{matrix}(x>0) \to y = x \\ (x \le 0) \to y = \infty\end{matrix}$.

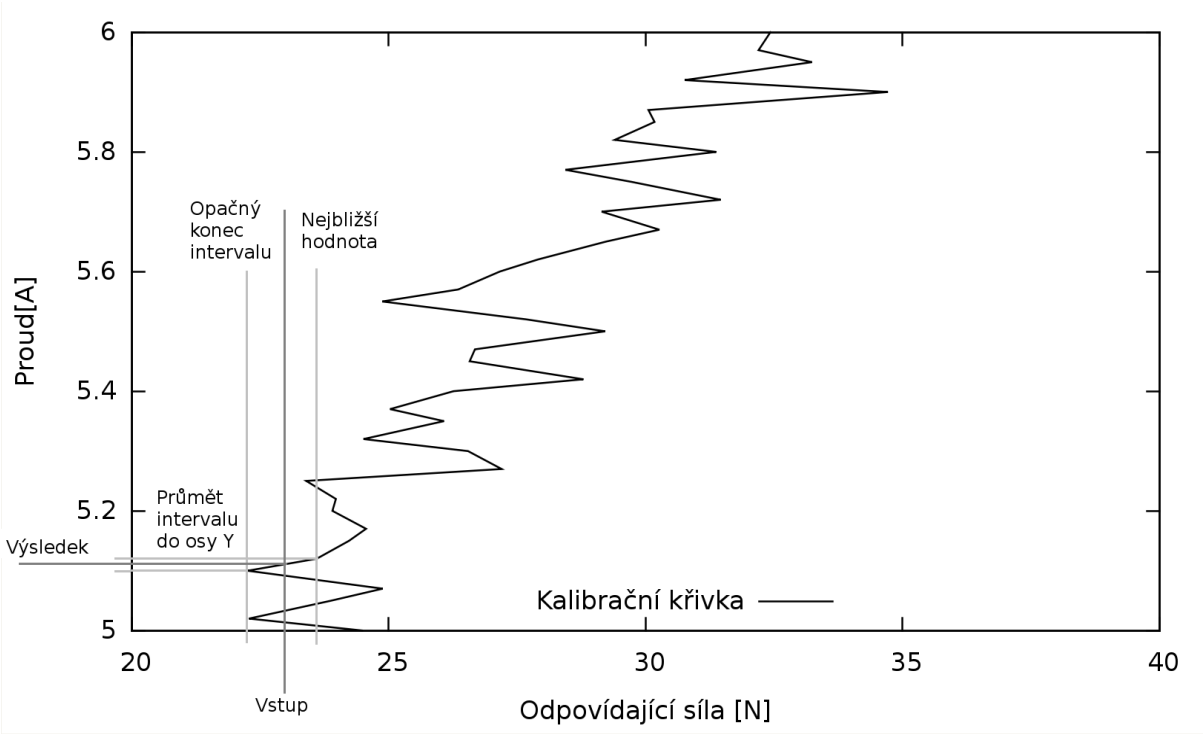
Graf 5 ukazuje závislost limitní síly na hodnotě omezení proudu. Hladina limitního proudu se lineárně zvedá ve směru proudové osy. Jak je ale vidět, síla nestoupá zcela lineárně spolu s hladinou limitního proudu. Dokonce je v grafu běžné, že vyššímu limitnímu proudu odpovídá nižší síla než u předchozího vzorku. Do značné míry je to způsobeno šumem vstupních hodnot. Síla se při jednotlivých stiscích při experimentování lišila o desítky procent od střední hodnoty. Pravděpodobně se ale na průběhu projevuje i samotný princip regulace, kdy díky aproximaci přímkou a výpočtu příští hodnoty může dojít k překročení hodnoty limitního proudu budoucím vzorkem o hodnotu, která převýší nejen aktuální omezení, ale i omezení následující. Tedy pro různé hodnoty omezujícího proudu blízko sebe může být síla stisku stejná. Roli také může hrát samotná regulace rychlosti pohybu prstů, u které není přesně známo, jak probíhá. Regulátor rychlosti pohybu prstů nějakým způsobem moduluje průběh okamžitého proudu, ale tento průběh je regulátorem síly stisku pravděpodobně podvzorkovaný. Může se tedy stát, že průběh navzorkovaný regulátorem síly stisku nebude odpovídat skutečnému průběhu do té míry, že i pro nižší sílu bude mít strmější náběh a bude tedy regulátorem síly stisku omezen dříve. V grafu je také patrný poměrně prudký nárůst síly v oblasti okolo 4A, který se postupně srovnává až do přibližně 4,5A. Tento skok je pravděpodobně způsobem sloučením výstupů jednotlivých cest při vyhodnocování překročení limitního proudu popsaném níže. Přesto, že byla snaha slučování cest co nejlépe odladit, není zdá se stále ideální.



Obrázek 6: Vnitřní struktura složeného bloku Omezovač síly stisku.



Graf 5: Kalibrační křivka - vztah maximální síly a limitního proudu.



Graf 6: Vizualizace výpočtu limitního proudu.

V grafu 6 je naznačen výpočet limitního proudu pro konkrétní zadanou sílu. Nejdříve je na osu síly přiveden vstup (tmavě šedá svislá čára), potom jsou určeny hranice intervalu, kam hodnota síly spadá (světle šedé svislé čáry), tento interval je zobrazen na osu proudu (světle šedé vodorovné čáry) a na základě podobnosti trojúhelníků je určena výsledná hodnota limitního proudu (tmavě šedá vodorovná čára).

V datovém souboru kalibrační tabulky jsou kromě tohoto průběhu uloženy i kalibrační parametry pro detektor kolize. Datový soubor je ve formátu CSV s unixovými konci řádků. Oddělovačem je mezera. Má následující tvar:

```
collision 66 260
350 7492
352 7424
...
```

Řádek začínající řetězcem „collision“ obsahuje parametry detektoru kolize. Ostatní řádky potom představují jednotlivé body kalibrační křivky. Parametry detektoru kolize jsou v pořadí offset prediktoru, limitní hodnota komparátoru. Jsou zapsány v setinách ampéru. U zbylých řádků je první sloupec hodnota limitního proudu rovněž v setinách ampéru a druhý sloupec hodnota odpovídající změřené síly v mN. Jednotky setiny A a mN jsou použity především proto, aby bylo možné hodnoty zapsat celým číslem. Celá čísla totiž zjednodušují načítání a zpracování dat.

Hodnotu proudu odpovídající zadané síle vypočtenou blokem *Výpočet limitního proudu* využívá blok *Výpočet váhového vektoru* a všechny bloky *Odhad příštího vzorku a vyhodnocení překročení limitu vzorkem*. Váhový vektor rozhoduje o tom, jaký význam budou mít jednotlivé paralelní cesty zpracování vstupního signálu na rozhodnutí o překročení limitního proudu. Pro různé hodnoty proudu je nutné pro dosažení co nejlepších výsledků použít různé způsoby zpracování signálu. Konkrétně pro nízké hodnoty se ukázalo být vhodnější průměrovat výstupy více cest využívajících vstupní filtraci extrakcí stejnosměrné složky. Je to způsobeno především značným šumem v průběhu proudu při nízkém zatížení prstů. Naopak pro vysoké hodnoty proudu je lepší použít vstup bez filtrace, protože při vyšších silách je šum znatelně nižší a také se přeskočením filtrace sníží zpoždění v náběhu proudu, který je zvláště pro větší síly strmý. Díky přeskočení filtrace reaguje regulátor na tyto strmé náběhy rychleji. Celkově výpočet váhového vektoru vypadá tak, že pro nízké hodnoty proudu do 3,5A používá pouze cesty s předzpracováním a váhu mezi ně dělí rovnoměrně, pro proud v rozsahu od 3,5A do 5,0A dělí váhu mezi cesty s předzpracováním a přímou cestu podle toho, v jak daleko se nachází od které krajní meze intervalu a při proudech nad 5,0A už cestám s předzpracováním přiřazuje nulovou váhu a rozhodování probíhá jen na základě výstupu přímé cesty. V oblasti, kde dělí váhu mezi cesty s předzpracováním a cestu bez předzpracování postupuje tak, že napřed rozdělí váhu na dva díly – jeden díl pro předzpracování a druhý pro přímou cestu. Díl, který náleží předzpracování, potom rovnoměrně rozdělí mezi cesty s předzpracováním. Výsledný vektor vah je potom odeslán bloku *Vážený průměr výstupů*. Konkrétní hodnoty mezí byly zjištěny experimentováním s regulátorem. Z matematického hlediska provádí blok následující zobrazení:

$$f(I_{IN}):\mathbb{R}\rightarrow\mathbb{R}^6:\quad \begin{array}{c} \boldsymbol{w}[0]=0,\boldsymbol{w}[1..5]=0{,}2\ pro\ I_{IN}\leq 3{,}5 \\ \boldsymbol{w}[0]=\frac{I_{IN}-3{,}5}{5{,}0-3{,}5},\boldsymbol{w}[1..5]=\frac{1-\boldsymbol{w}[0]}{5}\ pro\ 3{,}5>I_{IN}<5{,}0 \\ \boldsymbol{w}[0]=1,\boldsymbol{w}[1..5]=0\ pro\ I_{IN}\geq 5{,}0 \end{array} \qquad (5)$$

Kde:

w ... výstupní vektor vah, na který je vstup zobrazen.
I_{IN} ... vstupní hodnota proudu.

Blok *Vážený průměr výstupů* provádí čistě výpočet váženého průměru výstupů jednotlivých cest zpracování. Slučuje tak jednotlivé výstupy do jediného, který je vstupem komparátoru. Váhy

jednotlivých cest jsou dány položkami ve vektoru vah. Vektor vah je normalizovaný – součet vah ve vektoru je roven jedné. *Vážený průměr výstupů* proto pouze vynásobí hodnoty výstupů jednotlivých cest odpovídající váhou a všechny takovéto součiny sečte. Činnost bloku se řídí vzorcem:

$$OUT = \sum_{i=1}^{6} \boldsymbol{w}[i] * INPUT[i] \tag{6}$$

Kde:

OUT	... sloučený výstup cest zpracování.
INPUT[i]	... výstup i-té cesty.
w	... váhový vektor.

Výstup *Váženého průměru výstupů* zpracovává *Komparátor*. Tento blok provádí jednoduché porovnání vstupu s limitní úrovní a v případě překročení této úrovně vygeneruje výstupní událost překročení zadané síly. Na tuto událost následně reaguje blok *Řízení regulátoru*.

Nyní se zaměříme na popis jednotlivých cest zpracování vstupního signálu. Cesty zpracování lze rozdělit do dvou skupin podle přítomnosti bloku vstupní filtrace. Jedna skupina je tvořena cestou, která vstupní filtraci nemá. Druhá potom zbylými cestami. Cesty ve druhé skupině se liší pouze počtem vzorků, na kterých provádí filtraci. Počet vzorků je odvozen od konstanty *N*. Hodnota této konstanty je rovna počtu vzorků, jaký používá extrakce stejnosměrné složky v bloku *Detektor kolize*. Cesta s nejdelším úsekem předzpracování tedy využívá posledních 7 vzorků, cesta s nejkratším úsekem předzpracování pak už jen 3 vzorky. Blok *Extrakce SS* se interní strukturou i funkcí shoduje s blokem *Extrakce SS* popsaném v rámci rozboru složeného bloku *Detektor kolize*. Jeho úkolem je opět odstranit ze vstupního signálu vysokofrekvenční šum.

Dalším blokem zpracování přítomným na všech cestách je *Aproximace přímkou*. Tento blok se snaží vyhodnotit aktuální trend signálu proložením koncového úseku signálu přímkou. Blok pracuje s několika posledními vzorky signálu - v tomto případě s 13 včetně aktuálního vzorku. Tyto vzorky jsou vzaty jako řídicí body aproximační funkce. Parametry přímky aproximující koncový úsek signálu jsou počítány metodou nejmenších čtverců, kdy je snahou, aby součet všech druhých mocnin vzdáleností řídicích bodů od přímky byl co nejmenší. Aproximace průběhu přímkou tedy hledá takovou přímku, kterou bude možné nahradit původní průběh s co nejmenší chybou. Aproximací získáme koeficienty směrnicového tvaru přímky $y=px + q$, kde x je vstup, y výstup p a q jsou parametry. Dosazením za parametry získáme jednoduchou matematickou funkci popisující s určitou chybou původní průběh a to i v místech, kde není původní průběh explicitně definován. Parametry p a q jsou blokem *Aproximace přímkou* počítány dle vztahu vzniklého převzatého z literatorury [8] a [9]. Výpočetní vztah je následující:

$$p = \frac{n \cdot \sum_{i=1}^{n} x_i \cdot y_i - \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} y_i}{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i^2\right)} \quad , \quad q = \frac{\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} x_i \cdot y_i}{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i^2\right)} \tag{7}$$

Kde:

p	… směrnice přímky aproximující úsek průběhu.
q	… posuv přímky z nulové hodnoty.
n	… počet vzorků úseku průběhu.
x_i	… x-ová souřadnice i-tého vzorku.
y_i	… y-ová souřadnice i-tého vzorku.

Posledním blokem na každé z cest zpracování signálu je *Odhad příštího vzorku a vyhodnocení překročení limitu vzorkem*. Tento blok se snaží předpovědět, zda budoucí vzorek okamžitého proudu překročí stanovenou mez nebo ne. K predikci využívá právě aproximace vypočítané blokem

Aproximace přímkou. Přímku využije jako ukazatel trendu, ve kterém pokračuje i pro jeden budoucí vzorek. Hodnotu vzorku vypočítá dosazením do rovnice přímky $y=px+q$, kde parametry p a q získává od bloku *Aproximace přímkou*. Dosazení provádí pro čas v budoucnosti *t+délka vzorkovací periody*. Výsledek potom porovná s hladinou limitního proudu pro zadanou maximální sílu. Pokud je hladina překročena, generuje na výstupu hodnotu 1, jinak vrací hodnotu 0. Výpočet budoucího vzorku tedy probíhá podle vztahu:

$$i_{next} = p \cdot (L \cdot T_{sample} + T_{sample}) + q \tag{8}$$

Kde:

- i_{next} ... odhadovaná hodnota budoucího vzorku proudu.
- p ... směrnice přímkové aproximace.
- q ... posuv přímkové aproximace.
- L ... počet vzorků, nad kterými aproximace přímkou pracuje.
- T_{sample} ... perioda vzorkování.

Výsledek výpočtu vzorcem (8) je poté funkcí definovanou vzorcem (9) zobrazen do množiny {0,1}.

$$g(i_{next}, limit): \mathbb{R}^2 \rightarrow \{0,1\}: \begin{array}{l} g(i_{next}, limit) = 1 \; pro \; i_{next} \geq limit \\ g(i_{next}, limit) = 0 \; pro \; i_{next} < limit \end{array} \tag{9}$$

Kde:

- i_{next} ... odhadovaná hodnota budoucího vzorku proudu.
- limit ... maximální hladina proudu pro dodržení zadané limitní síly.

Funkce $g(i_{next}, limit)$ generuje výstup jedné cesty. Výstupní hodnoty této funkce pro každou z cest zpracovává blok *Vážený průměr výstupů* způsobem popsaným výše.

3.8 Formální popis regulátoru

V předchozích podkapitolách byly popsány exaktně matematické principy uvnitř regulátoru. Struktura regulátoru v nich však byla popsána jen neformálně. Proto se tato kapitola věnuje formálnímu popisu struktury a chování regulátoru. Vzhledem k tomu, že regulátor je diskrétní systém, byl pro popis zvolen formalismus DEVS. Celý regulátor je popsaný strukturovaným DEVS a jednotlivé komponenty atomickým DEVS. Zápis formalismu je oproti skutečnosti zjednodušený. Jednak v tom směru, že namísto konkrétních matematických rovnic popsaných výše jsou používány abstraktní funkce opisující svým názvem funkčnost, kterou zastávají a jednak je zjednodušen samotný zápis funkcí atomického DEVS tím, že funkce nejsou definovány pro chybné vstupy. Nastane-li tedy volání některé z funkcí pro vstupy, pro které není definována, implikuje to automaticky chybu v modelu. Popis vychází z následující definice formalismu DEVS (definice převzata z [10]):

Strukturovaný DEVS:

$$DEVN = (X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select)$$

Kde:

- X ... množina vstupních událostí.
- Y ... množina výstupních událostí.
- D ... množina odkazů na DEVS komponenty.
- $\{M_d\}$ ... množina DEVS modelů, platí: $\forall d \in D: \exists M_d \in \{M_d\}$.
- $\{I_d\}$ ... množina komponent ovlivňujících d: $\forall d \in D \cup N: I_d \in D \cup N, d \notin I_d$.

$\{Z_{i,d}\}$ ... propojení komponent – zobrazení:
$X \to X_d$ pro $i = N$
$Y_d \to Y$ pro $d = N$
$Y_i \to X_d$ pro $d \neq N, i \neq N$

Select ... funkce priority $2^D \to D$ – rozhoduje v případě výskytu interních událostí ve více komponentách současně.

Atomický DEVS:

$$DEVS = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

Kde:

X ... množina vstupních událostí.
Y ... množina výstupních událostí.
S ... množina vnitřních stavů modelu.
δ_{int} ... interní přechodová funkce $S \to S$.
δ_{ext} ... externí přechodová funkce $(S, X, e) \to S : e \in \langle 0, ta(s) \rangle, s \in S$.
Tedy *e* je délka časového intervalu od poslední interní události.
λ ... výstupní funkce $S \to Y$.
ta ... funkce posunu v čase $S \to \mathbb{R}^+ \cup \{\infty\}$.

Množina vnitřních stavů modelu je definována jako strukturovaná množina:

$$S = \{v \mid v \in S_1 \times S_2 \times \ldots \times S_n, S_1 \neq \emptyset, S_2 \neq \emptyset, \ldots, S_n \neq \emptyset\}$$

Kde:

v ... prvek množiny – podmnožina kartézského součinu množin S_X.
S_X ... libovolná neprázdná množina hodnot – datový typ.

Vnitřní stav atomického DEVS je tedy uspořádaná n-tice hodnot. Obor hodnot jednotlivých složek uspořádané n-tice je definován odpovídající množinou S_X. S_X tímto definuje datový typ složky n-tice *v*.

3.8.1 Strukturovaný DEVS regulátoru

Regulátor síly stisku $= (X, Y, D, M_d, I_d, Z_i, d, Select)$
$X = \{$ Vstup příkazu, Vstup vzorku $\}$
$Y = \{$ Výstup příkazu pro chapadlo, Výstup vzorku a stavových informací $\}$
$D = \{$ Detektor kolize, Komunikace s chapadlem, Omezovač síly stisku, Řízení regulátoru, Kalibrační tabulka $\}$

$\{I_d\} = \{$
$I_{\text{Detektor kolize}} = \{$ Řízení regulátoru, Kalibrační tabulka $\}$
$I_{\text{Omezovač síly stisku}} = \{$ Řízení regulátoru, Kalibrační tabulka $\}$
$I_{\text{Kalibrační tabulka}} = \{$ Řízení regulátoru $\}$
$I_{\text{Komunikace s chapadlem}} = \{$ Řízení regulátoru, Regulátor síly stisku $\}$
$I_{\text{Řízení regulace}} = \{$ Regulátor síly stisku, Komunikace s chapadlem, Omezovač síly stisku, Detektor kolize $\}$
$I_{\text{Regulátor síly stisku}} = \{$ Řízení regulátoru, Komunikace s chapadlem $\}$
$\}$

$$
\begin{aligned}
&\{Z_{i,d}\} = \{ \\
&\quad Z_{i,\text{Detektor kolize}} = \{ (\text{Řízení regulátoru . Výstup pro detektor kolize}, \\
&\qquad \text{Detektor kolize . Vstup vzorku}), \\
&\qquad (\text{Kalibrační tabulka . Kalibrace detektoru}, \\
&\qquad \text{Detektor kolize . Kalibrační data}) \} \\
&\quad Z_{i,\text{Omezovač síly stisku}} = \{ (\text{Řízení regulátoru . Výstup pro omezovač síly}, \\
&\qquad \text{Omezovač síly stisku . Vstup vzorku}), \\
&\qquad (\text{Kalibrační tabulka . Kalibrace omezovače}, \\
&\qquad \text{Detektor kolize . Kalibrační data}) \} \\
&\quad Z_{i,\text{Kalibrační tabulka}} = \{ (\text{Řízení regulátoru . Událost kalibrace}, \\
&\qquad \text{Kalibrační tabulka . Vstup události kalibrace}) \} \\
&\quad Z_{i,\text{Komunikace s chapadlem}} = \{ (\text{Řízení regulátoru . Výstup pro komunikaci s chapadlem}, \\
&\qquad \text{Komunikace s chapadlem . Vstup z řízení regulátoru}), \\
&\qquad (\text{Regulátor síly stisku . Vstup vzorku}, \\
&\qquad \text{Komunikace s chapadlem . Vstup chapadla}) \} \\
&\quad Z_{i,\text{Řízení regulátoru}} = \{ (\text{Regulátor síly stisku . Vstup příkazu}, \text{Řízení regulátoru . Vstup příkazu}), \\
&\qquad (\text{Komunikace s chapadlem . Výstup pro řízení regulace}, \\
&\qquad \text{Řízení regulátoru . Vstup z chapadla}), \\
&\qquad (\text{Detektor kolize . Výstup události kolize}, \\
&\qquad \text{Řízení regulátoru . Vstup z detektoru kolize}), \\
&\qquad (\text{Omezovač síly stisku . Překročení síly}, \\
&\qquad \text{Řízení regulátoru . Vstup z omezovače síly}) \} \\
&\quad Z_{i,\text{Regulátor síly stisku}} = \{ (\text{Řízení regulátoru . Výstup vzorku}, \\
&\qquad \text{Regulátor síly stisku . Výstup vzorku a stavových informací}), \\
&\qquad (\text{Řízení regulátoru . Událost dokončení}, \\
&\qquad \text{Regulátor síly stisku . Výstup vzorku a stavových informací}), \\
&\qquad (\text{Komunikace s chapadlem . Výstup příkazu pro chapadlo}, \\
&\qquad \text{Regulátor síly stisku . Výstup příkazu pro chapadlo}) \} \\
&\} \\
\end{aligned}
$$

Události *Výstup vzorku* a *Událost ukončení* indikující konec regulace jsou sloučeny do jediného výstupu. Toto je zjednodušení modelu oproti skutečnosti. Reálně jsou události oddělené různým rozhraním.

$$\text{SELECT} = \{(\{\text{Řízení regulace}, \text{Komunikace s chapadlem}\}, \text{Řízení regulace})\}$$

Komponenta *Řízení regulátoru* má přednost před jakoukoliv další komponentou. Prakticky se mohou setkat pouze s interní událostí v komponentě *Komunikace s chapadlem*. Jednotlivé prvky množiny $\{M_d\}$ reprezentují následující podkapitoly.

3.8.2 Atomický DEVS Detektoru kolize

$$\text{Detektor kolize} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

$X = \{$ Kalibrační data, Vstup vzorku $\}$

$Y = \{$ Výstup události kolize $\}$

$$S = \{(\text{SCORE}, \text{OVER}, \text{IN}, \text{OFFSET}, \text{LEVEL}) | \text{SCORE} \in \mathbb{R}^+ \cup \{0\}, \text{OVER} \in \{true, false\}, \text{OFFSET} \in \mathbb{R}, \text{LEVEL} \in \mathbb{R}\}$$

$$\delta_{\text{int}}((\text{SCORE}, \text{OVER}, \text{IN}, \text{OFFSET}, \text{LEVEL}))\{$$

return $(\text{SCORE}, \text{OVER}, false, \text{OFFSET}, \text{LEVEL})$*;*

$\}$

Interní přechodová funkce přepíná vnitřní stav na *false* v proměnné *IN*. Tím signalizuje dokončení výstupu. Externí přechodová funkce je potom pro přehlednost rozdělena na reakce na jednotlivé vstupní události.

$\delta_{ext}((\text{SCORE}, \text{OVER}, \text{IN}, \text{OFFSET}, \text{LEVEL}), \text{Kalibrační data}, e)\{$
  return$(0, \mathit{false}, \mathit{false}, \text{Kalibrační data}.\text{offset}, \text{Kalibrační data}.\text{level})$*;*
$\}$

Reakce na událost kalibrace je pouhé uložení nových kalibračních údajů a resetování vnitřního stavu – není generován žádný výstup.

$\delta_{ext}((\text{SCORE}, \text{OVER}, \text{IN}, \text{OFFSET}, \text{LEVEL}), \text{Vstup vzorku}, e)\{$
  $\text{TMPSC} = \mathit{Výpočet\_skóre}(\text{SCORE}, \mathit{Extrakce\_SS}(\text{Vstup vzorku}.\text{hodnota}), \text{OFFSET}, \text{LEVEL})$*;*
  if $(\text{TMPSC} > \text{LIMIT}) \vee (\text{Vstup vzorku}.\text{hodnota} > \text{LEVEL})$ ***then***
    return$(\text{TMPSC}, \mathit{true}, \text{true}, \text{OFFSET}, \text{LEVEL})$*;*
  else
    return$(\text{TMPSC}, \mathit{false}, \mathit{true}, \text{OFFSET}, \text{LEVEL})$*;*
  endif
$\}$

Reakce na vstup vzorku vypočítá nové interní skóre do proměnné *SCORE*, ověří, zda došlo k překročení limitu skóre *LIMIT* nebo zda vstupní průběh překračuje pevnou hodnotu *LEVEL,* a podle toho rozhodne o vyhlášení stavu kolize – proměnná *OVER*. Do proměnné *IN* nastaví hodnotu *true*, čímž vyvolá výstup. Hodnota *LIMIT* je pevně daná konstanta uvnitř regulátoru. Tato konstanta se v rámci simulačního běhu modelu nemění, proto není zahrnuta do vnitřního stavu.

$\lambda((\text{SCORE}, \text{OVER}, \text{IN}, \text{OFFSET}, \text{LEVEL}))\{$
  if $(\text{IN} = \mathit{true})$ ***then***
    if $(\text{OVER} = \mathit{true})$ ***then***
      return $\{$ Výstup události kolize $|$ Výstup události kolize$.$hodnota$= \mathit{true}\}$*;*
    else
      return $\{$ Výstup události kolize $|$ Výstup události kolize$.$hodnota$= \mathit{false}\}$*;*
    endif
  else
    return $\{\}$*;*
  endif
$\}$

Výstupní událost překročení síly je odesílána vždy. Informaci o překročení síly přenáší interní hodnotou. Je to proto, že řízení regulátoru v rámci modelu je synchronní a vyžaduje cyklické opakování posloupnosti událostí. Jedná se tedy o specifikum modelu.

$ta((\text{SCORE}, \text{OVER}, \text{IN}, \text{OFFSET}, \text{LEVEL}))\{$
  if $(\text{IN} = \mathit{true})$ ***then***
    return 0*;*
  else
    return ∞*;*
  endif
$\}$

Pokud má být generován výstup, naplánuje interní událost okamžitě. V opačném případě detektor čeká na vstup od řízení regulátoru.

3.8.3 Atomický DEVS Omezovače síly stisku

Omezení síly stisku $= (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$
$X = \{$ Omezení síly, Kalibrační data, Vstup vzorku $\}$
$Y = \{$ Překročení síly $\}$
$S = \{(\text{LIMIT}, \mathbf{history1}, \mathbf{history2}, \mathbf{history3}, \mathbf{history4}, \mathbf{history5}, \mathbf{history6}, \mathbf{w}, \text{CAL\_TAB}, \text{IN}, \text{OVER}) \mid \text{LIMIT} \in \mathbb{R}, \mathbf{history1\text{-}6} \in \mathbb{R}^{13}, \mathbf{w} \in \mathbb{R}^{6}, \text{CAL\_TAB} \in \{\mathbb{R}^{2}\}, \text{IN} \in \{true, false\}, \text{OVER} \in \{true, false\}\}$

Složka stavu *LIMIT* představuje omezení proudu, na kterou má omezovač síly stisku omezit odebíraný proud, aby nepřesáhl maximální sílu stisku. Vektory ***history1-6*** mají 13 položek a představují historii vzorků pro jednotlivé cesty zpracování vstupního signálu. Pro zjednodušení budou nadále zapisovány jako vektor těchto vektorů ***historyX***. Vektor ***w*** je vektorem vah výstupů jednotlivých cest.

$\delta_{\text{int}}((\text{LIMIT}, \mathbf{historyX}, \mathbf{w}, \text{CAL\_TAB}, \text{IN}, \text{OVER}))\{$
  return $(\text{LIMIT}, \mathbf{historyX}, \text{w}, \text{CAL\_TAB}, false, \text{OVER})$*;*
$\}$

Interní přechodová funkce jen změní stav tak, aby odpovídal odeslání výstupu.

$\delta_{ext}((\text{LIMIT}, \mathbf{historyX}, \mathbf{w}, \text{CAL\_TAB}, \text{IN}, \text{OVER}), \text{Omezení síly}, \text{e})\{$
  $\text{TMPL} =$ *Výpočet limitu proudu* $(\text{Omezení síly}.\text{limit}, \text{CAL\_TAB})$*;*
  $\mathbf{TMPW} =$ *Výpočet váhového vektoru* (TMPL)*;*
  return $(\text{TMPSL}, \mathbf{hisotryX}, \mathbf{TMPW}, \text{CAL\_TAB}, false, \text{OVER})$*;*
$\}$

Reakce na nastavení omezení síly pouze přepíše aktuální *LIMIT*. Nevyvolává žádný výstup.

$\delta_{ext}((\text{LIMIT}, \mathbf{historyX}, \mathbf{w}, \text{CAL\_TAB}, \text{IN}, \text{OVER}), \text{Kalibrační data}, \text{e})\{$
  return $(\text{LIMIT}, \mathbf{historyX}, \mathbf{w}, \text{Kalibrační data}.\text{kalibrační tabulka}, false, \text{OVER})$*;*
$\}$

Reakce na nastavení kalibračních dat – načtení nové kalibrační tabulky zaslané událostí na vstupu *Kalibrační data*.

$\delta_{ext}((\text{LIMIT}, \mathbf{historyX}, \mathbf{w}, \text{CAL\_TAB}, \text{IN}, \text{OVER}), \text{Vstup vzorku}, \text{e})\{$
  $\mathbf{TMPH1} =$ *Výpočet nové historie 1* $(\mathbf{history1}, \text{Vstup vzorku}.\text{hodnota})$*;*
  $\mathbf{TMPH2} =$ *Výpočet nové historie 2* $(\mathbf{history2}, \text{Vstup vzorku}.\text{hodnota})$*;*
  $\mathbf{TMPH3} =$ *Výpočet nové historie 3* $(\mathbf{history3}, \text{Vstup vzorku}.\text{hodnota})$*;*
  $\mathbf{TMPH4} =$ *Výpočet nové historie 4* $(\mathbf{history4}, \text{Vstup vzorku}.\text{hodnota})$*;*
  $\mathbf{TMPH5} =$ *Výpočet nové historie 5* $(\mathbf{history5}, \text{Vstup vzorku}.\text{hodnota})$*;*
  $\mathbf{TMPH6} =$ *Výpočet nové historie 6* $(\mathbf{history6}, \text{Vstup vzorku}.\text{hodnota})$*;*
  $\text{TMPEVAL} =$ *Vyhodnocení překročení síly* $(\mathbf{TMPH1}, \mathbf{TMPH2}, \mathbf{TMPH3}, \mathbf{TMPH4}, \mathbf{TMPH5}, \mathbf{TMPH6}, \mathbf{w}, \text{LIMIT})$*;*
  return $(\text{LIMIT}, \mathbf{TMPH1}, \mathbf{TMPH2}, \mathbf{TMPH3}, \mathbf{TMPH4}, \mathbf{TMPH5}, \mathbf{TMPH6}, \mathbf{w}, \text{CAL\_TAB}, true, \text{TMPEVAL})$*;*
$\}$

Reakce na vstup vzorku změní aktuální historie průběhů a vyhodnotí překročení limitu proudu.

$\lambda((\text{LIMIT}, \mathbf{historyX}, \mathbf{w}, \text{CAL\_TAB}, \text{IN}, \text{OVER}))\{$
  if $(\text{IN} = true)$ ***then***
    if $(\text{OVER} = true)$ ***then***
      return $\{$ Překročení síly $|$ Překročení síly.hodnota $= true\}$;
    else
      return $\{$ Překročení síly $|$ Překročení síly.hodnota $= false\}$;
    endif
  else
    return $\{\}$;
  endif
$\}$

$ta((\text{LIMIT}, \mathbf{historyX}, \mathbf{w}, \text{CAL\_TAB}, \text{IN}, \text{OVER}))\{$
  if $(\text{IN} = true)$ ***then***
    return 0;
  else
    return ∞;
  endif
$\}$

Při potřebě generovat výstup vrací *ta* hodnotu 0, jinak plánuje další událost do nekonečna – tedy čeká na vnější událost.

3.8.4 Atomický DEVS Komunikace s chapadlem

Komunikace s chapadlem $= (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$
$X = \{$ Vstup z řízení regulace, Vstup z chapadla $\}$
$Y = \{$ Výstup pro řízení regulace, Výstup příkazu pro chapadlo $\}$
$S = \{(\text{CMD\_RH}, \text{CMD\_HR}, \text{IN}) | \text{CMD\_RH} \in \{($ síla, pozice, rychlost, příkaz $) |$
síla $\in \mathbb{R}$, pozice $\in \mathbb{R}$, rychlost $\in \mathbb{R}$, příkaz $\in \mathbb{N}\}$,
CMD_HR $\in$ Množina příkazů protokolu chapadla, IN $\in \{$ rh, hr, both, none $\}\}$

$\delta_{\text{int}}((\text{CMD\_RH}, \text{CMD\_HR}, \text{IN}))\{$
  return $(\text{CMD\_RH}, \text{CMD\_HR}, \text{none})$;
$\}$

Interní přechodová funkce převádí vnitřní stav z provádění výstupu na klidový stav. Zajišťuje tak právě jedno provedení výstupu pro každou vstupní událost.

$\delta_{\text{ext}}((\text{CMD\_RH}, \text{CMD\_HR}, \text{IN}), \text{Vstup z řízení regulace}, \text{e}) = \{$
  if $(\text{IN} = \text{hr})$ ***then***
    return (převod příkazu r-h (Vstup z řízení regulace), CMD_HR, both);
  else
    return (převod příkazu r-h (Vstup z řízení regulace), CMD_HR, rh);
  endif
$\}$

δ_{ext}((CMD_RH , CMD_HR , IN) , Vstup z řchapadla , e) = {
  if (IN = rh) ***then***
    return (převod příkazu h-r (Vstup z chapadla) , CMD_HR , both) *;*
  else
    return (převod příkazu h-r (Vstup z chapadla) , CMD_HR , hr) *;*
  endif
}

Externí přechodová funkce volá funkce *převod příkazu h-r* a *převod příkazu r-h*. Jsou to abstraktní funkce pro převod tvaru příkazu z příkazu regulátoru na příkaz řídicí jednotky *(r-h)* chapadla a opačně *(h-r)*. Externí přechodová funkce iniciuje odeslání výstupů obklopujícím blokům.

λ((CMD_RH , CMD_HR , IN)) {
  if (IN = both) ***then***
    return { Výstup pro řízení regulace , Výstup pro chapadlo |
        Výstup pro řízení regulace = CMD_HR , Výstup pro chapadlo = CMD_RH } *;*
  elseif (IN = rh) ***then***
    return { Výstup pro řízení regulace | Výstup pro řízení regulace = CMD_HR } *;*
  elseif (IN = hr) ***then***
    return { Výstup pro chapadlo | Výstup pro chapadlo = CMD_RH } *;*
  else
    return { } *;*
  endif
}

ta ((CMD_RH , CMD_HR , IN)) {
  if (IN = none) ***then***
    return ∞ *;*
  else
    return 0 *;*
  endif
}

3.8.5 Atomický DEVS Řízení regulátoru

Řízení regulace = (X , Y , S , δ_{int} , δ_{ext} , λ , *ta*)
X = { Vstup příkazu , Vstup z chapadla , Vstup z detektoru kolize , Vstup z omezovače síly }
Y = { Výstup vzorku , Událost dokončení , Výstup pro komunikaci s chapadlem ,
  Výstup pro detektor kolize , Výstup pro omezovač síly , Událost kalibrace }
S = { (COLLISION_FLAG , STOP , SAMPLE_VAL , CMD , IN , REGULATING , T) |
  COLLISION_FLAG ∈ { *true* , *false* } , STOP ∈ { *true* , *false* } , SAMPLE_VAL ∈ ℝ ,
  CMD ∈ { (síla , pozice , rychlost , odesílat vzorky , odesílat kolizi , regulovat) |
  síla ∈ ℝ , pozice ∈ ℝ , rychlost ∈ ℝ , odesílat vzorky ∈ { *true* , *false* } ,
  odesílat kolizi ∈ { *true* , *false* } , regulovat ∈ { *true* , *false* } } , IN ∈ { none , hand , col_det ,
  force_lim , cmd , clk } , REGULATING ∈ { *true* , *false* } , T ∈ $\mathbb{R}^+ \cup \{0\}$ }

```
δint((COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, IN, true, T)) {
  if (STOP = false) then
    if (IN = cmd) then
      if (CMD.regulovat = true) then
        return (COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, clk, true, T);
      else
        return (COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, none, true, T);
      endif
    elseif (IN = none) then
      return (COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, clk, true, T);
    else
      return (COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, none, true, T);
    endif
  else
    return (COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, none, false, T);
  endif
}
```

Interní přechodová funkce není definována pro jakýkoliv stav z množiny stavů signalizujících neprobíhající regulaci. Pokud se tedy zavolá interní přechodová funkce i tehdy, kdy regulátor nepracuje, je to chyba. Interní přechodová funkce zajišťuje základní smyčku algoritmu regulace – viz algoritmus č.1. Hlavní smyčka algoritmu regulace je zahájena jen tehdy, pokud příkaz aktivující činnost regulátoru požadavek na regulaci obsahoval. Pokud se jednalo o příkaz k pohybu prstů bez regulace, zabrání interní přechodová funkce odeslání prvního dotazu na vzorek chapadlu a tím odstaví regulační smyčku.

```
δext((COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, IN, false, T), Vstup příkazu, e) {
  return (false, false, SAMPLE_VAL, Vstup příkazu, cmd, true, e);
}
```

Příkaz z nadřazeného systému může přijít jen pokud regulace neprobíhá. Pokud přijde příkaz během regulace, není chování regulátoru definované. V případě potřeby ukončení regulace je nutné napřed zastavit prsty pomocí rozhraní *m5api* a potom počkat na *Událost dokončení* odeslanou na výstup regulátoru, kterou řízení regulátoru signalizuje, že regulace skončila.

```
δext((COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, IN, true, T), Vstup z chapadla, e) {
  if (Vstup z chapadla.druh vstupu = vzorek) then
    return (COLLISION_FLAG, STOP, Vstup z chapadla.hodnota vzorku, CMD, hand, true, e);
  else if (Vstup z chapadla.druh vstupu = zatavení) then
    return (COLLISION_FLAG, true, SAMPLE_VAL, CMD, IN, true, e);
  endif
}
```

Reakce na vstup z chapadla závisí na druhu zprávy, jakou chapadlo posílá. Pokud přijde vzorek, což je synchronní událost vyvolaná řízením regulátoru, uloží se hodnota vzorku a vynutí se výstup změnou hodnoty složky stavu *IN* na *hand*. Pokud přijde informace o zastavení pohybu prstů, je vyhlášeno zastavení regulace nastavením *STOP* na hodnotu *true*. Tato externí událost může přijít jen během regulace – proto není pro stavy, v nichž je *REGULATING* rovno *false,* definována.

```
δ_ext((COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, IN, true, T), Vstup z detektoru kolize,
      e) {
  return (Vstup z detektoru kolize.hodnota, STOP, SAMPLE_VAL, CMD, col_det, true, e);
}
```

Reakce na vstup z detektoru kolize.

```
δ_ext((COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, IN, true, T), Vstup z omezovače síly,
      e) {
  return (COLLISION_FLAG, Vstup z omezovače síly.hodnota, SAMPLE_VAL, CMD,
          force_lim, true, e);
}
```

Reakce na vstup z omezovače síly.

```
ta((COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, IN, REGULATING, T)) {
  if (REGULATING = true) then
    if (IN = none) then
      if (CMD.regulovat = false) then
        return ∞;
      else
        return (40 − T);
      endif
    elseif (IN = clk) then
      return 40;
    else
      return 0;
    endif
  else
    return ∞;
  endif
}
```

Pokud je regulátor aktivní – tedy hodnota *REGULATING* je rovna *true*, časuje funkce *ta* interní události tak, aby se periodicky prováděl regulační cyklus. Pokud je funkce volána ve stavu, kdy *IN* má hodnotu *clk*, načasuje další událost za 40ms. Na stavy vyvolané externími událostmi plánuje odezvy interní událostí ihned. Pokud je funkce zavolána ve stavu *IN=none*, znamená to, že byla ukončena aktuální iterace regulace a další událost je naplánována, až doběhne zbytek vzorkovací periody. To ovšem jen v případě, že má regulátor regulovat. Pokud se jen pohybují prsty bez regulace síly stisku, naplánuje další interní událost na čas nekonečno, čímž odstaví regulační smyčku.

Výstup bloku *Řízení regulátoru* je opět pevně spjat s regulační smyčkou. Reaguje na vstupní události a přeposílá data mezi funkčními bloky regulátoru. Speciálním případem je výstup směrem k nadřazenému systému, který je parametrizován tím, jaké informace nadřazený systém požadoval. V modelu je v případě, že nebyla požadována signalizace kolize, vyplněn atribut *kolize* události *Výstup vzorku* vždy na hodnotu *false*. To se v praxi neděje. V praxi je tento atribut vždy vyplněn pravdivě a záleží na uživateli, zda jeho hodnotu využije. Takováto úprava má za následek zjednodušení rozhraní regulátoru – není nutné zadávat, zda uživatel hodnotu požaduje, či nikoliv. Výstupní funkce je opět definována jen ve stavech probíhající regulace. Pokud nastane výstup u regulátoru, který je nečinný, je to chyba.

```
λ((COLLISION_FLAG, STOP, SAMPLE_VAL, CMD, IN, true, T)){
  if (IN=cmd) then
    return {Výstup pro komunikaci s chapadlem, Událost kalibrace |
            Výstup pro komunikaci s chapadlem.pozice=CMD.pozice,
            Výstup pro komunikaci s chapadlem.rychlost=CMD.rychlost,
            Výstup pro komunikaci s chapadlem.příkaz=pohyb};
  elseif (IN=none) then
    return {Výstup pro komunikaci s chapadlem |
            Výstup pro komunikaci s chapadlem.příkaz=pošli vzorek};
  elseif (IN=hand) then
    return {Výstup pro detektor kolize |
            Výstup pro detektor kolize.hodnota vzorku=SAMPLE_VAL};
  elseif (IN=col_det) then
    return {Výstup pro omezovač síly |
            Výstup pro omezovač síly.hodnota vzorku=SAMPLE_VAL};
  elseif (IN=force_lim) then
    if (CMD.odesílat vzorky=true) then
      if (CMD.odesílat kolizi=true) then
        if (STOP=true) then
          return {Výstup vzorku, Událost dokončení |
                  Výstup vzorku.hodnota vzorku=SAMPLE_VAL
                  Výstup vzorku.kolize=COLLISION_FLAG};
        else
          return {Výstup vzorku |
                  Výstup vzorku.hodnota vzorku=SAMPLE_VAL
                  Výstup vzorku.kolize=COLLISION_FLAG};
        endif
      else
        if (STOP=true) then
          return {Výstup vzorku, Událost dokončení |
                  Výstup vzorku.hodnota vzorku=SAMPLE_VAL
                  Výstup vzorku.kolize=false};
        else
          return {Výstup vzorku |
                  Výstup vzorku.hodnota vzorku=SAMPLE_VAL
                  Výstup vzorku.kolize=false};
        endif
      endif
    else
      if (STOP=true) then
        return {Událost dokončení}
      else
        return {}
      endif
    endif
  endif
}
```

3.8.6 Atomický DEVS Kalibrační tabulky

$$\text{Kalibrační tabulka} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$
$$X = \{\text{Vstup události kalibrace}\}$$
$$Y = \{\text{Kalibrace detketoru}, \text{Kalibrace omezovače}\}$$
$$S = \{(\text{IN}) \mid \text{IN} \in \{true, false\}\}$$

$\delta_{\text{int}}((\text{IN}))\{$
  return $(false);$
$\}$

$\delta_{\text{ext}}((\text{IN}), \text{Vstup události kalibrace}, \text{e})\{$
  return $(true);$
$\}$

Externí přechodová funkce iniciuje výstup kalibrace k okolním blokům. Interní přechodová funkce přepne blok zpět do klidového stavu.

$\lambda((\text{IN}))\{$
  if $(\text{IN} = true)$ ***then***
    return $\{\text{Kalibrace detektoru}, \text{Kalibrace omezovače} \mid \text{Kalibrace detektoru.level} = \text{LEVEL},$
      $\text{Kalibrace detektoru.offset} = \text{OFFSET},$
      $\text{Kalibrace omezovače.kalibrační tabulka} = \text{CALIBRATION\_TABLE}\};$
  else
    return $\{\};$
  endif
$\}$

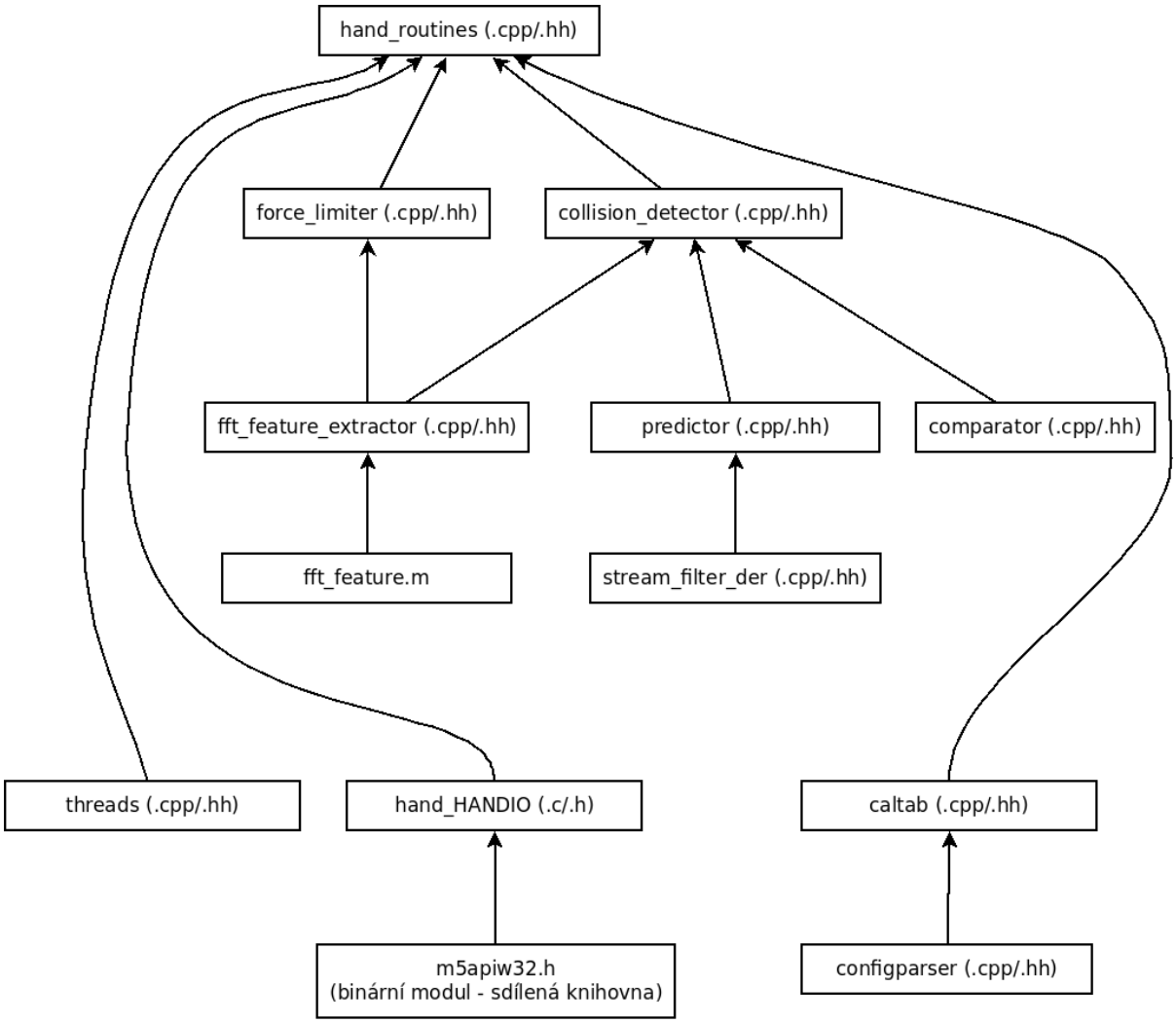
$ta((\text{IN}))\{$
  if $(\text{IN} = false)$ ***then***
    return $\infty;$
  else
    return $0;$
  endif
$\}$

3.9 Struktura souborů

Z návrhu funkčních bloků regulátoru je patrné, že některá funkcionalita se v regulátoru opakuje na více místech. Navíc se do této části návrhu promítají vlastnosti související s implementací regulátoru. Tyto skutečnosti byly při návrhu struktury souborů pro implementaci regulátoru zohledněny. Navržená struktura je popsána na obrázku č.8. Je to hierarchie souborů, kde tyto soubory představují nejmenší logicky souvislé celky funkcionality. Jednotlivé funkční bloky regulátoru tedy nelze jednoduše ztotožnit s konkrétním souborem. Jsou tvořeny typicky nějakým stromem uvnitř této struktury. Směr šipek představuje závislost modulů, kde moduly, ke kterým vedou šipky, závisí na těch, od kterých šipky odchází. Takovéto uspořádání umožňuje jednak snadno sdílet funkcionalitu mezi bloky regulátoru a jednak napomáhá přehlednosti, protože odstiňuje samotné algoritmy od nízkoúrovňového kódu dílčích operací.

Každý uzel struktury popisuje soubor s příponou *.c* nebo *.cpp* a zároveň soubor s rozhraním tohoto souboru s příponou *.h* nebo *.hh*, nebo jen soubor samotný (názvy souborů bez přípon v závor-

kách). Popisovaná struktura zahrnuje jen základní implementaci regulátoru coby knihovnu pro jazyk C++. Spustitelná binárka pro ovládání regulátoru ze skriptů využívající tuto knihovnu přidává k těmto souborům ještě další se vstupní funkcí *main* spustitelného souboru a zpracováním argumentů příkazové řádky. Ty ale ve struktuře nejsou zakresleny, protože nesouvisí s vlastním regulátorem. V rámci popisu jsou pro zjednodušení soubory i jejich rozhraní nazývány pouze názvem souboru bez přípony.



Obrázek 7: Struktura souborů pro implementaci regulátoru.

Vstupním bodem knihovny regulátoru je soubor *hand_routines*. Jeho rozhraní je rozhraním samotného regulátoru pro uživatele knihovny. Tento soubor obsahuje implementaci řídicího algoritmu regulace – viz algoritmus 1. Zároveň se zde také nachází inicializace celého regulátoru, časování událostí a správa vláken. Soubor *hand_routines* přímo závisí na *threads*, *force_limiter*, *collision_detector*, *caltab* a *hand_HANDIO*.

Soubor *threads* je pouze zapouzdřením rozhraní vláken a mutexů na platformách POSIX a WIN32 do jednotného rozhraní. Díky němu je možné odstínit zbylé části implementace od těchto pro platformu specifických záležitostí.

Soubor *force_limiter* je vstupním bodem pro omezovač síly. Obsahuje implementaci většiny interních bloků *Omezovače síly stisku*. V tomto souboru se nachází také instance datové struktury reprezentující interní stav omezovače síly. Některé bloky omezovače jsou však implementovány v dalších souborech. Konkrétně se jedná o blok *Extrakce SS* v souboru *fft_feature_extractor*. Tento

soubor obsahuje implementaci bloku *Extrakce SS* z obrázků 5 a 6. Extrakce stejnosměrné složky je sdílena mezi implementacemi bloků *Omezovače síly stisku* a *Detektoru kolize*. Z implementace byla implementace Fourierovy transformace vyčleněna do samostatného skriptu pro matematický systém Octave. Toto bylo učiněno jednak z důvodu snadné implementace Fourierovy transformace v prostředí Octave a také proto, že experimenty s předchozími verzemi regulátoru ukázaly potřebu způsob předzpracování signálu často měnit. Pro budoucí úpravy regulátoru by možná bylo vhodné vyčlenit i implementaci aproximace přímkou do samostatného souboru.

Podobně jako je *force_limiter* vstupním bodem implementace bloku *Omezovač síly stisku*, je *collision_detector* vstupním bodem bloku *Detektor kolize*. V tomto souboru se nachází inicializace detektoru kolizí a propojení interních bloků detektoru. Závisí na souborech *fft_feauture_extractor*, *predictor* a *comparator*. Soubor *fft_feature_extractor* byl popsán výše. Soubor *predictor* obsahuje implementaci strukturovaného bloku *Prediktor* z obrázku 5. Implementace derivačního filtru je oddělena do samostatného souboru *stream_filter_der*. Implementace derivačního filtru byla vyčleněna především z důvodu zjednodušení modifikace regulátoru. Derivační filtr je tak možné snadno použít při úpravách bloků regulátoru, nebo při vývoji rozšíření regulátoru. Při experimentování s předchozími verzemi regulátoru byl derivační filtr používán ve více modulech. Poslední ze souborů, ze kterých je blok *Detektor kolize* složen, je *comparator*. Tento soubor obsahuje implementaci komparátoru – viz (3).

Zbylé soubory připadají na poslední dva bloky regulátoru – *Kalibrační tabulka* a *Komunikace s chapadlem*. Blok *Komunikace s chapadlem* je implementován v souboru *hand_HANDIO*. Tento soubor je úzce spjat s knihovnou *m5api* a z implementačních důvodů, které se projevily u prototypů, je napsán v jazyce C. Je v něm obsažena inicializace knihovny *m5api* a pomocné funkce zjednodušující práci s knihovnou pro potřeby regulátoru. Tento soubor závisí na knihovně *m5api*, což je znázorněno závislostí *hand_HANDIO* na souboru rozhraní knihovny *m5apiw32.h*. Blok *Kalibrační tabulka* je implementován v souboru *caltab*. Tento soubor obsahuje implementaci algoritmů kalibrační tabulky pro výpočet hodnot limitního proudu a potřebné stavové proměnné. Závisí na souboru *configparser*, který obsahuje implementaci zpracování souborů s daty kalibrační tabulky. Tím je oddělena implementace algoritmů od implementace vstupně-výstupních operací kalibrační tabulky.

4 Implementace

Kapitola Implementace rozebírá detaily implementace navrženého regulátoru. Popisuje technologie použité při implementaci, problémy, se kterými se musela implementace potýkat a konkrétní řešení, která byla pro tyto problémy zvolena. Součástí kapitoly je také bližší popis některých použitých technologií a jejich možností.

4.1 Použité technologie a nástroje

Použité technologie a nástroje zde představují poměrně široký pojem. Tato podkapitola popisuje nejen samotné technologie a nástroje použité při implementaci, ale také softwarové a hardwarové prostředí, pro které byl regulátor implementován.

Navržený regulátor byl implementován v jazyce C++. Z hlediska implementace je jazyk C++ výhodný především proto, že umožňuje objektovou implementaci. Navržené funkční bloky regulátoru lze tedy přímo převést na objekty v implementačním jazyce. Vzhledem k tomu, že regulátor byl implementován opakovaně v rámci jednotlivých iterací vývoje, byla možnost rychlého převodu návrhu do implementačního jazyka jedním z klíčových parametrů při výběru jazyka. Druhým neméně významným parametrem byla snadná přenositelnost kódu v jazyce C++ na různé softwarové i hardwarové platformy. Tento fakt byl zohledněn především do budoucna, kdy není ještě zcela jisté, na jakém operačním systému a na jaké hardwarové architektuře bude regulátor provozován. Ačkoliv bude při portaci na jiné platformy kód vyžadovat nějaké změny, měly by být minimální. Třetím významným argumentem pro použití jazyka C++ byla dostupnost velkého množství knihoven s hotovou implementací mnoha algoritmů a nástrojů. To je v případě vývoje, kdy není předem známá množina použitých algoritmů, značnou výhodou.

Regulátor byl implementován na hardwarové platformě PC s procesorem architektury x86 s operačním systémem GNU/Linux. Při implementaci byla zohledněna i portace regulátoru na operační systém MS Windows. Portaci na jiné hardwarové architektury omezuje knihovna m5api, která je aktuálně k dispozici jen pro architekturu x86. GNU/Linux (konkrétně distribuce Ubuntu 10.04 LTS) byl zvolen proto, že je šířen pod svobodnou licencí a nabízí množství nástrojů pro vývoj a ladění software. Z nich byly používány především make, gcc a valgrind. Dále byly použity nástroje Wine a Octave. Rozhraní těchto nástrojů byla přímo slinkována s výsledným spustitelným souborem. Popis projektů Wine a Octave byl oddělen do samostatných podkapitol.

4.1.1 Wine

Projekt Wine vznikl v roce 1993. Jeho cílem bylo umožnit spouštět aplikace z MS Windows na operačním systému GNU/Linux a tím udělat GNU/Linuxu konkurenceschopný Solarisu s rozhraním Wubi. Od té doby se projekt výrazně rozrostl. Dnes podporuje více operačních systémů, jako GNU/Linux, různé větve BSD, Mac OS X nebo Solaris, a implementuje i některé proprietární nástroje a technologie vyvíjené firmou Microsoft pro platformu Windows, jako je rozhraní DirectX nebo nástroj pro úpravu registrů MS Windows regedit. Vývoj Wine je založen na reverzním inženýrství, proto implementace není v některých směrech dokonalá. Přesto ale umožňuje provozovat na podporovaných operačních systémech značné množství aplikací určených pro MS Windows. V době psaní práce byly v projektu Wine lépe podporovány 32-bitové aplikace. Podpora 64-bitových aplikací byla omezená.

V současné době představuje projekt Wine sice ne zcela kompletní avšak značně pokročilou implementaci API a ABI operačního systému MS Windows nad podporovanými operačními systémy. Ačkoliv projekt Wine nevyžaduje pro svou funkci instalaci MS Windows, nejedná se o emulátor. Wine je v jistém smyslu unikátní projekt virtualizace na aplikační úrovni. Pracuje tak, že nejdříve

zpracuje hlavičku spustitelného souboru, zjistí, na kterých knihovnách soubor závisí a které funkce z těchto knihoven požaduje, zapouzdří spustitelný soubor do samostatného procesu a spustí ho. Spuštěný soubor se chová jako obyčejný proces hostitelského operačního systému. V okamžiku, kdy je z tohoto procesu volána funkce rozhraní MS Windows nebo nějaké sdílené knihovny, je tato funkce přeložena na odpovídající volání hostitelského operačního systému a vykonána. Samozřejmě překlad volání není vždy jedna ku jedné – rozhraní operačních systémů se mohou výrazně lišit. V tomto směru je tedy rozhraní operačního systému MS Windows do jisté míry emulováno, ale spustitelného souboru jsou vykonávány nativně přímo na hardwaru. Díky tomu dosahují aplikace ve Wine výkonu v mnoha směrech alespoň řádově srovnatelného s výkonem běhu těchto aplikací přímo v MS Windows.

O překlad volání operačního systému a knihoven se u Wine stará oddělený proces označovaný jako wine-server. Tento proces se stará o překlad volání na obecně posloupnost volání hostitelského operačního systému, zajišťuje cachování překladu a různé výkonnostní optimalizace překladu. Mimo jiné umožňuje také zavádění nativních dynamických knihoven pro operační systém MS Windows. Právě možnosti načítání nativních dynamických knihoven implementace regulátoru v rámci této práce využívá pro práci s knihovnou m5api. Implementace používá rozhraní MS Windows implementované projektem Wine pro načtení dynamické knihovny a volání funkcí v ní obsažených. Výhodou tohoto řešení je, že je plně kompatibilní s rozhraním MS Windows. V případě překladu projektu na MS Windows tedy dojde pouze k nahrazení hlavičkových souborů z Wine hlavičkovými soubory z MS Windows a k výměně kompilátoru. Jakékoliv zásahy do kódu ale nejsou nutné. Detailní popis projektu Wine lze najít v odkazované literatuře [11].

4.1.2 Octave

Octave je interpret stejnojmenného vysokoúrovňového programovacího jazyka orientovaného na matematické výpočty. Octave je alternativou k nástroji Matlab. Jazyk Octave je do jisté míry kompatibilní jazykem nástroje Matlab, ale v obou jazycích existují vzájemně nekompatibilní konstrukce. Autoři Octave nicméně uvádějí, že portace kódu z Matlabu do Octave by neměla být náročná. Nástroj Octave původně vznikl jako jednoúčelový nástroj pro řešení pomocných výpočtů při návrhu chemických reaktorů.

Jazyk Octave je univerzálním programovacím jazykem. Obsahuje konstrukce podmíněného větvení kódu, cyklů, definice proměnných a volání funkcí. Základní knihovna umožňuje i práci se soubory. V čem se Octave od klasických programovacích jazyků liší je podpora pokročilých operací, podpora pro práci s maticemi a v porovnání s matematickou knihovnou C podstatně širší sada dostupných matematických funkcí. Další funkce lze poměrně snadno přidávat jako moduly na úrovni zdrojového kódu. Původně byl Octave pouze interaktivní interpret, později ale přibyla podpora pro neinteraktivní spouštění skriptů a objevila se také rozhraní pro další programovací jazyky, kterými bylo možné s Octave komunikovat. Jedním z jazyků, pro které rozhraní vzniklo, je i jazyk C. Právě rozhraní pro jazyk C využívá implementace regulátoru. Rozhraní zapouzdřuje volání funkcí Octave do konstrukcí hostitelského jazyka. Prostřednictvím tohoto rozhraní je tedy možné interpret Octave ovládat. Samotný interpret ale běží od zbytku programu odděleně. Rozhraním dostává kód, který potom interpretuje a stejným rozhraním potom vrací výsledky výpočtu. Toto řešení bylo pravděpodobně zvoleno z výkonnostních důvodů. Inicializace interpretu je poměrně náročná a není nutné ji dělat před vykonáním každého vstupu. Navíc má interpret vlastí správu paměti, která také může fungovat efektivněji, pokud není po každém zpracování vstupu restartována. Oddělení interpretu od zbytku programu má tedy své opodstatnění, je ale nutné s ním při volání Octave z jiného jazyka počítat a dopředu spuštění interpretu zajistit.

Interpretace jazyka obecně přináší určité zjevné nevýhody především v oblasti výkonu. Projekty jako Matlab a Octave tento problém částečně řeší překladem interpretovaného kódu do rychlejšího mezikódu, ale i přes tuto optimalizaci je z principu pomalejší, než vykonávání nativního kódu fyzickým procesorem počítače. Výhodou interpretovaných jazyků je ale obecně možnost větší svobody při používání proměnných a to jak z hlediska definice, kdy proměnná může být definována kdykoliv

za běhu, tak z hlediska typové kontroly, kdy proměnné mohou získat datový typ až okamžikem přiřazení. Další značnou výhodou interpretace je možnost zotavení po chybě v interpretovaném kódu. Na rozdíl od nativního kódu, kdy statickou chybu zachytí překladač a dynamickou operační systém nebo v horším případě hardware procesoru, u interpretovaného jazyka zachytí chybu interpret, který na ni typicky reaguje chybovým hlášením, ale nemusí při výskytu chyby nutně skončit a tím ztratit veškeré dosud vypočtené mezivýsledky. Detailní popis nástroje Octave lze najít v literatuře [12].

4.2 Problémy a omezení implementace

První problém, který bylo nutné při implementaci vyřešit, byla nedostupnost zdrojového kódu knihovny *m5api* nebo alespoň binární podoby knihovny pro GNU/Linux. K dispozici byla pouze binární podoba knihovny pro MS Windows na architektuře x86. Toto omezení bylo nutné obejít – navíc pokud možno tak, aby neznemožnilo portaci na jiné platformy. Projekt Wine tedy souvisí s blokem *Komunikace s chapadlem*, kde je využíván jako mezivrstva mezi knihovnou m5api a rozhraním bloku. Wine poskytuje nástroj pro obalení dynamické knihovny pro operační systém MS Windows wrapperem, který umožní využití této knihovny i na jiných operačních systémech, jako je GNU/Linux nebo FreeBSD.

Dalším problémem, který při implementaci nastal, byla chybějící podpora pokročilých matematických operací a algoritmů v základní knihovně C++. Především chyběla Fourierova transformace. Pro Fourierovu transformaci existují knihovny přímo pro jazyk C++, ale regulátor byl vyvíjen iterativně a nebylo od začátku zcela jasné, jakým způsobem budou vstupní data zpracovávána. V tomto směru představovala knihovna implementující pouze Fourierovu transformaci nebo nějakou omezenou sadu algoritmů a funkcí omezení, protože neumožňovala funkce libovolně měnit. Proto byl do implementace začleněn pomocí rozhraní pro C++ nástroj Octave a často se měnící netriviální části regulátoru byly nahrazeny funkcemi Octave. Octave obsahuje přímo pokročilé matematické funkce včetně Fourierovy transformace. Po optimalizacích zůstala v Octave jen implementace filtrace vstupního signálu Fourierovou transformací. Ostatní bloky byly převedeny přímo do jazyka C++. Implementace Fourierovy transformace v Octave sice zřejmě nedosahuje rychlosti srovnatelné s rychlostí nativního kódu, ale je podstatně kratší a jednodušší, proto byla ponechána.

Nedostupnost knihovny *m5api* pro jiné hardwarové platformy a jiné operační systémy představuje značné omezení. Knihovna je k dispozici pouze pro 32-bitovou architekturu x86. To přináší další problém, který bylo nutné v rámci implementace vyřešit. Většina dnešních osobních počítačů má totiž procesor architektury x86_64 – tedy 64-bitové rozšíření architektury x86. Architektura x86_64 je sice s původní x86 zpětně kompatibilní, ale binární rozhraní 64-bitových operačních systémů a knihoven z principu kompatibilní není. Pro běh aplikací je sice typicky k dispozici vrstva kompatibility, avšak ne vždy je takto dostupné i kompatibilní rozhraní knihoven. Pro zajištění funkčnosti implementovaného regulátoru na současných procesorech se tedy nabízela dvě řešení: obalení knihovny m5api 64-bitovým wrapperem tak, aby ji bylo možné používat jako nativní 64-bitovou knihovnu, nebo implementovat celý regulátor jako 32-bitovou aplikaci. Vzhledem k tomu, že knihovna m5api vyžaduje rozhraní MS Windows, a projekt Wine v době psaní práce nenabízel jednoduchý způsob, jak propojit 32-bitovou knihovnu s 64-bitovou aplikací, byla implementace tohoto wrapperu netriviální problém. Proto byla zvolena druhá varianta. Pro vývoj bylo vytvořeno samostatné 32-bitové prostředí uvnitř 64-bitového operačního systému GNU/Linux postavené na distribuci Ubuntu 10.04 LTS 32bit. Toto prostředí běželo v kompatibilním režimu nad 64-bitovým jádrem Linux, ale z pohledu aplikací poskytovalo plně 32-bitové rozhraní z hlediska vývojových nástrojů a knihoven. Prostředí obsahovalo nezávislou instalaci systémových knihoven, vývojových a ladicích nástrojů a nezávislou instalaci projektů Wine a Octave. V tomto prostředí, označovaném také jako bootstrap, bylo možné vyvíjet stejným způsobem, jakým by probíhal vývoj na 32-bitové architektuře. Takovéto řešení je principiálně proveditelné i na jiných operačních systémech podporovaných projektem Wine, jako je Solaris, nebo různé větve BSD, ale neumožňuje portaci regulátoru na hardwarové platformy jiné, než kompatibilní s x86.

5 Experimentování s implementovaným regulátorem

Jak název napovídá, tato kapitola se zabývá experimentováním s implementovaným regulátorem. Cílem experimentování je především ověřit skutečné vlastnosti navrženého regulátoru sérií testů. Dále je předmětem experimentů také vliv kalibrace na vlastnosti regulátoru, vliv mechanických vlastností objektu na přesnost regulace a změna vlastností chapadla během chodu od jeho zapnutí. Zvlášť byl také testován detektor kolize prstů s překážkou. Kapitola obsahuje popis jednotlivých experimentů, grafy změřených hodnot a shrnutí získaných výsledků.

5.1 Popis zapojení pro provádění experimentů

Experimentů bylo prováděno poměrně hodně. Pro jejich zrychlení byla tedy vytvořena automatizovaná experimentální smyčka. Experimentální smyčka se skládala z robotického chapadla, siloměru, terminálu a komunikačních linek. Každá z těchto částí mohla ovlivňovat přesnost experimentů. Chapadlo samotné bylo předmětem zkoumání. Proto bylo třeba vliv ostatních částí co nejvíce potlačit. Chapadlu byl do prstů vložen článek siloměru, který prsty opakovaně mačkaly a siloměrem byla měřena okamžitá síla. Úkolem terminálu bylo zadávat chapadlu příkazy k pohybu prstů a potom číst hodnoty ze siloměru, aby byla zachycena maximální síla stisku.

Pro dosažení co nejlepší opakovatelnosti bylo chapadlo ponecháno ve svislé poloze na rameni, které bylo ponecháno v klidu bez pohybu. Otřesy by mohly způsobit vibrace článku siloměru a tím zanášet do měření síly chybu. Měření tedy probíhalo v tomto směru v ideálním prostředí. V praxi bude pravděpodobně koordinován pohyb chapadla s pohybem ramene. Dále má na přesnost měření vliv siloměr a měřicí článek. Chybu měřicích přístrojů nelze v tomto případě jednoduše eliminovat, ale lze ji omezit na rozumně nízkou volbou přístrojů s malou chybou. Pro měření síly byl použit průmyslový siloměr s tenzometrickým článkem. Vlastnosti siloměru a článku popisují tabulky 7 a 8.

Parametr [jednotka]	**Hodnota**
Počet měření za sekundu	13
Maximální chyba měření (relativní vůči měřené hodnotě) [%]	0,5
Rozlišení [dílky]	10000
Rozsah [N]	2*rozsah článku (kladná i záporná síla)
Typ přístroje	Prominent 999990
Sériové číslo přístroje	030000
Napájecí napětí přístroje [V]	230
Komunikační rozhraní	RS-232

Tabulka 7: Vlastnosti siloměru.

Chyba měření siloměru je 0,5% z rozsahu článku, což je nejméně o jeden řád menší chyba, než chyba regulátoru, která se má dle navrhovaných parametrů pohybovat v řádu desítek procent. Neměla by tedy zásadním způsobem ovlivnit přesnost získaných hodnot.

Parametr [jednotka]	Hodnota
Rozsah síly [N]	500
Sériové číslo	050004
Typ článku	000090

Tabulka 8: Vlastnosti měřicího článku.

Vliv terminálu a komunikačních linek není tak zásadní, jako vliv siloměru a chapadla – respektive okolního prostředí, přesto je ale vhodné ho zmínit. Komunikační linky jsou digitální a nedochází u nich tedy ke zkreslení hodnoty, přesto mohou měření negativně ovlivnit díky zpoždění komunikace. Komunikační rozhraní chapadla i siloměru jsou RS-232. Obě zařízení komunikují rychlostí 9600 baudů za sekundu bez parity. Zpoždění samotné linky lze tedy vypočíst z délky zpráv komunikačních protokolů. Tento údaj je ale minimálním možným zpožděním komunikace. Mnohem zásadnější zpoždění může vznikat na straně terminálu. Jedním z důvodů tohoto zpoždění je buffer operačního systému. Zachycené byty zprávy se do něj ukládají během komunikace a aplikace v uživatelském prostoru tento buffer cyklicky čte. Aplikace se tedy k datům zprávy dostane v nejbližší iteraci po jejich doručení. Z výkonnostních důvodů však čas mezi iteracemi nemůže být nulový, protože by aplikace příliš zatěžovala terminál. V nejhorším případě se tedy zpoždění komunikace může prodloužit o dobu mezi iteracemi čtení a režii nutnou k přenosu dat z jádra OS k aplikaci. Toto zpoždění ještě zhoršuje fakt, že rozhraní RS-232 na terminálu byla realizována převodníky ze sběrnice USB. Je tedy nutné uvažovat ještě zpoždění způsobené bufferem převodníku, zpožděním na sběrnici USB, které vzhledem k paketové povaze sběrnice není garantované a zpoždění způsobené buffery implementace USB v operačním systému. Zpoždění tedy z povahy věci není spolehlivě shora omezené, ale stále by mělo být výrazně nižší, než nejkratší perioda vzorkování používaná při měření, a to 40ms u chapadla. Zkreslení způsobené zpožděním nesouvisí s hodnotou měřené veličiny, ale může způsobit posuv na časové ose mezi měřenými průběhy. Jedná se tedy o principiálně jinou chybu, než kterou způsobují siloměr a okolní podmínky, za kterých měření probíhá.

Terminál má pak na přesnost měření vliv v podobě zpoždění způsobeného výpočtem a ukládáním hodnot. Vzhledem k periodě čtení komunikačního bufferu je toto zpoždění poměrně málo významné. Je jen nutné vyhnout se v případě terminálu situacím, kdy by vznikl nedostatek prostředků a docházelo by k časově velmi náročnému odkládání paměti na disk. Toto bylo při měření zajištěno tím, že procesy, které by mohly způsobit takovýto stav, byly po dobu měření zastaveny.

5.2 Zkoumání změny vlastností chapadla během provozu od zapnutí

Úkolem tohoto experimentu bylo zjistit, jestli a případně jakým způsobem se mění vlastnosti chapadla během provozu těsně po jeho zapnutí. U mechanických systémů obecně bývá obvyklé, že ihned po zapnutí nemají stanovené parametry, ale že těchto parametrů dosáhnou až po určitém čase provozu. Dost často ustálení parametrů souvisí s teplotou maziva a tepelnou roztažností mechanických částí.

Princip tohoto experimentu spočívá v opakovaném pohybování prsty chapadla na prázdno ihned po zapnutí chapadla po tom, co delší dobu stálo. Během pohybu prstů byly zaznamenávány průběhy okamžitého odběru proudu. Nad těmito průběhy byly později vyhodnocovány statistiky, které by mohly odhalit případný trend změny vlastností během „zahřívací fáze“ činnosti chapadla. Konkrétně se jedná o rozsah vzorkovaných hodnot a stejnoměrnou složku průběhu. Dále se experiment zaměřoval na stanovení okamžiku, kdy se vlastnosti chapadla ustálí. Tento okamžik je významný pro regulátor síly stisku, protože až po jeho dosažení bude regulátor vykazovat definované

vlastnosti. Do té doby může mít regulace vlastnosti horší. Experimenty probíhaly při rychlosti odpovídající rychlosti prstů při regulaci.

Změřené výsledky byly znázorněny v grafech 7 a 8. Graf 7 zobrazuje změnu rozsahu hodnot a grafu 8 změnu stejnosměrné složky průběhů. Každý vzorek v grafu představuje statistiku nad průběhem okamžitého proudu při jednom stisku prstů chapadla.

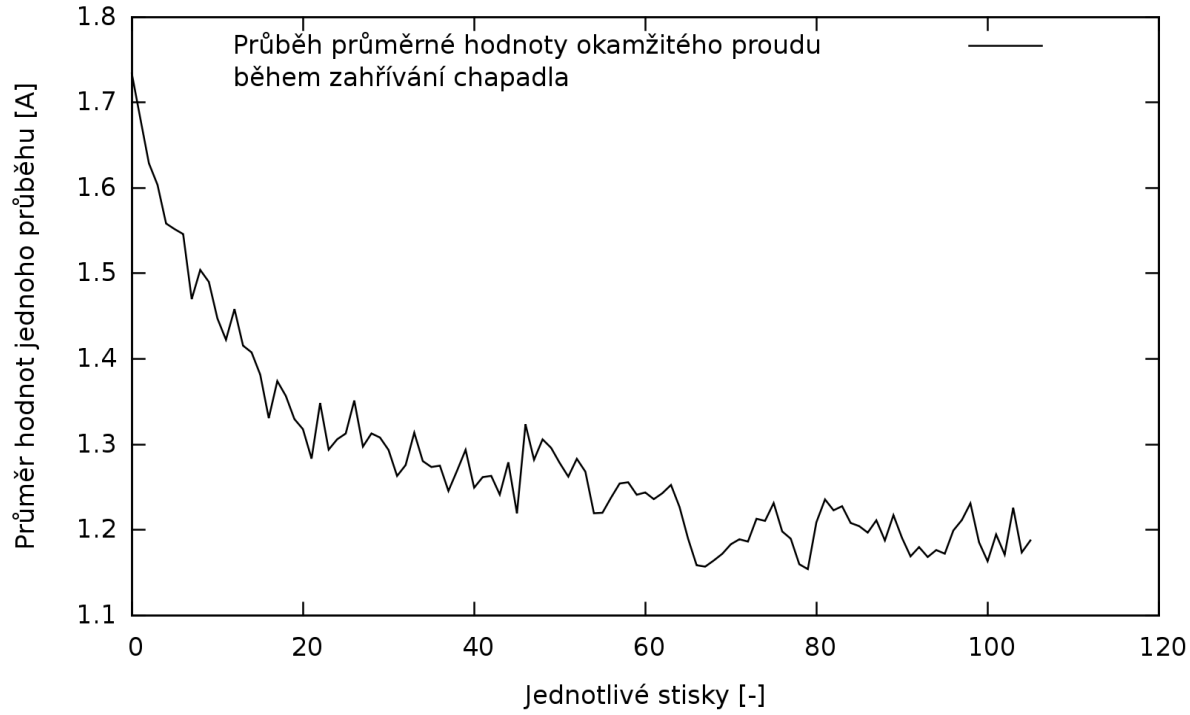
Průběh v grafu 7 je silně zvlněný. Zvlnění je způsobeno samotnou podstatou zobrazované informace. Rozdíl mezi minimální a maximální hodnotou průběhu je principiálně náhodný v nějakém rozsahu. Silné zašumění jednotlivých průběhů okamžitého proudu (viz graf 1) tuto skutečnost ještě zvýrazňuje. Zvlnění grafu významnou měrou znesnadňuje hledání trendu rozsahu hodnot v průběhu zahřívání chapadla. I přes zvlnění grafu lze říci, že žádná výrazná změna v rozsahu hodnot jednotlivých průběhů v průběhu zahřívání chapadla není patrná. Od vzorku 60 dále se v grafu nevyskytují hodnoty tak velké, jako v části grafu do vzorku 60, ale hodnoty kolísají kolem přibližně stále stejné hodnoty. Z grafu 7 lze tedy vyvodit závěr, že se rozsah hodnot okamžitého proudu v průběhu zahřívání chapadla výrazně nemění.

Graf 7: Změna rozsahu hodnot okamžitého proudu během zahřívání chapadla.

O poznání zajímavější je průběh v grafu 8, který zobrazuje změnu průměrné hodnoty proudu během zahřívání chapadla. Průměrná hodnota průběhu představuje v podstatě stejnosměrný posuv průběhu. Vypovídá tedy o tom, jaký je odběr chapadla při pohybu prstů konstantní rychlostí. Zde je na první pohled patrný klesající trend. To tedy znamená, že ze začátku je odběr chapadla při pohybu prstů na prázdno výrazně větší, než po zahřátí chapadla – v tomto případě o přibližně 0,5A, což odpovídá asi 40% z výsledné ustálené hodnoty. Stejnosměrný posuv má zásadní vliv na citlivost detekce kolize, kdy zvýšený proudový odběr způsobuje větší stejnosměrný posuv reálného signálu a díky tomu vyhodnocuje detektor kolize falešné kolize. Zvýšený odběr proudu také posouvá kalibrační tabulku pro omezovač síly stisku. Díky tomu bude častěji vyhodnoceno překročení limitní síly dříve, než k němu skutečně dojde. Regulátor tedy bude při regulaci s nedostatečně zahřátým chapadlem vykazovat četné falešné poplachy a často tak regulace nedosáhne požadované síly stisku. Pro malé síly se dokonce může stát, že bude vyhodnoceno překročení limitní síly i při běhu prstů na prázdno. Předčasně bude také nadřazenému systému zasílána událost signalizující kolizi prstů s překážkou. Popsané

problémy mohou tedy značně omezit funkčnost regulátoru. Pozitivem je, že tyto problémy nemohou z principu vést k překročení požadované síly stisku. Regulace bez dostatečného zahřátí mechanické části chapadla bude tedy zřejmě značně nespolehlivá, ale neměla by ohrozit uchopovaný předmět ani chapadlo samotné.

Z grafu 8 lze odhadnout okamžik, kdy se odběr chapadla stabilizuje. Tento okamžik nastává přibližně po 70 sevřeních a rozevřeních prstů. Od tohoto okamžiku se mění průměrný proud jen minimálně. Při rychlosti pohybu prstů 1mm/s, což je běžná rychlost prstů při regulaci, trvá sevření a rozevření chapadla přibližně půl minuty. Vykonání 70 sevření a rozevření prstů potřebných pro spolehlivou stabilizaci parametrů chapadla by tedy trvalo zhruba půl hodiny. To je pro praktické použití regulátoru nepříjemná prodleva. Proto byl minimální počet zahřívacích cyklů pro správnou činnosti regulátoru stanoven na 30. V oblasti od 30 sevření a rozevření prstů se už stejnosměrný posuv mění jen minimálně a výrazně se tím sníží čas potřebný k zahřátí chapadla na přijatelných 10 až 15 minut. Experimentálně bylo ověřeno, že změna stejnosměrného posuvu proudu mezi 30 a 70 zahřívacími cykly nemá zásadní vliv na omezení síly stisku regulátorem. Lze tedy říci, že po 30 cyklech rozevření a sevření prstů dosahuje regulátor udávaných vlastností.



Graf 8: Změna průměrné hodnoty průběhu proudu během zahřívání chapadla.

5.3 Přesnost regulace

Zkoumání přesnosti regulace je hlavní náplní experimentování s regulátorem. Cílem experimentů je stanovit průměrnou chybu, jaké se regulátor při regulaci dopouští vůči zadané hodnotě mezní síly stisku. Významným parametrem je také rozptyl změřených hodnot skutečné síly, který vypovídá o opakovatelnosti dosažených výsledků. Snahou při návrhu regulátoru přirozeně bylo oba tyto negativní parametry minimalizovat.

Experimentování probíhalo tak, že byl prsty chapadla opakovaně mačkán měřicí článek siloměru, během stisku byly zaznamenávány hodnoty okamžité síly až do zastavení prstů chapadla, kdy regulátor odeslal událost dokončení. Hodnoty byly průběžně ukládány a po zastavení prstů byla mezi

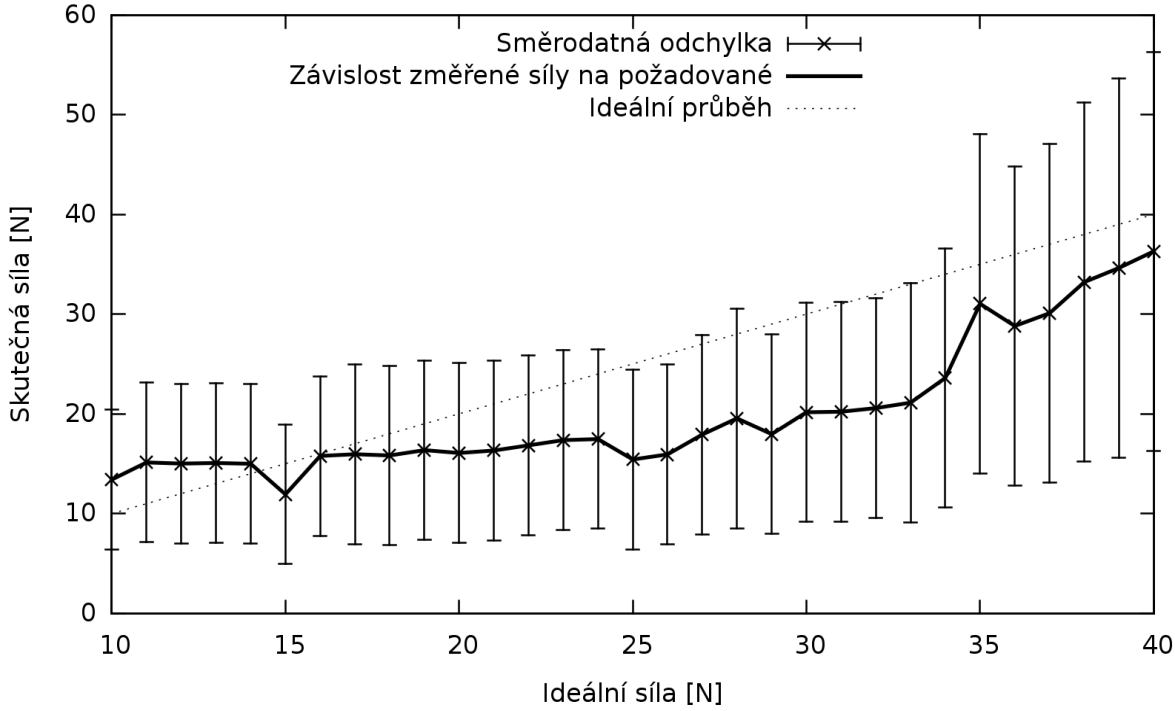
uloženými hodnotami aktuálního průběhu nalezena maximální hodnota. Tento způsob měření byl desetkrát zopakován a z výsledných maximálních hodnot potom byla stanovena střední hodnota a rozptyl. Celý postup byl zopakován pro každou zkoumanou hodnotu síly. Takto byl proměřen celý rozsah regulátoru od 10N po 40N. Síla byla navyšována s krokem 1N. Celkem tedy proběhlo 300 měření pro ověření přesnosti regulátoru v celém definovaném rozsahu regulace.

Klíčový význam pro přesnost regulace má správná kalibrace regulátoru. Kalibrace je uložena v datovém souboru kalibrační tabulky. Jak je možné vidět z grafu 4, kalibrační data jsou velmi zašuměná. Určit z nich správnou hodnotu proudového omezení pro regulátor je poměrně obtížné. Algoritmus popsaný vztahem (4) se sice snaží nějakým způsobem řešit šum v kalibračních datech, ale lze očekávat, že s kvalitnějšími vstupními daty bude dosahovat lepších výsledků. Na kvalitu získaných dat má zásadní vliv přístup, jakým jsou kalibrační data získávána. Proto bylo měření přesnosti regulátoru rozděleno do tří kapitol podle způsobu, jakým byla kalibrační data získána a případně předzpracována. Pro zjednodušení je v následujícím popisu pro kalibrační tabulku s daty určitého typu použito označení „kalibrační tabulka daného typu“. Kalibrační tabulkou tedy není pro tuto podkapitolu myšlena softwarová komponenta regulátoru, ale samotný datový soubor, se kterým tato komponenta pracuje. V tomto smyslu byly vyčleněny tři kalibrační tabulky: klasická kalibrační tabulka, filtrovaná klasická kalibrační tabulka a alternativní kalibrační tabulka.

5.3.1 Klasická kalibrační tabulka

Klasická kalibrační tabulka byla získána poměrně přímočarou cestou. Postupně po malých krocích byla měněna hodnota limitního proudu při stisku článku siloměru. Pro každou hodnotu proudu bylo provedeno opět 10 měření. V každém změřeném průběhu byla nalezena maximální hodnota. Z takto získaných maxim byla spočítána střední hodnota síly, která byla dosazena do řádku tabulky s odpovídajícím proudovým omezením. Tímto způsobem byla změřena odpovídající síla pro hodnoty od 3,5A do 7,5A s krokem 25mA. Hodnoty zaznamenané tímto způsobem jsou vizualizovány v grafu č.5. Z grafu je vidět, že hodnoty jsou silně zašuměné. I přesto, že je v kalibrační tabulce poměrně zřetelný stoupající téměř lineární růst limitní síly s limitním proudem, vykazuje regulace systematicky rostoucí chybu ve směru stoupající limitní síly. Průběh skutečné limitní síly je zobrazen v grafu č.9.

Z grafu je patrný rostoucí rozdíl křivky skutečné a ideální síly. V oblasti od 10N do 33N je dokonce křivka skutečné síly téměř vodorovná, ačkoliv by měla stoupat lineárně. Zároveň je většina hodnot skutečné síly pod ideální křivkou. To, že jsou hodnoty pod křivkou, nepředstavuje zásadní problém. Posunutí hodnot lze vyřešit přičtením stejnosměrného záporného posuvu k hodnotám síly v kalibrační tabulce. Výrazně větší problém představuje trend křivky skutečné síly, který je rostoucí jen minimálně a pro hodnoty zadané síly okolo 30N se odpovídající síle vzdaluje o více než 10N což je 30% ze zadané hodnoty. Teprve k této posunuté střední hodnotě se ještě přičítá chyba.
Vzhledem k principu regulátoru nelze jednoduše určit maximální možnou chybu regulace. Přesněji – tato maximální chyba regulace je dána hardwarovým limitem proudu pro pohon chapadla. Tento limit ale v případě hodnoty zadané síly u spodní hranice rozsahu tvoří stovky procent ze zadané hodnoty. Maximální chybu softwarové regulace ale nelze jednoduše určit. Proto byla do grafu namísto minimální a maximální hodnoty vynesena směrodatná odchylka změřených hodnot. Pro normální rozložení hodnot by měla vzdálenost jedné směrodatné odchylky od střední hodnoty pokrývat necelých 70% případů. Tedy ve většině případů se chyba regulace vejde do směrodatné odchylky, ale nejsou vyloučeny případy (cca 30%), kdy tato chyba bude překročena. Uvážíme-li pouze hodnoty spadající do 70% úspěšnějších případů, pohybuje se největší chyba regulace okolo 73% ze zadané hodnoty směrem dolů a 100% ze zadané hodnoty směrem nahoru. Je nutné podotknout, že tyto chyby se vyskytují u opačných konců rozsahu – na začátku rozsahu se projevuje především chyba přesahující zadanou sílu a na konci rozsahu spíše chyba pod zadanou sílu. Výsledky dosažené s klasickou kalibrační tabulkou byly vzaty jako referenční. Cílem dalších experimentů bylo dosáhnout výsledků lepších.



Graf 9: Závislost skutečné limitní síly na zadané - klasická kalibrační tabulka.

5.3.2 Filtrovaná klasická kalibrační tabulka

V grafu č.5 je v kalibračních datech patrný intenzivní šum. První modifikace kalibrační tabulky tedy spočívala v odstranění tohoto šumu. Filtrace odstraňující vliv šumu úplně příliš deformovala tvar kalibrační křivky, proto bylo třeba najít rovnováhu mezi amplitudou šumu a deformací křivky.

Filtrovaná klasická kalibrační tabulka tedy vychází z původních dat, ale snaží se je vhodným způsobem předzpracovat a tím zlepšit přesnost regulace. Pro odstranění šumu byl zvolen integrační filtr. Chování filtru popisuje vzorec (10).

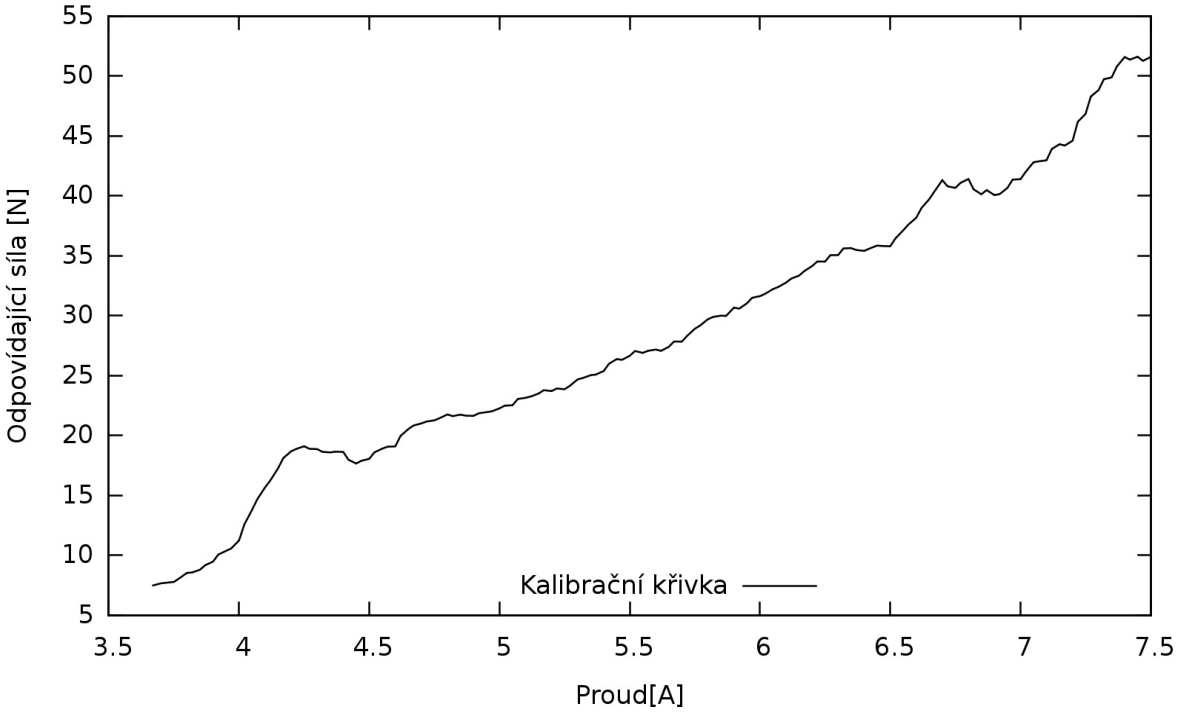
$$\textbf{\textit{for}}\ i = k\,..\,N\ \textbf{\textit{do}}: \frac{1}{N} \cdot \sum_{j=i-k}^{i} s_j \tag{10}$$

Kde:

k	... délka okna filtru.
N	... počet vzorků filtrovaného průběhu.
s_j	... konkrétní vzorek průběhu na indexu j.
i	... index výstupního vzorku.
j	... index zpracovávaného vzorku.

Filtr v podstatě pouze průměruje hodnoty v rámci úseku hodnot na filtrovaném průběhu. Úsek je volen plovoucím oknem pevné velikosti, které se s každým vypočteném filtrovaným vzorkem posouvá o jeden vzorek dál nad vstupním průběhem. Filtr tedy s každým krokem počítá hodnotu odpovídající nejpravějšímu vzorku okna. Takto filtr postupně prochází celý průběh od začátku až do konce.

Nevýhodou integračního filtru je nemožnost filtrovat průběh od prvního vzorku, protože pro prvních *k* vzorků, kde *k* je délka okna filtru, není dostatek předchozích vzorků pro výpočet filtrované hodnoty. V tomto případě to však nepředstavuje zásadní problém, protože kalibrační tabulka začíná na hodnotách, které by měly odpovídat silám pod spodní hranicí rozsahu regulátoru. Kalibrační tabulka po filtraci je zobrazena v grafu 10.



Graf 10: Kalibrační křivka - klasická kalibrační tabulka po filtraci integračním filtrem.

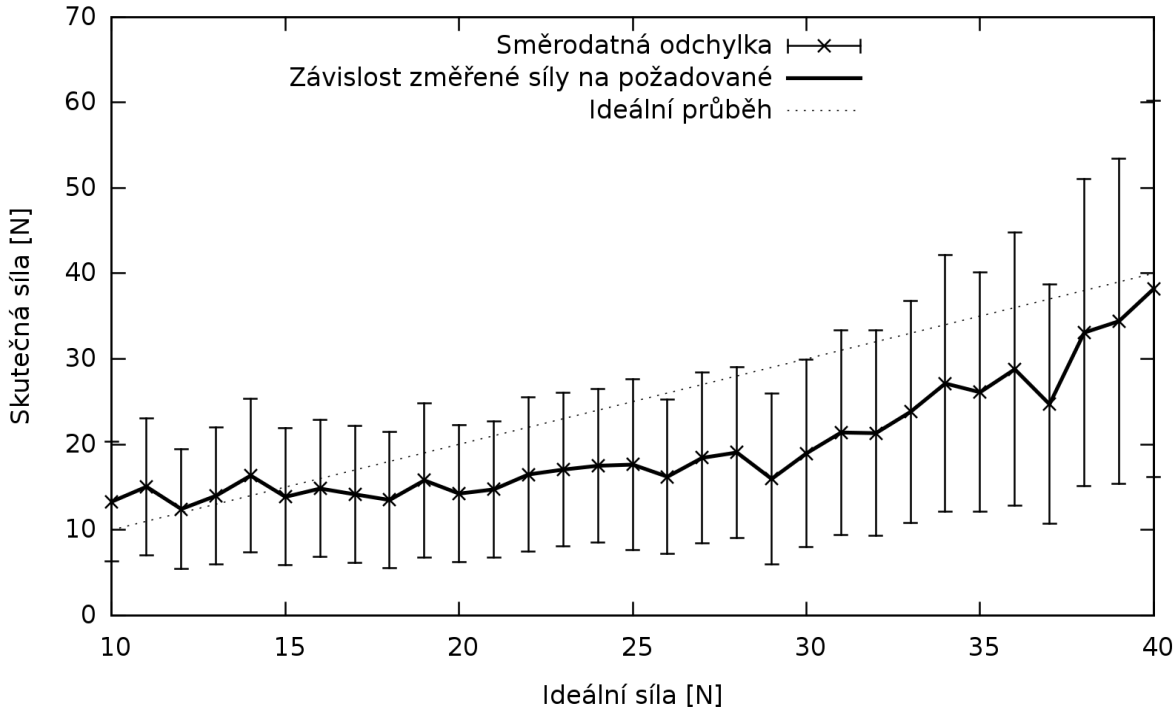
V grafu křivky kalibračních dat po filtraci je vidět, že potlačení šumu je poměrně účinné. Kalibrační křivka se teď svým tvarem mnohem více podobá přímce. To prakticky znamená, že je spolehlivější odečet hodnoty proudu pro konkrétní zadanou sílu, protože při inverzi funkce limitního proudu znázorněné křivkou je méně případů, kdy je hodnota limitního proudu pro zadanou sílu nejednoznačná a musí se algoritmicky určovat. Díky filtraci také lépe vynikl skutečný tvar kalibrační křivky. Vydutý tvar odchylující se od ideální přímky mezi hodnotou proudu 4A – 4,5A je nyní lépe patrný.

Průběh skutečné limitní síly s filtrovanou kalibrační tabulkou je zobrazen v grafu č.11. Hodnoty skutečné limitní síly se ani po úpravě kalibračních dat výrazně nepřiblížily ideálním hodnotám. Stále se skutečná křivka pohybuje poměrně hluboko pod tou ideální. Křivka má ale tvar lépe odpovídající ideální přímce. Je na ní lépe patrný nárůst skutečné síly ve směru zadávané síly. Méně patrná jsou také některá problematická místa. Především zlom v oblasti okolo 33N zadané síly už není téměř patrný. Křivka má teď okolo této oblasti spíše oblý tvar. V oblasti mezi 10N a 15N křivka lépe kopíruje ideální průběh. I přes zvlnění křivky je vidět rostoucí trend následující směr růstu zadané síly. Oproti křivce reálné síly s klasickou kalibrační tabulkou, kde se růst zadané limitní síly na skutečné síle prakticky neprojevoval, je to významné zlepšení.

Filtrace kalibračních dat neměla jak je z grafu vidět zásadní vliv ani na rozptyl skutečných hodnot. Rozptyl je zcela srovnatelný s původní kalibrační tabulkou. Toto ale není nikterak překvapivé zjištění. Rozptyl hodnot skutečné síly totiž principiálně nezávisí na kalibrační tabulce. Komponenta *Kalibrační tabulka* počítá limitní proud deterministickým algoritmem. I silně zašuměná kalibrační data vrací vždy stejnou hodnotu. Chyba hodnoty limitního proudu vypočteného *Kalibrační tabulkou* může tedy způsobit pouze posunutí hodnoty skutečné síly, ne však její kmitání. Z matematického hle-

diska má tedy chyba při výpočtu limitního proudu vliv pouze na střední hodnotu skutečné limitní síly, ne však na její rozptyl.

Shrneme-li tedy výsledky s filtrovanou kalibrační tabulkou, lze říci, že byly celkově lepší, než u klasické kalibrační tabulky. Především má křivka středních hodnot tvar bližší ideálnímu průběhu. Celkově je křivka sice stále posunutá pod ideální úroveň, to ale lze řešit korekcí hodnot odpovídající limitní síly v kalibrační tabulce.



Graf 11: Závislost skutečné limitní síly na zadané - filtrovaná klasická kalibrační tabulka.

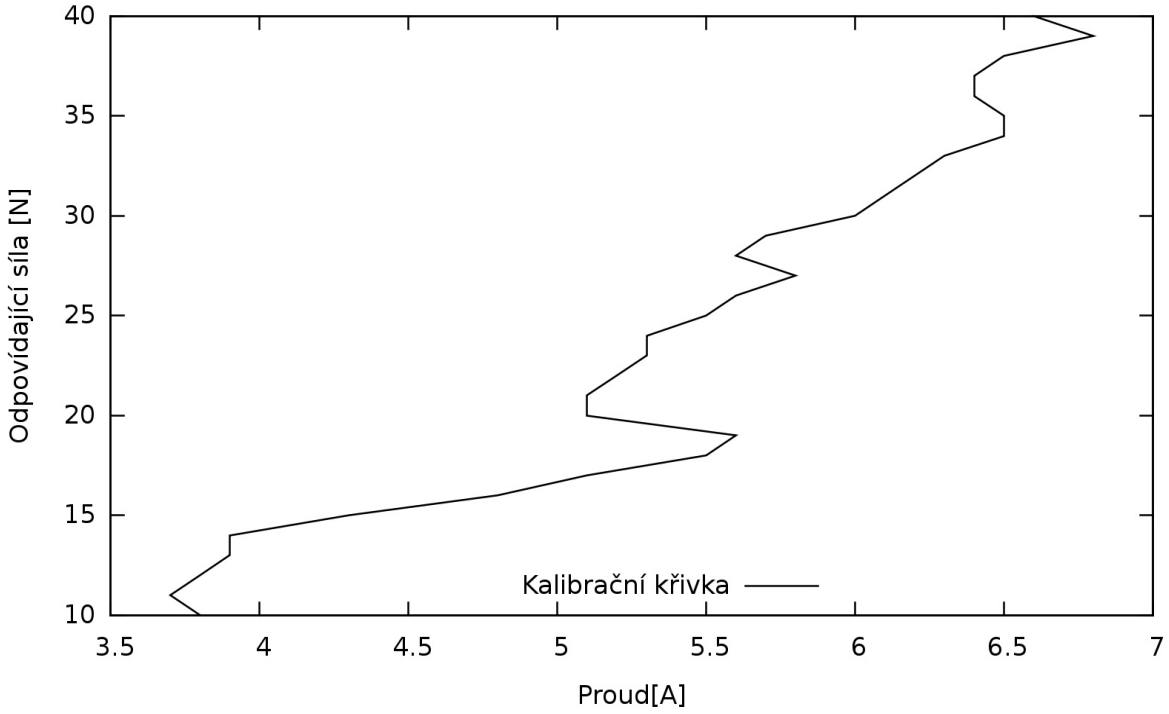
5.3.3 Alternativní kalibrační tabulka

Další možností, jak zlepšit kvalitu kalibračních dat, bylo použití jiného přístupu ke generování kalibrační tabulky. Současné metody generovaly kalibrační tabulku v podstatě slepou metodou. Generovaly hodnoty limitního proudu pro dopředu neznámé hodnoty síly. Mezi těmito hodnotami pak algoritmus *Kalibrační tabulky* odhadoval hodnoty požadované síly.

Experimenty s klasickou kalibrační tabulkou ale ukazují, že nemá smysl zadávat limitní sílu v jednotkách menších než jednotky N. Pro další metodu byl tedy zvolen nejmenší krok, o který je možné měnit limitní sílu regulátoru na 1N. I tak chyba regulace tento krok změny síly dalece převyšuje. Proto byl navržena alternativní metoda generování kalibrační tabulky v podstatě opačná vzhledem ke stávající. Zatímco stávající metoda měnila po krocích limitní proud a k tomuto proudu určovala odpovídající sílu, alternativní metoda postupuje po krocích síly a snaží se určit limitní proud pro konkrétní síly. Alternativní metoda generování kalibrační tabulky tedy funguje tak, že pro konkrétní hodnotu síly iterativně upravuje po stanoveném kroku hodnotu limitního proudu. Hodnotu zvyšuje nebo snižuje podle toho, zda předchozí měření přesáhlo požadovanou hodnotu či nikoliv. Hodnota limitní síly je maximum z průběhu hodnot síly změřených při každém sevření měřicího článku prsty chapadla. Vzhledem k tomu, že maximální hodnoty mají značný rozptyl, nelze čekat, až se skutečná maximální hodnota dostatečně přiblíží požadované. Proto je přizpůsobování ukončeno po pevně stanoveném počtu iterací – v tomto konkrétním případě bylo iterací dvacet. Tímto způsobem se

střední hodnota maxim skutečné síly přiblíží požadované hodnotě. Hodnota proudu po poslední iteraci je potom výslednou hodnotou limitního proudu odpovídající požadované síle. Takto je potom procházen postupně po krocích 1N celý rozsah požadované síly regulátoru od spodní hranice k horní. Průběh kalibrační křivky získané tímto způsobem je znázorněn v grafu č.12.

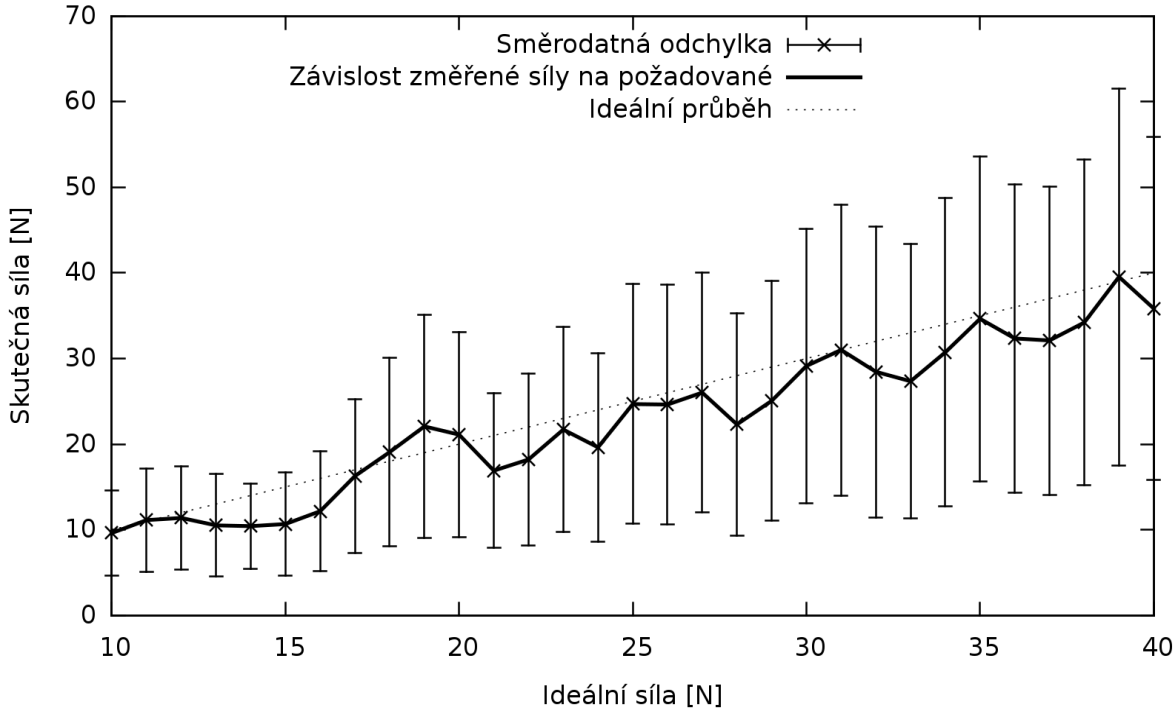
Zvyšování hodnoty limitní síly po krocích a hledání odpovídajícího limitního proudu má oproti klasickému přístupu tu výhodu, že v oblastech, kde malá změna limitního proudu vyvolá velkou změnu síly, je hodnot na proudové ose kalibrační křivky více, než u klasické metody. Naopak v oblastech, kde i výrazné zvýšení limitního proudu způsobuje jen minimální změny síly je hodnot méně. Klasická metoda měla na proudové ose hodnoty vždy stejně husté.



Graf 12: Kalibrační křivka - alternativní kalibrační tabulka.

Ačkoliv v tomto případě je závislost získaných dat opačná oproti klasické kalibrační tabulce (u alternativní kalibrační tabulky závisí limitní proud na požadované síle), je graf pro lepší srovnání ve stejném tvaru, jako u klasické kalibrační tabulky. Závislá osa je v tomto případě ale osa x. Z grafu je vidět, že kalibrační křivka je opět zvlněná. Oproti klasickému přístupu má ale průběh zvlnění nižší frekvenci a větší amplitudu. Pro síly od přibližně 20N výš připomíná kalibrační křivka přímku, ale pro nižší hodnoty síly je zvlnění silné. Mezi 4A a 15N až po 5,5A a cca 20N je značný rozdíl na proudové ose (přibližně 1,5A), kterému odpovídá jen malá změna síly (asi 5N). V případě klasického přístupu má kalibrační křivka v této oblasti tvar spíše opačný – viz graf č.10. To je pravděpodobně způsobeno značným rozptylem hodnot skutečné síly, kdy 10 vzorků nestačí klasické metodě pro správné určení síly odpovídající konkrétnímu proudu v této oblasti. Toto tvrzení podporuje i fakt, že s klasickou kalibrační tabulkou je v průběhu skutečné limitní síly v této oblasti téměř konstantní úsek, kdy regulace reaguje na změnu požadované síly jen minimálně. Hodnoty změřené klasickou kalibrační metodou v této oblasti tedy pravděpodobně nejsou zcela správné. Klasická metoda by pravděpodobně potřebovala v této oblasti více vzorků pro přesnější určení hodnoty. Alternativní metoda je v této oblasti úspěšnější. Je to způsobeno jednak jiným přístupem k určení dvojice limitní síla – limitní proud, ale také tím, že alternativní metoda pro každou hodnotu síly iteruje dvaceti měřeními, zatímco klasická metoda průměruje pouze výsledky deseti měření. Menším počtem vzorků je klasická metoda značně znevýhodněna, ale vzhledem k počtu hladin limitního proudu, pro které klasická metoda vy-

hodnocuje limitní sílu, přináší zvýšení počtu měření pro každou hladinu značné komplikace. Zatímco klasická metoda potřebuje v rozsahu od 10N do 40N s krokem 1N při 20 iteracích pro každý krok 620 měření, což je poměrně hodně, klasická metoda potřebuje pro rozsah limitního proudu od 3,5A do 7,5A s krokem 0,05A při 10 měřeních pro každou hladinu 800 měření. Navýšení počtu měření pro každou hladinu na 20 a více by zvýšilo celkový počet měření na minimálně 1600. Každé měření znamená sevření a rozevření prstů chapadla. Experiment s takovým počtem měření znamená poměrně velkou zátěž pro chapadlo, ale především je časově velmi náročný. Proto nebyl realizován. Výsledky dosažené s alternativní kalibrační tabulkou jsou znázorněny v grafu č.13.



Graf 13: Závislost změřené síly na požadované – použita alternativní kalibrační tabulka.

Z grafu je vidět, že výsledky dosažené s alternativní kalibrační tabulkou se od těch s klasickou kalibrační tabulkou výrazně liší. Především je zde rozdíl v tom, že křivka skutečné síly se oproti průběhům s klasickou kalibrační tabulkou velmi blíží ideální přímce naznačené v grafu a to jak tvarem, tak hodnotou. Oproti klasické metodě je ale více zvlněná. Především se zvlnění projevuje v oblasti mezi 15N a 20N. Toto zvlnění svou polohou odpovídá zvlnění kalibrační křivky. Tedy ani alternativní metoda kalibrace nemá v této oblasti ideální výsledky. Je to pravděpodobně způsobeno tím, že pro nižší hodnoty je průběh okamžitého proudu silně zvlněný. Z hlediska překročení limitní síly stisku je tato oblast jako jediná problematická, protože v ní hodnoty skutečné síly přesahují požadované hodnoty. Ačkoliv *Omezovač síly stisku* vliv šumu kompenzuje váženým průměrem výstupů více cest zpracování s různými vlastnostmi, nedokáže tato kompenzace, jak je vidět z výsledků experimentů, vliv šumu zcela potlačit. Možná řešení tohoto problému by mohla být buďto navrhnout dokonalejší omezovač síly, který bude lépe kompenzovat vliv šumu, nebo dodatečně upravit kalibrační tabulku. Vzhledem k rozptylu hodnot skutečné síly je ale odchylka v problematickém úseku rozsahu poměrně malá. Navrhované úpravy proto nebyly realizovány. Dle očekávání je rozptyl hodnot i v tomto případě srovnatelný s ostatními s předchozími. To potvrzuje domněnku, že kalibrační tabulka nemá teoreticky a jak se ukázalo ani prakticky na rozptyl pozorovatelný vliv.

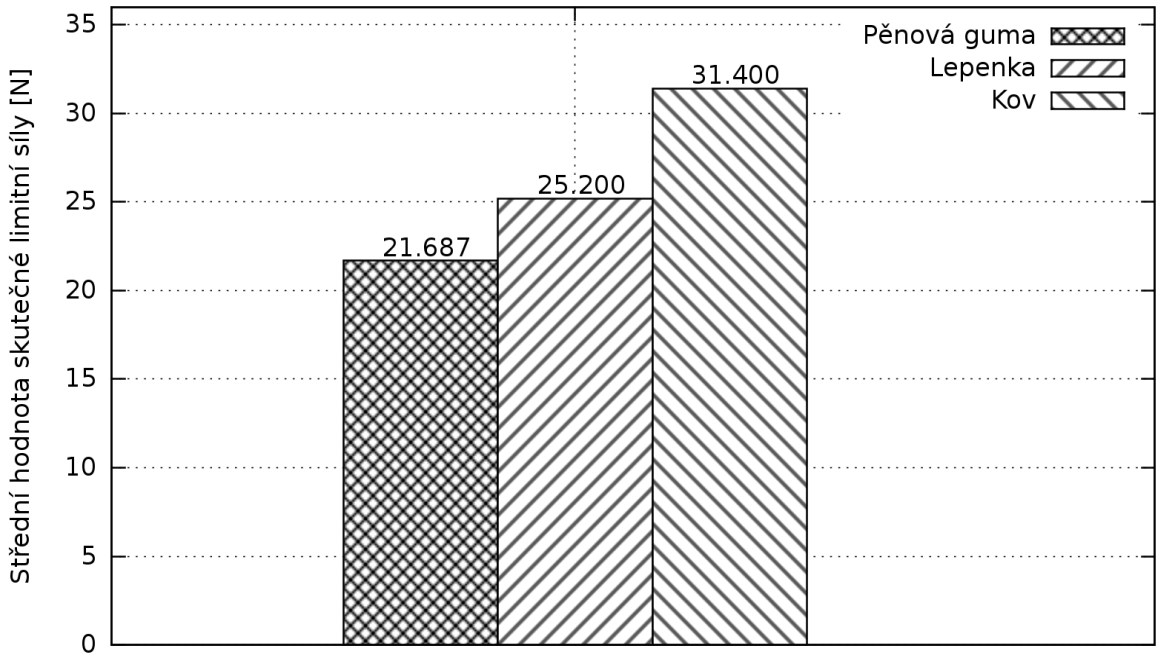
Alternativní metodu vytváření kalibrační tabulky lze tedy celkově z použitých metod tvorby a předzpracování kalibrační tabulky prohlásit za nejúspěšnější. Alternativní kalibrační tabulka dosahuje jednoznačně nejlepších výsledků při reálných experimentech. Významný je u této metody také

fakt, že oproti klasické metodě vyžaduje pro dosažení srovnatelných výsledků nejmenší počet měření. Díky tomu je kalibrace regulátoru ve všech směrech méně náročná operace. Výhodou alternativní metody je také to, že kalibrační křivka je funkcí limitní síly. Převod limitní síly na proud je tedy vždy jednoznačný.

5.4 Vliv tuhosti materiálu na přesnost regulace

Jak bylo ve specifikaci parametrů regulátoru v kapitole týkající se návrhu naznačeno, regulátor reaguje na překročení limitní síly s určitým zpožděním. Toto zpoždění nesouvisí s citlivostí regulátoru, má ale přesto vliv na limitní sílu stisku. Pokud totiž regulátor zareaguje na překročení limitní síly stisku se zpožděním, nedojde k zastavení prstů ve správný okamžik. Díky opožděnému zastavení mohou pohybující se prsty vyvinout výrazně větší sílu stisku, než jaká byla požadována. Zvláště u tvrdých materiálů může být tento nárůst značný. Teoreticky u dokonale nepružných materiálů a v případě dokonale nepružných prstů vyroste síla během nenulového zpoždění od okamžiku kolize do nekonečna. V praxi samozřejmě přinejmenším prsty chapadla pruží, ale i tak lze předpokládat významný vliv tvrdosti materiálu na výslednou limitní sílu. Proto byl vliv tuhosti ověřen experimentem.

Experiment fungoval tak, že byl prsty chapadla opakovaně svírán článek siloměru a mezi prsty a siloměr byla vkládána vrstva materiálu určitých vlastností. Zde je vhodné podotknout, že při všech ostatních experimentech byla mezi prsty a měřicí článek vkládána vrstva pokud možno ideálně pružného materiálu právě z důvodu vyloučení vlivu zpoždění regulátoru na výsledky experimentů. S každým materiálem proběhlo několik sevření a rozevření prstů (konkrétně 10) a pro všechna jednotlivá sevření byl zaznamenán průběh síly. V průběhu byla potom nalezena maximální dosažené síly. Z výsledků jednotlivých experimentů pro konkrétní materiál byl na závěr spočítán průměr. Tedy celý experiment fungoval stejně, jako experiment ověřující přesnost regulace. Z hlediska tvrdosti materiálu byl samotný kovový článek siloměru považován za maximálně tvrdý materiál. V tomto směru není článek siloměru od ideálu daleko. Sice se vlivem působení síly deformuje, ale tato deformace je zcela



Graf 14: Vizualizace vlivu tvrdosti materiálu na střední hodnotu limitní síly.

minimální vzhledem ke vzdálenosti, jakou urazí prsty chapadla za vzorkovací periodu. Výsledky experimentu pro jednotlivé materiály od nejpružnějšího k nejtvrdšímu byly shrnuty do grafu č.14.

V grafu č.14 je zobrazena střední hodnota limitní síly pro materiály různé tvrdosti. Zadaná síla, na kterou měl regulátor síly stisku omezit, byla 25N. Použita byla alternativní kalibrační tabulka. Vzhledem k tomu, že průběh v tabulce je průměrně mírně posunutý pod přímku ideálního průběhu (byl tak ponechán z bezpečnostních důvodů), měla by se skutečná síla stisku v tomto experimentu pohybovat těsně pod hranicí 25N. Pro pěnovou gumu zastavil regulátor prsty spíše předčasně – průměrná síla stisku se pohybovala mezi 21N a 22N. Záporná chyba vzhledem k zadané hodnotě síly byla přibližně -13%. Pro lepenku (karton) se limitní síla zvýšila na přibližně 25N. Zde je průměrná chyba regulace zcela minimální. Mnohem větší chyba regulace nastává v situaci, kdy je svírán samotný siloměr – tedy jsou odstraněny všechny pružné vrstvy mezi prsty a siloměrem a při sevření naráží kov na kov. Průměrná hodnota limitní síly se zvýšila vlivem odstranění pružných vrstev na 31N, což představuje kladnou chybu přesahující 25%.

Z experimentu lze tedy vyvodit jednoznačný závěr, že tvrdost materiálu má významný vliv na přesnost regulace. S tímto vlivem je potřeba počítat a v případě, že je nutné uchopovat nepružné předměty bez pružné mezivrstvy, který by poskytla potřebnou toleranci pro zpoždění reakce regulátoru, je třeba nárůst síly zohlednit v kalibrační tabulce.

5.5 Vlastnosti detektoru kolize

Předchozí testy, ačkoliv se týkaly regulátoru jako celku, testovaly především vlastnosti omezovače síly stisku. Proto byl zvlášť otestován i samotný detektor kolize. Testována byla citlivost detektoru kolize a rychlost reakce – tedy čas uplynulý od skutečné kolize prstů s překážkou do vyhodnocení kolize detektorem. Dle parametrů regulátoru stanovených při návrhu by měl být detektor kolize schopen sílu působící proti pohybu prstů nad 5N spolehlivě vyhodnotit jako kolizi. Rychlost reakce detektoru kolize sice není návrhem přesně specifikována, ale je jisté, že nesmí překročit maximální zpoždění celého regulátoru, které bylo návrhem stanoveno na 600ms (údaje z tabulky č.3). Výstupem experimentu jsou skutečné hodnoty výše popsaných vlastností detektoru a zhodnocení, zda získané hodnoty splňují specifikaci parametrů v návrhu regulátoru, či nikoliv.

Měření parametrů probíhalo tak, že byl prsty chapadla opakovaně svírán měřicí článek siloměru až do okamžiku, kdy vyhlásil detektor kolizi. V tomto okamžiku byly prsty zastaveny. Průběh síly během každého sevření prstů byl zaznamenáván. Maximální sílu, při které byla kolize vyhodnocena, bylo možné určit snadno – v zaznamenaném průběhu síly byla nalezena maximální hodnota a tato byla prohlášena za maximální sílu stisku. Určení zpoždění bylo poněkud obtížnější a méně přesné. Za zpoždění při detekci kolizí byl prohlášen čas mezi zachycením prvního vzorku překračujícího minimální hodnotu siloměrem po okamžik vyhlášení kolize detektorem. Vzhledem k nízké vzorkovací frekvenci siloměru ale při tomto měření může vzniknou významná chyba v řádu jednotek až desítek procent ze změřené hodnoty – vzorkovací perioda odpovídá 13% ze specifikované hodnoty zpoždění. Přesnější způsob měření však nebyl k dispozici. Výsledky experimentů jsou shrnuty v tabulce 9.

Parametr [jednotka]	**Střední hodnota / směrodatná odchylka**	**Splněna specifikace**
Citlivost detekce [N]	3,5/1,1	ANO
Rychlost reakce [ms]	388/205	ANO

Tabulka 9: Změřené vlastnosti detektoru kolize.

Výsledky měření ukazují, že navrhované parametry byly implementovaným detektorem kolize do značné míry splněny. Střední hodnoty experimentů se pohybují spolehlivě pod hranicí stanovenou návrhem. Jistým nedostatkem je ale fakt, že návrh definuje parametry jako horní mez, zatímco u skutečné implementace nejsou tyto parametry jednoznačně omezeny. Směrodatná odchylka uvedená

u obou parametrů naznačuje, s jakou pravděpodobností bude konkrétní sevření prstů parametry splňovat, ale pravděpodobnost překročení limitu je z principu nenulová. V případě citlivosti detekce je pravděpodobnost dodržení limitu poměrně velká – předpokládáme-li gaussovské rozložení hodnot, dosahuje přibližně 80%. O poznání horší je situace týkající se zpoždění detektoru. Zde velikost směrodatné odchylky omezuje pravděpodobnost dodržení limitu na přibližně 70% výsledků. Ve více než čtvrtině případů tedy nebude maximální zpoždění regulátoru vlivem zpoždění rozpoznávače dodrženo. Celkové zpoždění regulátoru je díky značnému rozptylu hodnot skutečné síly obtížné změřit – v mnoha případech regulátor prsty zastaví dříve, než limitní síly stisku dosáhnou. Zpoždění regulátoru je ale zdola omezené zpožděním detektoru kolize. Omezovač síly pracuje paralelně s detektorem kolize a tak by jeho zpoždění nemělo mít zásadní vliv. Ze zpoždění detektoru je tedy možné celkové zpoždění regulátoru odhadnout.

6 Možnosti rozšíření regulátoru

Tato kapitola se věnuje možnostem rozšíření regulátoru a způsobům využití dat získávaných během regulace. Cílem kapitoly je prozkoumat možnosti zdokonalení regulace za hranice samotné implementace regulátoru a také navrhnout případná využití informací poskytovaných regulátorem nadřazeným systémem. Popis konkrétních možností rozšíření je rozdělen do podkapitol.

6.1 Detekce poruch

Vzhledem k tomu, že regulátor poskytuje vzorky okamžitého proudu, bylo by možné tyto informace využít k detekci zvýšeného odběru proudu signalizujícího poruchu. Zvýšený odběr proudu lze zjistit ze zachyceného průběhu – stačí určit velikost stejnosměrného posuvu průběhu na proudové ose. Problémem, se kterým by se ale musel takový detektor poruch vypořádat, je volba vhodných úseků průběhu, ze kterých by bylo možné poruchu detekovat. Jakmile se totiž prsty střetnou s překážkou, odběr pohonu výrazně stoupá. Takové zvýšení by zcela zastínilo stejnosměrný posuv celého průběhu. Dále by také detektor musel udržovat informaci o čase provozu chapadla. Pokud se totiž chapadlo rozběhne po delší době nečinnosti, je jeho odběr během zahřívání zvýšený. Všechny tyto okolnosti by musel detektor poruch vyloučit, aby mohl ze zachyceného průběhu spolehlivě vyhodnotit, zda porucha nastala. Určení správných okamžiků je ale poměrně problematické. Zvláště pak v reálném čase, kdy není předem známo, kdy nastane kolize prstů s překážkou. Detekci poruch by tedy bylo lepší provádět z historických průběhů doplněných o informace o okamžiku kolize a okamžiku vyhodnocení překročení limitní síly. Takto by mohl detektor snáze vyloučit ovlivněné úseky zaznamenaných průběhů.

Jiným přístupem k detekci poruch je dlouhodobá statistika. Zatímco výše popsané řešení by dokázalo okamžitě detekovat poruchy projevující se prudkou změnou odběru proudu, s rozpoznání pomalých změn způsobených opotřebením nebo nedostatkem maziva by mělo problémy. Zde by naopak byla vhodnější dlouhodobá statistika, která by průměrovala všechny validní úseky průběhů za delší časové období – například jeden týden, nebo měsíc. Vzhledem k dostatečnému množství vzorků by nebyl problém s odstraněním vlivu šumu na přesnou hodnotu stejnosměrného posuvu. Díky tomu by bylo možné rozpoznat i velmi malý posuv a tak v mnoha případech poruchy nejen detekovat, ale také jim předcházet.

6.2 Optická navigace prstů chapadla

Vzhledem k velmi pomalému pohybu prstů chapadla při regulaci síly by bylo vhodné posunout prsty bez omezení síly co nejblíže předmětu tak, aby se dal potom v krátkém čase s omezením síly uchopit. Podobně funguje i chování živých tvorů při uchopování křehkých předmětů. Když se například pokusíme uchopit sklenici na víno, nejdříve ruku bez přesnější kontroly přiblížíme ke sklenici a potom s citem pomalu sevřeme prsty kolem stopky. Kdybychom stejně pomalu pohybovali rukou po celou dobu od začátku záměru uchopit sklenici, uchopení by trvalo podstatně déle. Pro navigaci ruky do blízkosti sklenice používáme zrak.

Na stejném principu by mohlo fungovat i uchopování předmětů robotickým chapadlem. V tomto případě ale nejde o navigaci celého ramene, nýbrž pouze prstů chapadla. Na chapadle jsou totiž umístěny dvě kamery, které by měly poskytnout operátorovi pracujícímu s ramenem vzdáleně přehled o uchopovaném předmětu. Tyto dvě kamery vidí jak prsty chapadla, tak uchopovaný předmět. Mohly by tedy sloužit jako umělé oči, které by řízení robota dávaly informaci o tom, jak moc se ještě může k uchopovanému předmětu přiblížit, aby pak měl ještě dost prostoru pro jeho bezpečné uchopení. To by

značně urychlilo především uchopování neznámých předmětů, u kterých není možné dopředu bezpečnou vzdálenost prstů od předmětu definovat, a nebo předmětů, jejichž přesná poloha není známa.

Samotný problém optické navigace prstů lze rozdělit na dva samostatné podproblémy. Tím prvním je lokalizace prstů v obraze a tím druhým lokalizace uchopovaného předmětu. Lokalizace prstů nepředstavuje zásadní problém. Prsty mají jediný stupeň volnosti. Jejich pravděpodobnou polohu lze tedy omezit na jedinou přímku v prostoru. Na této přímce je možné hledat prsty například korelací s předem připravenou předlohou. Pro vyloučení záměny prstů s jiným objektem lze prsty označit signálními samolepkami, nebo jiným způsobem zvýšit počet významných bodů na prstech. O poznání složitější je lokalizace neznámého předmětu. Korelaci v tomto případě nelze použít, protože pro neznámý předmět není k dispozici předloha. I v případě, že bychom uchopovaný předmět předem znali, může se vůči kameře různě otočit. Navíc udržovat databázi předloh pro rozpoznání každého předmětu, se kterým ruka kdy pracovala, je poměrně náročné. V případě lokalizace neznámého předmětu lze ale využít skutečnosti, že kamery jsou umístěny velmi blízko prstům. Díky tomu totiž musí být kamery zaostřeny na velmi malou vzdálenost. To způsobuje, že veškeré předměty, které jsou blíž nebo dál od kamer než prsty chapadla, jsou rozostřené. Pomocí ostrosti objektu v obraze lze tedy vyloučit z obrazu příliš vzdálené předměty. Dále je možné určit polohu prstů způsobem popsaným výše. Po těchto krocích se oblast hledání neznámého objektu značně omezuje. Je zřejmé, že uchopovaný předmět se musí v obraze nacházet mezi prsty a ve stejné vzdálenosti od kamery, jako prsty samotné. Samotný objekt by pak nemuselo být příliš náročné najít detekcí založenou na významných bodech objektu. U objektu nemusíme ani znát jeho střed – klíčová je z hlediska navigace prstů poloha hran objektu a jejich vzdálenost od prstů chapadla. Vzdálenost prstů od hran objektu zjistíme jednoduše tak, že spočítáme pixely mezi prsty chapadla a hranou objektu a převedeme vynásobením odpovídajícím koeficientem na mm.

Tento návrh je pouze teoretický. Při jeho realizaci mohou nastat různé problémy, díky kterým se nakonec celý návrh může ukázat jako nesprávný a bude muset být vytvořen návrh jiný. Přesto by ale bylo vhodné optickou navigaci prstů vůči uchopovanému předmětu implementovat, protože její přínos by mohl být velmi praktický.

6.3 Rozpoznávání materiálu

Rozpoznávání materiálu pomocí robotického chapadla si klade za cíl poznat podle průběhu síly při stiskání materiálu, o jaký materiál se jedná. Vzhledem k tomu, že chapadlo nemá přímé měření síly, bylo nutné jako vstup pro rozpoznávač použít vzorky naměřených hodnot proudového odběru jeho motoru. K trénování a testování rozpoznávače byly využity záznamy průběhů odběru proudu chapadlem.

6.3.1 Vstupní filtr

Vzhledem k tomu, že průběh okamžitého odběru proudu je velmi zašuměný (viz graf č.1), bylo nutné vstup filtrovat. Navržený vstupní filtr, je složen ze dvou sériově spojených filtrů – integračního a prahového. Integrační filtr vyhladí zašuměný průběh odebíraného proudu potlačením vysokých frekvencí. Váha integrační složky filtru je však shora omezená nutností detekovat rychlé změny proudu u tvrdých materiálů. Z toho důvodu byl zapojen za integrační filtr filtr prahový. Ten nahradí integrovaný klidový proud dokonalou nulou a tím silně omezí chyby rozpoznání vlivem šumu v klidovém proudu. Hodnotu prahu je nutné zvolit tak, aby co nejméně zasahovala do oblasti, kde se odebíraný proud vlivem zátěže prstů zvyšuje. Prahový filtr je možné použít ke zlepšení rozlišitelnosti vstupů právě díky specifické vlastnosti pohonu chapadla, kdy při malém zatížení prstů má šum sice velkou amplitudu, ale s rostoucím zatížením amplituda šumu rychle klesá. Prahový filtr tedy prakticky z průběhu odstraní oblast, kterou integrační filtr nezvládal dostatečně dobře zpracovat.

Filtr	Parametr [jednotka]	Hodnota parametru
Integrační	Počet průměrovaných vzorků [-]	20
Prahový	Práh nulování [A]	2,5

Tabulka 10: Parametry filtrů pro předzpracování vstupu rozpoznávače.

Parametry filtrů byly určeny experimentováním nad množinou anotovaných vzorků. Pro každou konfiguraci filtrů byla provedena série pokusných rozpoznání a vyhodnocena úspěšnost. Do grafů byly zaznamenány průběhy proudu při postupné aplikaci jednotlivých již odladěných filtrů. I lidským okem je viditelné, že charakteristické rysy průběhů jsou po aplikaci filtrace zřetelnější.

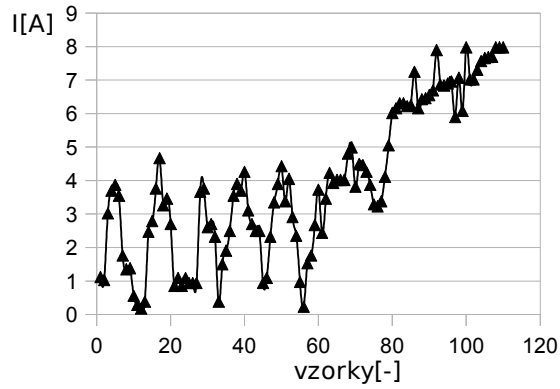
6.3.2 Neuronová síť

K vlastnímu rozpoznávání byla použita neuronová síť RCE. Algoritmus RCE vymezí pro každou třídu hyperkulový prostor, který představuje prostor pro zobecňování. Hyperkoule různých tříd se mohou i překrývat, a pokud se pak vstupní vzorek „trefí“ do prostoru sdíleného hyperkoulemi různých tříd, lze to interpretovat jako nejistotu – materiál s nějakou pravděpodobností náleží každé z protínajících se tříd, což je přirozenější, než ostrá hranice. Vstupem neuronové sítě je výstup vyfiltrovaného vstupního signálu – posledních k hodnot před tím, než ruku zastaví interní omezení proudu (nejlepší výsledky byly získány s $k = 110$). Blokové schéma celého rozpoznávače je zobrazeno na obrázku č.8. Chování neuronové sítě lze ladit následujícími parametry popsanými v tabulce č.11.

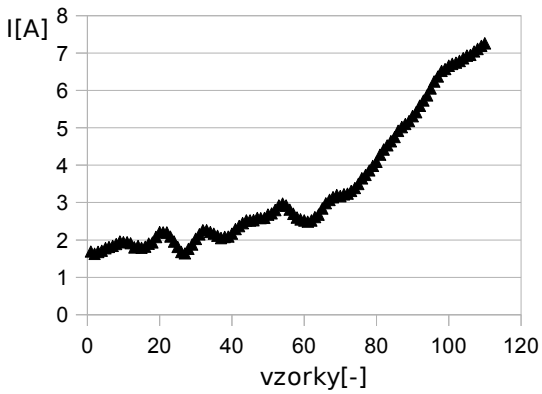
Parametr [jednotka]	Hodnota parametru
Délka vektoru vstupů k [-]	110
Výchozí velikost poloměru hyperkoule [-]	35

Tabulka 11: Parametry neuronové sítě rozpoznávače.

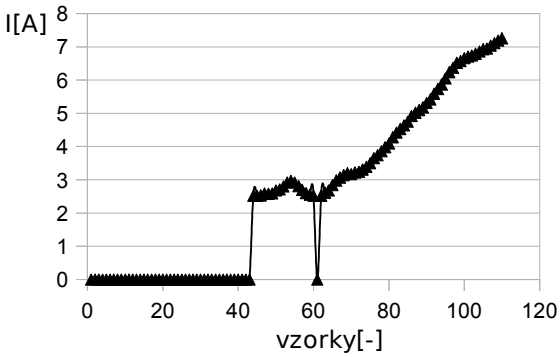
Poloměr hyperkoule neměl během testování na úspěšnost rozpoznávače zásadní vliv. Pouze v případě, že byl příliš malý, docházelo k tomu, že rozpoznávač nedokázal materiál zařadit. Hodnota poloměru byla tedy nastavena tak, aby s rozumnou rezervou překračovala práh, pod kterým tento nepříznivý jev nastával. Naopak délka vektoru vstupů měla na úspěšnost rozpoznávání vliv zásadní. Bylo nutné tento parametr nastavit tak, aby pokrýval úsek signálu, kde docházelo k deformaci objektu prsty chapadla. Právě v této části se totiž průběhy od sebe lišily. Rozpoznávač pracuje vždy nad posledními k vzorky po tom, co prsty zastaví omezující proud. Vstupní vektor délky k by tedy měl ideálně pokrýt úsek signálu od okamžiku střetu prstů s objektem. To nelze u všech materiálů zaručit, protože u každého materiálu stoupá vlivem deformace síla jinak a často i po jiných přírůstcích. Proto byl experimentálně stanoven kompromis, při kterém se dařilo materiály nejlépe rozpoznat. Na parametr k má vliv také vstupní filtrace. Integrační filtr vyhlazuje ostré změny v signálu a tím způsobuje zpoždění projevů nárůstu odebíraného proudu vlivem odporu materiálu. Úsek vstupního signálu s různými vlivy filtrace je zobrazen v grafu č.15.



1. Bez vstupní filtrace



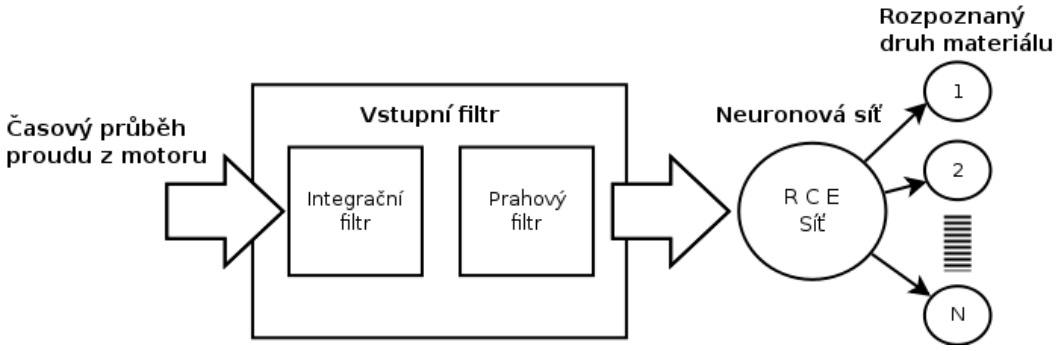
2. Filtrace integračním filtrem



3. Filtrace integračním a následně prahovým filtrem

V grafu č.15 jsou ukázány průběhy proudového signálu před a po filtraci - grafy popisují závislost okamžité velikosti proudu odebíraného motorem chapadla [A] na normovaném čase [bezrozměrný].

Graf 15: Průběhy proudu po aplikaci různých filtrů.



Obrázek 8: Princip rozpoznávače

6.3.3 Návrh tříd pro rozpoznávač

Při návrhu rozpoznávače byl použit objektově orientovaný přístup. Tato podkapitola popisuje třídy navržené pro neuronovou síť. Nejsou zde rozepsány vstupy a výstupy hodnot. To jednak proto, že z hlediska dokumentace principu nejsou podstatné a také proto, že se při různých použitích rozpoznávače mění podle toho, jakým rozhraním je rozpoznávač k okolním systémům připojen. Rozhraní

vstupního filtru je rovněž ovlivněno konkrétním způsobem vstupu a tak zde není uvedeno. Popis tříd je v jazyce C++. V tomto jazyce byl také rozpoznávač implementován.
Třída pro jeden vzorek. Vzorek je pro rozpoznávač úsek vyfiltrovaného průběhu, jak bylo popsáno výše. Každý vstupní vektor je charakterizován třídou, do které patří – atribut *clasID*.

```
class InputVector
{
   protected:
      int classID;
      std::vector<double> vals;

   public:
      InputVector(std::string datafile, int class_id); /* konstruktor */
      int ClassID(); /* vrátí ID vektoru */

   friend class Neuron;
};
```

Třída reprezentující jeden neuron sítě. Neuron je reprezentován středem *S* o dimenzionalitě vstupního vektoru a poloměrem hyperkoule *R*.

```
class Neuron
{
   protected:
      std::vector<double> S; /* střed neuronu - N hodnot */
      double R; /* poloměr hyperkoule */

   public:
      void ReduceSphere(InputVector); /* redukce hyperkoule na polovinu
vzdál. středu a zadaného vektoru */
      bool SphereEval(InputVector); /* ohodnoceni náležitosti vektoru do
hyperkoule neuronu */
      Neuron(InputVector); /* konstruktor se středem ve vektoru V */
};
```

Třída reprezentující jednu klasifikační třídu sítě. Každá klasifikační třída obsahuje seznam neuronů *Neurons* obsahující neurony, které do dané třídy patří a identifikátor třídy *classID*.

```
class InputClass
{
   protected:
      std::list<Neuron> Neurons; /* množina neuronu ve třídě neuronu */
      int classID; /* každá třída má svou jedinečnou identifikaci */

   public:
      InputClass(); /* konstruktor třídy  neuronu */
      void ID(int); /* zadání celočíselného ID třídy */
      InputClass(int); /* konstruktor třídy  neuronu – přímo se zadáním
      celočíselného ID třídy */
      int ClassID(); /* vrátí ID třídy */
      void InsertNeuron(Neuron); /* vložení neuronu do třídy */
      bool VectorFit(InputVector); /* ověří, zda je vstupní vektor
      přijímán touto třídou neuronu - OR výstupní neuron */

    friend class NNetwork;
    friend void LearnRCE(NNetwork&, std::string);
```

```
  };
```

Třída reprezentující celou neuronovou síť. Obsahuje hashovací tabulku tříd podle identifikátoru třídy. Tato třída poskytuje rozhraní pro ohodnocení neznámého vzorku. Učení je realizováno samostatnou funkcí, která očekává jako parametr referenci na neuronovou síť a cestu k seznamu trénovacích dat.

```
class NNetwork
{
   private:
      std::map<int, InputClass> Classes; /* K tříd */
      void AddClass(int); /* vytvoření třídy s daným ID */
      bool FindClass(int); /* najde třídu daného ID */

   public:
      void InsertNeuron(Neuron, int); /* vloží neuron a zařadí ho podle ID
      do správné třídy */
      std::vector<int> EvalInput(InputVector); /* ohodnocení vstupu
      a vyhlášení, do kterých tříd patří */

   friend void LearnRCE(NNetwork&, std::string);
};
```

6.3.4 Reálné experimenty

Rozpoznávač byl laděn a testován na sadě vzorků pěti materiálů: dřeva, tvrzené gumy, lepenky (kartonu), polystyrénu a pěnové gumy (postupně od nejtvrdšího k nejměkčímu). Záměrně byly vybrány materiály, které lze rozlišit jen na základě silového působení. Robotická ruka nevnímá vlastnosti povrchu a proto jejím prostřednictvím nelze rozpoznat některé lidským hmatem snadno rozpoznatelné materiály, jako jsou například tvrdé dřevo a sklo. Výsledky experimentů jsou uvedeny v následujících dvou tabulkách.

Vstupní filtr	**Chybně rozpoznáno / nerozpoznáno [%]**
Žádný	0/93,3
Integrace	0/13,3
Integrace a prahování	0/6,7

Materiál	**Chybně klas. / neklasifikováno. [%]**
Dřevo	0/0
Tvrzená guma	0/0
Lepenka	0/0
Polystyrén	0/33,3
Pěnová guma	0/0

[vlevo]: Vliv filtrace na úspěšnost rozpoznávání
[vpravo]: Úspěšnost rozpoznávání jednotlivých materiálů

Tabulka 12: Experimenty s detekcí materiálu

6.3.5 Zhodnocení rozpoznávače

Reálné experimenty ukázaly, že rozpoznávač dosahuje úspěšnosti použitelné pro další nadřazené systémy. Přesto se však u rozpoznávače objevují nežádoucí vlastnosti, jako občasná neschopnost klasifikovat materiál. Vhodnější by byla neurčitá klasifikace více materiálů současně. Lze také očekávat sníženou úspěšnost rozpoznávače pro materiály, které mají podobné mechanické vlastnosti. Prakticky to znamená, že rozpoznávač bude nutné doplnit o další vstupy (například obraz z kamer) pro dostatečně spolehlivou klasifikaci libovolného materiálu.

Současné řešení rozpoznávače nabízí prostor pro další vývoj. Vývoj rozpoznávače by se měl zaměřit na neuronovou síť, kdy připadá v úvahu vícevrstvá síť a také jiná forma vstupu signálu do neuronové sítě. Současné řešení totiž vyžaduje, aby pohon chapadla dosáhl určité úrovně, aby mohlo rozpoznání proběhnout. Ideální rozpoznávač pro poskytování průběžných informací regulátoru síly stisku by byl takový, který by dokázal ohodnotit průběh po každém vzorku, nebo alespoň po každém krátkém úseku vzorků. Toto řešení rozpoznávače poskytuje informaci o druhu materiálu jen jednou během uchopování objektu. Pokud se ale nastaví hladina, při které rozpoznání proběhne, na rozumnou hodnotu, může být i tato jedna informace dobře použitelná.

V konečném výsledku by měl rozpoznávač sloužit jednak jako „poradní hlas“ při regulaci síly stisku chapadla a také jako další zdroj informací pro nadřazený systém. Propojení rozpoznávače s regulátorem nebylo v rámci této práce z časových důvodů implementováno. Současné řešení pracuje s off-line záznamy průběhů uloženými v souborech. Bude tedy potřeba implementovat propojení rozpoznávače s regulátorem pro rozpoznávání v reálném čase, stanovit vhodnou hladinu proudu pro provedení rozpoznání, stanovit pravidla pro jednotlivé druhy materiálu, která budou ovlivňovat průběh regulace, a nakonec ověřit vliv rozpoznávání materiálu na regulaci sadou testů. Použití rozpoznávače by ale mohlo být užitečné především v případě uchopování tvrdých předmětů, kde bylo možné na základě rozpoznání tvrdého materiálu zastavit prsty s předstihem tak, aby nedošlo vlivem zpoždění regulátoru k překročení limitní síly.

7 Závěr

Závěrečná kapitola je shrnutím celé práce. Hodnotí postup práce a dosažené výsledky. Zmiňuje také překážky, které se v rámci práce vyskytly, a způsob, jakým byly překonány. Dále se závěr věnuje nedostatkům, které byly u současného řešení zjištěny a možnostmi jejich odstranění. Součástí závěru je také perspektiva budoucího využití práce včetně možností dalšího rozšíření.

7.1 Postup práce

Zadáním práce bylo nastudovat ovládání robotického chapadla, zapojit a zprovoznit chapadlo, navrhnout a implementovat regulátor síly stisku a ověřit vlastnosti implementovaného regulátoru. Těmito body zadání byl postup práce řízen. Nejdříve bylo třeba zprovoznit zapojit a oživit chapadlo a zprovoznit komunikaci s chapadlem. Tato část práce probíhala průběžně v předchozích semestrech a byla dokončena už v rámci semestrálního projektu. Ačkoliv zapojení chapadla není složité, byla tato část práce poměrně náročná. Bylo třeba prostudovat poměrně rozsáhlou dokumentaci chapadla i ramene a navrhnout a zrealizovat fyzické propojení chapadla s řídicím terminálem a napájecím zdrojem a to včetně vytipování napájecího zdroje a zajištění potřebného materiálu. Největším problémem této části práce byla dokumentace k chapadlu, která byla příliš obecná a neobsahovala detailní popis propojovací desky chapadla. Některé údaje tak bylo nutné určit měřením, nebo experimentálně.

Druhá část práce se potom zabývala návrhem a implementací regulátoru. Tato část práce byla řešena z části v rámci semestrálního projektu a z části potom v posledním semestru. Vývoj regulátoru probíhal iterativně podle spirálového modelu. Na začátku byl navržen prototyp regulátoru, který byl implementován, testován, modifikován a postupně rozšiřován, až bylo dosaženo současného výsledku. Návrh regulátoru probíhal s cílem udělat regulátor co nejsnáze portovatelný mezi operačními systémy GNU/Linux a MS Windows. Značnou překážkou proto byla nedostupnost knihovny m5api pro jiný operační systém, než pro MS Windows. Podle dokumentace sice knihovna přímo pro GNU/Linux existovala, ale pouze v binární podobně pro starší verze některých distribucí GNU/Linuxu. Navíc tato verze nebyla k dispozici ani na CD se softwarem a dokumentací k chapadlu, ani na odkazovaných serverech na internetu. Přenositelnost knihovny na jiné operační systémy byla tedy vyřešena projektem Wine. Samotný návrh regulátoru byl pak náročný především časově. Každý navržený a implementovaný přístup bylo nutné otestovat sadou experimentů. Zlepšení u nových přístupů bylo často jen malé a na první pohled ne zcela patrné. Návrh se nejdříve soustředil na definici rozhraní a parametrů regulátoru. Následně byl zahájen vývoj detektoru kolize prstů s překážkou. U toho bylo potřeba zajistit především dostatečnou citlivost a spolehlivost. V další fázi byl hledán postup, který by dokázal omezit sílu stisku dostatečně stabilně zatím bez ohledu na to, na jakou konkrétní hodnotu síly omezuje. Po nalezení takového postupu byl teprve řešen problém s kalibrací. Při kalibraci byl již nasazen i detektor kolize, který pomáhal potlačit falešná překročení limitního proudu pro nízké hodnoty zadané síly. Závěr práce na regulátoru se týkal experimentování s regulátorem a ladění parametrů jednotlivých funkčních bloků tak, aby regulátor co nejlépe fungoval jako celek.

Poslední bod zadání se týkal možností rozšíření regulátoru. Možná rozšíření byla popsána v kapitole 6. Z těchto rozšíření byl implementován rozpoznávač materiálu pomocí stisku prstů chapadla. Rozpoznávač byl vyvíjen nezávisle na samotném regulátoru síly stisku. Jeho vývoj proběhl už dříve v rámci předmětu Základy umělé inteligence.

7.2 Zhodnocení dosažených výsledků

Z hlediska dosažených výsledků lze práci opět rozdělit na dvě části. První částí bylo zapojení a zprovoznění chapadla, druhou potom navržení a implementace regulátoru síly stisku. Během první části

práce se podařilo úspěšně zrealizovat připojení chapadla a zprovoznit komunikaci s chapadlem. V rámci této části byla otestována funkčnost pohonu chapadla, různé režimy pohybu prstů a funkčnost nastavení proudového omezení pohonu chapadla. V tomto směru lze tedy prohlásit první část práce týkající se hardwaru za úspěšnou.

Největší význam ale mají výsledky dosažené v rámci druhé části práce. Především se pak jedná o úspěšnost vývoje a vlastnosti implementovaného regulátoru. Z pohledu bodů zadání lze i tuto část prohlásit za dokončenou. Regulátor byl navržen, implementován a otestován. Stejně tak proběhl návrh možných rozšíření regulátoru, z nichž byl rozpoznávač materiálu i implementován. Předmětem diskuze jsou ale vlastnosti implementovaných řešení. Na začátku návrhu byly stanoveny vlastnosti a parametry, které by regulátor měl mít. Kromě definovaného rozhraní se jednalo především o maximální chybu regulace, rychlost reakce regulátoru, citlivost detekce kolize a rozsah síly, ve kterém má regulátor pracovat. Z hlediska těchto parametrů lze objektivně říci, že rozsah síly byl dodržen spolehlivě, ale rychlost reakce regulátoru, citlivost detekce kolize a maximální chybu regulace se dodržet nepodařilo. Citlivost detekce kolize se nepodařilo dodržet víceméně v tom, že princip regulátoru neumožňuje zaručit jednoznačnou horní mez síly působící proti pohybu prstů, při které je vyhlášena kolize. Střední hodnota experimentů s detekcí kolize se pohybuje poměrně spolehlivě pod stanovenou hranicí 5N, ale směrodatná odchylka hodnot naznačuje, že hranice může být překročena. U rychlosti reakce byl problém především v tom, že se ji nepodařilo spolehlivě změřit. Spolehlivé měření zpoždění bylo možné provést pouze u detektoru kolize. Tam výsledek měření naznačoval, že se reálné zpoždění regulátoru bude pohybovat okolo stanovené maximální hodnoty a může ji s nemalou pravděpodobností překročit, ale překročení doby zpoždění by nemělo být příliš velké. Maximální chyba regulace byla implementovaným regulátorem překročena zcela jednoznačně. I v případě nejlepší použité metody kalibrace regulátoru překračuje stanovenou maximální chybu i střední hodnota experimentálních měření nehledě na rozptyl hodnot. Z hlediska chyby lze tedy říci, že původní návrh 20% ze zadané hodnoty byl příliš optimistický. Střední hodnota změřených hodnot se v nejhorších případech odchyluje od zadané o přibližně 30%. Směrodatná odchylka pak naznačuje, že s nemalou pravděpodobností mohou při regulaci vzniknout chyby dalece překračující 50% zadané hodnoty. Nenulová pravděpodobnost chyb překračujících 50% znamená spíše to, že regulace není zcela spolehlivá. Mohou tedy nastat případy, kdy regulace zareaguje příliš pozdě nebo naopak příliš brzy výsledná síla stisku se tak výrazně liší od zadané. Pro případ úplného selhání regulace je pohonu chapadla nastaven hardwarový limit proudu. Ten však odpovídá síle překračující 100N. Jedná se tedy spíše o ochranu samotného chapadla před poškozením. Uchopované předměty už mohou být touto silou poškozeny. V praxi však k takto závažnému selhání regulace dochází víceméně jen vlivem zásadní poruchy, jako je předčasné ukončení procesu regulátoru, nebo výpadek komunikace mezi regulátorem a chapadlem.

Výsledný regulátor má také některé nedostatky, které jsou výsledkem nesprávných rozhodnutí ve fázi návrhu. Asi největší problémy způsobuje použití knihovny m5api. Knihovna sice ušetřila poměrně hodně práce při implementaci tím, že nebylo nutné implementovat protokol pro komunikaci s chapadlem, zároveň je ale zdrojem mnoha omezení stávající implementace především co se týče možností portace výsledného regulátoru na jiné platformy. Řešením by bylo implementovat generování a zpracování potřebných zpráv protokolu vlastními silami. Takové řešení ale není příliš systematické. Vhodnější by bylo vytvořit otevřenou implementaci knihovny m5api nebo alespoň její části pro komunikaci s jednotkami rozhraním RS-232. O něco méně závažným nedostatkem je potom nepříliš optimální implementace. Ta nepřímo vyplývá z experimentální povahy celého řešení, kde byl brán zřetel spíše na možnosti budoucích modifikací, než na výkon výsledné implementace. Výkon by se dal jednoznačně zvýšit například použitím nativní implementace předzpracování vstupního signálu namísto současného řešení, které částečně využívá interpretovaného jazyka Octave. I přes omezenou přesnost a výše popsané nedostatky je regulátor funkční a neměl by být žádný problém s jeho začleněním do komplexního systému pro řízení a monitorování celého robotického ramene.

7.3 Další vývoj práce

Další vývoj práce okolo regulátoru by se měl zaměřit především na koordinaci řízení ramene a chapadla. Výsledkem by mělo být jednoduché rozhraní pro práci s ramenem, které by umožňovalo uživateli pohybovat s ramenem a zadávat rameni program, který má vykonat. Součástí řízení je i zajištění bezpečnosti ramene a ostatního vybavení pracoviště a zaručení celkové odolnosti systému vůči chybám i záměrným útokům. Rameno s jednoduchým rozhraním pro ovládání by pak mělo sloužit jako nástroj pro experimenty z oblasti robotiky a umělé inteligence.

Prostor k dalšímu rozvoji nabízí i samotný regulátor. Jak už bylo naznačeno výše, bylo by do budoucna vhodné nahradit knihovnu m5api otevřenou implementací a celou implementaci regulátoru zmenšit a zoptimalizovat. Prostor pro další výzkum nabízí i samotné řešení regulátoru. Není vyloučeno, že použitím přístupu jiného, než je stávající, by bylo možné dosáhnout lepších výsledků především v oblasti přesnosti regulace. Pravděpodobně největší prostor pro další vývoj představují možnosti rozšíření regulátoru. Především by bylo vhodné dokončit propojení rozpoznávače materiálu s regulátorem a implementovat navigaci prstů pomocí kamer. To by umožnilo inteligentnější a hlavně rychlejší uchopování předmětů. Kromě těchto dvou navrhovaných rozšíření zůstává ještě celá řada možností, jak regulátor zdokonalit. Směr dalšího vývoje pravděpodobně také významnou měrou ovlivní zkušenosti s regulátorem při praktickém nasazení.

Literatura

[1] Schunk GmbH & Co KG, Lauffen/Neckar: PG70 Assembly and Operating Manual, EN, 2006.

[2] Mitsubischi Melfa: RV-6S Series Standard Specifications Manual, EN, 2005, BFP-A8322D.

[3] Mitsubischi Melfa: CR1/CR2/CR3/CR4/CR7/CR8/CR9 Controller INSTRUCTION MANUAL Detailed explanations of functions and operations, EN, 2005, BFP-A5992-K.

[4] Kolektiv autorů: Introduction to Profibus-DP [online]. 2000 [cit. 28.12.2010]. Dostupný z WWW: <http://www.automation.com/resources-tools/articles-white-papers/fieldbus-serial-bus-io-networks/introduction-to-profibus-dp>.

[5] Kolektiv autorů: CAN - Controller Area Network [online]. [cit. 28.12.2010]. Dostupný z WWW: <http://www.pp2can.wz.cz/pages/Can.htm**>.**

[6] Olmr V.: Sériová linka RS-232 [online]. 12.12.2005 [cit. 28.12.2010]. Dostupný z WWW: <http://hw.cz/rs-232>.

[7] Schunk GmbH & Co KG (Tschakarow R., Gaiser F.), Lauffen/Neckar: Plustronic Manual complete, EN, 2003.

[8] Mařík R.: Metoda nejmenších čtverců [online]. 22.1.2006 [cit. 28.4.2011]. Dostupný z WWW: <http://user.mendelu.cz/marik/prez/mnc-cz.pdf>.

[9] Kolektiv autorů: Metoda nejmenších čtverců [online]. [cit. 28.4.2011]. Dostupný z WWW: <http://herodes.feld.cvut.cz/mereni/mnc/mnc.php>.

[10] Peringer P.: Simulační nástroje a techniky, Vysoké učení technické v Brně – Fakulta informačních technologií, 2011.

[11] Kolektiv autorů: Wine-wiki [online]. 22.4.2011 [cit. 20.5.2011]. Dostupný z WWW: <http://wiki.winehq.org/>.

[12] Eaton J. W.: GNU Octave Manual, Network Theory Limited, EN, 2002, 0-9541617-2-6.

Seznam příloh

Příloha 1. CD s elektronickou podobou dokumentu, zdrojovými soubory dokumentu, zdrojovými soubory rozpoznávače materiálu a získanými záznamy průběhů proudu ve formátu CSV.

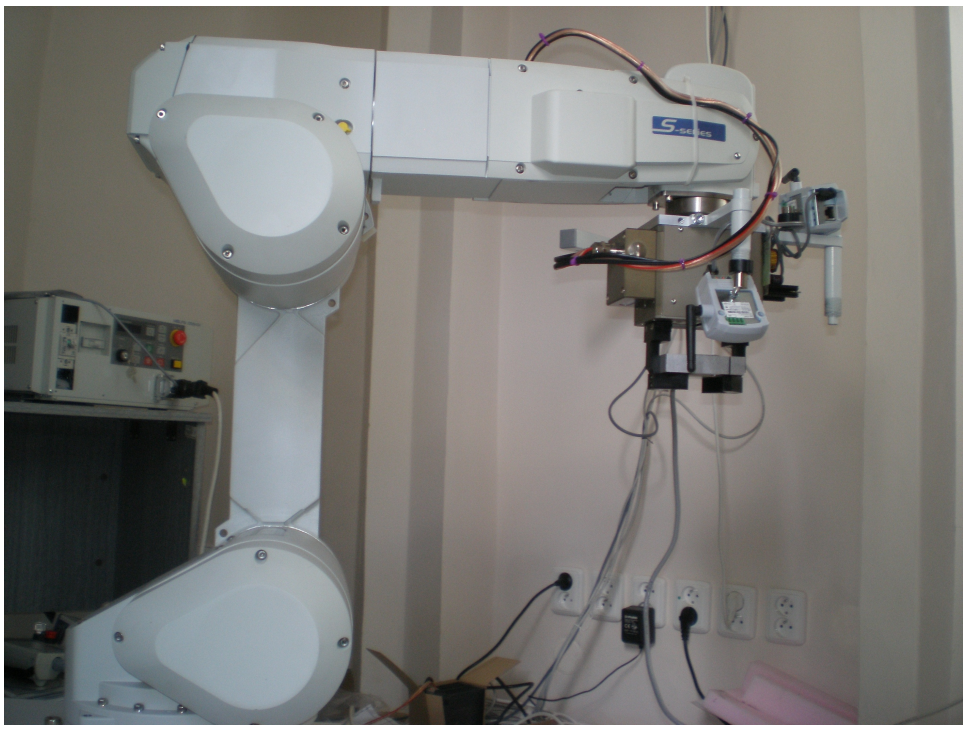
Příloha 2. Fotodokumentace – fotografie chapadla, ramene, kontroléru a siloměru.

Příloha 3. Postup instalace – popis jednotlivých kroků nutných k nainstalování prostředí pro překlad a běh regulátoru.

Příloha 4. Návod k použití k programům HAND_cmd.exe a measure.

Příloha 5. Návrh tříd pro regulátor síly stisku.

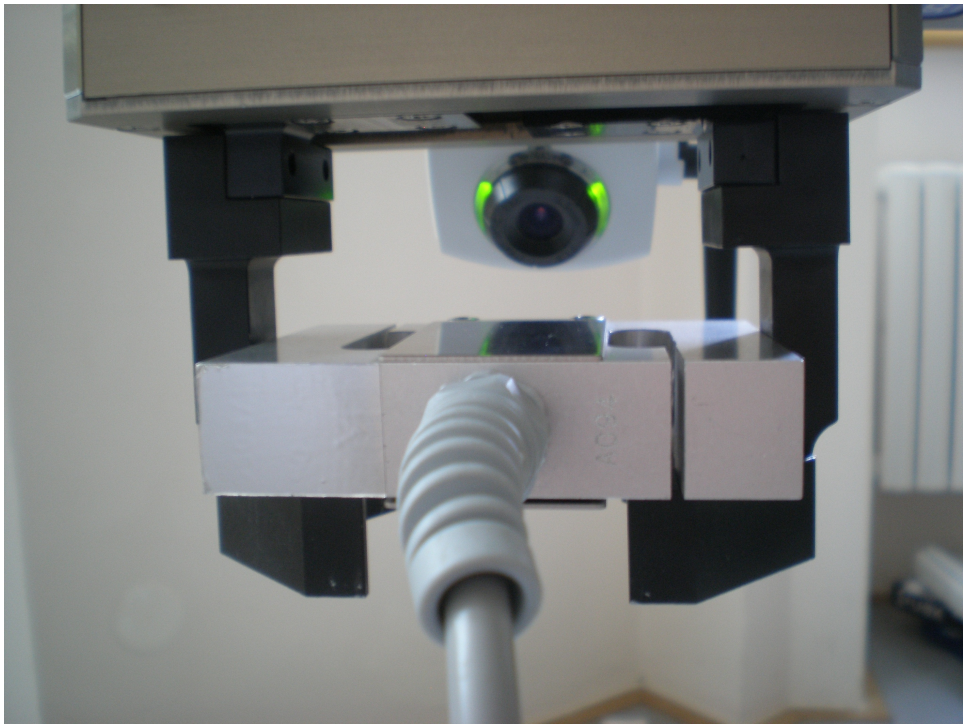
Příloha 2: Fotodokumentace



Fotografie 1: Rameno MELFA RV-S6.

Fotografie 2: Kontrolér ramene CR2B-574.

Fotografie 3: Siloměr.



Fotografie 4: Měřicí článek siloměru v prstech chapadla.

Příloha 3: Postup instalace

Toto je stručný návod k nainstalování prostředí pro překlad a spuštění regulátoru na distribuci Ubuntu. Testování proběhlo na Ubuntu 10.04 LTS. Návod by však měl fungovat i na novějších verzích. Symbol $ nebo # znamená začátek řádku. Řádek pokračuje až do dalšího symbolu nebo do konce bodu postupu. Symbol $ představuje účet běžného uživatele, symbol # účet uživatele root.

1. Instalace 32-bitového chroot prostředí Ubuntu v 64-bitovém Ubuntu

U 32-bitového systému je také vhodné použít chroot kvůli ochraně hostitelského OS.

```
$ sudo apt-get install dchroot debootstrap
$ sudo mkdir /chroot/
$ sudo debootstrap --arch i386 maverick /chroot/
http://archive.ubuntu.com/ubuntu
$ sudo chroot /chroot/
# dpkg-reconfigure locales
```

2. Konfigurace zdrojů software

(v jiném terminálu)
```
$ sudo gedit /chroot/etc/apt/sources.list
```

Je nutné vložit následující řádky:
deb http://archive.ubuntu.com/ubuntu maverick main restricted universe multiverse
deb http://security.ubuntu.com/ubuntu maverick-security main restricted universe multiverse
deb http://ppa.launchpad.net/ubuntu-wine/ppa/ubuntu maverick main
deb-src http://ppa.launchpad.net/ubuntu-wine/ppa/ubuntu maverick main

3. Aktualizace systému

Tento krok není povinný, ale je silně doporučený kvůli konzistenci softwarového prostředí.

(v chroot terminálu)
```
# apt-get update ; apt-get upgrade
```

4. Instalace potřebných balíčků

(v chroot terminálu)
Připojení /proc je třeba provést při každém přihlášení.
```
# mount none /proc -t proc
# apt-get install wine1.3 wine1.3-dev octave3.2-headers g++
libc6-dev gcc-4.5 libgmp3-dev
```

Je třeba potvrdit i instalaci závislostí. Některé balíky také mohou upozorňovat na neplatné klíče.

U mobilních stanic je potřeba také přenést při změně konfigurace sítě nové adresy DNS serverů (v jiném terminálu):
```
$ cp /etc/resolv.conf /chroot/etc/
```

5. Konfigurace Wine

Nastavení domovského adresáře:

```
# export HOME="/root/"
```

Nastavení domovského adresáře je potřeba provést po každém přihlášení.

```
# mkdir /root/.wine/
# winecfg
```

Objeví se okno s konfigurací Wine. Stačí potvrdit současné volby.

Pro konfiguraci klasického sériového portu:

```
# mknod /dev/ttyS0 c 4 64
# ln -s /dev/ttyS0 ~/.wine/dosdevices/com1
```

Pro konfiguraci sériového portu na USB převodníku:

```
# mknod /dev/ttyUSB0 c 188 0
# ln -s /dev/ttyUSB0 ~/.wine/dosdevices/com1
```

6. Vytvoření portu pro measure

Program measue používá samostatnou linku RS-232. Bude pro ni potřeba vytvořit souborové rozhraní.

(v chroot terminálu)

Pro konfiguraci klasického sériového portu:

```
# mknod /dev/ttyS1 c 4 65
```

Pro konfiguraci sériového portu na USB převodníku:

```
# mknod /dev/ttyUSB1 c 188 1
```

7. Zkopírování zdrojových souborů regulátoru

(v jiném terminálu)

```
# sudo cp -r /media/cdrom/* /chroot/
```

8. Překlad a spuštění

(v chroot terminálu)

```
$ cd /src/regulator/
$ make
$ ./HAND_cmd.exe -h
```

Příloha 4: Návod k použití k programům HAND_cmd.exe a measure

Popis obou programů v rámci tohoto manuálu není vyčerpávající. Především se tento návod nevěnuje řešení případných problémů. Jedná se spíše o rychlý návod k ovládání zmíněných programů. V případě použití programů v chroot prostředí je nutné po vstupu do nového kořenového adresáře změnit cestu k domovskému adresáři a připojit /proc následujícím způsobem:

```
# export HOME="/root/"
# mount none /proc -t proc
```

HAND_cmd.exe

Spustitelný soubor regulátoru *HAND_cmd.exe* je program pro příkazovou řádku. Je ovládán pomocí parametrů na příkazové řádce. Detailnější popis parametrů se nachází v kapitole 3.2. Změna rychlosti prstů může vést ke značnému zvýšení chyby regulace a ke změně limitní síly určené hardwarovým limitem proudu. Může tedy dojít k poškození chapadla a uchopovaného předmětu. Program akceptuje tyto parametry (pořadí zadávaných parametrů nehraje roli):

`-heatup`	... zahřátí pohonu chapadla pro dosažení provozních parametrů. Provádí také vystavení prstů do výchozí polohy.
`-reset`	... reset řídicí jednotky chapadla.
`-f <value>`	... limitní síla [N].
`-pos <value>`	... cílová poloha prstů relativně vůči „domovské“ poloze [m].
`-clt <path>`	... cesta k datům kalibrační tabulky – výchozí 'clt0.001.clt'.
`-initstr <i>`	... inicializační řetězec m5api knihovny - výchozí: 'RS232:1,9600'.
`-h`	... tiskne nápovědu k programu.
`-calibrate`	... spustí generování kalibračních dat. Tento režim vyžaduje spustitelný soubor k programu pro ovládání siloměru *measure*.
`-v`	... rychlost pohybu prstů [m/s]. **NEBEZPEČNÝ PARAMETR – jeho změna může vést k poškození hardwaru.**

Po zapnutí chapadla je nutné uvést prsty do výchozí polohy a pak zahřát mechanickou část chapadla na provozní teplotu. Tyto úkony zajistí spuštění *HAND_cmd.exe s* parametrem *-heatup*. Bez provedení těchto operací po zapnutí chapadla není možné prsty pohybovat. Pro pohyb prstů s regulací síly stisku je nutné minimálně zadat cílovou polohu a sílu stisku prstů.

Measure

Program *measure* slouží ke komunikaci s použitým siloměrem. Oproti HAND_cmd.exe je velmi jednoduchý. Jeho jediný úkolem je čtení dat ze siloměru a jejich vypisování na standardní výstup. Program akceptuje jediný parametr a tím je cesta k souborovému rozhraní portu rozhraní RS-232. Při volání z *HAND_cmd.exe* je cesta k souborovému rozhraní programu *measure* zadána. Je napevno nastavena na */dev/ttyUSB1*. V případě, že je potřeba ji změnit, lze použít symlink na jiné souborové rozhraní. Generování kalibračních dat samo o sobě souvisí spíše se servisem chapadla a regulátoru – při běžném používání regulátoru by nemělo být potřeba. Příklad použití *measure*:

```
# ./measure /dev/ttyUSB1
```

Příloha 5: Návrh tříd regulátoru

Zápis tříd regulátoru tak, jak byly navrženy. Interní návrh regulátoru je objektový, přestože rozhraní regulátoru objektové není. Stejně tak není objektové rozhraní knihovny m5api. Objektová struktura regulátoru tedy navazuje z obou stran na neobjektová rozhraní. Proto není interní struktura regulátoru plně objektová. Kromě rozhraní celého regulátoru nemá objektové zapouzdření knihovna m5api. Následuje popis navržených tříd:

Třída reprezentující rozhraní CSV souboru. Obsahuje metody pro otevření, zpracování a čtení informací z CSV souboru. V implementaci je použita pro zpracování dat kalibrační tabulky.

```
class Config {

    private:
      std::vector<std::vector<std::string> > data;
      bool failure;
      std::string file_path;
      char sep;

    public:
      Config(std::string file, char separator = '|');
      std::string GetData(unsigned int X, unsigned int Y);
      unsigned int GetCSVHeight();
      unsigned int GetRowWidth(unsigned int row);
      bool Error();
      std::string GetDataByKey(std::string key, unsigned int col = 1 ,
      unsigned int key_pos = 0);
      void Write();
  };
```

Třída reprezentující kalibrační tabulku. Definuje rozhraní komponenty *Kalibrační tabulka*. Rozhraní obsahuje metody pro otevření a zpracování kalibračních dat a následné čtení těchto dat ostatními bloky komponentami regulátoru. Objekt kalibrační tabulky uchovává kalibrační data jako vnitřní stav.

```
struct CollisionCalibration {
    double offset; /* offset prediktoru */
    double static_level; /* limit proudu pro komparátor */
  };

class CalTab {

    private:
      std::map<double,double> force_to_cur_table; /* data *P
      CollisionCalibration coll_params; /* parametry pro detektor */

    public:
      CalTab(std::string filename);
      double GetCurForForce(double force); /* převod síly na proud */
      CollisionCalibration GetCollisionParams();
      Error();
  };
```

Blok regulátoru *Detektor kolize* je reprezentován následující třídou. Detektor má pouze dvě metody pro komunikaci s okolními bloky – metody pro kalibraci detektoru a metodu pro zpracování vstupního vzorku.

```
class CollisionDetector {

    private:
      FeatureExtractor *fe;
      Predictor *pred;
      Comparator *cmp;

    public:
      CollisionDetector();
      ~CollisionDetector();
      void SetCallibration(double predictor_offset, double
      comparator_max_level);
      double Process(double raw_val); /* zpracování jednoho vzorku  */
  };
```

Následující třída reprezentuje blok *Komparátor* obsažený v bloku *Detektor kolize*. Komparátor má za úkol vyhodnotit rozdílnost dvou průběhů signálu. Při vyhodnocování pracuje s historickými daty, které objekt této třídy uchovává jako interní stav. Třída definuje metody pro nastavení limitní úrovně komparátoru a zpracování vstupního vzorku přímo s ohodnocením tohoto vzorku. Navíc je možné číst interní spojité hodnocení komparátoru. Jakmile toto hodnocení překročí nastavenou limitní hladinu, komparátor vyhlásí rozdílnost průběhů.

```
class Comparator {

    private:
      double last_prediction;
      double last_raw_val;
      double last_raw_delta;
      double score_mixed;
      double max_level;
      double act_rank; /* hladina hodnocení, při které se průběhy liší */

    public:
      Comparator(double activation_rank = 0.1, double limit_level = 0.0);
      double Process(double val_predicted, double val); /* zpracování
      vzorku */
      void SetMaxLevel(double level);
      double GetRank(); /* čtení vnitřního ohodnocení */
  };
```

Třída Predictor reprezentuje blok *Prediktor*, který je součástí detektoru kolize. Úkolem bloku je předpovídat průběh signálu takový, jaký by byl beze změny stejnosměrné složky. Třída definuje metody pro vytvoření a inicializaci prediktoru, pro nastavení offsetu a pro zpracování vstupního vzorku. Prediktor umožňuje zesílením upravit amplitudu predikovaného signálu.

```
class Predictor {

    private:
      Filter_Der *filter;
      double amp;
      double off;
```

```
    bool collision_detected;
    double current_max;
    double tmp_curr_max;

  public:
    Predictor(double aplifier, double offset = 0 , double curr_max =
    0 );
    ~Predictor();
    double Process(double); /* krok predikce */
    void SetOffset(double offset); /* nastavení offsetu prediktoru */
};
```

Blok extrakce stejnosměrné složky reprezentuje třída FeatureExtractor. Objekt této třídy představuje vlastně dolnopropustní filtr signálu. Třída definuje metody pro inicializaci objektu a zpracování vstupního vzorku. Při inicializaci je nastavena délka úseku vzorků, nad kterým bude zpracování probíhat.

```
class FeatureExtractor {
  private:
    std::deque<double> tmpvals;
    int tmp_length; /* délka filtru */
    double min,max;
    int sample_counter;

  public:
    static bool octave_inicialized;
    double Process(double);
    FeatureExtractor(int length);
    ~FeatureExtractor();
};
```

Derivační filtr použitý v bloku *Prediktor* je reprezentován třídou Filter_Der. Třída definuje metodu pro vytvoření konkrétní instance filtru i s inicializací a metodu pro zpracování vzorku. Podobně jako u třídy FeatureExtractor i zde je jedním z parametrů filtru délka úseku, nad kterým filtr pracuje.

```
class Filter_Der {

  private:
    std::deque<double> tmpvals;
    int integral_length; /* celková délka filtru */

  public:
    /* konstruktor s délkou integrace */
    Filter_Der(int int_length, double default_value = 0.0);
    double Process(double input); /* metoda pro zpracování jednoho
    vzorku filtrem */
};
```

Třída ForceLimiter reprezentuje blok regulátoru *Omezovač síly stisku*. Objektu této třídy je v konstruktoru předána hodnota limitního proudu. Objekt čte vstup a převádí ho na vnitřní spojitý stav filtrací a dalším zpracováním. Při překročení limitního proudu vnitřním stavem tuto skutečnost signalizuje na výstupu.

```
class ForceLimiter {

    private:
      int sample_count; /* čítač vzorků */
      std::deque<double> tmpvals[6]; /* historické hodnoty */
      FeatureExtractor *fi_s[5];  /* filtry */
      double weights[6]; /* váhy pro filtry */
      double level_raw; /* hladina proudu, kterou hlídá */
      double final_result; /* proměnná pro akumulaci výstupní hodnoty */

    public:
      ForceLimiter(double level);
      ~ForceLimiter();
      double Process(double); /* zpracování vzorku */
      double GetRank(); /* čtení spojitého vnitřního stavu */
  };
```