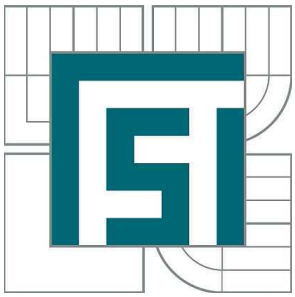


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ŘÍZENÍ POHYBŮ DVOURAMENNÉHO ŽONGLOVACÍHO STROJE

DOUBLE ARM JUGGLING MACHINE MOVEMENT COTROL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN JANEČEK
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ZDENĚK NĚMEC, CSc.
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2011/2012

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Jan Janeček

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Řízení pohybů dvouramenného žonglovacího stroje

v anglickém jazyce:

Double arm juggling machine movement cotrol

Stručná charakteristika problematiky úkolu:

Ukázka dynamiky pohonů na příkladě žonglovacího stroje. Nevržený a realizovaný stroj bude zařazen do soutěže, kterou vypsala firma Phonix Contact.
Ukol je rozdělen do dvou diplomovových prací, tato práce řeší vlastní řízení pohybu ramen podle zadávaných povelů z druhé kooperující úrovně řízení.

Cíle diplomové práce:

1. Dokonale se seznámit s přístrojovým a programovým vybavením, které bude použito pro realizaci žonglovacího stroje.
2. V součinnosti s druhým řešitelem úkolu navrhnout celkovou koncepci stroje, algoritmy činnosti a algoritmy řízení pohybu ramen stroje.
3. Zpracovat programové vybavení pro řízení dvou stejnosměrných motorů prostřednictvím daného programovatelného automatu, výkonové elektroniky, atd.
4. Ověřit funkčnost a analyzovat kvalitu a omezující faktory dynamiky pohonů s ohledem na potřeby žonglování s míčkem.

Seznam odborné literatury:

[1] Švarc, I.; Šeda, M.; Vítečková, M. Automatické řízení. Brno: CERM, 2007. 324 s. ISBN 978-80-214-3491-2.
[2] Noskievič, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999.
[3] Firemní materiály k programovatelným automatům fy Phoenix Contact.

Vedoucí diplomové práce: doc. Ing. Zdeněk Němec, CSc.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2011/2012.

V Brně, dne

L.S.

Ing. Jan Roupec, Ph.D.	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Ředitel ústavu	Děkan fakulty

ABSTRAKT

Tématem projektu, jehož součástí je tato diplomová práce, je vytvoření dvouramenného žonglovacího stroje. Projekt je rozdělen do dvou diplomových prací. Tato práce se zabývá řízením pohybů ramen žonglovacího stroje. V první části diplomové práce jsou informace o použitých hardwarových a softwarových prostředcích. V druhé části je vysvětlena činnost použitého regulačního obvodu, použitých komunikačních protokolů a samotná implementace celého řízení.

ABSTRACT

The diploma thesis is a part of the project Intelligent double arm juggling machine. The project is divided into two diploma thesis. This thesis deals with the motion control of juggling arms. Information about hardware and software resources used in this project is in the first part of thesis. There is explanation of the control loop, explanation of used communication protocols and implementation of program proceeding in the second part of thesis.

KLÍČOVÁ SLOVA

Žonglovací stroj, řízení DC motoru, PLC, programovatelný automat, servozesilovač, regulátor, regulace rychlosti, regulace polohy, regulace proudu, Phoenix Contact

KEYWORDS

Juggling machine, DC motor control, PLC, programmable logic controller, servo amplifier, controller, speed control, position control, current control Phoenix Contact

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem diplomovou práci *Řízení pohybů dvouramenného žonglovacího stroje* vypracoval samostatně pod vedením doc. Ing. Zdeňka Němce, CSc. s využitím materiálů uvedených v seznamu použité literatury.

Dne 25.5.2012 Jan Janeček

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce doc. Ing. Zdeňku Němcovi, CSc. za odborné vedení a poskytnutí cenných rad při řešení problémů souvisejících s vypracováním diplomového projektu.

Poděkování patří také Bc. Danielu Mertlíkovi, který se podílel na zpracování druhé poloviny projektu, díky jehož nemalému přispění se úspěšně podařilo zprovoznit žonglovací zařízení.

Jan Janeček

BIBLIOGRAFICKÁ CITACE

JANEČEK, J. *Řízení pohybů dvouramenného žonglovacího stroje.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012. 77 s. Vedoucí diplomové práce doc. Ing. Zdeněk Němec, CSc..

Obsah:

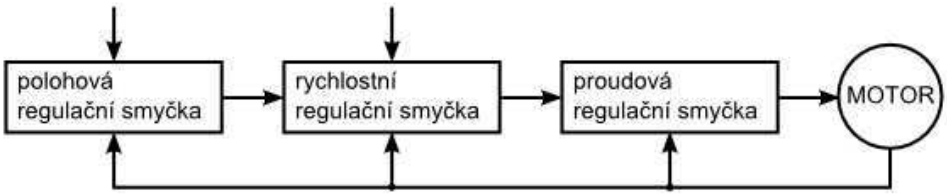
Zadání závěrečné práce....3
Abstrakt....5
Prohlášení o originalitě....7
1 Úvod....11
2 Rozbor problému....13
2.1 Popis základního triku....13
2.2 Prováděné triky....15
3 Použitá zařízení....17
3.1 Zdroje napájení....18
3.2 Programovatelný automat ILC 350ETH....19
3.3 Servozesilovač IB IL EC AR 48/10A....22
3.3.1 Operační režimy servozesilovače....22
3.3.2 Připojení servozesilovače....23
3.3.3 Parametrizace servozesilovače....25
3.3.4 Operační stavy servozesilovače....26
3.3.5 Funkční struktura....28
3.4 DC motor Transmotec....29
3.5 Inkrementální snímač HEDS....30
4 Použitý software....33
4.1 PC WORX....33
4.1.1 Základní ovládací prvky....33
4.1.2 Programové organizační jednotky....36
4.1.3 Typy úloh....37
4.1.4 Podporované datové typy....38
5 Komunikace....39
5.1 PCP komunikace....39
5.1.1 PCP_CONNECT....39
5.1.2 PCP_READ....39
5.1.3 PCP_WRITE....40
5.2 Ethernetová komunikace....40
5.2.1 IP_CONNECT....40
5.2.2 IP_URCV....40
5.2.3 IP_USEND....41
6 Řešení....43
6.1 Programy....43
6.1.1 Program Main....43
6.1.2 Program ETHSetting....46
6.1.3 Program Juggling....51
6.2 Popis regulací....58
6.2.1 Proudový regulační obvod....60
6.2.2 Rychlostní regulační obvod....63
6.2.3 Polohový regulační obvod....66
6.3 Poznatky z řešení....68
6.3.1 Vliv tření....68
6.3.2 Vliv vůle....69
6.3.3 Analýza dopadu míčku při triku....70
7 Závěr....73
Seznam použité literatury....75
Seznam příloh....77

1 ÚVOD

V dnešní době, kdy je stále více technických aplikací řízeno počítači resp. programovatelnými jednotkami, jako může být programovatelný automat, je jasné, že množství použitých řídicích systému se stále zvyšuje. Řídicí systém, jak ho dnes známe, se skládá z procesů a řídicích prostředků jako programovatelné automaty a regulátory. Nasazování automatických řídicích systémů práci nejen zkvalitňuje, ale zrychluje, dělá ji bezpečnější a příjemnější. Dnes je automatizace a použití automatických řídicích systémů nedílnou součástí lidského života. Bez řídicích systémů se neobejde téměř žádný výrobní proces v moderní době. Řídicí systémy ovšem nepronikly pouze do průmyslového prostředí, ale úspěšně jsou nasazovány v dalších odvětvích jako stavebnictví, péče o prostředí, rekreace atd.

Použití řídicího systému při rekreaci nebo zábavě je hlavní smysl projektu Inteligentní dvou ramenný žonglovací stroj, jehož součástí je tato diplomová práce. Žonglování je zdánlivě jednoduchá věc, která spočívá ve vyhození míčku a jeho opětovném chycení. Pro žonglujícího člověka je ovšem velice snadné pohybovat končetinami v libovolném směru, aby správně zachytil padající míč. Při této schopnosti není obtížné zachytit míč i při nepovedeném vyhození. Schopnosti rotačního stroje ovšem takovou škálou pohybů nedisponují. V možnostech rotačního stroje je pouze rotační pohyb, proto realizace žonglování na takovémto stroji je značně komplikovaná. Pro vyhození míčku je zapotřebí roztočit rameno stroje a následným brzděním vyhodit míček po tečně rotačního pohybu. Chycení míčku je mnohem snadnější. Stačí natočit rameno na místo dopadu. Dopad míčku do přesného místa ovšem klade značné nároky na chod celého stroje. Složitost celého projektu souvisí právě s dynamickými nároky, s kritickým časovým vyhodnocováním a s mechanickými vlivy.

Vzhledem k náročnosti projektu je řešení rozděleno do dvou na sebe navazujících částí. Každá z částí projektu je řešena jedním diplomantem. Tato práce řeší vlastní řízení pohybu ramen žonglovacího stroje na základě povelů z druhé úrovně řízení. Hlavní cíle pro úspěšnou realizaci části projektu, který naleží právě této práci, jsou správná obsluha komunikační vazby mezi oběma úrovněmi řízení a implementace řídícího algoritmu společně s nastavením výkonového členu. Výkonový člen je hlavní řídicí jednotka, která obsahuje uspořádání regulačních smyček podle obrázku 1.1 a všechny smyčky jsou při řešení projektu využity.



Obrázek 1.1 Uspořádání regulačních smyček ve výkonovém členu.

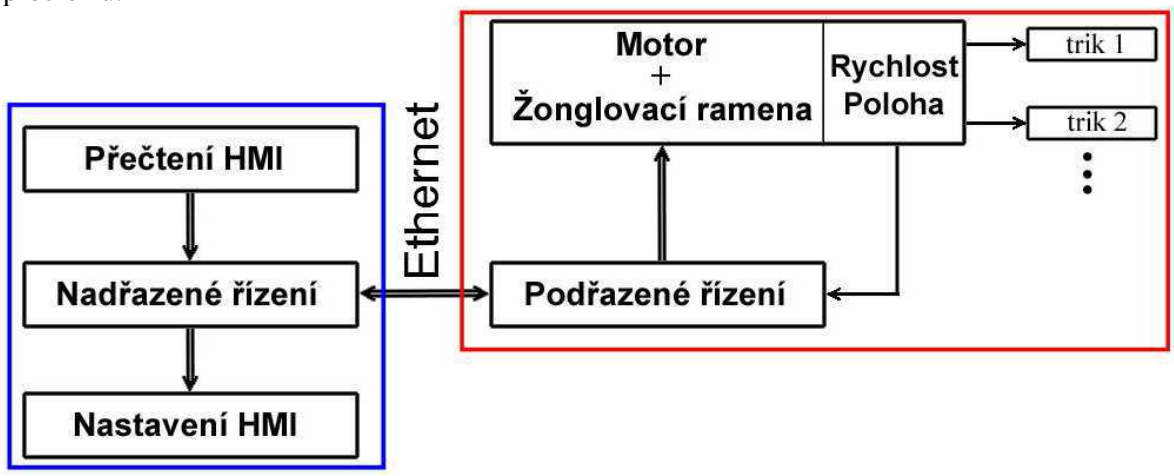
Cílem projektu je realizace žonglovacího stroje pomocí hardwarových a softwarových prostředků od firmy Phoenix Contact. Projekt se účastnil soutěže xplore New Automation Award 2012, kterou pořádá právě firma Phoenix Contact.

2 ROZBOR PROBLÉMU

Existují technické aplikace, které nemusí mít výhradně průmyslové uplatnění, ale mohou například sloužit pro pobavení. Přesto mohou být tyto aplikace velice náročné. Náročnost může spočívat jak v technické realizaci, tak ve složitosti řízení a ovládání daného zařízení. Příkladem takové složité úlohy je právě realizace žonglovacího stroje.

Žonglování je realizováno pomocí vyhazování míčku jedním ramenem a chytání míčku stejným nebo druhým ramenem. Pro realizaci hodu a následného chycení je nutné mít na obou ramenech připevněné košíčky. Ramena jsou přimontována k motoru, který s nimi vytváří rotační pohyb. Díky tomuto rotačnímu pohybu a následnému brzdění je realizováno vyhození míčku. Chycení míčku nastává v určené poloze při zastaveném ramenu. Jednotlivé sekvence házení a chytání míčku jsou nazvány triky.

Celý projekt je zpracován jako dvě diplomové práce. Jedna práce se zabývá konstrukčním řešením zařízení, která je třeba vyrobit, nadřazeným řízením (ovládání celého zařízení, vizualizace a komunikace s uživatelem) a vzájemnou komunikací použitých programovatelných automatů. Tato práce realizuje samotné řízení pohonu pro zajištění jednotlivých triků. Pohyby motoru jsou ovládány díky informacím, které jsou přijaty z nadřazeného řídícího systému. Pro zpracování obou částí jsou tedy použity dva programovatelné automaty. Tím je zajištěna separace obou dílčích problémů.



Obrázek 2.1 Schéma řízení celého stroje.

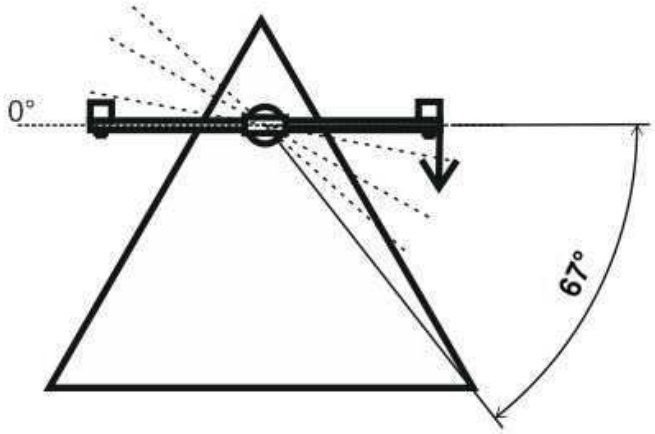
Podrobnější rozdělení celého projektu je zobrazeno na obrázku 2.1. Modrá oblast se týká přidružené diplomové práce a má na starosti nadřazené řízení. Červená oblast je téma této práce a realizuje pohyb ramen a vytváří triky.

Systém je navržen tak, že programovatelný automat má pomocí lokální sběrnice INTERBUS připojen výkonový člen – servozesilovač. Tento prvek v sobě má výkonovou elektroniku, která je schopna ovládat motor podle zadaných parametrů. Provádí změny napájecího proudu a napětí a tím vytváří žádané otáčky a sílu motoru. Servozesilovač je tedy klíčovým prvkem pro realizaci celého projektu.

2.1 Popis základního triku

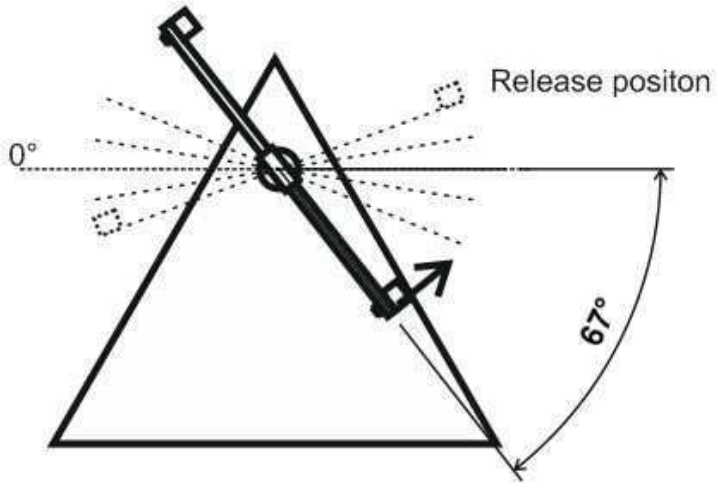
Každý trik se skládá ze základního triku a jeho modifikací. Základní trik spočívá ve vyhození míčku požadovanou rychlostí z požadovaného úhlu. Požadovaná rychlost vyhození a požadovaný úhel jsou hlavními parametry. Těmito dvěma parametry lze určit jak výšku letu míčku tak místo dopadu. K dokončení základního triku je zapotřebí také míček chytit, což je realizováno zastavením ramene s košíčkem na místě dopadu.

Základní trik je dále rozdělen do sekvence různých úkonů. Prvním z nich je vystřelení. Aby bylo možné míček vystřelit z požadovaného úhlu potřebuje rameno dostatečně velký úhel pro rozběh na požadovanou rychlost. Úplně první činností tedy je nastavení ramene na počáteční úhel. Pro nastavení ramene do této polohy je použita polohová regulace. Tuto skutečnost ilustruje obrázek 2.2. Hodnota počátečního úhlu byla experimentálně zjištěna pro zajištění maximálního rozběhu a zároveň dostatečně svislé polohy, aby nevypadl míček před rozběhem.



Obrázek 2.2 Nastavení ramene na počáteční úhel základního triku.

Další činností, po nastavení ramene do počáteční polohy, je výstřel míčku. Pro vystřelení míčku je zapotřebí dosáhnout konstantních otáček. Na tuto činnost se mnohem lépe hodí rychlostní regulace, je tedy nezbytné do této regulace přepnout. Po přepnutí do rychlostní regulace už následuje samotné vystřelení míčku. To je realizováno roztočením ramene na požadované konstantní otáčky a po dosažení požadovaného úhlu intenzivním brzděním, což zajistí vyhození míčku. Roztáčení ramene je ilustrováno na obrázku 2.3.



Obrázek 2.3 Vystřelení míčku požadovanou rychlostí z požadovaného úhlu.

Pro chytání míčku je mnohem vhodnější polohová regulace místo rychlostní. Během letu míčku je tedy nutné opět přepnout do polohové regulace, aby bylo možné chytit míček. Chytání míčku probíhá nastavením ramene do polohy dopadu míčku (obvykle 0°). Po dopadu míčku je základní trik ukončen.

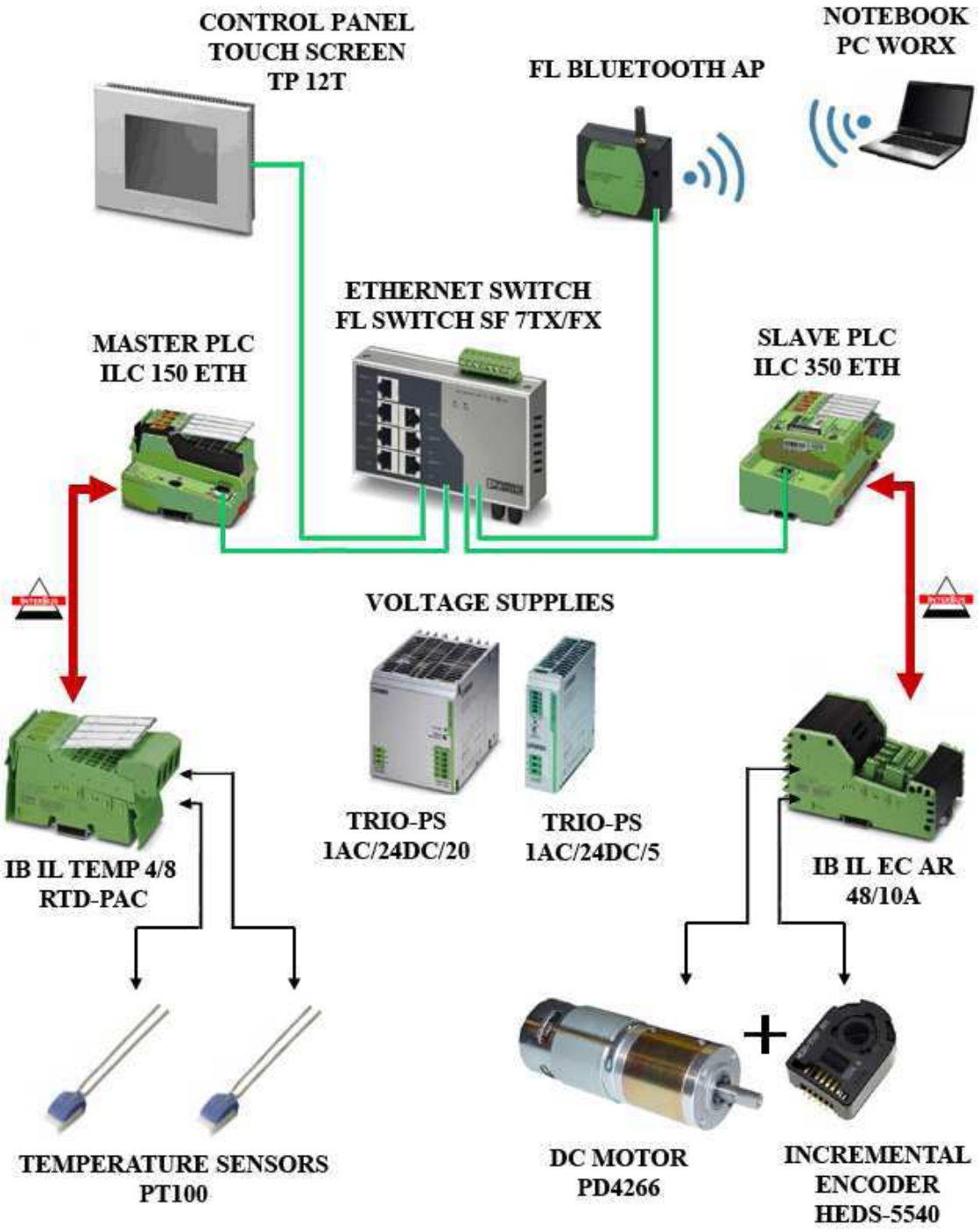
2.2 Prováděné triky

Modifikace základního triku je využita v každé z možných házecích funkcí. Zařízení umožňuje několik funkcí resp. triků:

- **Jeden hod** – jedná se o funkci, během které je míček jednou vyhozen. Je možné zadat libovolné hodnoty odhozového úhlu a rychlosti. Tyto hodnoty lze zadat libovolně bez vnitřních výpočtů a provázání obou hodnot, proto není zaručeno chycení míčku. Pro realizaci tohoto triku je použita smyčka základního triku bez modifikací. Pomocí tohoto triku lze provést svislý hod (vyhození míčku a chycení míčku pouze jedním ramenem). Provedení svislého hodu spočívá pouze v nastavení nulového odhozového úhlu a libovolné rychlosti. Míček je vyhozen pouze ve směru svislé osy, proto dopadne do stejného košíčku, ze kterého byl vyhozen.
- **Přehození s nastavitelnou výškou** – u tohoto triku je jediným zadávaným parametrem maximální výška letu míčku. Z této hodnoty je pomocí fyzikálního principu dopočítána hodnota odhozové rychlosti a odhozového úhlu, které jsou nezbytné pro provedení základního triku. Tento výpočet probíhá v nadřazeném systému řízení a automatu, který ovládá motor, jsou zaslány pouze hodnoty odhozového úhlu a rychlosti. Pro realizaci triku je opět použita pouze smyčka základního triku s tím rozdílem, že dosazované hodnoty odhozového úhlu a rychlosti jsou pomocí výpočtu provázány a spočítány tak aby byl zaručen dopad do košíčku na druhém rameni.
- **Žonglování** – jedná se o automatický režim, při kterém probíhá neustálé přehazování z jedné strany na druhou a opačně. Pro realizaci triku je použita smyčka základního triku s přednastavenými hodnotami odhozových parametrů. Po provedení každého přehozu jsou hodnoty upraveny pro hod opačným směrem. Tento postup se provádí dokud není trik ukončen uživatelem. Žonglovaní navíc obsahuje další drobně modifikovaný režim žonglování s krokováním. Žonglování s krokováním funguje stejně jako žonglování, ale je navíc opatřeno přerušeními. Při provádění triku dojde na každém bodu přerušení k zastavení vykonávání triku a čeká se na povel od uživatele. Při stisku tlačítka uživatelem je navázáno na provádění přerušeného triku a vykoná se další část triku.

3 POUŽITÁ ZAŘÍZENÍ

Vzhledem k tomu, že byl projekt přihlášen do soutěže xplore New Automation Award 2012 uspořádané firmou Phoenix Contact, byly v rozvaděči použity prvky výhradně od této firmy. Na obrázku 3.1 je vidět kompletní schéma použitých zařízení pro celý projekt. Informace uvedené v této kapitole jsou čerpány z firemních dokumentů. Použité údaje byly přeloženy z anglických originálů.



Obrázek 3.1 Schéma použitých zařízení.

Této diplomové práce se týkají pouze některé ze zobrazených zařízení:

Zařízení	Označení
Programovatelný automat	ILC 350ETH
Servozesilovač	IB IL EC AR 48/10A
DC motor	PD4266
Inkrementální enkodér	HEDS-5540

Tabulka 3.1 Seznam použitých zařízení.

Ve schématu jsou zobrazena spojení jednotlivých zařízení pomocí různě barevných čar. Každá barva reprezentuje jiný typ připojení:

Barva	Typ připojení
zelená	Ethernet
červená	INTERBUS
černá	silové vedení, vedení enkodéru, vedení čidel
modrá	bezdrátové Bluetooth

Tabulka 3.2 Typy použitých spojení.

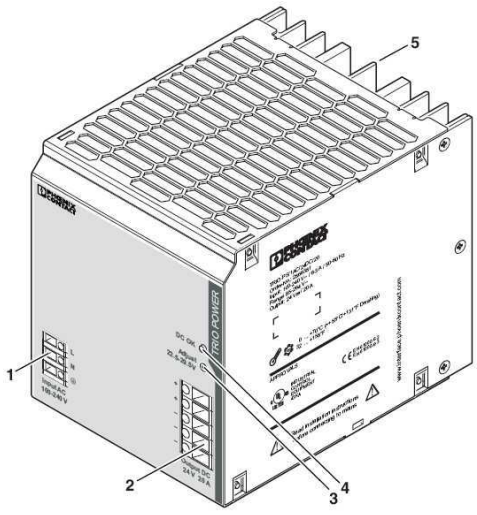
3.1 Zdroje napájení

Pro napájení zařízení jsou použity dva zdroje, výkonnější TRIO-PS 1AC/24DC/20 pouze pro servozesilovač a méně výkonný TRIO-PS 1AC/24DC/5 pro programovatelný automat. Údaje o zdrojích jsou čerpány z firemních materiálů [1] a [2].

TRIO-PS 1AC/24DC/20 je 24 voltový zdroj s maximálním proudovým výstupem 20 ampér. Nejdůležitější parametry zdroje jsou uvedeny v tabulce 3.3.

Jmenovité vstupní napětí	100V AC … 240V AC
Rozsah vstupního napětí	85V AC … 264V AC
Jmenovité výstupní napětí	24V DC ±1%
Nastavitelný rozsah výstupního napětí	22,5V DC … 29,5V DC
Výstupní proud	20A (0℃ … 55℃)

Tabulka 3.3 – Parametry zdroje TRIO-PS 1AC/24DC/20.



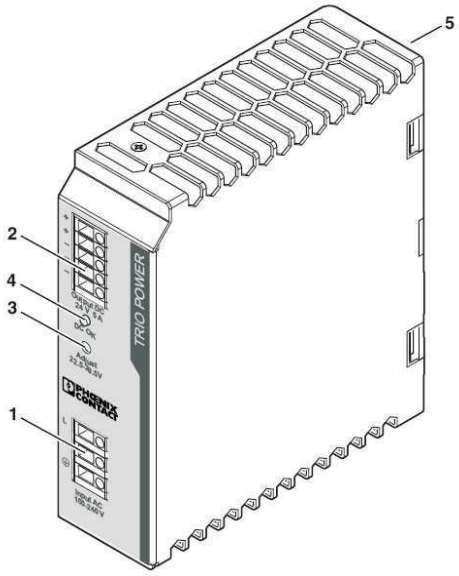
1	AC vstup
2	DC výstup
3	Potenciometr: 22,5V DC – 29,5V DC
4	„DC OK“ LED
5	Úchyt na DIN lištu

Obrázek 3.2 Zdroj TRIO-PS 1AC/24DC/20.

TRIO-PS 1AC/24DC/5 je 24 voltový zdroj s maximálním proudovým výstupem 5 ampér. Nejdůležitější parametry zdroje jsou v tabulce 3.4.

Jmenovité vstupní napětí	100V AC … 240V AC
Rozsah vstupního napětí	85V AC … 264V AC
Jmenovité výstupní napětí	24V DC ±1%
Nastavitelný rozsah výstupního napětí	22,5V DC … 29,5V DC
Výstupní proud	5A (-25℃ … 55℃)

Tabulka 3.4 Parametry zdroje TRIO-PS 1AC/24DC/5.



1	AC vstup
2	DC výstup
3	Potenciometr: 22,5V DC – 29,5V DC
4	„DC OK“ LED
5	Úchyt na DIN lištu

Obrázek 3.3 Zdroj TRIO-PS 1AC/24DC/5.

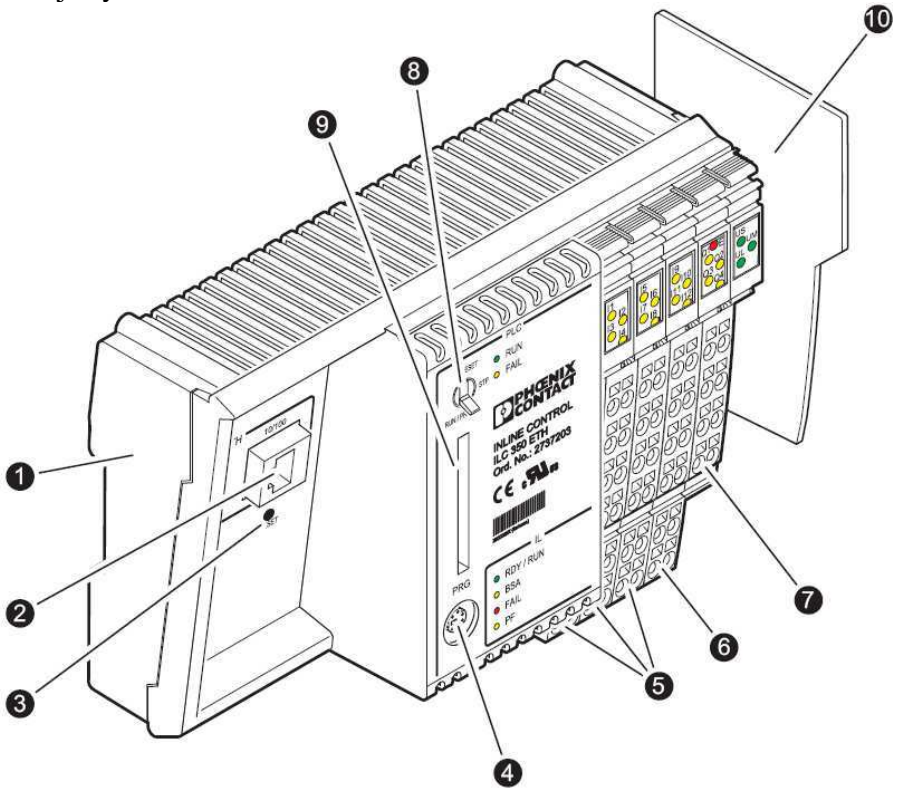
3.2 Programovatelný automat ILC 350ETH

Programovatelný automat ILC 350ETH je kompaktní automat s integrovaným Ethernetovým rozhraním a INTERBUS rozhraním. Ethernetové rozhraní je realizováno klasickým krouceným kabelem a slouží k zpřístupnění zařízení pomocí Ethernetu a TCP/IP. INTERBUS rozhraní slouží k připojení vstupně/výstupních zařízení k programovatelnému automatu. Údaje o programovatelném automatu jsou čerpány z firemního materiálu [3].



Obrázek 3.4 Programovatelný automat ILC 350ETH.

Automat obsahuje ovládací a připojovací prvky. Tyto prvky jsou zobrazeny na obrázku 3.5 a jejich význam je vysvětlen v tabulce 3.5.



Obrázek 3.5 Ovládací a připojovací prvky automatu.

1	Tělo automatu
2	Ethernetové rozhraní
3	Tlačítko reset
4	RS-232 rozhraní
5	Konektory 1 až 3: vstupní svorky
6	Konektor 4: výstupní svorky
7	Konektor 5: napájecí svorky
8	Přepínač pro výběr režimu automatu
9	Zdířka pro pamětovou kartu
10	Bočnice

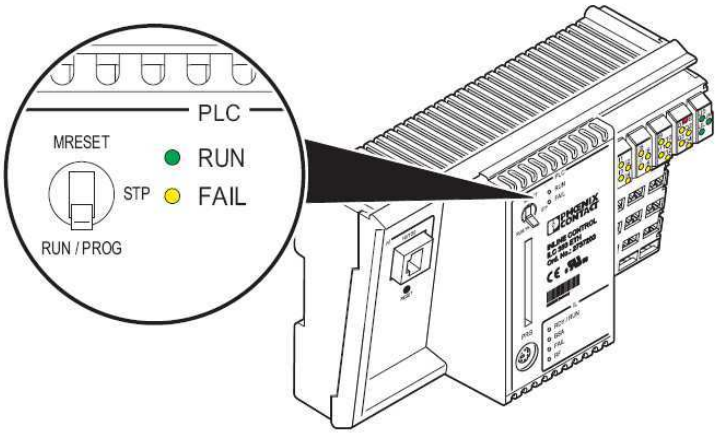
Tabulka 3.5 Vysvětlivky ovládacích a připojovacích prvků automatu.

Na obrázku 3.5 (odkaz 8) je zobrazen přepínač pro výběr režimu automatu. Tento přepínač slouží k výběru provozního stavu automatu a programu. Přepínač lze přepnout do 3 různých stavů:

- RUN/PROG – program nahraný v automatu běží. Lze provádět změny v programu a konfiguraci. Lze také využít možnosti monitorování stavu proměnných a programu.
- STP – program nahraný v automatu neběží, automat je v režimu STOP.
- MRESET – data a program nahraný v automatu jsou vymazány

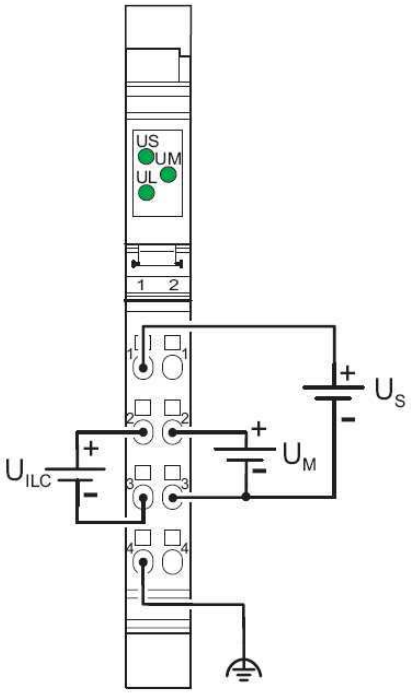
Přepínač v režimu RUN/PROG a STP má aretační funkci. Při přepnutí do těchto stavů, zůstane přepínač na svém místě a automat v požadovaném stavu.

Při přepnutí přepínače do režimu MRESET jsou resetována data a program. Po restartu se přepínač a automat přepne do režimu STP automaticky.



Obrázek 3.6 Přepínač módů automatu.

Na obrázku 3.5 (odkaz 7) jsou zobrazeny napájecí svorky automatu. Pro napájení automatu se používá 24 voltů stejnosměrných. Přípustný rozsah napájecího napětí je od 20,4V DC do 30V DC. Typická spotřeba automatu bez připojeného dalšího vstupně/výstupního zařízení ke sběrnici INTERBUS je 4,8W pro napájecí napětí 24V.



Obrázek 3.7 Napájecí svorkovnice automatu.

Pro napájení automatu slouží 3 druhy napájecích napětí, všechny se shodnou velikostí napájecího napětí 24V:

- U_S – 24 voltové napájení segmentu
- U_{ILC} – 24 voltové napájení pro generování 7,5 voltu komunikačního napájení pro automat a zařízení připojené na lokální sběrnici INTERBUS.
- U_M – 24 voltové hlavní napájení

3.3 Servozesilovač IB IL EC AR 48/10A

Servozesilovač je univerzální rychlostní nebo polohový regulátor s výstupem pro permanentně buzené stejnosměrné motory s kartáčky (DC motor) nebo bez kartáčků (EC motor). Toto zařízení má v sobě přímo implementovány vstupy pro inkrementální snímače. Pro řízení motoru tedy není třepa připojovat k programovatelnému automatu další zařízení pro snímání natočení rotoru. Technické údaje o servozesilovači jsoučerpány z technické dokumentace [4].



Obrázek 3.8 Servozesilovač IB IL EC AR 48/10A.

Jedná se o univerzální zařízení pro připojení širokého spektra motorů. Tomu odpovídá i rozsah výstupních elektrických veličin. Servozesilovač je schopen dodávat do připojeného motoru až 10A při napětí v rozsahu 12V až 48V. Maximální spotřeba energie může dosáhnout až 450W.

Zařízení disponuje IxR kompenzací, která vyrovnává rychlostní odchylky způsobené měnícím se zatížením. Dalším nezbytným vybavením zařízení jsou bezpečnostní ochrany. Servozesilovač obsahuje ochranu proti: proudovému přetížení, napěťovému přetížení, přehřátí servozesilovače, zkratu na vodičích motoru a zkratu na napájecích vodičích. Při zjištění některé z těchto zásadních poruch je okamžitě vypnuto výstupní napájení.

3.3.1 Operační režimy servozesilovače

Podle zvoleného operačního režimu servozesilovače se změní celkové chování zařízení a pro každý režim se uvede v činnost jiný typ regulátoru. Servozesilovač obsahuje tyto režimy:

- **Polohový režim** – pokud je v činnosti tento režim, stane se servozesilovač polohovým řídícím systémem schopným otáčet osou motoru z místa na požadované místo. Pokud servozesilovač pracuje v polohovém režimu, jsou v činnosti 3 regulátory: polohový regulátor, rychlostní regulátor a proudový regulátor. Všechny 3 regulátory jsou kaskádovitě uspořádány ve zmíněném pořadí. Nejdůležitějším regulátorem při aktivním polohovém režimu je polohový regulátor. Je realizován pomocí číslicového PID regulátoru a je hardwarově implementován v servozesilovači, proto běží s velice malou periodou 1ms. Pro funkci regulátoru je třeba nastavit žádanou hodnotu polohy, regulátor si pomocí informace o aktuální poloze z inkrementálního snímače zjistí regulační odchylku a provede potřebný zásah.
- **Rychlostní režim 1** – při tomto režimu se servozesilovač stane systémem schopným otáčet osou motoru požadovanou rychlostí. Jsou využívány dva regulátory: rychlostní regulátor a proudový regulátor. Regulátory jsou opět kaskádovitě seřazeny v uvedeném pořadí. Využívaný rychlostní regulátor v tomto režimu není identický s tím, který je využíván v polohovém režimu a pracuje s periodou 500μs. Pro zjišťování aktuální rychlosti otáčení pro výpočet regulační odchylky lze využít připojeného inkrementálního snímače, pomocí něhož jsou snímány dvě po sobě jdoucí polohy a následně vypočítána aktuální rychlost otáčení. Další možností je snímat přímo napájecí napětí motoru. Hodnota napětí přímo odpovídá rychlosti otáčení motoru. Výhodou

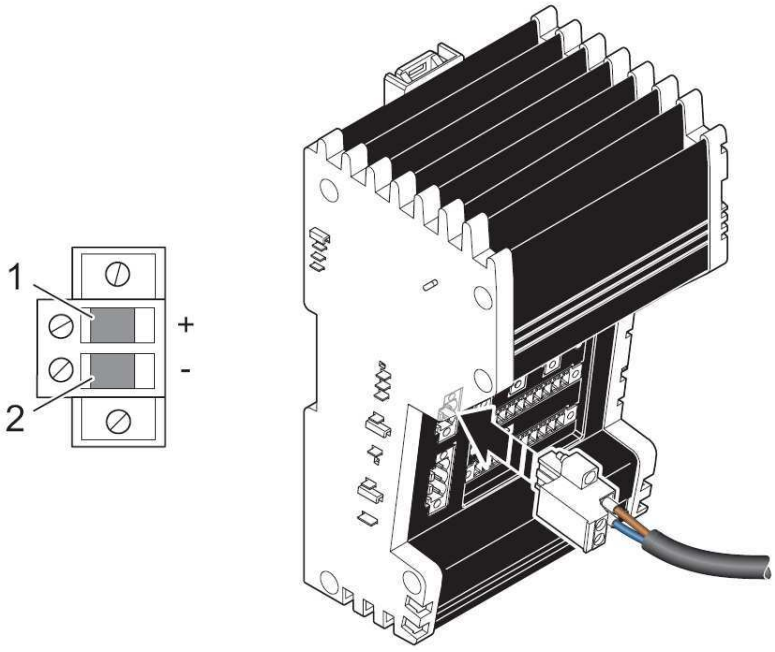
tohoto snímání rychlosti je, že není třeba využívat žádný snímač polohy osy motoru. Nevýhodou zjišťování aktuální rychlosti pomocí napájecího napětí je vliv zatížení. Při stejném napájecím napětí se při zvyšujícím se zatížení snižuje rychlost otáčení. Pro odstranění tohoto jevu lze zapnout IxR kompenzaci, která na základě změn proudu díky změně zatížení, upravuje hodnotu napájecího napětí pro optimalizaci rychlosti.

- **Rychlostní režim 2** – Jedná se o podfunkci polohového režimu a do činnosti jsou zapojeny stejné regulátory. V tomto režimu jsou neustále automaticky generovány nové žádané polohy pro polohový regulátor za účelem dosažení požadované konstantní rychlosti. Výhodou tohoto režimu je, že využívá stejných parametrů jako polohový režim, proto není nutné konfigurovat rychlostní režim 1. Nejčastější využití tohoto režimu je v aplikacích orientovaných na polohové řízení.
- **Režim Homing** – Další režim, který je podfunkcí polohového režimu. Slouží k přiřazení obvykle nulové polohy určitému natočení motoru.
- **Režim kroutícího momentu** – Tento režim uvádí v činnost pouze proudový regulátor. Slouží k vyvolání a udržování konstantního kroutícího momentu motoru. To lze využít v aplikacích vyžadujících vytváření konstantní síly.

3.3.2 Připojení servozesilovače

Servozesilovač disponuje konektory pro připojení napájecího napětí, stejnosměrného motoru, 2 inkrementálních snímačů a 3 polohových dorazů.

- **připojení napájecího napětí**

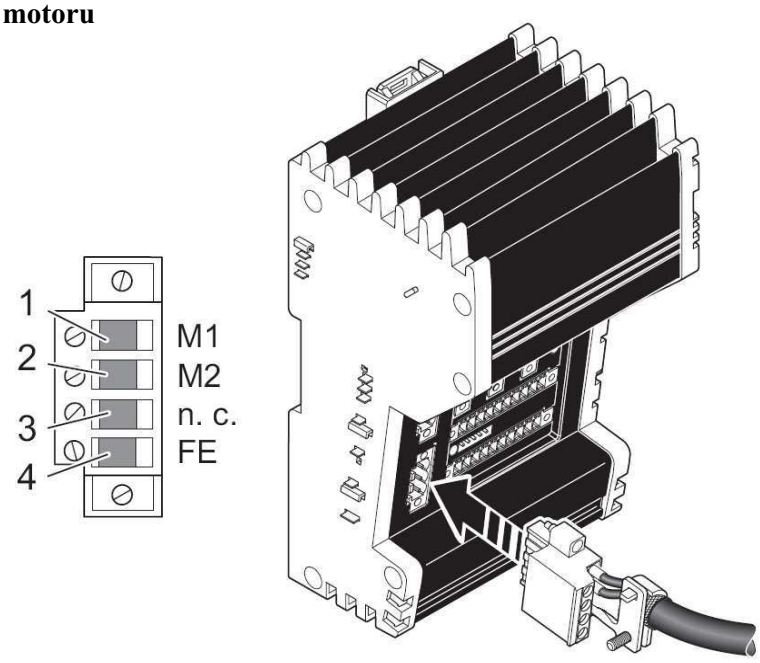


Obrázek 3.9 Připojení napájecího napětí k servozesilovači.

Zdířka	Význam
1	Us +
2	Us –

Tabulka 3.6 Popis napájecího konektoru servozesilovače.

- **připojení motoru**

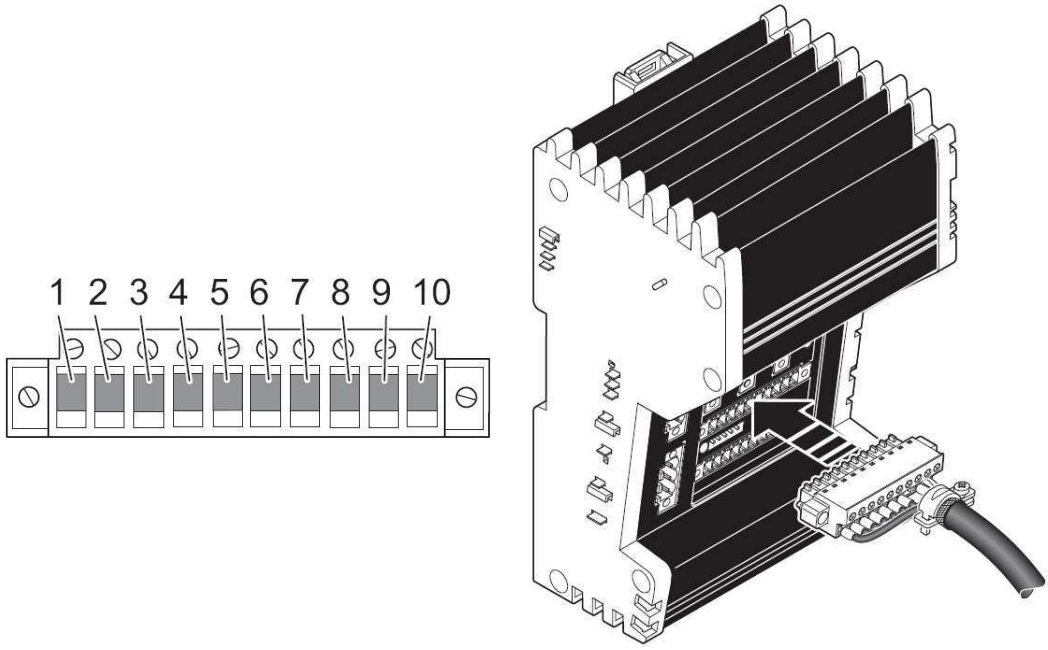


Obrázek 3.10 Připojení motoru k servozesilovači.

Zdířka	Význam
1	M1 (Motor +)
2	M2 (Motor –)
3	Nepoužito
4	Uzemnění (FE)

Tabulka 3.7 Popis konektoru pro připojení motoru k servozesilovači.

- **připojení inkrementálního snímače**



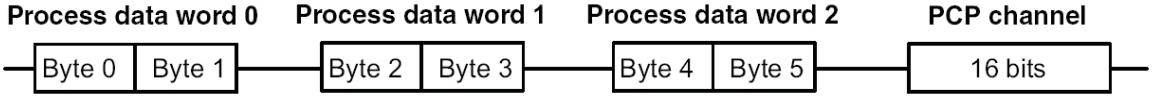
Obrázek 3.11 Připojení inkrementálního snímače k servozesilovači.

Zdířka	Význam
1	A
2	negace A
3	B
4	negace B
5	Z
6	negace Z
7	GND
8	+5V
9	+24V
10	Uzemnění (FE)

Tabulka 3.8 Popis konektoru pro připojení inkrementálního snímače k servozesilovači.

3.3.3 Parametrizace servozesilovače

Pro ovládání a konfiguraci servozesilovače se využívá sběrnice INTERBUS, kterou je servozesilovač připojen k programovatelnému automatu. Sběrnice INTERBUS pracuje na principu kruhové topologie. V tomto kruhu má servozesilovač rezervováno 16 bitů pro PCP kanál a 48 bitů pro každý z datových směrů. Pro oba komunikační směry platí stejné procesní schéma zobrazené na obrázku 3.12. Procesní data jsou zasílána v každém cyklu kruhu. Naopak PCP komunikace funguje na vyžádání – data jsou zaslána nebo přijata pouze při požadavku. PCP komunikace slouží zpravidla pro inicializaci parametrů servozesilovače, které se v průběhu činnosti systému dále nemění.



Obrázek 3.12 Procesní data pro oba komunikační směry.

Každé z procesních slov lze nakonfigurovat pro nastavování a odečítání různých parametrů servozesilovače.

Přednastavené hodnoty vstupních procesních dat jsou:
- process data word 0 = Status word (stav servozesilovače)
- process data word 1 = Speed actual value (aktuální hodnota rychlosti otáčení)
- process data word 2 = nevyužito

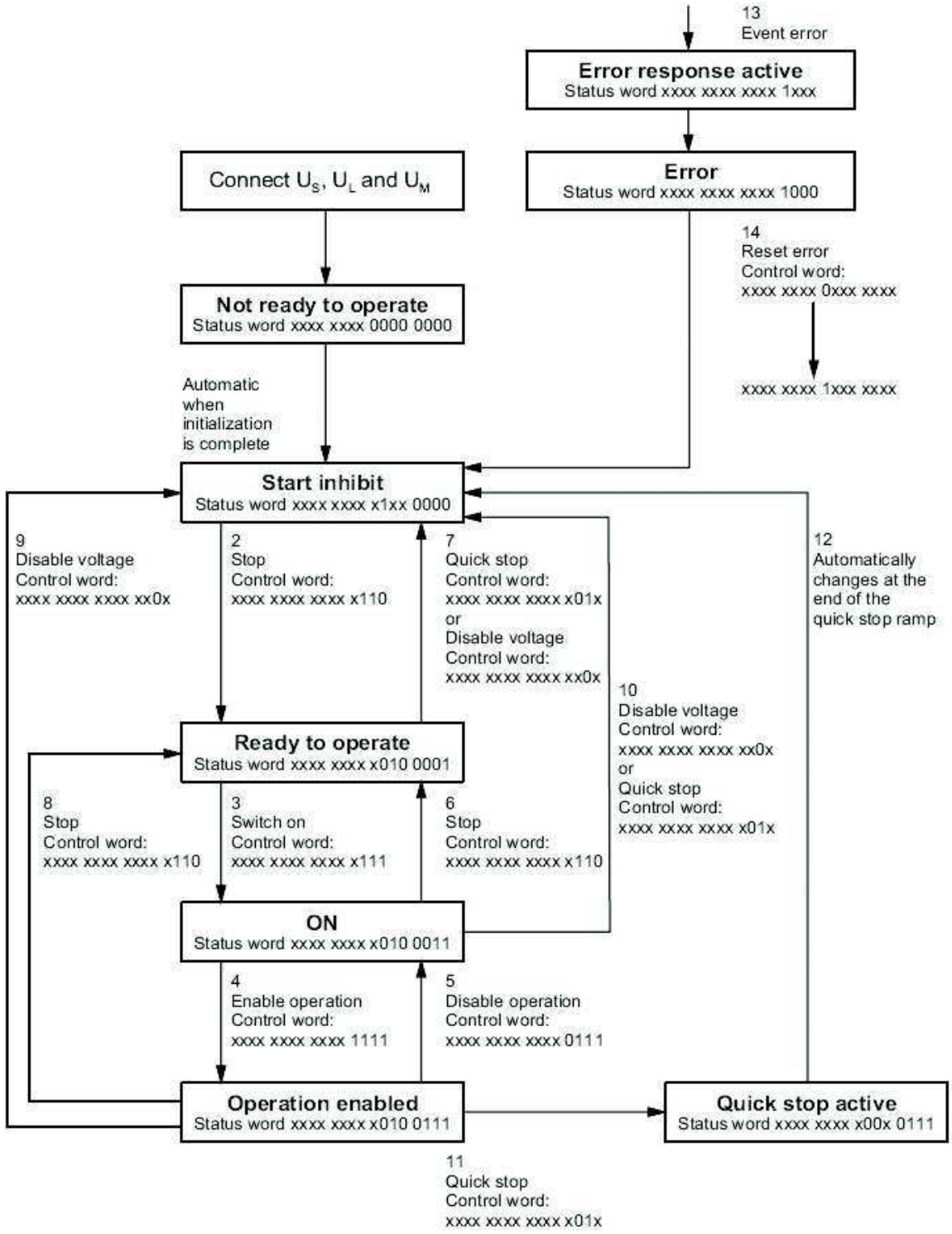
Pro výstupní procesní data jsou přednastavené hodnoty:
- process data word 0 = Control word (řízení servozesilovače)
- process data word 1 = Speed setpoint (žádaná hodnota rychlosti otáčení)
- process data word 2 = nevyužito

Jak již bylo řečeno pro parametrizaci servozesilovače se často používá PCP komunikace. Servozesilovač podporuje několik PCP služeb. Pro parametrizaci jsou nejdůležitější:

- **„Initiate“ (connect)** – slouží k navázání komunikace mezi automatem a servozesilovačem. Navázání komunikace vždy vyvolává automat a při úspěšném navázání komunikace odpoví servozesilovač kladným potvrzením.
- **„Read“** – tato služba poskytuje programovatelnému automatu přístup ke všem parametrům servozesilovače s povoleným přístupem pro čtení.
- **„Write“** – tato služba poskytuje programovatelnému automatu možnost zápisu do všech parametrů servozesilovače s povolením zápisu. Pokud se automat snaží zapsat do nesprávného parametru, servozesilovač vytvoří chybovou hlášku s podrobnými informacemi o způsobené chybě.

3.3.4 Operační stavy servozesilovače

Operační stavy servozesilovače odpovídají specifickým vnitřním a vnějším vlivům. Pro změnu stavů zařízení slouží procesní data a vnitřní události. Pro změnu operačního stavu servozesilovače se používá řídící slovo (Control word). Operační stavy servozesilovače je nutné projít v přesné sekvenci, jak je zobrazeno na obrázku 3.13. Přechod mezi jednotlivými stavy je realizován nastavením odpovídajících bitů v řídicím slově a informace o aktuálním stavu jsou neustále zobrazovány ve stavovém slově (Status word).



Obrázek 3.13 Diagram operačních stavů servozesilovače.

Každý ze stavů servozesilovače má různé unikátní vlastnosti a jednotlivým stavům servozesilovače odpovídají přesně nastavené bity ve stavovém slově. Nastavení těchto bitů je zobrazeno v tabulce 3.9 a vlastnosti jsou popsány níže.

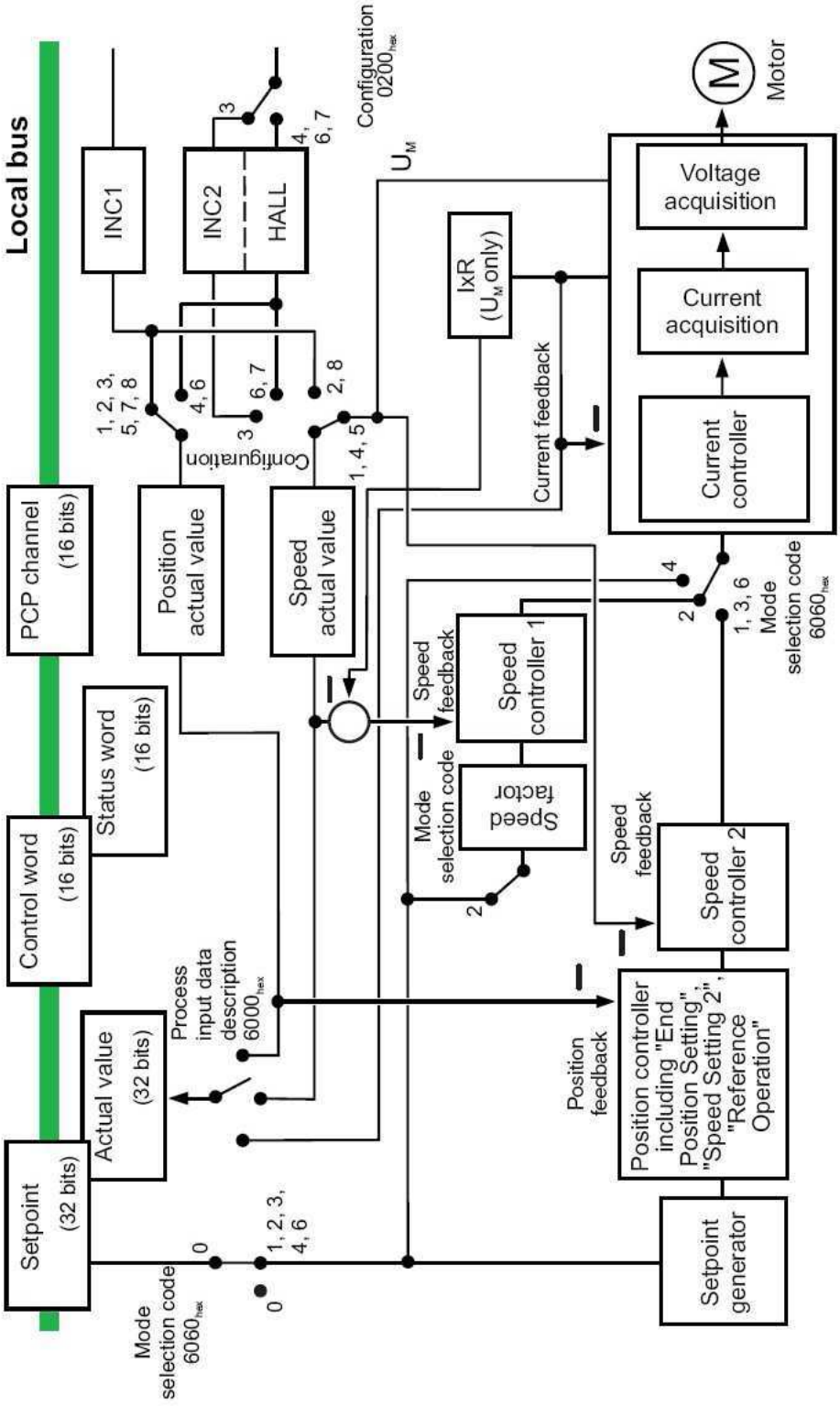
Stav	Bit 6	Bit 5	Bit 3	Bit 2	Bit 1	Bit 0
	Start inhibit	Quick stop	Error	Operation enabled	ON	Ready to operate
Not ready to operate	0	0	0	0	0	0
Start inhibit	1	X	0	0	0	0
Ready to operate	0	1	0	0	0	1
ON	0	1	0	0	1	1
Operation enabled	0	1	0	1	1	1
Error	0	X	1	0	0	0
Error response active	0	X	1	1	1	1
Quick stop active	0	0	0	1	1	1

Tabulka 3.9 Indikace stavů servozesilovače ve Status word.

- **Not ready to operate** – je stav, ve kterém byl servozesilovač právě připojen k napájecímu napětí. Servozesilovač není schopen přijímat příkazy ze sběrnice. Během tohoto stavu běží test zařízení a inicializace, je odpojeno napájení motoru a vypnuty řídicí funkce. Po skončení inicializačních operací přechází servozesilovač automaticky do stavu Start inhibit.
- **Start inhibit** – v tomto stavu jsou ukončeny softwarové a hardwarové inicializační operace, servozesilovač je nakonfigurován s výchozími hodnotami, je umožněno nastavovat servozesilovač a je vypnuto napájení a řídicí funkce.
- **Ready to operate** – v tomto stavu lze provádět nastavování servozesilovače, řídicí funkce jsou vypnuty a napájení je zapnuto.
- **ON** – v tomto stavu lze provádět nastavování servozesilovače, řídicí funkce a napájení jsou vypnuty.
- **Operation enabled** – servozesilovač je v operačním stavu. Jsou aktivovány řídicí funkce a napájení, servozesilovač může být nastavován a je možné nastavit žádanou hodnotu pro uvedení motoru do pohybu.
- **Quick stop active** – tento stav je vyvolán „quick stop“ příkazem, motor je brzděn po quick stop rampě. Po dokončení brzdění přechází servozesilovač automaticky do stavu Start inhibit.
- **Error response active** – při detekování chyby je aktivován tento stav. Dojde k odpojení napájení, rozsvícení „ERR“ diody a nastavení „Error“ bitu ve stavovém slově.
- **Error** – v tomto stavu je vypnuté napájení a čeká se na rozpoznání a odstranění vzniklé chyby.

3.3.5 Funkční struktura

Na obrázku 3.14 je znázorněn strukturní diagram polohového řízení, rychlosti otáčení, rychlostního řízení, proudového řízení, dokování, řídicích funkcí, monitorování, varovných a chybových signálů. Diagram zobrazuje vazby mezi jednotlivými systémy a použití hlavních nastavovacích parametrů: parametru na volbu režimu servozesilovače (index 6060_{hex}) a parametru na volbu konfigurace použité při zjišťování aktuálních údajů o motoru.



Obrázek 3.14 Funkční struktura servozesilovače.

3.4 DC motor Transmotec

Pro potřeby tohoto projektu je využit stejnosměrný komutátorový motor s kovovými kartáči od firmy Transmotec. Technické údaje o motoru jsou dostupné na webové stránce výrobce [5].



Obrázek 3.15 Stejnosměrný motor PD4266 od firmy Transmotec.

Firma dodává motor taktéž s enkodérem. Tato možnost však nebyla využita a motor je osazen jiným enkodérem. Podrobnější informace o použitém enkodéru jsou v další kapitole.

Motor lze provozovat při teplotě -10...60 °C a při relativní vlhkosti 20...85%. Motor je osazen planetovou převodovkou s ozubenými koly z oceli s převodovým poměrem 24:1. Vůle naprázdno nepřesahuje 3°, povolené radiální zatížení nesmí překročit 8 kg a axiální zatížení nesmí překročit 3 kg. Hlavní vlastnosti motoru jsou uvedeny v tabulce 3.10.

Jm. napětí [V]	24
Jm. moment [g-cm]	570
Jm. otáčky [ot/min]	5900
Jm. proud [mA]	≤2100
Otáčky naprázdno [ot/min]	7000
Proud naprázdno [mA]	≤500
Jm. výkon [W]	34,7

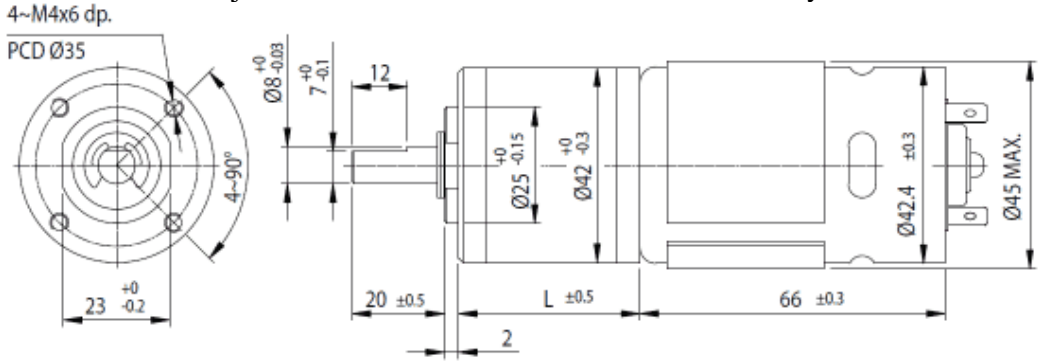
Tabulka 3.10 Vlastnosti motoru.

Vzhledem k tomu, že motor obsahuje převodovku, jsou skutečné provozní vlastnosti za převodovkou odlišné. Skutečné vlastnosti jsou uvedeny v tabulce 3.11.

Převodový poměr	24:1
Počet prevodových stupňů	2
Jm. moment [kg-cm]	9,5
Jm. otáčky [ot/min]	240
Délka L [mm]	39,2
Váha [g]	551

Tabulka 3.11 Vlastnosti motoru s převodovkou.

Na obrázku 3.16 je zobrazeno schéma motoru s funkčními rozměry.



Obrázek 3.16 Rozměrové schéma motoru.

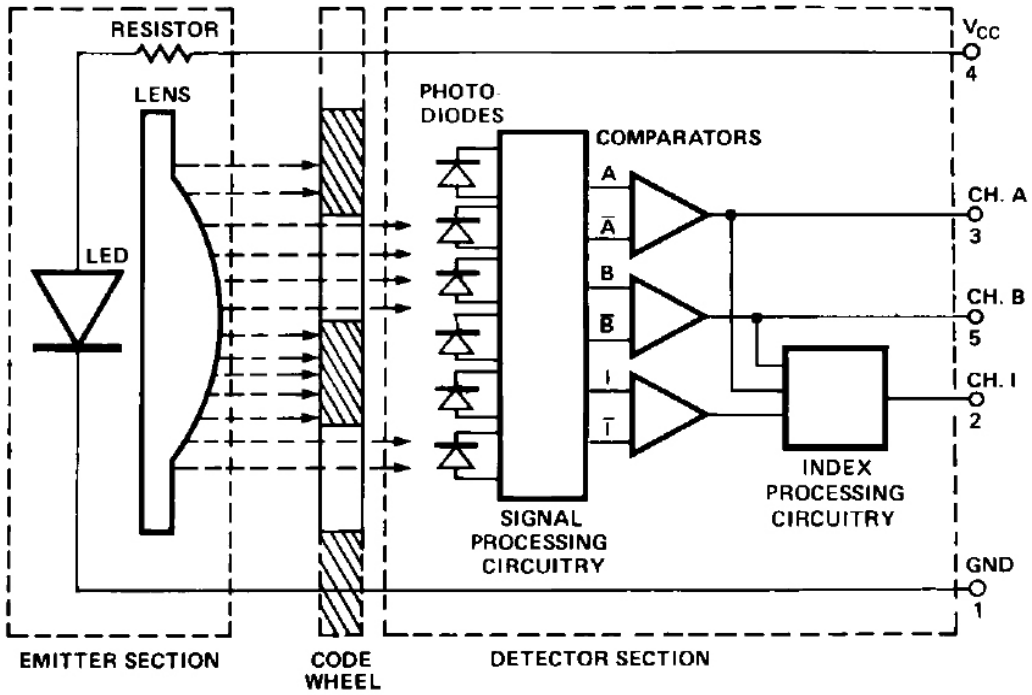
3.5 Inkrementální snímač HEDS

V projektu byl použit inkrementální enkodér HEDS-5540 s 500 pulzy na otáčku, obsahující 3 kanály. Datasheet k enkodéru z kterého je čerpáno v této kapitole je dosupným z [6].



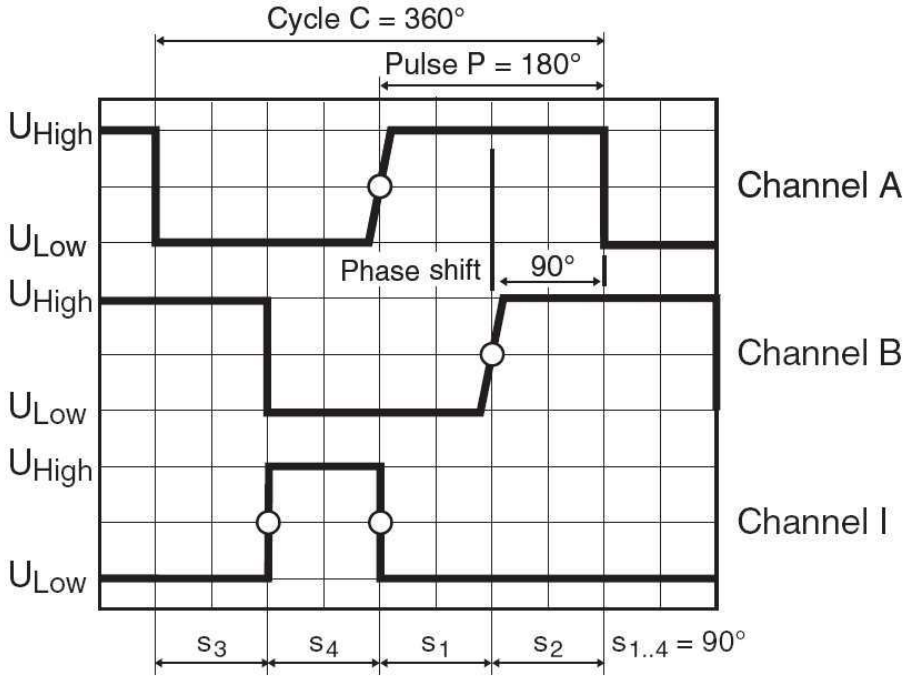
Obrázek 3.17 Inkrementální enkodér HEDS-5540.

Inkrementální enkodér slouží k převodu rotačního pohybu hřídele na 3 kanálový digitální signál. Převod je realizován světelnou metodou. Enkodér obsahuje světelný zářič (obvykle LED diodu). Světlo je pomocí čočky nasměrováno do paralelního paprsku. Paprsek svítí na soustavu světelných detektorů. Mezi světelným vysílačem a přijímačem se ještě vyskytuje děrované kolečko. Kolečko při otáčení přerušuje paprsek světla. Dopadající paprsek na soustavu světelných detektorů je tedy na některých detektorech přerušený a na některých ne. Na osvětlených detektorech je generován signál a na neosvětlených nikoli. Tyto signály putují do komparátorů a vytváří pulzy na kanálech A a B. Blokové schéma tohoto principu je zobrazen na obrázku 3.18.



Obrázek 3.18 Blokové schéma Inkrementálního enkodéru.

Enkodér vytváří pulzy na všech 3 kanálech. Kanály A a B jsou fázově posunuty o 90°. Jestliže se enkodér otáčí proti směru hodinových ručiček (z pohledu od strany enkodéru připojeného k motoru), kanál A předbíhá kanál B jak je zobrazeno na obrázku 3.19. Pokud se enkodér otáčí ve směru hodinových ručiček předbíhá kanál B kanál A.



Obrázek 3.19 Generované pulzy při otáčení proti směru hodinových ručiček.

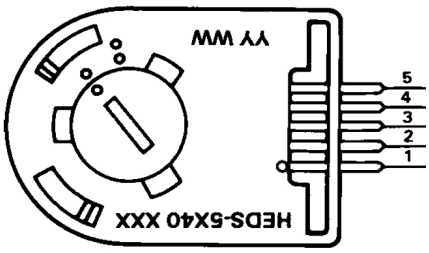
Pro vybraný inkrementální snímač jsou charakteristické vlastnosti z tabulky 3.12.

Napájecí napětí	5V ± 10%
Výstupní signál	TTL kompatibilní
Pulzy na otáčku	500
Počet kanálů	3
Maximální provozní frekvence [kHz]	100
Maximální otáčky [rpm]	12000
Fázový posun	90° ± 45°
Čas náběžné hrany	180 ns
Čas sestupné hrany	40 ns
Provozní teplota	-40...+100℃
Výstupní proud pro kanál	min. -1 mA, max. 5 mA

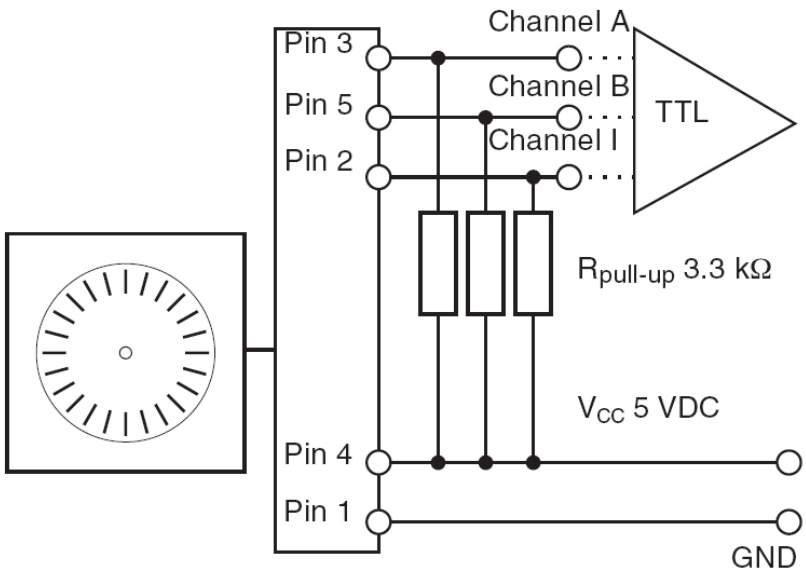
Tabulka 3.12 Vlastnosti inkrementálního snímače HEDS-5540.

Pro připojení enkodéru se používá více-žilový kabel. Konkrétně v tomto případě pěti-žilový, jedna žíla pro každý z výstupních signálů a 2 žíly pro napájení. Enkodér je přímo propojen se servozesilovačem, který díky enkodéru určuje požadované informace jako aktuální polohu nebo aktuální rychlost otáčení a zároveň poskytuje napájecí napětí pro enkodér. Při připojování enkodéru je velice důležité dodržet správné připojení k jednotlivým výstupním pinům. Při případném špatném připojení hrozí zničení enkodéru. Význam jednotlivých výstupních pinů je zobrazen na obrázku 3.20 a 3.21.

Pin 1	GND
Pin 2	kanál I
Pin 3	kanál A
Pin 4	napájecí napětí (5V)
Pin 5	kanál B



Obrázek 3.20 Připojovací piny enkodéru.



Obrázek 3.21 Schéma výstupní části enkodéru.

4 POUŽITÝ SOFTWARE

Cílem této diplomové práce je řízení stejnosměrného motoru pomocí programovatelného automatu a připojeného výkonového členu v podobě servozesilovače. Pro řízení těchto zařízení je dostačujícím programovým nástrojem program PC WORX. Tento software byl použit jako jediný pro celé programování.

4.1 PC WORX

Jedná se o software pro programování programovatelných automatů založený na normě IEC-61131 vyvinutý firmou Phoenix Contact. Program v sobě kombinuje nástroje pro konfiguraci, programování a vizualizaci automatů. Je vhodný pro použití jak pro distribuované řízení tak pro centrální řízení.

PC WORX nabízí 5 programovacích jazyků podle normy IEC-61131, tyto jazyky lze při programování volně kombinovat v rámci jediného projektu. Díky složitosti programového kódu je však v tomto projektu použit výhradně strukturovaný text.

- Strukturovaný text (ST)
- Instrukční list (IL)
- Funkční bloky (FB)
- Ladder diagram (LD)
- Sekvenční funkční diagram (SFC)

Pro potřeby zobrazování a načítání dat z uživatelského rozhraní, např. z dotykového panelu, lze propojit PC WORX s OPC serverem a snadno získávat a nastavovat data pro uživatele.

4.1.1 Základní ovládací prvky

Celý program má standardní uživatelské rozhraní, s kterým se pracuje velice intuitivně. Obsahuje obvyklé ovládací prvky jako stromovou strukturu projektu nebo rozhranní pro vkládání funkcí, atd. Detailní popis uživatelského rozhraní a ovládacích prvků je zobrazen na obrázku 4.1.

(1) Okno zdrojového kódu – slouží k zobrazování a úpravě zdrojového kódu právě otevřené programové organizační jednotky.

(2) Okno struktury projektu – zobrazuje celou strukturu projektu: datové typy, programové organizační jednotky, úlohy, proměnné, atd.

(3) Okno funkcí a funkčních bloků – v tomto okně je seznam všech dostupných funkcí a funkčních bloků, které lze použít při programování. Nachází se zde jak předpřipravené funkce tak i uživatelsky definované. Pomocí tohoto okna lze libovolné funkce vkládat do zdrojového kódu.

(4) Okno systémových zpráv – slouží k zobrazování informací po kompilaci programu. Informace zahrnují průběh kompilace, nalezené varování nebo chyby, atd.

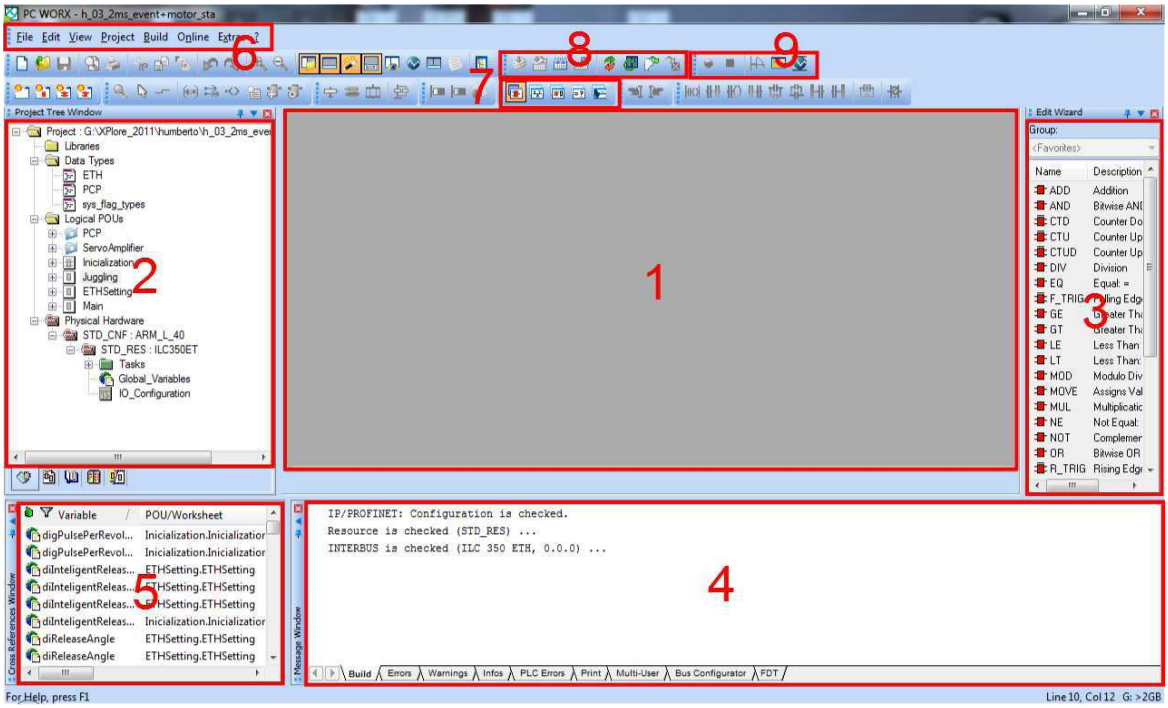
(5) Okno odkazů – obsahuje všechny proměnné použité v projektu a konkrétní informace o použití každé proměnné.

(6) Menu programu

(7) Ovládací tlačítka pro přepínání mezi prostředími programu – zleva obsahuje tlačítka pro přepnutí do: programového prostředí, konfiguračního prostředí a prostředí pro konfiguraci vstupů a výstupů.

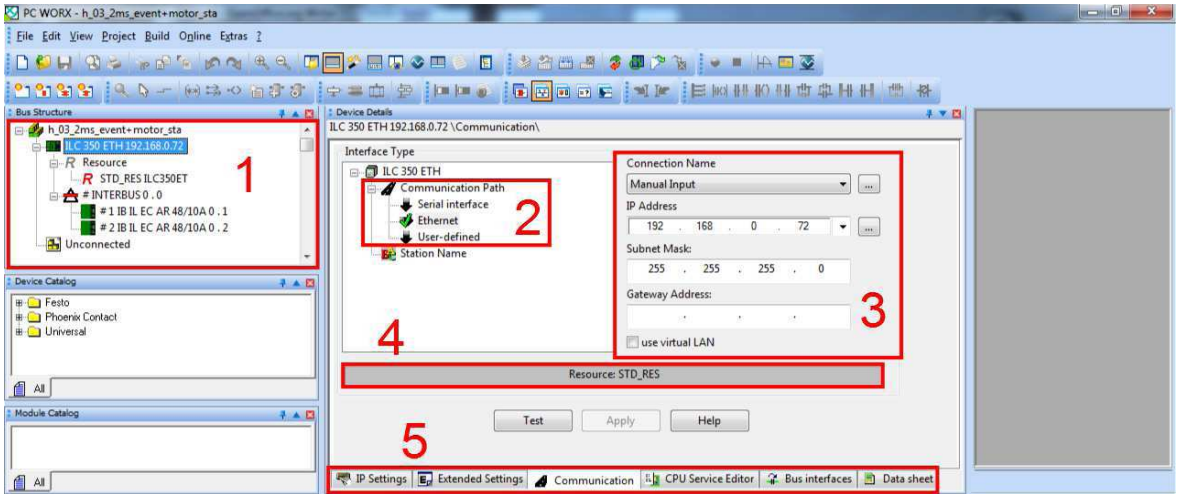
(8) Oblast ovládacích tlačítek pro kompilaci a debugování – nejdůležitějšími ovládacími prvky tohoto panelu jsou tlačítka pro: kompilaci, zapnutí/vypnutí debug režimu a spouštění ovládacího rozhraní.

(9) Ovládací tlačítka pro analyzátor – analyzátor slouží k nahrávání hodnot proměnných a následné vykreslování těchto hodnot do grafu. Jedná se o velice důležitý prvek, který je často využíván při debugování.



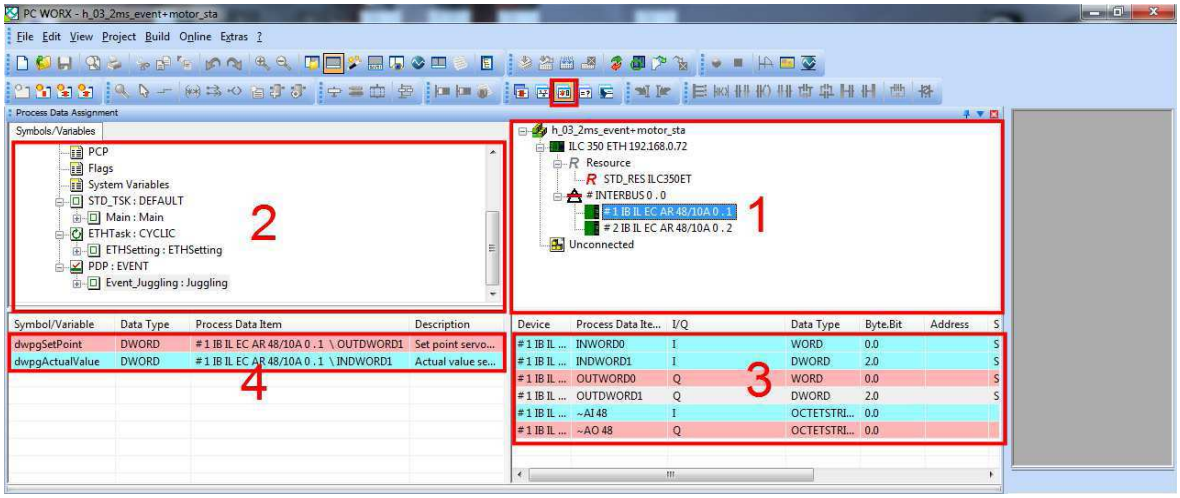
Obrázek 4.1 Základní ovládací prvky programu.

Při použití tlačítek na ovládacím panelu číslo 7 se lze dostat do nastavovacích prostředí programu. Konfigurační prostředí zobrazené na obrázku 4.2 slouží k zjištění továrních informací a nastavení parametrů automatu a připojených zařízení k lokální sběrnici. Další konfigurační prostředí je zobrazeno na obrázku 4.3 a slouží ke konfiguraci vstupů a výstupů zařízení připojených k programovatelnému automatu pomocí lokální sběrnice.



Obrázek 4.2 Konfigurační prostředí programu.

(1) Zobrazená zařízení připojená k lokální sběrnici.
(2) Výběr různých typů komunikace pro propojení s automatem.
(3) Nastavení Ethernetové komunikace (IP adresy a masky sítě).
(4) Informace o stavu komunikace – navázána/nenavázána
(5) Panel pro přepínání mezi různými nastavovacími obrazovkami – zobrazená obrazovka je nastavení komunikace.



Obrázek 4.3 Konfigurační prostředí pro nastavení vstupů a výstupů připojených zařízení.

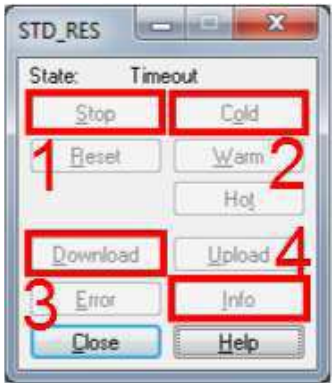
(1) Zobrazená zařízení připojená k lokální sběrnici.

(2) Dostupné proměnné v projektu pro připojení vstupu nebo výstupu.

(3) Vstupy a výstupy vybraného zařízení připojeného ke sběrnici.

(4) Přiřazení vstupu nebo výstupu vybrané proměnné.

Dalším velice důležitým ovládacím prvkem je ovládací rozhraní automatu. Jedná se o malý ovládací panel s několika tlačítky. Hlavní funkce panelu je zapínání/vypínání běhu programu v programovatelném automatu a nahrávání/stahování programu do/z automatu. Toto jednoduché ovládací rozhraní je zobrazeno na obrázku 4.4.



Obrázek 4.4 Ovládací rozhraní automatu.

(1) Tlačítko STOP – slouží k zastavení vykonávání programu, který je nahrán v automatu. Tlačítko má stejnou funkci jako STP režim přepnutý pomocí přepínače na těle automatu. S tím rozdílem, že tlačítko na těle automatu má větší prioritu.

(2) Tlačítko COLD – slouží ke spuštění programu nahraného v automatu.

(3) Tlačítko DOWNLOAD – slouží k nahrání otevřeného projektu do automatu. Pro úspěšné nahrání projektu je zapotřebí projekt předem zkompilovat. V opačném případě se objeví chybová hláška.

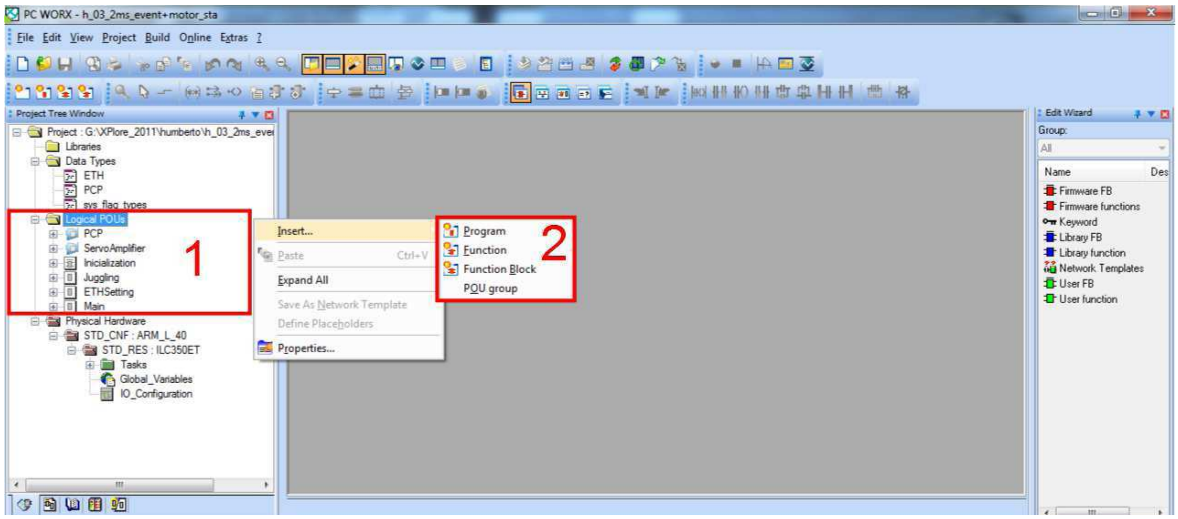
(4) Tlačítko INFO – jedná se o čistě informativní tlačítko. Zobrazuje informace o právě nahraném projektu v automatu.

4.1.2 Programové organizační jednotky

Programové organizační jednotky (POU – program organization unit) jsou nejdůležitějšími prvky celého programu. Norma IEC 61131 rozlišuje 3 typy programových organizačních jednotek: funkce, funkční blok a program. Dalo by se říci, že uvedené pořadí organizačních jednotek určuje jejich hierarchické postavení. Žádná z organizačních jednotek nemůže volat sama sebe ani zprostředkovaně, je tedy zakázáno používat rekurzivní metody programování, místo nich je třeba použít jiných metod například cyklů.

- **Funkce** – jedná se o nejnižší programovou jednotku. Funkce neobsahuje paměť a může volat pouze funkce, nikoli funkční bloky.
- **Funkční blok** – je velice podobný funkci, ale obsahuje navíc paměť. Tato paměť slouží k uchování dat z předchozího zavolání funkčního bloku. Tato vlastnost je velice užitečná a lze ji využít například pro čítače. Další výhodou funkčního bloku je, že může volat jak funkce tak funkční bloky.
- **Program** – jedná se v podstatě o hlavní programovou jednotku. Obsahuje statickou paměť stejně jako funkční blok a stejně tak může volat jak funkční bloky tak funkce. Programy nejsou spouštěny voláním jako funkce a funkční bloky, ale jsou navázány na úlohy a jsou spouštěny podle typu úlohy, ke které jsou připojeny.

V projektu, který je vytvořen v PC WORX, jde mimo tyto tři organizační jednotky vytvářet navíc POU skupina. Skupina nemá žádný programový význam, slouží pouze k organizaci a přehlednosti hierarchie jednotlivých organizačních jednotek ve stromu celého projektu. Na obrázku 4.5 je ve výseči číslo 1 zobrazena oblast logických programových jednotek a jejich umístění v hierarchii projektu. Ve výseči číslo 2 je zobrazeno menu pro přidání další programové jednotky do struktury projektu. Jak je vidět, lze přidat všechny 3 zmíněné typy POU a navíc POU skupinu.



Obrázek 4.5 Programové organizační jednotky v projektu.

4.1.3 Typy úloh

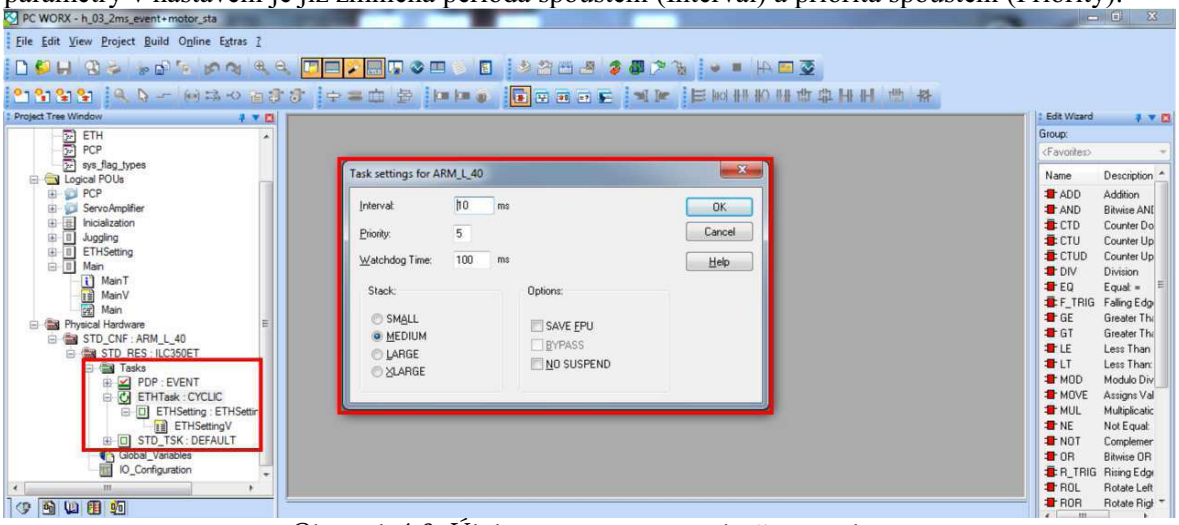
Jak již bylo řečeno pro spouštění programů slouží tzv. úlohy. Úloha je řídicí prvek. V PC WORX lze použít dva typy úloh: periodicky spouštěný a neperiodicky spouštěný.

- **Periodicky spouštěná úloha** – je aktivována v přesně daný časový okamžik. Spouštění tedy probíhá s konstantní periodou.
- **Neperiodicky spouštěná úloha** – je aktivována pokud je na řídící proměnnou přivedena náběžná hrana signálu. Jedná se o takzvané událostní spouštění.

Dalším typem je defaultní úloha, která může být v projektu obsažena jen jednou a je spouštěna v cyklu. Tato úloha spouští hlavní program a má nejnižší prioritu.

Posledním typem úlohy je systémová úloha. Jedná se v podstatě o událostně spouštěnou úlohu. Spuštění probíhá na základě přechodu automatu do určitého systémového stavu, např. režim stop, chyba komunikace, atd.

Při spouštění úloh hraje velkou roli priorita, která je přiřazena každé úloze. Úloha s prioritou 0 má nejvyšší prioritu a úloha s prioritou 31 nejnižší. Priorita určuje, která úloha bude spouštěna přednostně při požadavku na spuštění více úloh ve stejný časový okamžik. Na obrázku 4.6 je zobrazeno umístění úloh v projektu a nastavení periodicky spouštěné úlohy. Hlavními parametry v nastavení je již zmíněná perioda spouštění (Interval) a priorita spouštění (Priority).



Obrázek 4.6 Úlohy ve stromové struktuře projektu.

4.1.4 Podporované datové typy

Datový typ	Popis	Velikost [Bit]	Rozsah	Počáteční hodnota
BOOL	Boolean	1	0...1	0
SINT	Short integer	8	-128...127	0
INT	Integer	16	-32,768...32,767	0
DINT	Double integer	32	-2,147,483,648 až 2,147,483,647	0
USINT	Unsigned short integer	8	0 až 255	0
UINT	Unsigned integer	16	0 až 65,535	0
UDINT	Unsigned double integer	32	0 až 4,294,967,295	0
REAL	Real numbers	32	-3.402823466 E+38 až -1.175494351 E-38 a +1.175494351 E-38 až +3.402823466 E+38	0.0
LREAL	Long real numbers	64	-1.7976931348623158 E+308 až -2.2250738585072014 E-308 a +2.2250738585072014 E-308 až +1.7976931348623158 E+308	0.0
TIME	Duration	32	0... 4,294,967,295 ms	t#0s
BYTE	Bit string of length 8	8	0...255 (16#00...16#FF)	0
WORD	Bit string of length 16	16	0...65,535 (16#00...16#FFFF)	0
DWORD	Bit string of length 32	32	0...4,294,967,295 (16#00....16#FFFFFFFF)	0

Tabulka 4.1 Povolené datové typy v PC WORX.

5 KOMUNIKACE

Pro potřeby parametrizace a komunikace se servozesilovačem je využita procesní komunikace po sběrnici INTERBUS a komunikační protokol periferií (PCP komunikace). Dalším využitým komunikačním rozhraním je Ethernet, který je použit pro zajištění komunikace mezi automaty, mezi automatem a dotykovým panelem a pro komunikaci s okolím za účelem programování a údržby. V této kapitole budou vysvětleny pouze komunikační příkazy použité pro nastavování servozesilovače (PCP) a pro komunikaci mezi automaty (Ethernet).

5.1 PCP komunikace

PCP komunikace slouží pro acyklickou komunikaci s moduly připojenými k programovatelnému automatu. Konkrétně je použita pro parametrizaci servozesilovače. PCP protokol disponuje příkazy pro navázání komunikace, čtení, zápis a ukončení komunikace. Pro usnadnění práce jsou v softwaru PC WORX předpřipraveny funkční bloky, které tyto příkazy obsluhují: PCP_CONNECT pro navázání spojení mezi účastníky komunikace, PCP_READ pro čtení dat a PCP_WRITE pro zápis dat.

5.1.1 PCP_CONNECT

PCP_CONNECT slouží pro navázání a udržování komunikace mezi automatem a připojeným modulem. Každý modul musí mít navázané spojení samostatným blokem PCP_CONNECT. Pokud je na vstupu EN_C logická hodnota TRUE, blok se snaží navázat a udržovat komunikaci s modulem definovaným pomocí parametru PARTNER. Vstupně výstupní proměnná PARTNER musí být vždy navázána na řetězec. Pokud je hodnota výstupního parametru VALID na TRUE poté výstupní číselný parametr ID odpovídá zařízení, s kterým je navázána komunikace. Tato číselná hodnota slouží pro rozpoznání zařízení při použití dalších PCP příkazů. Pokud blok nedokázal navázat komunikaci je indikována chyba pomocí výstupního parametru ERROR a příčina chyby obsažena ve vstupně výstupním parametru ADD_ERROR.

PCP_CONNECT_1(EN_C:=(* BOOL *), ADD_ERROR:=(* ANY *), PARTNER:=(* ANY *));
(* BOOL *) :=PCP_CONNECT_1.VALID;
(* BOOL *) :=PCP_CONNECT_1.ERROR;
(* ANY *) :=PCP_CONNECT_1.ADD_ERROR;
(* ANY *) :=PCP_CONNECT_1.PARTNER;
(* INT *) :=PCP_CONNECT_1.ID;

5.1.2 PCP_READ

PCP_READ slouží ke čtení dat ze zařízení identifikovaného parametrem ID, získaného z bloku sloužícího pro navázání komunikace. Důležitým vstupně výstupním parametrem je VAR_1, který definuje INDEX a SUBINDEX požadovaných dat. Blok je spouštěn vzestupnou hranou parametru REQ a indikuje nově příchozí data obsažena ve výstupním parametru RD_1 pomocí vzestupné hrany v parametru NDR. Případně místo přijatých dat se na výstupu ERROR objeví chybová hláška a obsah chyby ve vstupně výstupním parametru ADD_ERROR. Posledním výstupním parametrem je DATA_CNT udávající velikost přijatých dat v bajtech.

PCP_READ_1(REQ:=(* BOOL *), ID:=(* INT *), ADD_ERROR:=(* ANY *), VAR_1:=(* ANY *), RD_1:=(* ANY *));
(* BOOL *) :=PCP_READ_1.NDR;
(* INT *) :=PCP_READ_1.DATA_CNT;
(* BOOL *) :=PCP_READ_1.ERROR;
(* ANY *) :=PCP_READ_1.ADD_ERROR;
(* ANY *) :=PCP_READ_1.VAR_1;
(* ANY *) :=PCP_READ_1.RD_1;

5.1.3 PCP_WRITE

PCP_WRIETE slouží k zápisu dat do modulu identifikovaného vstupním parametrem ID. Blok je opět spouštěn vzestupnou hranou vstupního parametru REQ. VAR_1, stejně jako u bloku sloužícího pro čtení, určuje INDEX a SUBINDEX místa kam má být proveden zápis. Zapisovaná data se přiřazují do vstupního parametru SD_1 a jejich velikost do parametru DATA_CNT. Po úspěšném zápisu je tato skutečnost indikována ve výstupním parametru DONE. V opačném případě je indikována chyba v parametru ERROR a důvod chyby ve vstupně výstupním parametru ADD_ERROR.

PCP_WRITE_1(REQ:=(* BOOL *), ID:=(* INT *), DATA_CNT:=(* INT *), ADD_ERROR:=(* ANY *), VAR_1:=(* ANY *), SD_1:=(* ANY *));
(* BOOL *) :=PCP_WRITE_1.DONE;
(* BOOL *) :=PCP_WRITE_1.ERROR;
(* ANY *) :=PCP_WRITE_1.ADD_ERROR;
(* ANY *) :=PCP_WRITE_1.VAR_1;
(* ANY *) :=PCP_WRITE_1.SD_1;

5.2 Ethernetová komunikace

Ethernetová komunikace je použita pro komunikaci mezi programovatelnými automaty. Princip používání příkazů je velice podobný jako u PCP komunikace. PC WORX obsahuje opět před chystané funkční bloky pro obsluhu Ethernetové komunikace, proto je práce s ní velice jednoduchá. Obsahuje funkční bloky pro navázání komunikace, čtení a zápis. Informace o funkci a parametrizaci funkčních bloků pro ovládání Ethernetové komunikace jsou z technické dokumentace [7].

5.2.1 IP_CONNECT

IP_CONNECT slouží k navázání Ethernetové komunikace mezi zařízeními (automaty). Blok se pokouší navázat komunikaci při logické hodnotě TRUE ve vstupním parametru EN_C. Druhým vstupním parametrem je PARTNER, který obsahuje řetězec s IP adresou protějšího zařízení. Informaci o navázání komunikace nebo o chybě jsou indikovány v proměnných VALID a ERROR. Hlavním výstupním parametrem je ID, který obsahuje identifikační číslo navázané komunikace pro další funkční bloky ETH komunikace.

IP_CONNECT_1(EN_C:=(* BOOL *), PARTNER:=(* STRING *));
(* BOOL *) :=IP_CONNECT_1.VALID;
(* BOOL *) :=IP_CONNECT_1.ERROR;
(* INT *) :=IP_CONNECT_1.STATUS;
(* INT *) :=IP_CONNECT_1.ID;

5.2.2 IP_URCV

IP_URCV slouží k přijímání dat. Blok je aktivní a přijímá data když má vstupní proměnná EN_R hodnotu TRUE. Data jsou přijímána od zařízení, které má navázanou komunikaci pod číslem z proměnné ID. Přijatá data nebo chyba je indikována výstupní proměnnou NDR resp. ERROR. Přijatá data jsou obsažena ve vstupně výstupní proměnné RD_1.

IP_URCV_1(EN_R:=(* BOOL *), ID:=(* INT *), R_ID:=(* STRING *), RD_1:=(* ANY *));
(* BOOL *) :=IP_URCV_1.NDR;
(* BOOL *) :=IP_URCV_1.ERROR;
(* INT *) :=IP_URCV_1.STATUS;
(* ANY *) :=IP_URCV_1.RD_1;

5.2.3 IP_USEND

IP_USEND slouží k odeslání dat v navázané komunikaci specifikované vstupním parametrem ID. Blok pošle data v okamžiku, kdy se ve vstupní proměnné REQ objeví vzestupná hrana. Posílaná data jsou ve vstupně výstupní proměnné SD_1. Funkční blok indikuje úspěšné poslání dat proměnnou DONE, nebo chybu proměnnou ERROR.

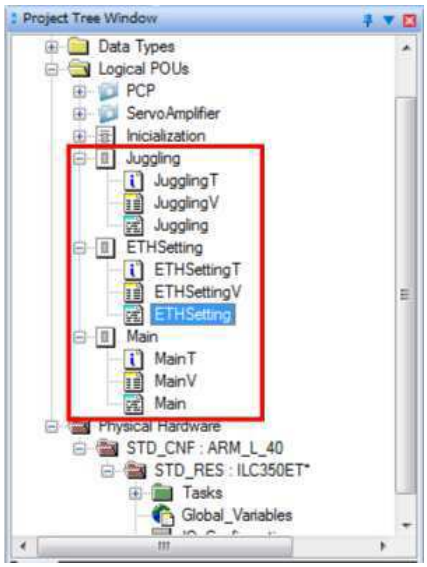
IP_USEND_1(REQ:=(* BOOL *), ID:=(* INT *), R_ID:=(* STRING *), SD_1:=(* ANY *));
(* BOOL *):=IP_USEND_1.DONE;
(* BOOL *):=IP_USEND_1.ERROR;
(* INT *):=IP_USEND_1.STATUS;
(* ANY *):=IP_USEND_1.SD_1;

6 ŘEŠENÍ

Řešení části projektu, která ovládá pohon, je realizováno pomocí jediného programu, který je nahrán v programovatelném automatu. Tento program obsahuje 3 hlavní programové smyčky, které realizují všechny operace.

6.1 Programy

Jednotlivé programy neboli programové smyčky jsou spouštěny s různými prioritami a různým způsobem. V projektu jsou využity 3 typy spouštění: default, cyklické a událostní. Pro každou programovou smyčku je použita jiná priorita spouštění pro zajištění správného vyhodnocování událostí.



Obrázek 6.1 Hlavní programové smyčky programu.

Smyčka s nejnižší prioritou se jmenuje *Main* a provádí pouze nutné počáteční operace jako inicializace proměnných, počáteční nastavení servozesilovače, navázání komunikace, atd. Další smyčka má vyšší prioritu a jmenuje se *ETHSetting*. Tato programová smyčka je navázána na cyklicky prováděnou úlohu s periodou provádění 10 ms. Úkolem smyčky je obsluha Ethernetové komunikace a zpracování příkazů přijatých právě touto komunikací. Poslední smyčka s nejvyšší prioritou má název *Juggling*. Jedná se o hlavní smyčku provádějící samotné triky. Smyčka je navázána na úlohu, která je spouštěna událostí. Událost je vyvolána vždy při příchodu dat po sběrnici INTERBUS. Smyčka je tedy spouštěna s periodou 1,85 ms a je synchronní s INTERBUS komunikací.

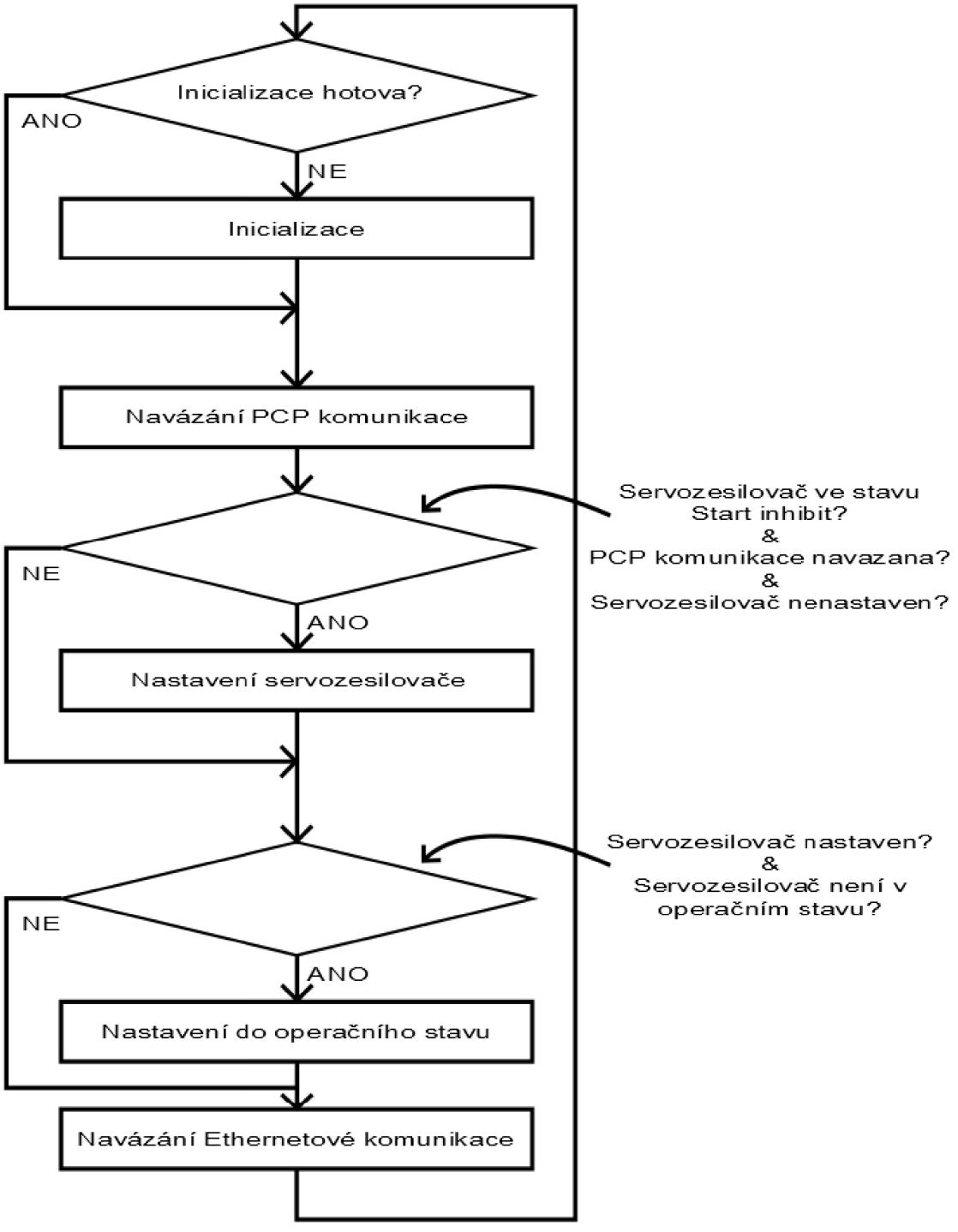
6.1.1 Program Main

Programová smyčka *Main* je hlavní programovou smyčkou. Tuto smyčku musí obsahovat každý program. Je spouštěna stále dokola. Vykonávání je přerušeno pouze na dobu nutné prodlevy mezi jednotlivými vykonáváními smyčky a pokud se má vykonat jiný task s vyšší prioritou. Tyto vlastnosti přesně vyhovují použití smyčky *Main* v projektu.

Smyčka *Main* slouží hlavně k obslužným a inicializačním operacím. Vykonávání smyčky se dá rozdělit do 5 dílčích operací:

- inicializace – slouží k nastavení počátečních hodnot proměnných a parametrů.
- navázání PCP komunikace
- nastavení parametrů servozesilovače – parametrizace servozesilovače pomocí PCP.
- uvedení servozesilovače do operačního stavu – průchod všech operačních stavů podle schématu uvedeného na obrázku 3.13.
- navázání Ethernetové komunikace

Operace se vykonávají v uvedené posloupnosti a provádí různé operace inicializačního charakteru. Blokové schéma programové smyčky je zobrazeno na obrázku 6.2.



Obrázek 6.2 Blokové schéma programové smyčky Main.

Celá programová smyčka je navržena tak, aby všechny dílčí operace proběhli jen jednou. U inicializace, nastavení parametrů servozesilovače a nastavení servozesilovače do operačního stavu je tato skutečnost realizována pomocí testovacích podmínek. U zbylých dvou komunikačních operací je to zajištěno hodnotami vstupních proměnných. Více informací o komunikačních příkazech je uvedeno v kapitole 5. Výsledný efekt toho principu je, že po vykonání všech inicializací a počátečních nastavení jsou nastaveny kontrolní proměnné na splněné hodnoty a testovací podmínky dále neprovádějí vnořené procedury. Programová smyčka Main proto poté, kromě vyhodnocování podmínek, nic neprovádí a šetří výpočetní výkon.

Nejsložitějšími operacemi jsou: uvedení servozesilovače do operačního stavu a nastavení parametrů servozesilovače. Při uvádění servozesilovače do operačního stavu je nutné postupně projít všechny 4 stavy servozesilovače. Po připojení zařízení k napájecímu napětí je po vnitřních inicializacích servozesilovač automaticky uveden do stavu Start Inhibit. Pro přepínaní mezi stavy je nutné nastavit procesní slovo Control word na definovanou hodnotu.

- stavu Start Inhibit odpovídá Status word ve tvaru – xxxx xxxx x1xx 0000 a pro přechod ze Start Inhibit do Ready to Operate je nutné nastavit Control word na – xxxx xxxx xxxx x110

  IF (wStatusWord.X0 = FALSE) **&** (wStatusWord.X1 = FALSE) **&** (wStatusWord.X2 = FALSE) **&** (wStatusWord.X3 = FALSE) **&** (wStatusWord.X6 = TRUE)
  THEN wControlWord.X0 := FALSE;
  wControlWord.X1 := TRUE;
  wControlWord.X2 := TRUE;
  END_IF;

- stavu Ready to Operate odpovídá Status word ve tvaru – xxxx xxxx x010 0001 a pro přechod z Ready to Operate do ON je nutné nastavit Control word na – xxxx xxxx xxxx x111

  IF (wStatusWord.X0 = TRUE) **&** (wStatusWord.X1 = FALSE) **&** (wStatusWord.X2 = FALSE) **&** (wStatusWord.X3 = FALSE) **&** (wStatusWord.X4 = FALSE) **&** (wStatusWord.X5 = TRUE) **&** (wStatusWord.X6 = FALSE)
  THEN wControlWord.X0 := TRUE;
  wControlWord.X1 := TRUE;
  wControlWord.X2 := TRUE;
  END_IF;

- stavu ON odpovídá Status word ve tvaru – xxxx xxxx x010 0011 a pro přechod z ON do Operation Enabled je nutné nastavit Control word na – xxxx xxxx xxxx 1111

  IF (wStatusWord.X0 = TRUE) **&** (wStatusWord.X1 = TRUE) **&** (wStatusWord.X2 = FALSE) **&** (wStatusWord.X3 = FALSE) **&** (wStatusWord.X4 = FALSE) **&** (wStatusWord.X5 = TRUE) **&** (wStatusWord.X6 = FALSE)
  THEN wControlWord.X0 := TRUE;
  wControlWord.X1 := TRUE;
  wControlWord.X2 := TRUE;
  wControlWord.X3 := TRUE;
  END_IF;

Uvedené zdrojové kódy jsou součástí podprogramu, který realizuje uvedení servozesilovače do operačního stavu.

Při přechodu do nového stavu je taktéž automaticky změněn údaj v procesním slově Status word, které definuje v jakém stavu se servozesilovač nachází. Před přechodem do dalšího stavu je tedy nejdříve zkontrolováno, zda se servozesilovač nachází v požadovaném stavu, ze kterého se bude přepínat do nového. Tuto skutečnost je nutné testovat, protože přepnutí servozesilovače do nového stavu trvá určitou dobu, která je způsobena činností servozesilovače a prodlevou procesní komunikace. Pokud se servozesilovač dostane do stavu Operation Enabled je připraven

pro provoz a proměnná, která rozhoduje o příštím provádění této dílčí operace, je nastavena tak, aby byl tento úsek v příštím cyklu vynechán.

Nejdůležitější a nejdelší na dobu vykonávání je nastavování parametrů servozesilovače. Při této operaci je nutné nastavit desítky parametrů servozesilovače. Při nastavování je použita PCP komunikace, jejíž detailní popis je uveden v kapitole 5. Všechny parametry v servozesilovači mají své výchozí hodnoty, ale pro správnou funkci takto dynamicky složité úlohy je potřeba většinu hlavních parametrů překonfigurovat. Překonfigurovat bylo zapotřebí nominální hodnoty připojeného motoru, s kterými servozesilovač pracuje, jako jmenovité napětí motoru, jmenovitý proud motoru, jmenovité otáčky motoru a další. Dále bylo potřeba překonfigurovat parametry týkající se enkodéru, dynamických vlastností řízení, limitních provozních hodnot a hlavně všechny parametry použitých regulátorů. Detailnímu popisu procesu nastavování regulátorů bude věnována kapitola 6.2.

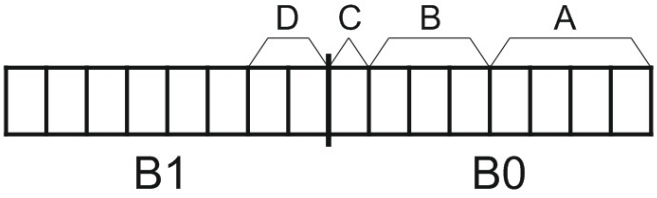
6.1.2 Program ETHSetting

Programová smyčka ETHSetting je smyčka, která obsluhuje Ethernetovou komunikaci. Zpracovává přijatou komunikaci od nadřízeného řídicího systému a převádí přijaté informace do povelů ovládajících pohyb motoru.

Data přijatá Ethernetovou komunikací mají podobu struktury. Datový typ vytvořený pro jejich identifikaci má tvar:

```
TYPE
    stDataETH : STRUCT
        Mode : DWORD;
        AngleRelease : DWORD;
        SpeedRelease : DWORD;
    END_STRUCT;
END_TYPE
```

Struktura obsahuje 3 záznamy. Všechny 3 záznamy jsou typu double word o velikosti 32 bitů. Struktura obsahuje 2 údaje nesoucí čistě numerická data: *AngleRelease* pro hodnotu odhozového úhlu a *SpeedRelease* pro hodnotu odhozové rychlosti. Třetím záznamem je řídicí slovo *Mode*. Slouží k identifikaci operací, které byly provedeny na uživatelském rozhraní. Obsahuje informace o režimu, v kterém se zařízení nachází, o aktuálně stisknutém tlačítku, o směru házení a o aktuálním kroku v krokovém režimu. Pro informace o jednotlivých položkách jsou v řídicím slově *Mode* vyhrazeny určité bity. Detailní popis řídicího slova *Mode* je na obrázku 6.3.



Obrázek 6.3 Kontrolní slovo Mode.

I když má slovo *Mode* velikost 4 bajty, využity jsou pouze první dva, konkrétně bajt 0 (B0) a bajt 1 (B1). Jednotlivé bity mají tento význam:

- oblast A (bit 0 až bit 3) – stisknuté tlačítko
- oblast B (bit 4 až bit 6) – režim zařízení
- oblast C (bit 7) – směr žonglování
- oblast D (bit 8 až bit 9) – aktuální krok při režimu s krokováním

Při zpracování řídicího slova *Mode* v programu *ETHSetting* je nejdříve testován režim, v kterém je zařízení provozováno. Poté je testováno, které tlačítko bylo stisknuto. Zařízení lze provozovat ve 4 režimech a lze navíc rozpoznat jeden režim servisní:

- **jeden hod** – tomuto režimu odpovídá bitová hodnota 001 (dekadicky 1). V tomto režimu jsou aktivní tlačítka pro nastavení hodnot zadaných uživatelem (tlačítko SET) a tlačítko pro odstartování provádění triku (tlačítko JUGGLE). Při nastavování hodnot od uživatele je také použita hodnota určující směr házení.

  IF wMode.X4 **&** NOT wMode.X5 **&** NOT wMode.X6
  THEN
  (*testování stisku tlačítka*)
  END_IF;

- **automatické žonglování** – tomuto režimu odpovídá bitová hodnota 010 (dekadicky 2). Pro tento režim je přípustné pouze jedno tlačítko (tlačítko JUGGLE). Tlačítko pro nastavení hodnot je přístupné pouze ze servisní obrazovky a při běžném provozu není přístupné, protože se jedná o automatický režim, který má přednastavené ideální parametry.

  IF NOT wMode.X4 **&** wMode.X5 **&** NOT wMode.X6
  THEN
  (*testování stisku tlačítka*)
  END_IF;

- **automatické žonglování s krokováním** – tomuto režimu odpovídá bitová hodnota 011 (dekadicky 3). Jedná se opět o automatický režim. Disponuje tlačítkem pro nastavení triku (tlačítko STEP MODE) a tlačítkem pro provádění jednotlivých kroků (tlačítko STEP). Pomocí tlačítka STEP se v kontrolním slově *Mode* mění oblast D z bitové hodnoty 01(dekadicky 1) na 10 (dekadicky 2) a naopak. Tyto dvě hodnoty aktivují 2 body přerušení, které zajišťují krokování.

  IF NOT wMode.X4 **&** wMode.X5 **&** wMode.X6
  THEN
  (*testování stisku tlačítka*)
  END_IF;

- **žonglování s nastavitelnou výškou** – tomuto režimu odpovídá bitová hodnota 001 (dekadicky 1) stejně jako pro režim jednoho hodu. Oba triky jsou ve své podstatě identické, proto lze pro řízení použít první režim. Liší se pouze v získání odhozových hodnot a to má na starosti druhá část projektu. Z hlediska řízení se v obou případech jedná o trik vyhození míčku žádanou rychlostí z žádaného úhlu.

- **servisní režim** – tomuto režimu odpovídá bitová hodnota 111 (dekadicky 7). V podstatě se nejedná o režim provozu, ale o režim nastavení parametrů automatických režimů. K tomuto slouží servisní obrazovka, ke které má přístup pouze obsluha.

  IF wMode.X4 **&** wMode.X5 **&** wMode.X6
  THEN
  (*testování stisku tlačítka*)
  END_IF;

Jak již bylo zmíněno oblast A slouží k rozpoznání aktuálně stisknutého tlačítka. V programu je rozlišeno 6 různých tlačítek:

- **tlačítko SET** – tomuto tlačítku odpovídá bitová hodnota 0001 (dekadicky 1). Tlačítko slouží k nastavení hodnot od uživatele do provozních proměnných jednotlivých režimů. Tlačítko SET funguje napříč režimy a o tom k jakému režimu právě patří je rozhodnuto na základě informace o právě provozovaném režimu zařízení.

```
IF wMode.X0 & NOT wMode.X1 & NOT wMode.X2 & NOT wMode.X3
        THEN
                (*nastavení provozních proměnných daného režimu*)
END_IF;
```

- **tlačítko JUGGLE** – tomuto tlačítku odpovídá bitová hodnota 0010 (dekadicky 2). Tlačítko slouží ke spuštění provádění daného triku. Funguje multifunkčně stejně jako tlačítko SET. O tom, který trik se má provést, je rozhodnuto opět podle režimu provozu. Tlačítko STEP je také reprezentováno bitovými hodnotami 0010, ale navíc se v automatickém režimu s krokováním zohledňuje obsah oblasti D v kontrolním slově *Mode*.

```
IF NOT wMode.X0 & wMode.X1 & NOT wMode.X2 & NOT wMode.X3
        THEN
                (*spuštění provádění triku*)
END_IF;
```

- **tlačítko STOP** – tomuto tlačítku odpovídá bitová hodnota 0011 (dekadicky 3). Tlačítko funguje v každém režimu stejně – zastaví právě provádějící trik a vrátí rameno do počáteční polohy.

```
IF wMode.X0 & wMode.X1 & NOT wMode.X2 & NOT wMode.X3
        THEN
                (*zastavení právě prováděného triku*)
END_IF;
```

- **tlačítko SET UP** – tomuto tlačítku odpovídá bitová hodnota 0111 (dekadicky 7). Tlačítko je přístupné pouze ze servisní obrazovky a slouží k nastavení procesních proměnných automatických režimů žonglování.

```
IF wMode.X0 & wMode.X1 & wMode.X2 & NOT wMode.X3
        THEN
                (*nastavení parametrů do procesních proměnných automatických režimů*)
END_IF;
```

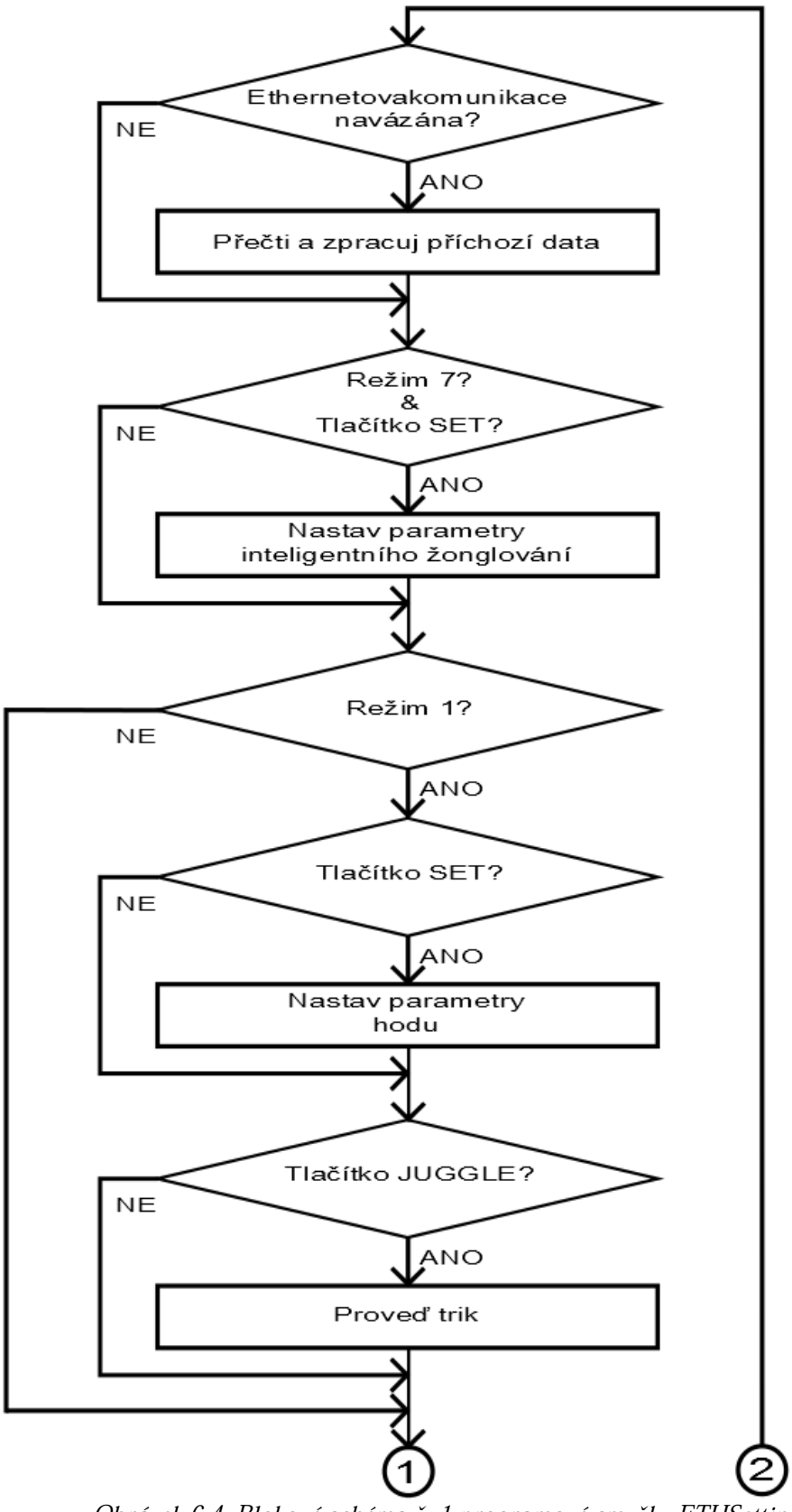
- **tlačítko TOTAL STOP** – tomuto tlačítku odpovídá bitová hodnota 1111 (dekadicky 15). Tlačítko slouží k uvedení stroje do havarijního režimu. To znamená okamžité zastavení stroje a odpojení napájení motoru. Během havarijního režimu není možné stroj provozovat.

```
IF wMode.X0 & wMode.X1 & wMode.X2 & wMode.X3
        THEN
                (*přechod do havarijního stavu, odpojení napájení motoru*)
END_IF;
```

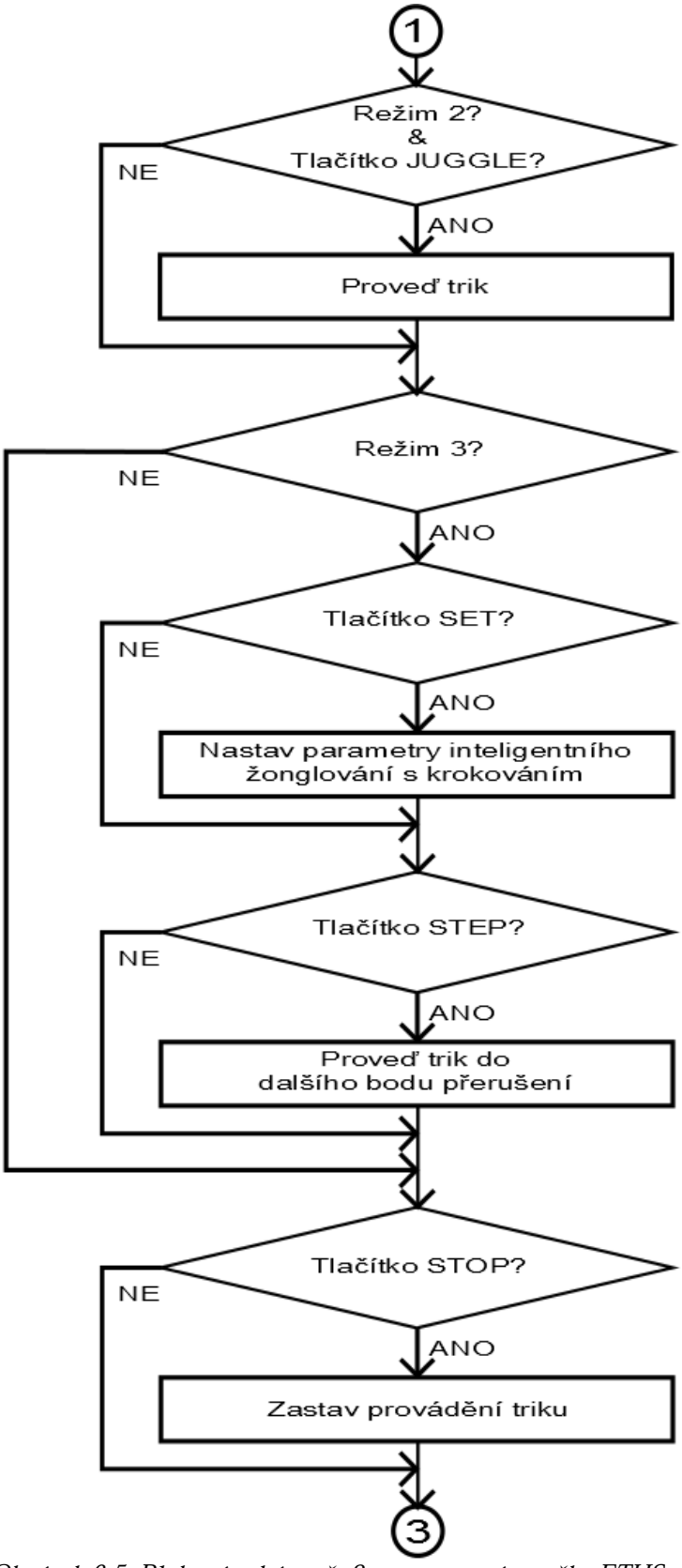
- **tlačítko CONTINUE** – tomuto tlačítku odpovídá bitová hodnota 1110 (dekadicky 14). Tlačítko slouží k uvedení stroje do opětovného provozu po havarijním stavu.

```
IF NOT wMode.X0 & wMode.X1 & wMode.X2 & wMode.X3
        THEN
                (*uvedení stroje do provozního stavu po havarijním zastavení*)
END_IF;
```

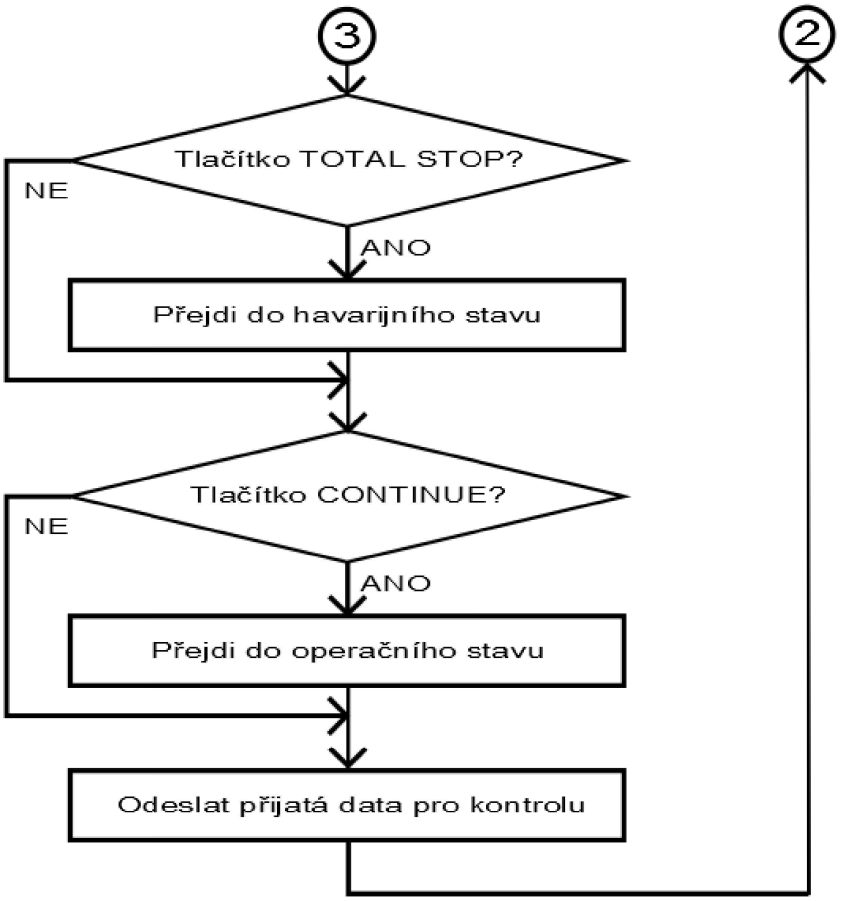
Pro názornější pochopení činnosti programu *ETHSetting* je na obrázcích 6.4, 6.5 a 6.6 zobrazeno blokové schéma celé programové smyčky.



Obrázek 6.4 Blokové schéma č. 1 programové smyčky ETHSetting.



Obrázek 6.5 Blokové schéma č. 2 programové smyčky ETHSetting.



Obrázek 6.6 Blokové schéma č. 3 programové smyčky ETHSetting.

6.1.3 Program Juggling

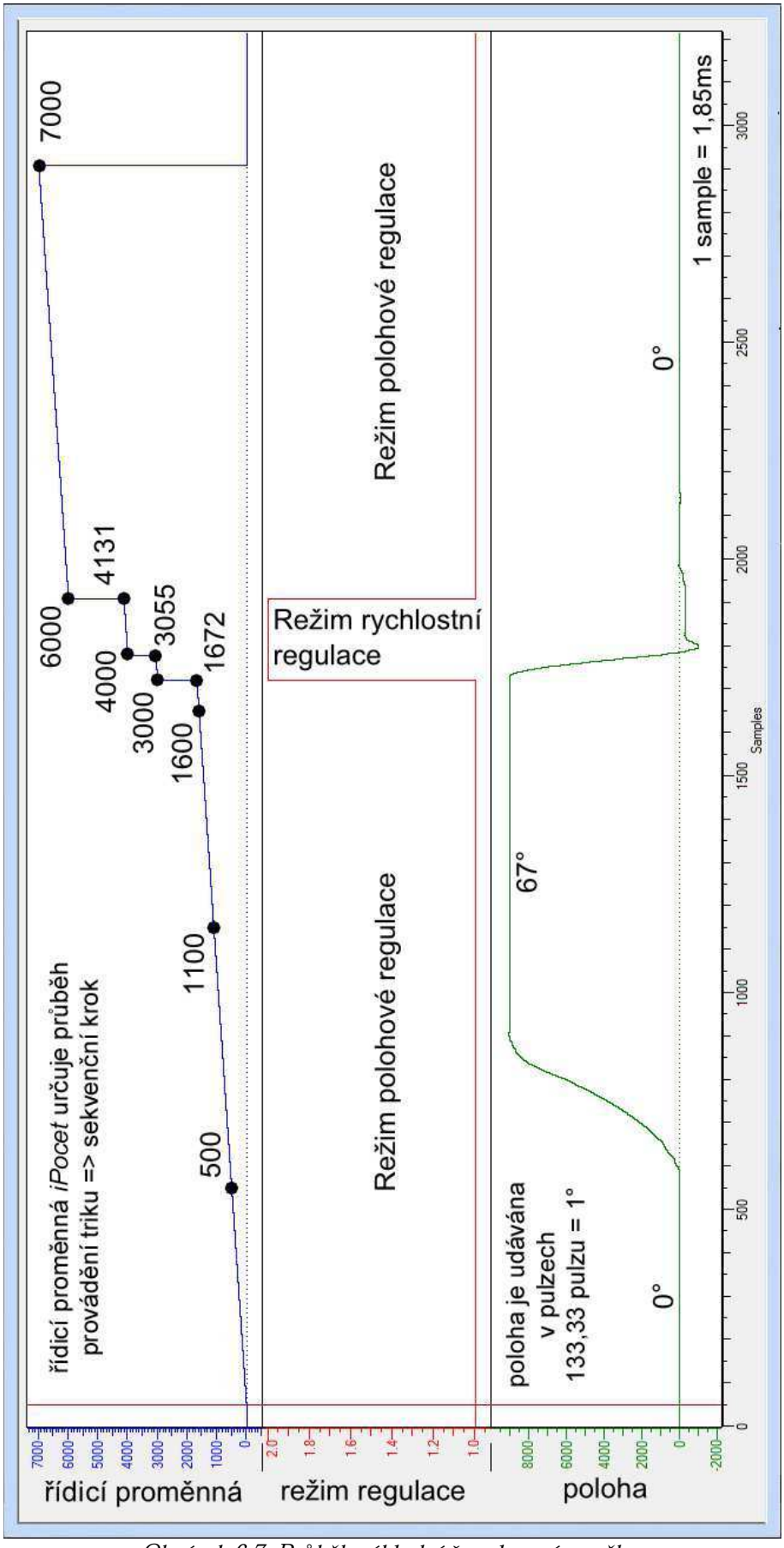
Programová smyčka Juggling má nejdůležitější funkci. Realizuje samotné pohyby ramene pomocí správného nastavení servozesilovače. Hlavní částí programové smyčky je realizace základní žonglovací smyčky.

Realizace základní žonglovací smyčky spočívá v sekvenčním provádění příkazů pro ovládání a nastavování servozesilovače. Vzhledem k tomu, že program v programovatelném automatu je prováděn cyklicky a celou žonglovací smyčku nelze provést v jednom programovém cyklu, je vytvořena řídicí proměnná *iPocet*, která určuje jakou operaci ze sekvenční posloupnosti je nutné právě vykonávat. Řídicí proměnná *iPocet* se v každém cyklu zvětšuje o jedničku. Jednotlivým operacím je přiřazeno přímo číslo nebo interval, které určuje kdy lze danou operaci provést. Čísla se nepřekrývají, jsou tedy unikátní a operace jsou očíslovány v pořadí, ve kterém je nutné je vykonat. Pokud řídicí proměnná *iPocet* dosáhne hodnoty pro určitou operaci, je tato operace vykonána v právě běžícím programovém cyklu. V každém programovém cyklu je tedy vykonána jedna nebo žádná operace. Provádění triku není přesně navazující sekvencí úkonů, při vykonávání triku je potřeba také určitých prodlev. Z tohoto důvodu nejsou operace přiřazeny přímo po sobě jdoucím hodnotám řídicí proměnné *iPocet*. Pomocí rozdělení hodnot řídicí proměnné *iPocet* jednotlivým operacím nebo „pauzám“ lze určovat časový průběh jednotlivých operací při vykonávání žonglovacího triku. Jeden programový cyklus trvá 1,85ms proto každých 1000 provedených cyklů odpovídá 1,85 sekundy. Každých 1000 hodnot řídicí proměnné *iPocet* tedy odpovídá času 1,85 sekundy. Pro zjednodušení časových údajů při vysvětlování funkce programu je použito pro 1000 hodnot řídicí proměnné *iPocet* časový údaj 2 sekundy.

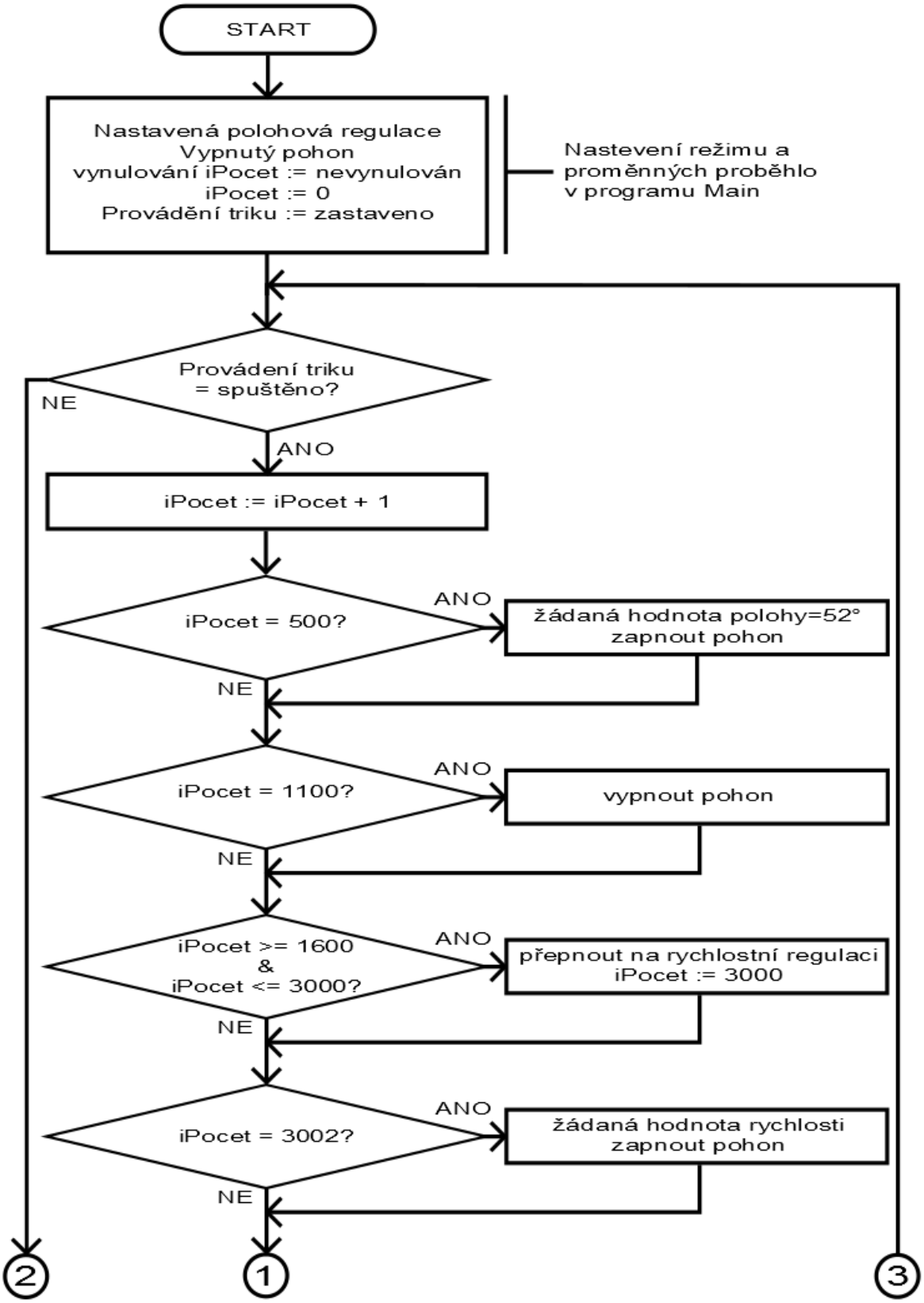
Po obdržení povelu pro začátek provádění triku je stanovena přibližně sekundová prodleva než začne samotný trik probíhat, to odpovídá hodnotě 500 řídicí proměnné *iPocet*. Pro provedení celé základní žonglovací smyčky je nutné provést tyto příkazy:

1. Nastavení žádané hodnoty polohy na 67°, zapnutí pohonu. Po zapnutí pohonu se začne rameno pohybovat na žádanou pozici. Na nastavení hodnot do servozesilovače a provedení pohybu je vyhrazen čas přibližně 1 sekundy, což odpovídá rozdílu hodnoty řídicí proměnné *iPocet* pro druhou operaci a hodnoty řídicí proměnné *iPocet* pro první operaci (1100 – 500 = 600). Tento čas je dostatečný pro ustálení ramene na požadované poloze.
2. Vypnutí pohonu přibližně na dobu jedné sekundy a ponechání pohonu ve vypnutém stavu.
3. Přepnutí režimu polohové regulace na rychlostní. Přepnutí trvá určitou dobu. Prodleva je způsobena dobou přenosu dat po lokální sběrnici INTERBUS a dobou zpracování dat servozesilovačem. Pro ideální časový průběh provádění této operace je pro ni nastaven interval hodnot řídicí proměnné *iPocet,* v kterém je testováno potvrzení z odeslaného příkazu pro přepnutí regulačního režimu. V okamžiku obdržení kladného potvrzení je řídicí proměnná *iPocet* nastavena na horní mez intervalu a v příštím programovém cyklu proto pokračuje dalšími operacemi. Díky tomu není při přepínání zbytečná časová prodleva, která nesouvisí se samotným přepnutím režimu.
4. Nastavení žádané hodnoty odhozové rychlosti, zapnutí pohonu. Po zapnutí pohonu je rameno urychlováno na žádanou hodnotu v co nejkratším čase, aby v době odhozu byla rychlost otáčení ramene na stanovené hodnotě.
5. Testování požadované odhozové polohy. Testování probíhá v dalším programovém cyklu od počátku pohybu ramene po dobu přibližně 2 sekund. Tento čas je mnohonásobně předimenzován, protože rameno dosáhne požadované odhozové polohy pouze ve zlomku sekundy. Pokud je kladně vyhodnoceno dosažení odhozového úhlu, je jako žádaná hodnota rychlosti nastavena rychlost opačného směru a vysoké hodnoty, a řídicí proměnná *iPocet* je nastavena na horní mez intervalu, ve kterém je testováno dosažení požadované polohy. Opačná rychlost způsobuje brzdění. Pro odhození míčku je nejdůležitější okamžik, ve kterém se brzdit začíná, proto je brzdná rychlost nastavena jen na dobu 30 programových cyklu (přibližně 60 ms).
6. Nastavení žádané hodnoty rychlosti na nulu. To způsobí zastavení motoru.
7. Přepnutí režimu do polohové regulace. Přepnutí režimu je realizováno identicky jako při přepínání z polohové regulace na rychlostní. Nejprve je odeslán příkaz pro přepnutí režimu do servozesilovače, poté je testováno potvrzení ze servozesilovače. V okamžiku kladného potvrzení je ukončeno testování nastavením řídicí proměnné *iPocet* na hodnotu horní meze intervalu, v kterém se tato operace může provádět.
8. Nastavení žádané hodnoty polohy na nulu, tedy na klidovou polohu. Tato poloha odpovídá místu dopadu míčku. V programu je nastavena maximální doba pohybu ramena na nulovou polohu na přibližně 2 sekundy. Pro zajištění správného vykonání triku je však zapotřebí dosáhnout dopadové polohy přibližně v polovičním čase. Časová rezerva je ponechána z důvodu ustálení ramene na nulové hodnotě i po dopadu míčku, který působí jako poruchová veličina v regulačním obvodu a je potřeba, aby se s ní polohový regulátor vypořádal a uvedl rameno na požadovanou nulovou polohu.
9. Zastavení provádění triku. Provádění triku je zastaveno nastavením proměnné určující zda se má trik provádět či nikoli na hodnotu, která odpovídá neprovádění triku. Dále je vynulována řídicí proměnná *iPocet*. Tato proměnná je v každém cyklu testována na nulovou hodnotu a v případě že ji obsahuje je udržován pohon ve vypnutém stavu. Toto opatření zabraňuje zbytečnému přetěžování pohonu a udržování pohonu pod napájecím napětím i v době kdy není prováděn trik.

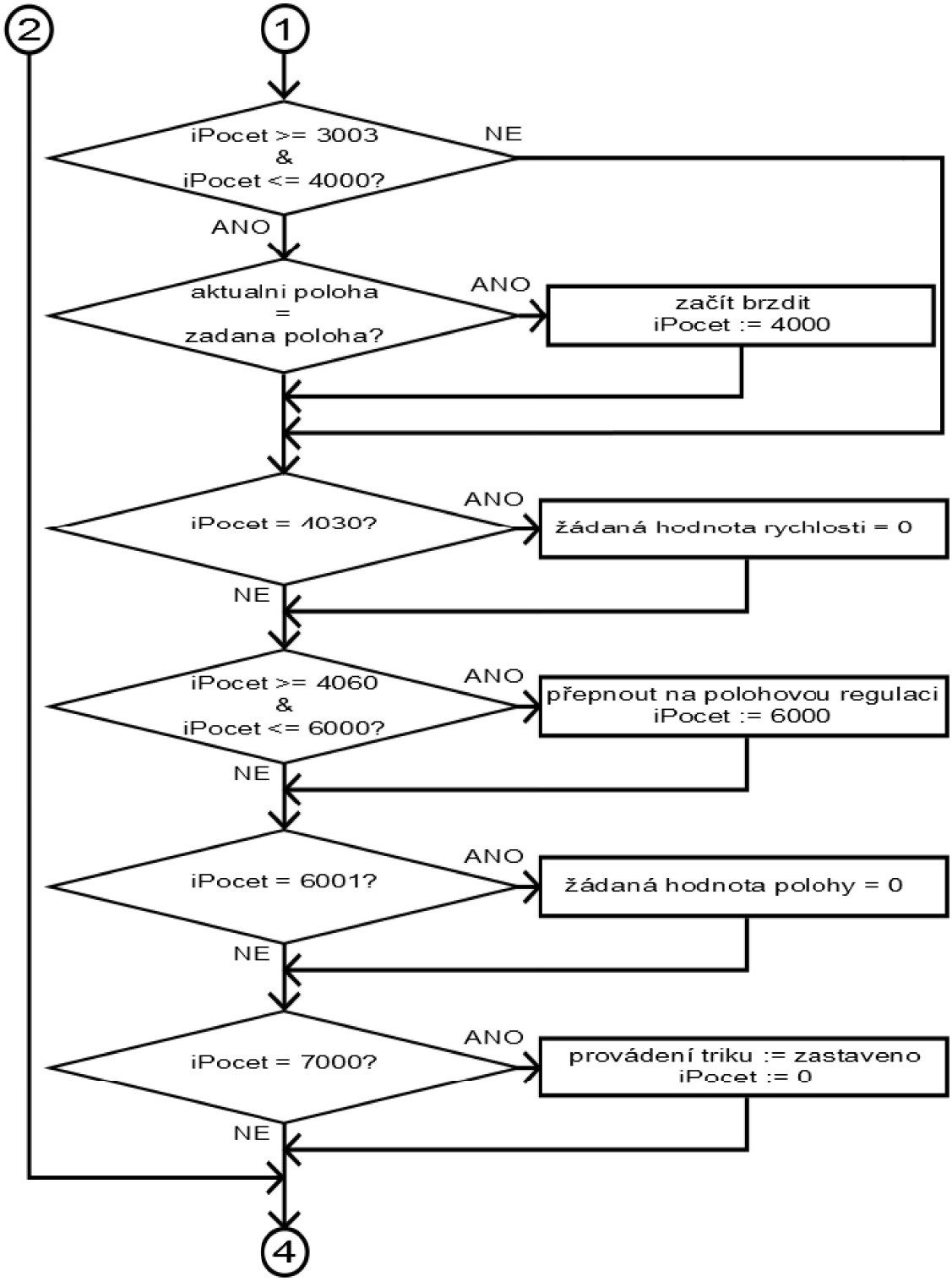
Popsaná sekvence operací, které jsou nutné pro provedení základní žonglovací smyčky jsou graficky znázorněny na obrázcích 6.8 a 6.9. Testování řídicí proměnné *iPocet* v nulové hodnotě je zobrazeno jako poslední podmíněný příkaz v blokovém schématu 3, tedy na obrázku 6.10. Na obrázku 6.7 je znázorněn průběh polohy, přepínání režimů a hodnoty řídicí proměnné *iPocet* během vykonávání základní žonglovací smyčky.



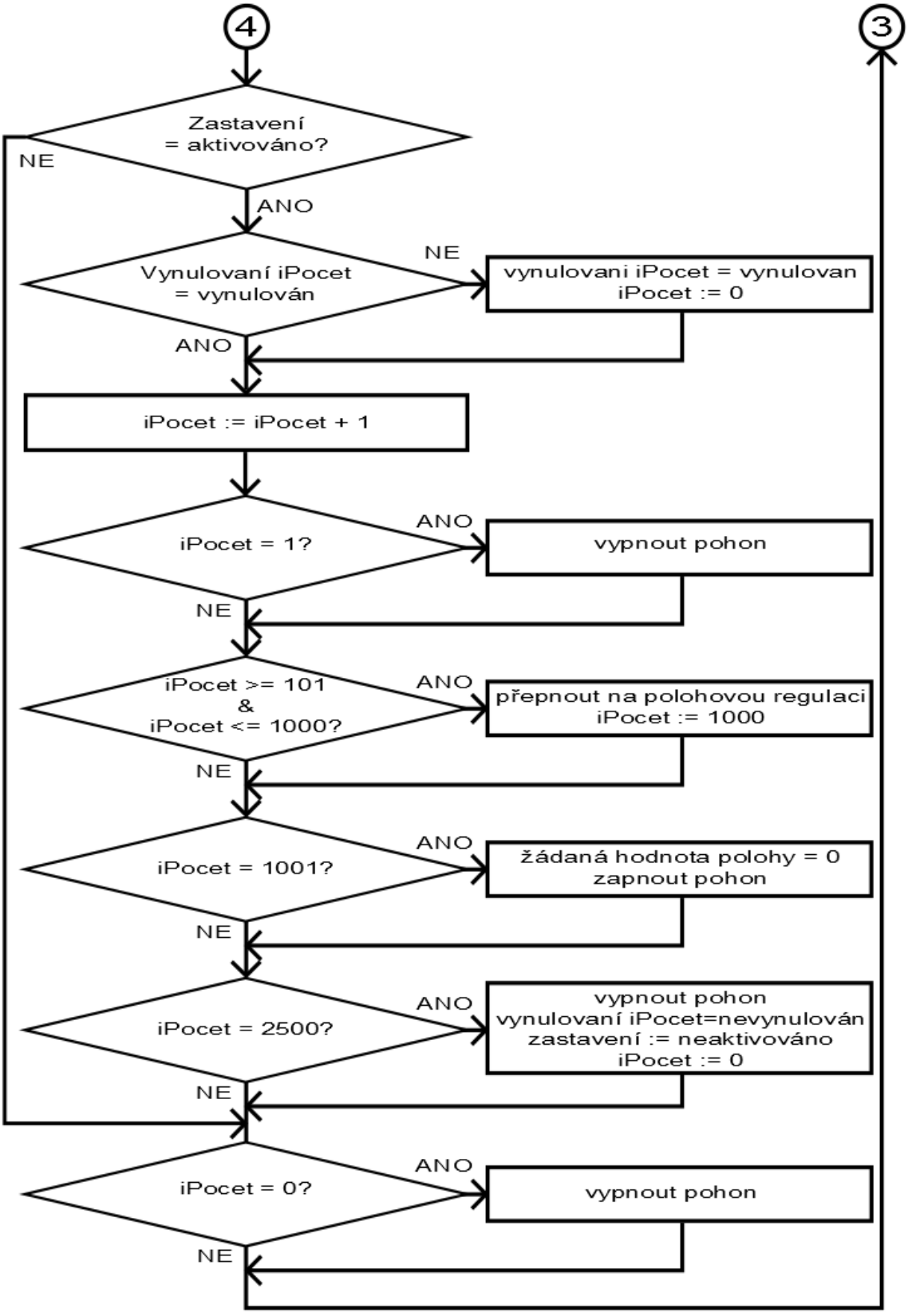
Obrázek 6.7 Průběh základní žonglovací smyčky.



Obrázek 6.8 Blokové schéma č. 1 programové smyčky Juggling.



Obrázek 6.9 Blokové schéma č. 2 programové smyčky Juggling.



Obrázek 6.10 Blokové schéma č. 3 programové smyčky Juggling.

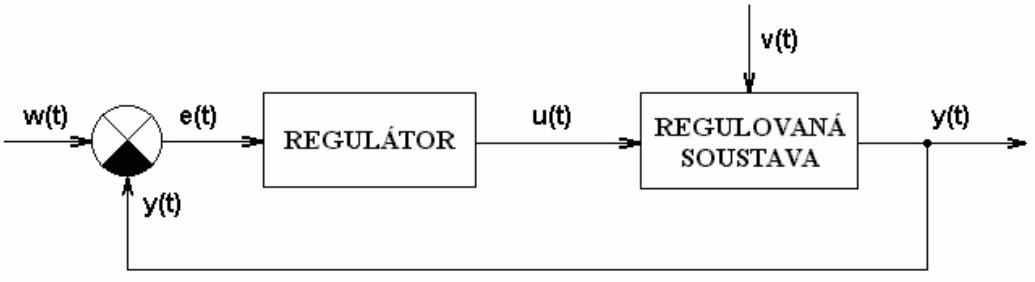
Na posledním obrázku blokového schématu zobrazujícího programovou smyčku *Juggling* je graficky znázorněna sekvence operací, které jsou prováděny při zastavování právě prováděného triku pomocí tlačítka STOP. Vzhledem k tomu, že tlačítko jen nepřeruší trik, ale po přerušení vykonávání triku přesune rameno na nulovou polohu, je nutné provést řízení pohonu v polohovém regulačním režimu. Realizace sekvenčního vykonávání operací je stejná jako u základní žonglovací smyčky, tedy pomocí řídicí proměnné *iPocet*. Po stisku tlačítka STOP je zastaveno provádění triku nastavením proměnné, která určuje zda se trik provádí či nikoli na hodnotu, při které se trik neprovádí. Zároveň je kladně nastavena proměnná určující, že se bude zastavovat. Vzhledem k tomu, že tlačítko stop lze stisknout v jakýkoliv okamžik provádění triku, obsahuje řídicí proměnná *iPocet* neznámou hodnotu, je tedy nutné proměnnou vynulovat, aby mohla sekvence pro zastavení začít od začátku. K vynulování musí dojít právě jednou při prvním průchodu stop smyčky. Proto je použita pomocná proměnná, která zajistí provedení vynulování pouze v prvním kroku. Této operaci odpovídá první podmíněný příkaz ve „stop smyčce“ zobrazené na 3. části blokového schématu programové smyčky *Juggling* na obrázku 6.10. Pro úspěšné provedení „stop smyčky“ je nutné provést v uvedeném pořadí tyto operace:

1. Vypnutí pohonu. To má z následek brzdění.
2. Přepnutí na režim polohové regulace. Přepnutí do tohoto režimu je velice důležité, tlačítko STOP lze stisknout v kterékoliv fázi triku, proto nemusí být polohová regulace právě aktivní. Přepnutí je realizováno stejně jako v základní žonglovací smyčce. Servozesilovači je poslán příkaz, který přepne režim regulace. Po obdržení kladného potvrzení o provedení příkazu je řídicí proměnná *iPocet* nastavena na horní mez intervalu pro vykonávání operace přepínání režimu. V dalším cyklu tedy program pokračuje vykonáváním další operace.
3. Nastavit žádanou hodnotu polohy na nulovou a zapnout pohon. Po zapnutí pohonu se začne rameno pohybovat na nulovou polohu, tedy do vodorovné pozice. Jedná se o klidovou polohu, proto je zvolena jako poloha, na kterou je rameno přesunuto po zastavení provádění triku.
4. Vypnutí pohonu, zastavení provádění stop smyčky, přenastavení pomocné nulovací proměnné a vynulování řídicí proměnné *iPocet*. Přenastavení pomocné nulovací proměnné je nutné pro zajištění správného vynulování řídicí proměnné pří dalším stisknutí tlačítka stop. Vynulováním řídicí proměnné *iPocet* je udržován vypnutý pohon při neprovádění triku, stejně jako tomu je při ukončení základní žonglovací smyčky.

Tlačítko stop lze použít i mimo provádění triku. Stop smyčka i z klidového stavu stroje nataví rameno na nulovou polohu. Tohoto lze využít pro ustálení ramene ve vodorovné poloze před provedením samotného triku v případě, že rameno je z nějakého důvodu v nestandardní poloze.

6.2 Popis regulací

Regulátory použité v tomto žonglovacím stroji resp. v servozesilovači mají velký vliv na přesnost ovládání ramene zařízení a tedy na kvalitu žonglování. Regulátory mají za úkol udržovat regulovanou veličinu na požadované hodnotě i při případném působení poruch. Schéma obecného regulátoru je zobrazeno na obrázku 6.11. Obecné poznatky o regulátorech jsou čerpány z [8].



Obrázek 6.11 Blokové schéma obecného regulačního obvodu.

Jednotlivé veličiny ve schématu na obrázku 6.11 mají význam:

- **w(t)** – žádaná hodnota
- **e(t)** – regulační odchylka
- **u(t)** – akční veličina
- **v(t)** – poruchová veličina
- **y(t)** – regulovaná veličina

Při regulaci je v regulačním obvodu vyhodnocován rozdíl žádané hodnoty a aktuální hodnoty regulované veličiny, který vytváří regulační odchylku.

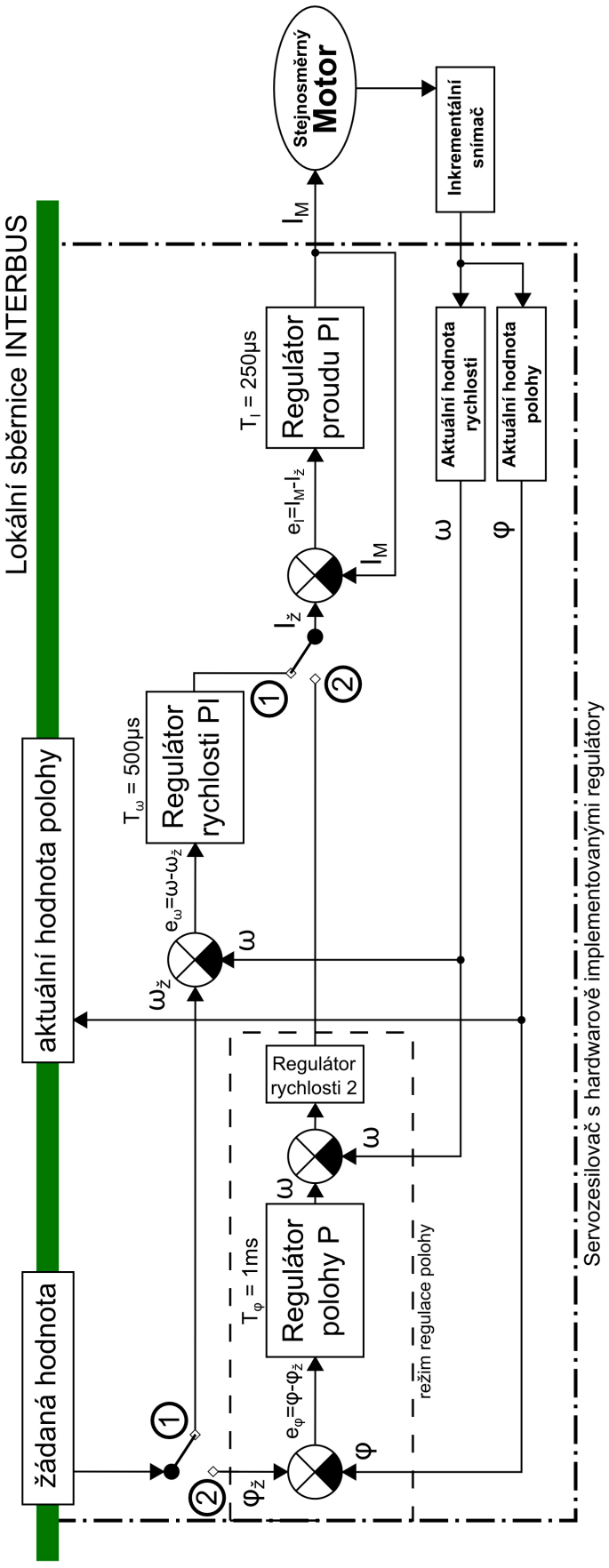
$$e(t) = w(t) - y(t) \tag{6.1}$$

Pokud aktuální hodnota regulované veličiny a žádaná hodnota nemají stejnou velikost, má regulační odchylka nenulovou hodnotu. Pokud je nenulová regulační odchylka, reaguje na to regulační obvod akční veličinou, kterou se snaží upravit regulovanou veličinu na velikost žádané hodnoty.

Vzhledem k tomu, že v servozesilovači je více regulátorů a všechny jsou využity při řešení, je schéma celého regulačního obvodu komplikovanější než na obrázku 6.11. Schéma obsahující použité regulátory a vazby na proměnné, které realizují vnější styk se servozesilovačem, je na obrázku 6.12. Pro režim rychlostní regulace, jsou v činnosti regulátory rychlosti a proudu. Tato konfigurace je použita při přepnutém přepínači do polohy 1. Při přepnutí přepínače do polohy 2 je v činnosti režim polohové regulace, který využívá proudový, rychlostní a polohový regulátor. V tomto případě se nejedná o stejný rychlostní regulátor jako při konfiguraci 1. Polohový regulátor využívá odlišný rychlostní regulátor s jinými parametry.

Vysvětlivky k blokovému schématu na obrázku 6.12:

- T_φ – perioda vyhodnocování polohového regulátoru = 1ms
- T_ω – perioda vyhodnocování rychlostního regulátoru = 500μs
- T_I – perioda vyhodnocování proudového/momentového regulátoru = 250μs
- I_M – aktuální hodnota proudu motoru
- ω – aktuální hodnota rychlosti motoru
- φ – aktuální hodnota polohy motoru
- $I_Ž$ – žádaná hodnota proudu generovaná jedním z rychlostních regulátorů
- $\omega_Ž$ – žádaná hodnota rychlosti přijatá z programovatelného automatu
- $\varphi_Ž$ – žádaná hodnota polohy přijatá z programovatelného automatu
- e_I – regulační odchylka proudu
- e_ω – regulační odchylka rychlosti
- e_φ – regulační odchylka polohy



Obrázek 6.12 Regulační obvod žonglovacího zařízení.

6.2.1 Proudový regulační obvod

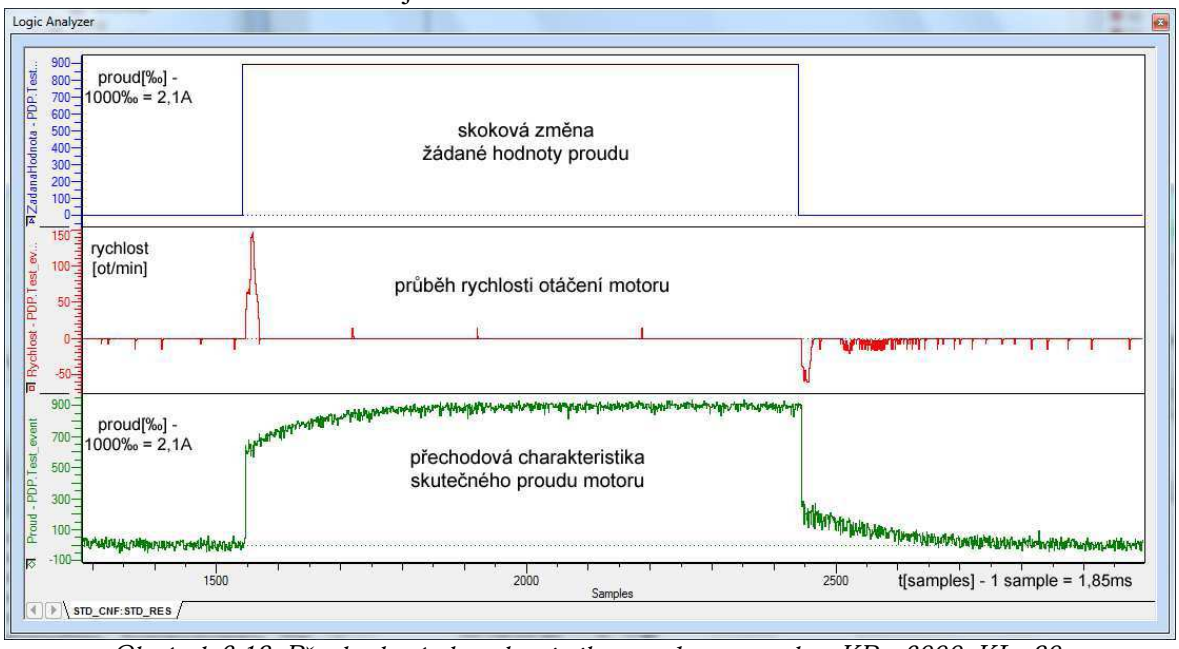
Proudový regulační obvod je poslední obvod, který je zapojen v celé regulační smyčce. Je v činnosti po celou dobu provozu zařízení, protože ho používají jak polohový režim tak rychlostní. Akční zásah, který vytváří, ovládá přímo motor.

Díky tomu, že proudový regulátor je hardwarově realizován v servozesilovači, běží s velice malou periodou regulace, která je 250µs. Výrobcem přednastavené konstanty regulátoru mají hodnoty:

- proporcionální zesílení – $KP_I = 200$
- integrační časová konstanta – $KI_I = 30$

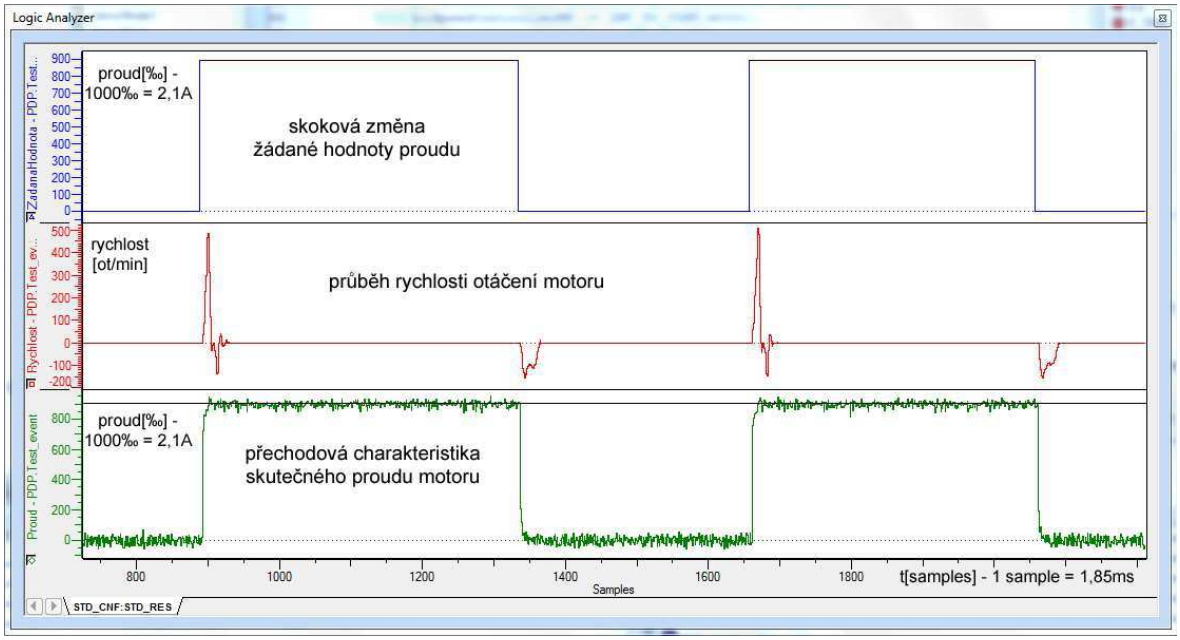
Při použití přednastavených konstant má proudový regulátor velice špatné dynamické vlastnosti. Vzhledem k povaze stroje je nutné využívat maximální dynamický potenciál regulátorů. Z tohoto důvodu je zapotřebí konstanty proudového regulátoru výrazně zvýšit.

První úprava proběhla u proporcionálního zesílení, které je potřeba zvýšit pro rychlejší nástup regulované veličiny. Jako dostačující hodnota proporcionálního zesílení je stanoveno $KP_I = 6000$. Pro zobrazování přechodových charakteristik při nastavování regulátoru proudu je využit analyzátor, který je obsažen v softwaru PC WORX. První vykreslovaná veličina (modrá) je žádaná hodnota proudu v promilích jmenovitého proudu motoru. Druhá vykreslovaná veličina (červená) je rychlost otáčení motoru v otáčkách za minutu a poslední vykreslovaná veličina (zelená) je proud motoru opět v promilích jmenovitého proudu motoru, tedy nejdůležitější veličina při nastavování proudového regulátoru. Pro nastavování regulátoru je zvolena žádaná hodnota proudu o velikosti 90% jmenovitého proudu motoru, tedy 1,85 ampéry. Při nastavování proudového regulátoru je vhodné mít zafiksované rameno aby nedocházelo k jeho pohybu. Při nastavování bylo rameno zafiksováno, ale k pohybu ramene i tak došlo. Bylo to způsobeno nedostatečnou tuhostí ramene stroje a fiksačního zařízení.



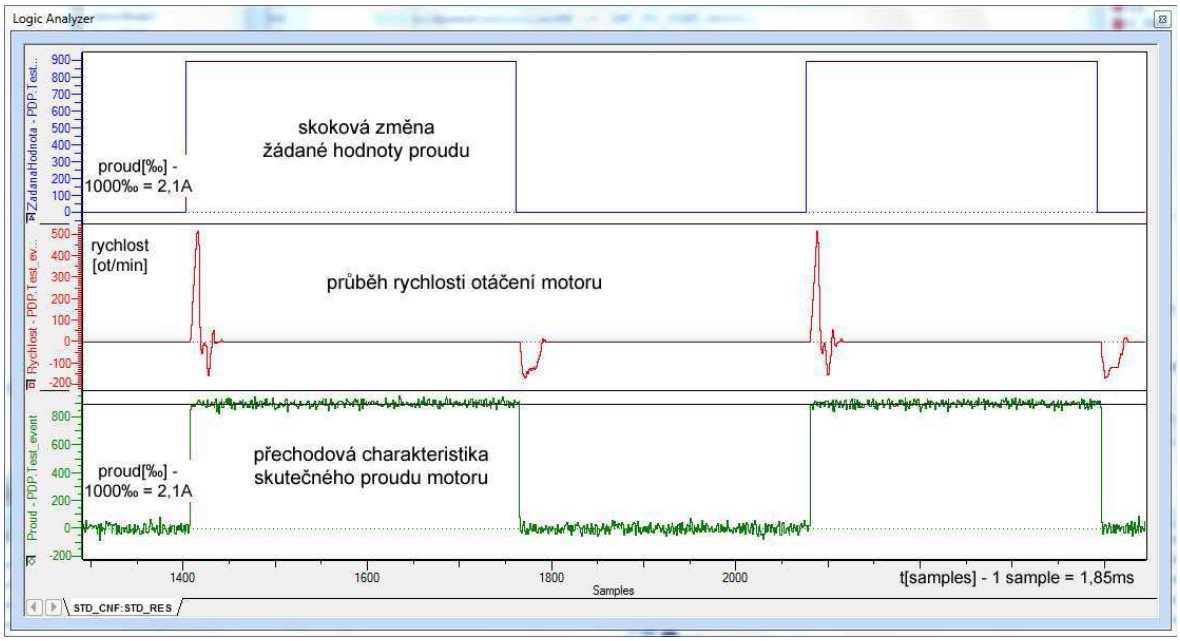
Obrázek 6.13 Přechodová charakteristika regulace proudu s KP_I=6000, KI_I=30.

Při ještě větším zesílení už nebyl průběh proudu výrazně lepší, proto je jako proporcionální zesílení zvolena hodnota $KP_I = 6000$. Přechodovou charakteristiku s hodnotami proporcionálního zesílení $KP_I = 6000$ a integrační časové konstanty $KI_I = 30$, tedy přednastavená velikost, je zobrazena na obrázku 6.13. Pro další zlepšení průběhu proudu je proto potřeba upravit integrační časovou konstantu. Opět zvýšením stejně jako u proporcionálního zesílení. Ke zvyšování integrační časové konstanty došlo několikrát. Výrazného zlepšení průběhu proudu je dosaženo při hodnotě konstanty $KI_I = 480$, kdy je průběh už velice dobrý. Přechodová charakteristika regulátoru s právě touto hodnotou časové integrační konstanty je zobrazena na obrázku 6.14.



Obrázek 6.14 Přechodová charakteristika regulace proudu s KP_I=6000, KI_I=480.

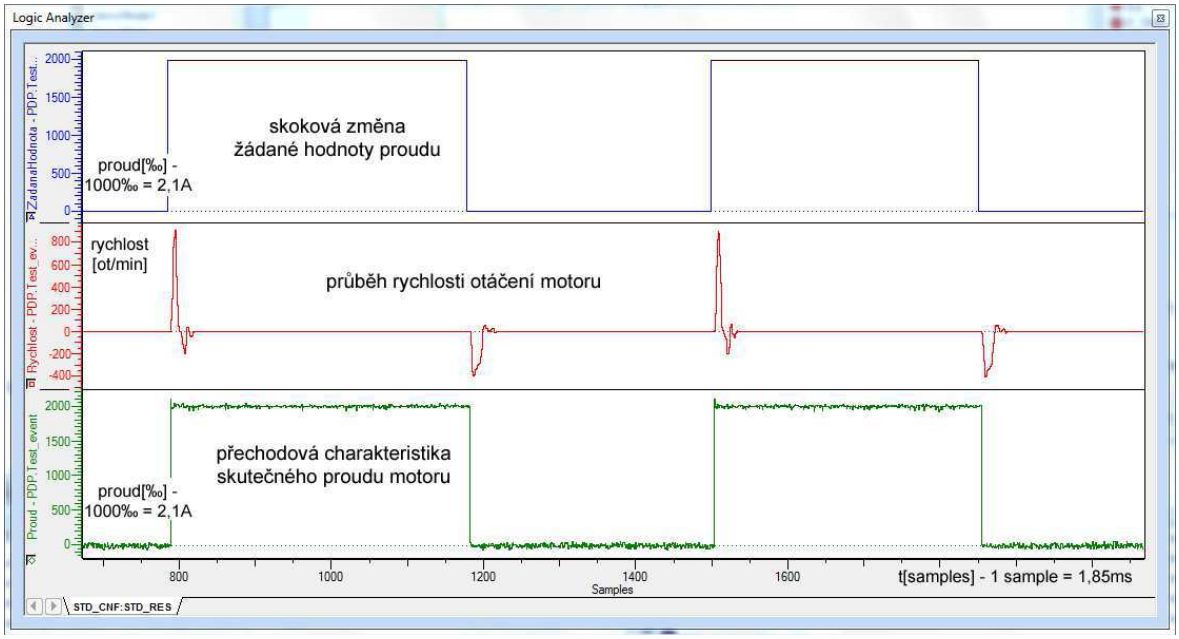
Jak je vidět na přechodové charakteristice z obrázku 6.14, došlo k výraznému zlepšení průběhu proudu. Průběh ale stále není ideální a s dalším zvýšením integrační časové konstanty lze dosáhnout ještě lepšího náběhu regulované veličiny. Hodnota konstanty se při zvyšování zastavila na čísle KI_I = 2000. Při tomto nastavení regulátoru má charakteristika proudu téměř ideální průběh. Přechodová charakteristika s tímto natavením je zobrazena na obrázku 6.15.



Obrázek 6.15 Přechodová charakteristika regulace proudu s KP_I=6000, KI_I=2000.

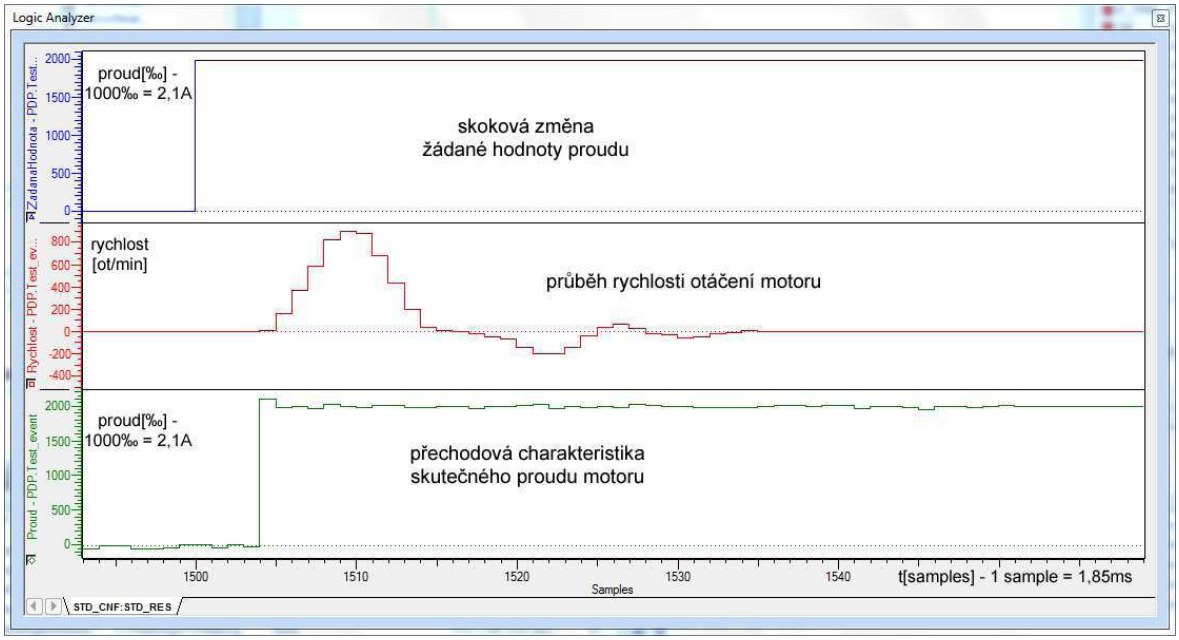
Další zvyšování integrační časové konstanty už vedlo k zhoršení průběhu. Právě proto jsou jako konečné parametry nastavení regulátoru proudu zvoleny hodnoty proporcionálního zesílení KP_I = 6000 a integrační časové konstanty KI_I = 2000. Pro ověření správné funkčnosti žonglovacího stroje je nutné zkontrolovat průběh proudu při vyšších žádaných hodnotách. Při provozu zařízení se proudy nepohybují pouze v tak malých hodnotách. Pro ověření funkčnosti regulátoru v celém spektru možných žádaných hodnot proudu je další velikost žádané hodnoty stanovena na 200% jmenovitého proudu motoru, tedy 4,2 ampéry. Přechodová charakteristika je zobrazena na obrázku

6.16. Charakteristika proudu sice již obsahuje drobný překmit, ale pro zajištění maximální dynamiky je výhodnější překmit než pozvolnější náběh regulované veličiny.



Obrázek 6.16 Přechodová charakteristika regulace proudu s vyšší žádanou hodnotou.

Pro lepší posouzení průběhu proudu při náběhu je na obrázku 6.17 zobrazen detail přechodové charakteristiky z obrázku 6.16. Na charakteristice je zřetelně vidět drobný překmit a velice rychlé ustálení na žádané hodnotě.



Obrázek 6.17 Detail přechodové charakteristiky regulace proudu.

Konečným nastavením proudového regulátoru s nejlepším průběhem regulované veličiny je:

- proporcionální zesílení – $KP_I = 6000$
- integrační časová konstanta – $KI_I = 2000$

6.2.2 Rychlostní regulační obvod

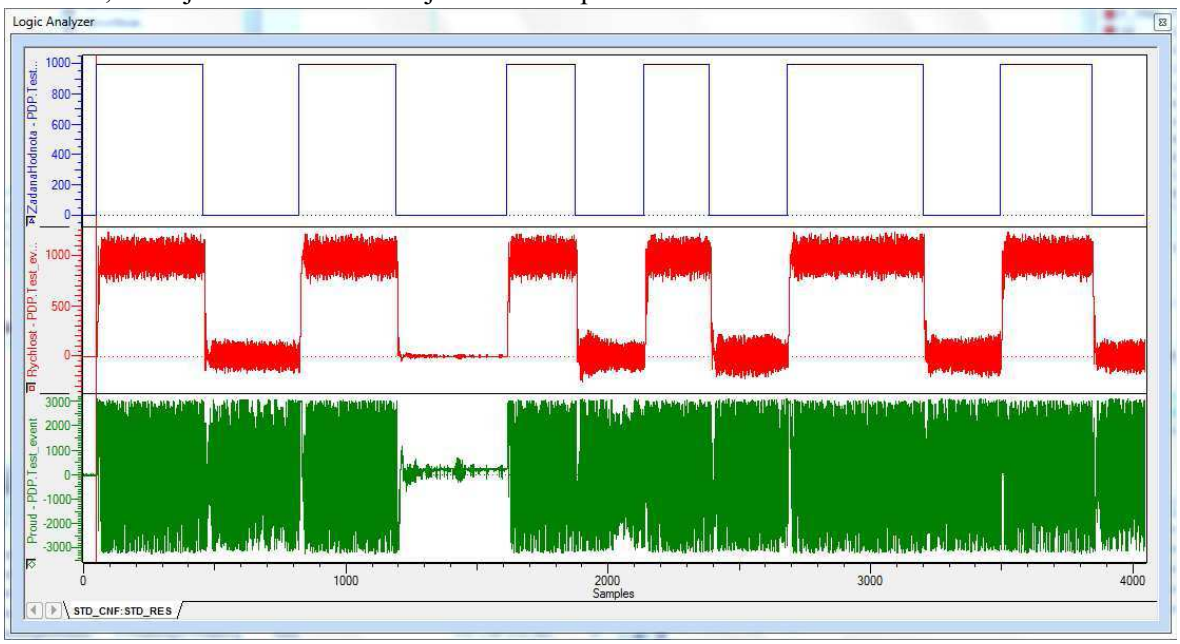
Rychlostní regulační obvod je nejdůležitější regulační obvod při realizaci žonglování. Má největší vliv na přesnost odhození míčku, protože realizuje hodnotu odhozové rychlosti, která je pro odhození míčku zásadní.

Rychlostní regulátor je opět hardwarově implementován v servozesilovači a proto disponuje velice malou periodou činnosti, která činí 500µs. Regulátor má přednastavené hodnoty regulačních konstant:

- proporcionální zesílení – $KP_\omega = 800$
- integrační časová konstanta – $KI_\omega = 100$

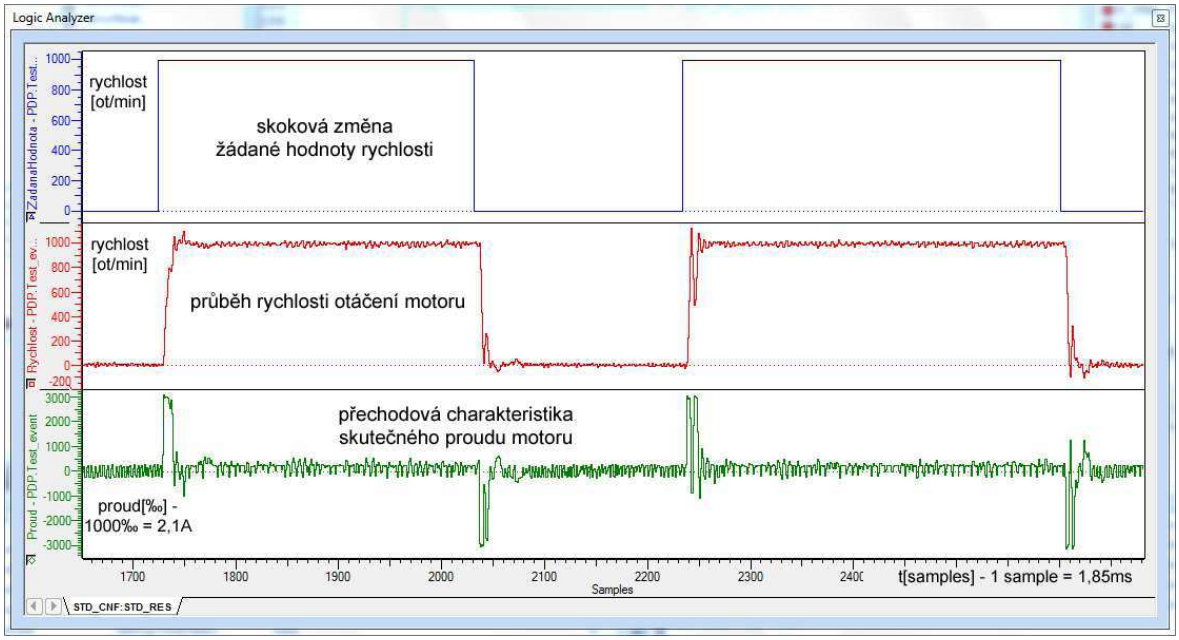
Pro zobrazování přechodových charakteristik je opět využit analyzátor implementovaný v softwaru PC WORX. V analyzátoru jsou zachyceny 3 proměnné. První je žádaná hodnota rychlosti (modrá) v otáčkách za minutu. Druhá veličina je skutečná velikost rychlosti (červená) také v otáčkách za minutu. Poslední veličinou je skutečný proud motoru (zelená) v promilích jmenovitého proudu motoru. Vzhledem k důležitosti rychlostního regulátoru v aplikaci žonglovacího stroje je zapotřebí dosáhnout maximálně rychlého průběhu přechodového děje a ustálení regulované veličiny na co nejpřesnější hodnotě v co nejkratším čase.

Situaci zkomplikovala přítomnost rozběhové rampy v rychlostním obvodu, která nejde v servozesilovači vypnout. Rozběhová rampa určuje strmost náběžné hrany, proto je nastavena na maximální možnou hodnotu. Další komplikací je vůle v převodovce a celé mechanické konstrukci motoru s rameny. Vůle do celého soustrojí vnáší nelinearitu a v některých okamžicích působí jako poruchová veličina. Značně komplikuje celé nastavování regulátoru. Bližší pohled na vliv vůle je uveden v kapitole 6.3. Ukázka nepředvídatelného vlivu vůle je na obrázku 6.18, který zobrazuje rychlostní regulátor v nestabilním stavu a obsahuje ustálenou oblast, která se objevila náhle, bez změny provozních podmínek. Regulátor je v nestabilním stavu při velice vysokých hodnotách proporcionálního zesílení. Konkrétní hodnota proporcionálního zesílení je $KP_\omega = 6000$ a integrační časové konstanty $KI_\omega = 100$. Rozkmit jak rychlosti tak proudu je na neúnosných hodnotách a proud dokonce kmitá do tak vysokých hodnot, že naráží na proudové omezení, které je na hodnotě 300% jmenovitého proudu motoru.



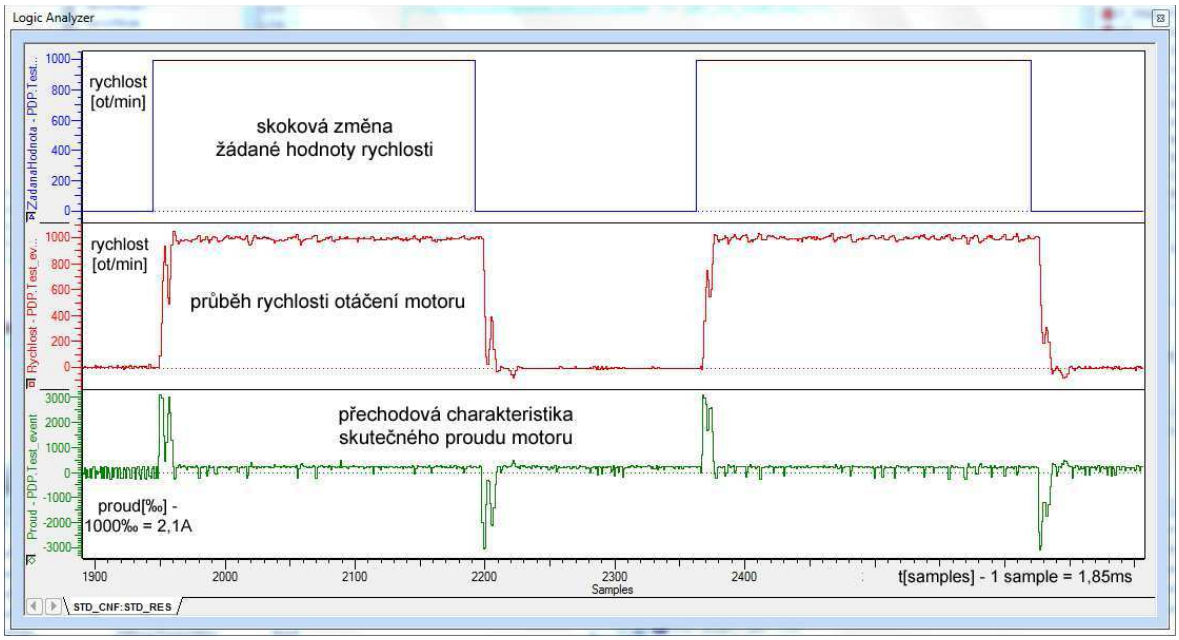
Obrázek 6.18 Nestabilní rychlostní regulační obvod.

Pro odstranění nestability rychlostního regulačního obvodu je zapotřebí snížit hodnoty regulačních konstant. Proto došlo ke snížení proporcionálního zesílení na polovinu a integrační časové konstanty na čtvrtinu. Větší zeslabení integrační časové konstanty je z důvodu zajištění menšího vlivu při nastavování proporcionálního zesílení.



Obrázek 6.19 Přechodová charakteristika regulace rychlosti s KP_ω=3000, KI_ω=25.

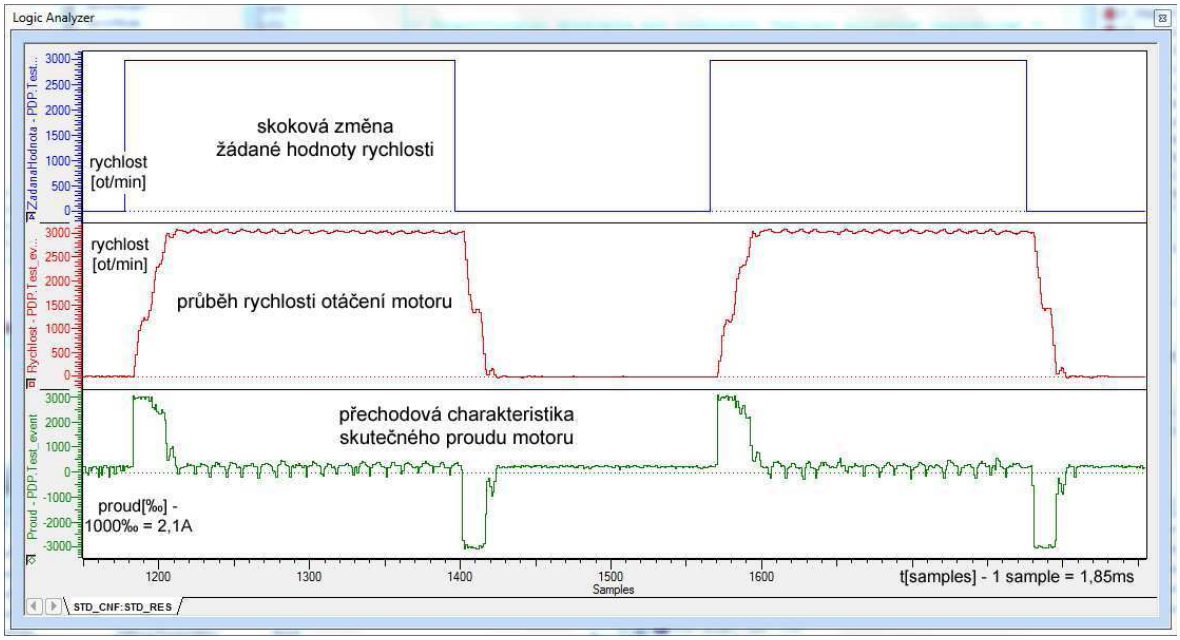
Obrázek 6.19 zobrazuje přechodovou charakteristiku s proporcionálním zesílením $KP_\omega = 3000$ a integrační časovou konstantou $KI_\omega = 25$. Průběh rychlosti není špatný, má rychlý náběh, ale obsahuje překmit. Překmit není u takto důležitého regulátoru žádoucí.



Obrázek 6.20 Přechodová charakteristika regulace rychlosti s KP_ω=1500, KI_ω=25.

Dalším snížením proporcionálního zesílení je odstraněn překmit při zachování strmosti náběhu regulované veličiny. Jak je vidět na obrázku 6.20, je i průběh proudu výrazně lepší, to svědčí o lepším nastavení.

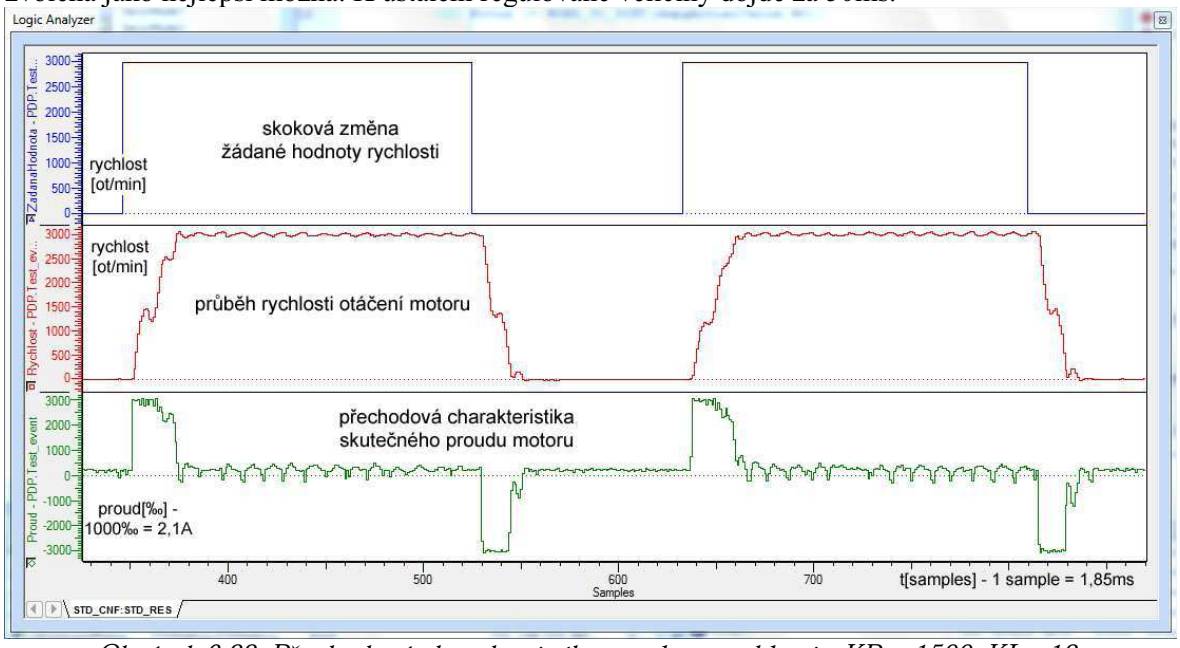
Strmost vzestupné hrany rychlosti je ovšem zavádějící. Dosavadní nastavování rychlostního regulátoru je při velikosti žádané hodnoty rychlosti motoru 1000 otáček za minutu. Tato hodnota je výrazně menší než provozní hodnoty. Při provozu žonglovacího zařízení se otáčky motoru pohybují kolem 3000 otáček za minutu. Je zapotřebí tedy ověřit dobrý průběh i při vyšší žádané hodnotě rychlosti.



Obrázek 6.21 Přechodová charakteristika regulace rychlosti s vyšší žádanou hodnotou.

Provozní rychlosti zařízení se pohybují kolem 3000 otáček za minutu, proto byla jako nová testovací rychlost zvolena právě tato hodnota. Při zachování proporcionálního zesílení na hodnotě $KP_\omega = 1500$ a integrační časové konstanty na hodnotě $KI_\omega = 25$ je na obrázku 6.21 vidět opět dobrý průběh. Charakteristika neobsahuje překmity a má stabilní průběh. Je také vidět pomalejší náběh regulované veličiny, to je způsobeno vyšší žádanou hodnotou.

Při dalším snížení proporcionálního zesílení už dochází ke zpomalení náběhu regulované veličiny. Tato skutečnost není žádoucí, proto je hodnota proporcionálního zesílení $KP_\omega = 1500$ zvolena jako nejlepší možná. K ustálení regulované veličiny dojde za 50ms.



Obrázek 6.22 Přechodová charakteristika regulace rychlosti s KP_ω=1500, KI_ω=12.

Poslední úpravou k nejlepšímu možnému stavu je snížení integrační časové konstanty. Průběh se sníženou konstantou je na obrázku 6.22. Při zachování rychlosti náběhu regulované veličiny dojde ke zlepšení ustálení rychlosti na žádané hodnotě. Při $KP_\omega = 1500$ a $KI_\omega = 12$ je rozkmit ustálené hodnoty rychlosti 2,3%. Tato hodnota je společně s rychlostí ustálení za 50ms nejlepší možnou, které bylo dosaženo.

6.2.3 Polohový regulační obvod

Posledním regulačním obvodem použitým při řešení je polohový regulační obvod. Regulační obvod je zapojen v kaskádě – proudový regulační obvod, rychlostní regulační obvod a polohový regulační obvod. Při využití polohového regulačního obvodu je v činnosti odlišný rychlostní regulační obvod než tomu bylo při použití pouze rychlostního režimu.

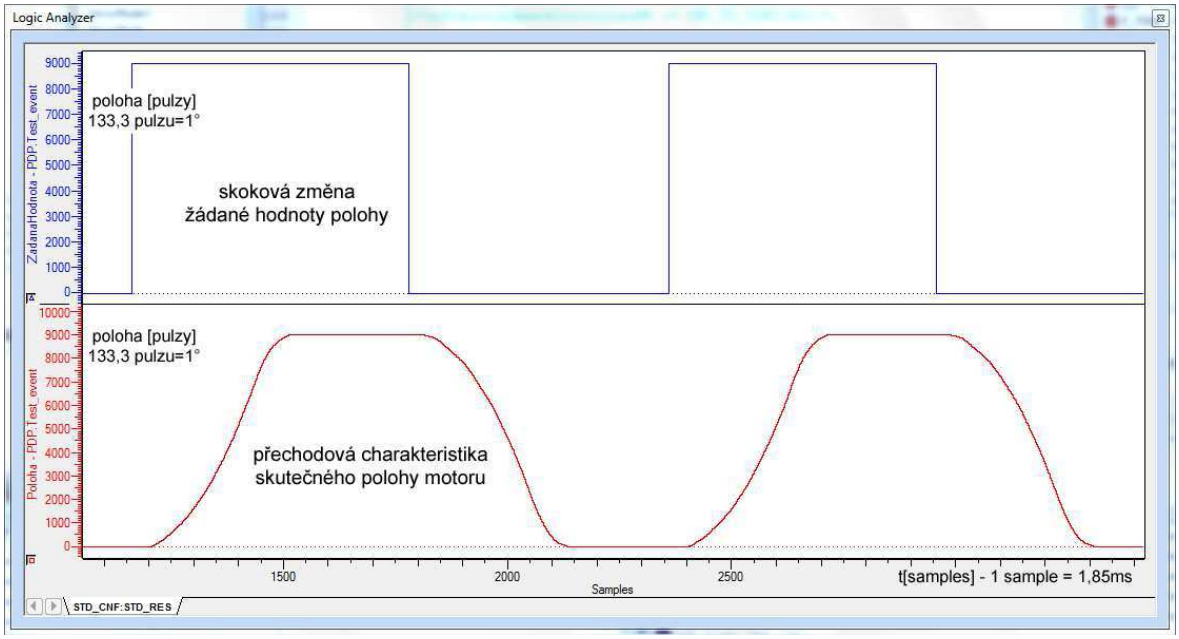
Jak polohový regulátor tak rychlostní regulátor č. 2 jsou opět hardwarově implementovány. Perioda jejich vyhodnocování je opět velice malá, konkrétně má hodnotu 1ms. Přednastavené parametry regulátorů od výrobce mají hodnoty:

- proporcionální zesílení regulátoru polohy – $KP_{\varphi} = 1000$
- integrační časová konstanta regulátoru polohy – $KI_{\varphi} = 0$
- derivační časová konstanta regulátoru polohy – $KD_{\varphi} = 0$
- proporcionální zesílení regulátoru rychlosti č. 2 – $KP_{\omega 2} = 40$
- integrační časová konstanta regulátoru rychlosti č. 2 – $KI_{\omega 2} = 20$

Stejně jako tomu je u předchozích regulátorů i u tohoto je pro zobrazování přechodových charakteristik použit analyzátor implementovaný v softwaru PC WORX. Vzhledem k tomu, že aktuální poloha zabírá 4 bajty v procesní komunikaci, není na jinou veličinu již místo a nelze ji tedy vykreslovat. Z tohoto důvodu obsahují obrázky pouze žádanou hodnotu polohy (modrá) a aktuální hodnotu polohy (červená). Obě proměnné jsou udávány v pulzech. Jedna otáčka ramene o 360° provede 48000 pulzů, 1° tedy odpovídá 133,33 pulzu.

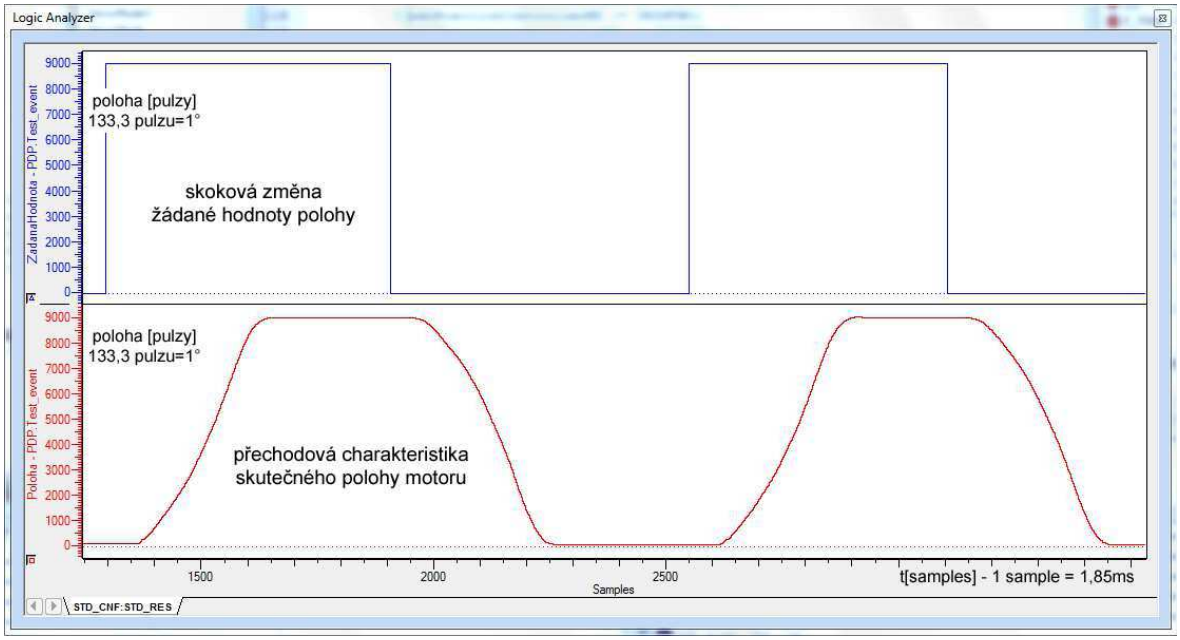
Při nastavování polohového regulátoru je nutné nastavit i rychlostní regulátor č. 2, který nelze nastavovat separátně. Je tedy zvolen postup, při kterém se nejdříve nastaví polohový regulátor a případné korekce se provedou pozdějším nastavením rychlostního regulátoru č. 2.

Pro testování byla zvolna jako žádaná hodnota polohy 9000, která odpovídá 67°. Tento úhel zároveň odpovídá počátečnímu úhlu základního triku.



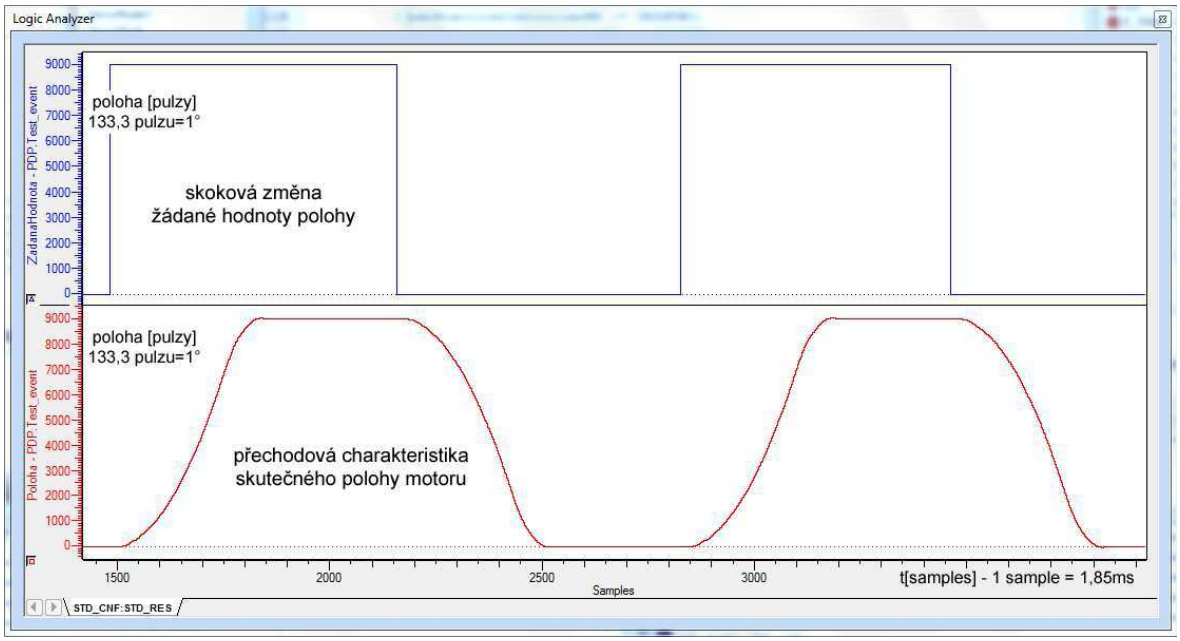
Obrázek 6.23 Přechodová charakteristika regulace polohy s KP_{φ}=1000.

Obrázek 6.23 zobrazuje první zaznamenanou charakteristiku s přednastavenými hodnotami pro proporcionální zesílení $KP_{\varphi} = 1000$, integrační časové konstanty $KI_{\varphi} = 0$ a derivační časové konstanty $KD_{\varphi} = 0$. Jedná se tedy pouze o regulátor typu P, protože integrační i derivační složky jsou vyřazeny. Charakteristika má velice dobrý průběh. Pro činnost tohoto regulátoru není tak důležitá dynamika. Regulátor proto bude mít pozvolnější náběh regulované veličiny. Důležitý však je co nejklidnější průběh a dobré doregulování na požadovanou hodnotu.



Obrázek 6.24 Přechodová charakteristika regulace polohy s KP_{φ}=500.

Na další přechodové charakteristice na obrázku 6.24 je vidět průběh při snížení proporcionálního zesílení na poloviční hodnotu. Průběh je stále dobrý, ale přeci jen už obsahuje drobný překmit a odeznění přechodového děje trvá déle. Z toho vyplývá že zlepšení průběhu charakteristiky nelze dosáhnout snižováním proporcionálního zesílení.



Obrázek 6.25 Přechodová charakteristika regulace polohy s KP_{φ}=2000.

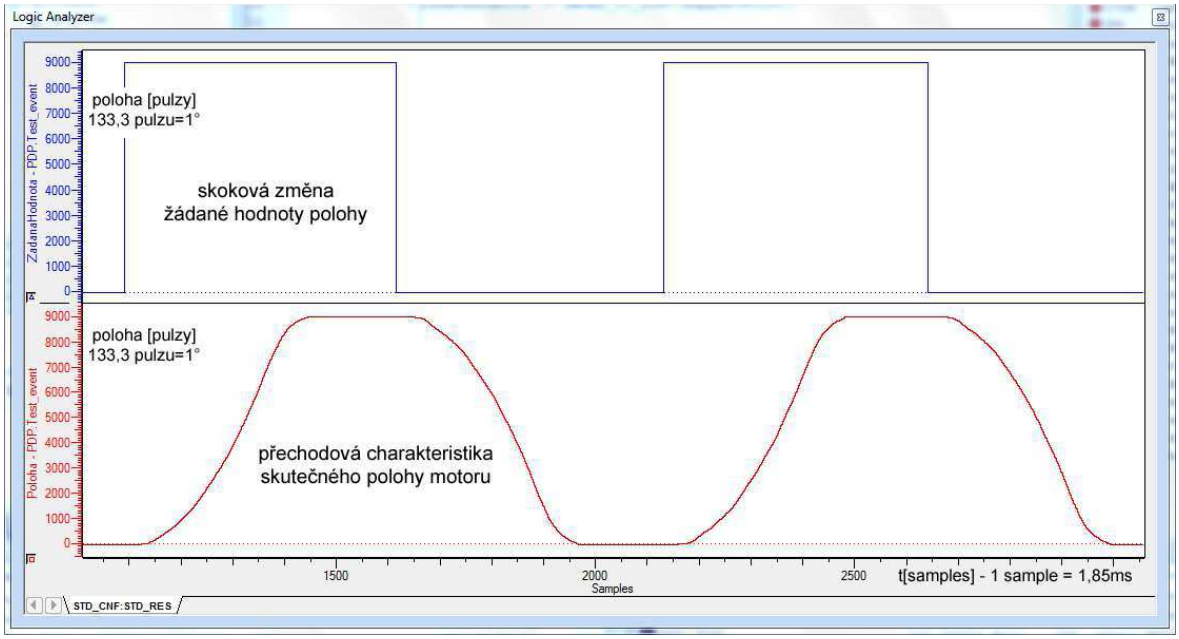
Pokud snižování proporcionálního zesílení nevede k lepšímu průběhu, lze proporcionální zesílení zvýšit, jak ilustruje další obrázek číslo 6.25. Přechodová charakteristika s větším zesílením má nejhorší průběh z provedených měření. Obsahuje jak drobný překmit tak má nejpomalejší ustálení přechodového děje. Z poznatku, že ani zvyšování ani snižování proporcionálního zesílení nevede ke zlepšení charakteristiky je vyvozeno, že nejlepší průběh je s KP_{φ} = 1000.

Nastavení regulátoru s tímto parametrem je velice dobré, ale motor při provozu vydává pronikavý zvuk o vysoké frekvenci. Tato skutečnost naznačuje špatné provozní podmínky a přetěžování motoru. Přetěžování motoru také odpovídala zvýšená provozní teplota motoru.

Řešením hlukového projevu je snížení koeficientů rychlostního regulátoru č. 2. Postupným snižováním koeficientů bylo dosaženo utlumení hluku, který vydával motor, při nezměněném průběhu přechodové charakteristiky. Konečné hodnoty rychlostního regulátoru č. 2 jsou:

- proporcionální zesílení – $KP_{\omega 2} = 10$
- integrační časová konstanta – $KI_{\omega 2} = 5$

Došlo tedy ke snížení parametrů na čtvrtinu. Průběh přechodové charakteristiky nastaveného polohového regulátoru je na obrázku 6.26.



Obrázek 6.26 Přechodová charakteristika nastaveného polohového regulátoru.

6.3 Poznatky z řešení

Při řešení jsem narazil na několik problémů. Některé většího a některé menšího rozsahu. Ač se některé z počátku zdály téměř neřešitelné a nesmyslné, myslím že se mi podařilo všechny zdárně vyřešit.

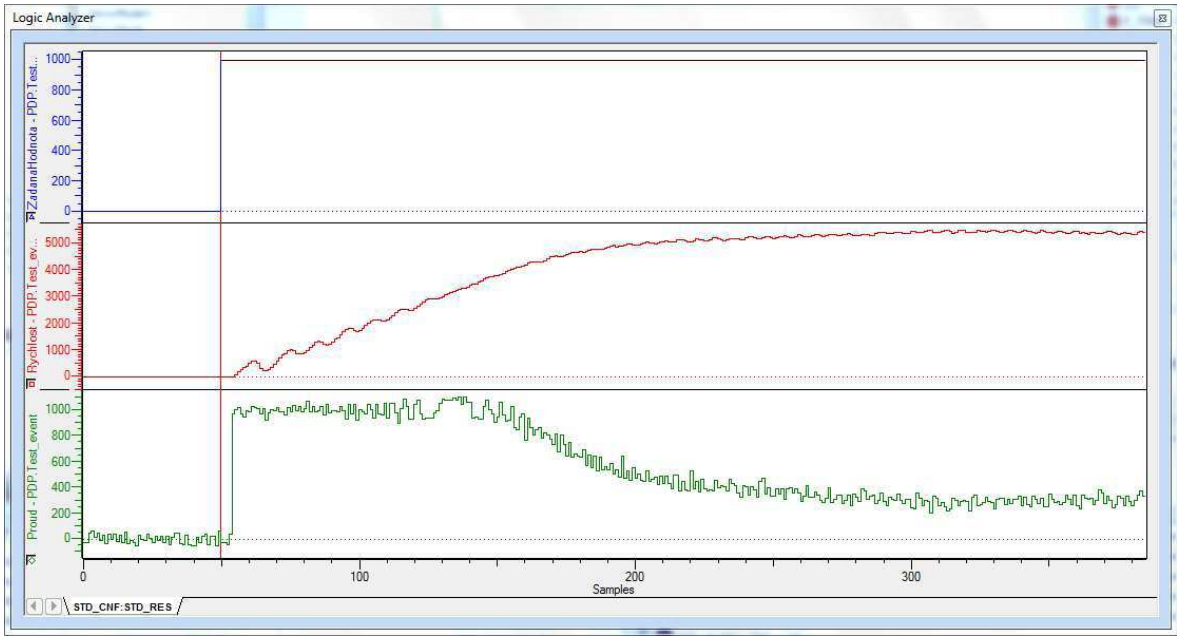
6.3.1 Vliv tření

Stejnosměrné motory mají zásadní neblahou vlastnost, kterou je tření. Hlavní podíl na velikost celkového tření má tření v komutátoru. Na celkové velikosti se však podílí další součástky jako ložiska. Vliv tření není zanedbatelný a působí na pohyb rotoru. Při roztáčení pohonu je nutné tření překonat. U stejnosměrných motorů muže být proud na překonání vlivu tření značně velký. Konkrétní velikost proudu pro překonání vlivu tření u použitého motoru Transmotec je 0,5 ampéry. Tato hodnota odpovídá 23,8% jmenovitého proudu motoru. Naštěstí lze vliv tření velice lehce eliminovat při použití servozesilovače. Servozesilovač obsahuje funkci, která právě tento problém řeší. Lze v něm nastavit hodnotu proudu, pro kterou není motor stále v pohybu, tedy minimální hodnotu proudu, při kterém se začne motor roztáčet.

6.3.2 Vliv vůle

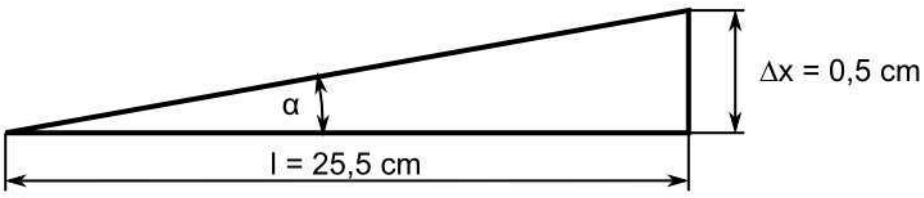
Jak již bylo zmíněno, velkým problémem při nastavování regulátorů a při provozu samotného zařízení je mechanická vůle v konstrukci motoru, převodovky a ramene. Kmity, které v soustavě způsobuje právě vůle, jsou vidět na všech snímcích přechodových charakteristik při natavování rychlostního regulátoru, konkrétně se jedná o kmity na náběžné hraně.

Pro ověření, zda na kmitání má vliv vůle, byl proveden test. Při proudovém resp. momentovém režimu byla zadána žádaná hodnota proudu bez zafixovaného ramene. Předpokládaný průběh rychlosti měl být hladký a v počátcích přechodového děje lineární, než se ustálí na hodnotě odpovídající žádanému proudu. Skutečnost, jak je zobrazena na obrázku 6.27, je však jiná. Kmitání se i v tomto nejjednodušším režimu objevilo. Proto je připisováno právě vůli zařízení, kdy při rozběhu motoru dochází k vymezování a k uvolnění vymezení vůle, které se opakuje do ustálení. Vůle má tedy neblahý vliv na průběh charakteristik a provoz zařízení. Vnáší do systému nelinearitu a působí jako poruchová veličina.



Obrázek 6.27 Test vlivu vůle pro skokovou změnu proudu.

Na obrázku 6.28 je zobrazen geometrický princip zjištění skutečné hodnoty vůle celého zařízení. Veličina *l* je délka ramene a Δx je změna polohy ramene při vymezování krajních poloh vůle.



Obrázek 6.28 Skutečná hodnota vůle.

$$\alpha = \arctan\left(\frac{\Delta x}{l}\right) = \arctan\left(\frac{0{,}5}{25{,}5}\right) = \arctan\left(0{,}019\right) = 1\,°8\,´$$

Skutečná hodnota vůle celého zařízení je tedy 1°8´.

6.3.3 Analýza dopadu míčku při triku

Při režimu žonglování s nastavitelnou výškou dochází u nadřízeného automatu k výpočtu hodnot odhozových parametrů podle fyzikálního principu šikmého vrhu. Výpočet je prováděn pro zaručení dopadu míčku v místě, kde se bude nacházet rameno žonglovacího stroje právě v době dopadu. Při tomto výpočtu se neuvažují žádné nepřesnosti. Místo dopadu je tedy jen teoretickým místem dopadu. Při skutečném odhození míčku však působí několik faktorů, které dopad ovlivňují. Prvním faktorem je již zmíněná vůle, která do odhodu vnáší předem nespecifikovatelnou nepřesnost. Nelze ji teď dost dobře předvídat a kvantifikovat. Dalšími významnými vlivy jsou: vliv časového zpoždění vyhodnocování odhozového úhlu v programu a nepřesnost odhozové rychlosti díky regulační odchylce rychlostního regulátoru.

Pro vyhodnocení vlivů nepřesnosti je použit výpočet místa dopadu míčku s určitými odhozovými parametry. Pro výpočet je použita odhozová rychlost 3500 otáček motoru za minutu, která je přibližně střední hodnota provozních rychlostí a tomu odpovídající odhozový úhel. Odchylka rychlostního regulátoru činí 2,3%. Tato odchylka vstupuje do řízení přímo v podobě odchylky odhozové rychlosti. Při maximální kladné odchylce regulace rychlosti je tedy odhozová rychlost 3580,5 otáček za minutu a při maximální záporné odchylce regulace rychlosti je 3419,5.

Druhým výrazným vlivem je nepřesnost vyhodnocování aktuální polohy ramene pro spuštění brzdění při odhozu míčku. Nepřesnost je způsobena periodou vykonávání programové smyčky *Juggling*, která má na starosti právě porovnávání žádané hodnoty polohy a aktuální hodnoty polohy. Programová smyčka pracuje s periodou 1,85ms. Při aplikaci vzorečku 6.2 na otáčky motoru, které jsou uvedeny v otáčkách za minutu, dostaneme úhel za milisekundu. Tuto hodnotu urazí rameno za čas 1ms.

$$\omega_{ramene} = \frac{\frac{rpm_{motoru}}{60000} * 360}{24} \tag{6.2}$$

Při vydělení otáček motoru v otáčkách za minutu hodnotou 60000 dostaneme otáčky motoru v otáčkách za milisekundu. Vynásobením 360° dostaneme otáčky motoru ve stupních za milisekundu. Poslední operace je vydělení převodovým poměrem pro převod otáček motoru na otáčky ramene. Po aplikaci vzorečku 6.3 dostaneme již hodnotu úhlu, který rameno urazí při maximálním zpoždění vyhodnocení aktuální hodnoty polohy. Aplikací vzorečku 6.3 vynásobíme otáčky ramene uvedené v milisekundě periodou opakování programové smyčky a vydělíme hodnotu 2. Uvažujeme pouze poloviční vliv zpoždění vyhodnocování, protože při porovnávání žádané hodnoty není testována spodní mez intervalu, ale střední hodnota.

$$\alpha = \frac{\omega_{ramene} * 1{,}85}{2} \tag{6.3}$$

Při výpočtu vzdálenosti dopadu míčku byla použita série vzorečku 6.4, 6.5 a 6.6. Vypočítané hodnoty jsou uvedeny v tabulce 6.1. Důležité jsou hodnoty dostřelu ve sloupci minimální velikost a maximální velikost. Tyto dvě hodnoty určují interval dostřelu míčku při shodě obou hlavních nepřesností v maximální míře.

$$rpm_{ramene} = \frac{rpm_{motoru}}{i} = \frac{rpm_{motoru}}{24} \tag{6.4}$$

Vzorec 6.4 převádí otáčky motoru na otáčky ramene vydělením otáček motoru převodovým poměrem.

$$v_0 = 2 * \pi * r * rpm_{ramene} \tag{6.5}$$

Vzorec 6.5 slouží k výpočtu obvodové rychlosti ramene. Tato hodnota vstupuje do výpočtu dostřelu jako počáteční rychlost vystřelení míčku.

$$x = \frac{v_0^2}{g} * \sin(2 * \alpha) \tag{6.6}$$

Vzorec 6.6 vychází z fyzikálního principu šikmého vrhu. Výsledná hodnota odpovídá vzdálenosti místa dopadu.

Veličina	Značka	Velikost			Jednotka
		minimální	střední	maximální	
délka ramene	r	0,225	0,225	0,225	[m]
otáčky motoru	rpm motoru	3419,500	3500,000	3580,500	[ot/min]
maximální odchylka úhlu	α max	0,791	0,809	0,828	[°]
odhozový úhel	α	79,791	79,000	78,172	[°]
převodový poměr	i	24,000	24,000	24,000	
otáčky ramene	rpm ramene	142,479	145,833	149,188	[ot/min]
otáčky ramene	rpm ramene	2,375	2,431	2,486	[ot/sec]
obvodová rychlost ramene	v0	3,357	3,436	3,515	[m/s]
dostřel	x	0,401	0,451	0,505	[m]

Tabulka 6.1 Analýza dostřelu.

Z tabulky výsledků je zřejmé, že šířka celého intervalu dopadu míčku je 0,505 – 0,401 = 0,104 metru, tedy 10,4 centimetru. Vzhledem k tomuto údaji byla také volena velikost dopadových košíčků na ramenech. Průměr košíčků byl vzhledem ke konstrukčním požadavkům ještě upraven na velikost 9 centimetrů. Vlivem této skutečnosti a působením dalších nepřesností, jako například nepředvídatelné vymezování vůli, má za následek chyby při provádění triku. Při stávajícím nastavení a konstrukci celého stroje nelze tedy zaručit 100% jistotu chycení míčku.

7 ZÁVĚR

Diplomový projekt byl řešen pomocí prvků firmy Phoenix Contact. Jedná se hlavně o software na programování automatů a o programovatelný automat s výkonovým členem. Díky výuce jsem byl seznámen s programováním a ovládáním programovatelného automatu, který byl použit pro ovládání motoru. Servozesilovač, nejdůležitější část žonglovacího stroje, byl ovšem pořízen jako nová součástka. Proto bylo potřeba se s tímto novým zařízením podrobně seznámit. Složitost celého výkonového členu spočívala hlavně v nutnosti nakonfigurovat velké množství parametrů a ve správném nastavení celého regulačního obvodu.

Konfigurace servozesilovače probíhala pomocí PCP komunikace. O tomto komunikačním protokolu není k dispozici mnoho podkladů, proto bylo obtížné naučit se správně používat funkční bloky obsluhující PCP komunikaci a realizovat tedy správnou funkci komunikace.

Další komplikací bylo správné nastavení celé regulační smyčky. V servozesilovači jsou hardwarově implementovány 4 regulátory, které jsou zapojeny v kaskádě. Jednotlivé regulátory jsou v činnosti při určitých režimech a musí být správně nastaveny. Správné nastavení regulátorů zabralo nejvíce času.

Zmíněné komplikace byly spíše technického charakteru a bylo zapotřebí je tak jako tak zvládnout. Horší jsou vlivy, které nemají z počátku žádné vysvětlení a chovají se nepředvídatelně. Nejzásadnější neblahý vliv na chod stroje má bezpochyby mechanická vůle. Je způsobena hlavně převodovkou, ale na její velikosti mají podíl i další konstrukční prvky. Největší problémy činila vůle při nastavování regulátorů a způsobovala kmitání. Vliv vůle se nedá nikdy zcela eliminovat, protože mechanické zařízení vždy nějakou mít bude, ale správným nastavením zařízení se nám podařilo její vliv snížit.

Ačkoliv se jedná o zařízení velice náročné na dynamické vlastnosti a časově kritické řízení, myslím, že i bez špičkových komponent určených pro tak náročné úlohy se nám podařilo zdárně sestrojit a naprogramovat funkční žonglovací stroj. Díky zmíněným vlivům není zaručena 100% úspěšnost provedení triku, ale i tak se pohybuje na vysoké úrovni.

Projekt Intelligent double arm juggling machine, jehož je tato diplomová práce součástí, se účastnil mezinárodní soutěže xplore New Automation Award 2012 pořádané firmou Phoenix Contact. Projekt byl zařazen do kategorie rekreace. I při vysoké konkurenci jsme byli vybráni do finálového kola, které se konalo v sídle firmy v Bad Pyrmontu v Německu. I když jsme se s projektem neumístili mezi prvními třemi týmy v kategorii, myslím, že za velký úspěch lze považovat již účast ve finálovém kole.

SEZNAM POUŽITÉ LITERATURY

[1] db_en_trio_ps_1ac_24dc_20_102779_02_en.pdf [citováno dne: 04.04.2012], Dostupné z: <http://select.phoenixcontact.com/phoenix/dwld/db_en_trio_ps_1ac_24dc_20_102779_02_en.pdf?cp=y&asid2=4666817865899>.

[2] db_en_trio_ps_1ac_24dc_5_102777_en_03.pdf [citováno dne: 04.04.2012], Dostupné z: <http://select.phoenixcontact.com/phoenix/dwld/db_en_trio_ps_1ac_24dc_5_102777_en_03.pdf?cp=y&asid2=6364988831103>.

[3] um_en_ilc_330_350_6959_en_05.pdf [citováno dne: 05.04.2012], Dostupné z: <http://select.phoenixcontact.com/phoenix/dwld/um_en_ilc_330_350_6959_en_05.pdf?cp=y&asid2=8759812164291>.

[4] um_en_ib_il_ec_ar_48_10a_6977_en_01.pdf [citováno dne: 11.04.2012], Dostupné z: <http://select.phoenixcontact.com/phoenix/dwld/um_en_ib_il_ec_ar_48_10a_6977_en_01.pdf?cp=y&asid2=2882498533792>.

[5] GROSS, J. Transmotec CZ [online]. 2012, [citováno dne: 15.04.2012], Dostupné z: <http://www.transmotec.cz/dc-motory-prevodovka-planetova/serie-pd4266.html>.

[6] HEDS-5540-A06 [citováno dne: 19.04.2012], Dostupné z: <http://pdf1.alldatasheet.com/datasheet-pdf/view/88919/HP/HEDS-5540-A06.html>.

[7] ah_en_fb_tcp_ip_communication_6982_en_03.pdf [citováno dne: 25.04.2012], Dostupné z: <http://select.phoenixcontact.com/phoenix/dwld/ah_en_fb_tcp_ip_communication_6982_en_03.pdf?cp=y&asid2=9939248966280>.

[8] Švarc, I.; Šeda, M.; Vítečková, M. Automatické řízení. Brno: CERM, 2007. 324 s. ISBN 978-80-214-3491-2.

[9] Noskievič, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999.

SEZNAM PŘÍLOH

Příloha 1: Fotografie zařízení
Příloha 2: Program řízení v textové podobě
Příloha 3: CD, které obsahuje:

- Řízení pohybů dvouramenného žonglovacího stroje.pdf (elektronická podoba diplomové práce)
- Program řízení pro programovatelný automat ILC 350ETH (vytvořeno v softwaru PC WORX)
- Demonstrační video žonglovacího stroje (video soubor)

PŘÍLOHA 1


CERTIFICATE
xplore
xplore
New Automation Award 2012
PHOENIX CONTACT







PŘÍLOHA 2

- **INICIALIZACE**

```
sPartnerPCP := 'D1.0.1';
sPartnerETH := '/IP=192.168.0.71';

iHomeAngleABS := 9000;
iReverseSpeedABS := 1000;

(* nastaveni defauiltnich hodnot trikovych parametru pro jednotlive hody *)
iHomeAngle := iHomeAngleABS;
diReleaseAngle := DINT#1681;
iSpeed := -3161;
iReverseSpeed := iReverseSpeedABS;
xDirection := TRUE;

(* nastaveni defaultnich hodnot trikovych parametru pro inteligentni zonglovani *)
iInteligentHomeAngle := iHomeAngleABS;
diInteligentReleaseAngle := DINT#1681;
iInteligentSpeed := -3161;
iInteligentReverseSpeed := iReverseSpeedABS;
xInteligentDirection := TRUE;

(* konstanty motoru a encoderu pro vypocty *)
igEncoderResolution := 2000;
igGearValue := 23;
digPulsePerRevolution := INT_TO_DINT(igEncoderResolution);
rdelay := 22.0;  (* [ms] *)(*24.0*)
rkoef := 1.35;
rgPreBrakingAngle := DINT_TO_REAL(digPulsePerRevolution)
                     /60000.0          (* 60[sec]*1000[ms] *)
                     *rdelay (* zpozdeni zpracovani nove zadane hodnoty *)
                     *rkoef; (* koeficient pro korekci *)

(* inicializace dokoncena*)
xInicialization := TRUE;
```

- **NASTAVENÍ PARAMETRŮ SERVOZESILOVAČE** – v servozesilovači bylo třeba nastavit desítky parametrů. Programový kód, který slouží k nastavení parametrů, je značně dlouhý, proto je zde uvedena jen část. Konkrétně se jedná o funkční blok realizující přepínání mezi režimy provozu servozesilovače.

```
IF iReset = 0 THEN
xSetMode := FALSE;
xSetPoint := FALSE;
ELSE
(* Parametry zadane hodnoty a aktualni hodnoty pro jednotlive rezimy *)
        CASE iModeSelectionCode OF
                1:      wActualValueValue :=WORD#16#6064;       (*6064*)
                        wSetPointValue :=WORD#16#607A;          (*607A*)
                2:  wActualValueValue := WORD#16#6064;    (*6044*)
                        wSetPointValue := WORD#16#6042;         (*6042*)
```

```
                3:      wActualValueValue := WORD#16#606C;      (*606C*)
                        wSetPointValue := WORD#16#6081;         (*6081*)
                4:      wActualValueValue := WORD#16#6078;      (*6078*)
                        wSetPointValue := WORD#16#6071;         (*6071*)
        END_CASE;

        (* __ NASTAVENI MODU SERVOZESILOVACE __ *)
        IF NOT xSetMode
                THEN
                        awIndex[1]:=WORD#16#6060;
                        awIndex[2]:=WORD#16#01;
                        PCP_Writing_1(xReq:=TRUE,
                                        iID:=iIDPCP,
                                        iCount:=2,
                                        awIndexSubindex:=awIndex,
                                        awWriteData[1]:=
                                        INT_TO_WORD(iModeSelectionCode));

                        xSetMode:=PCP_Writing_1.xWritingDone;
        END_IF;

        (* __ NASTAVENI AKTUALNI HODNOTY SERVOZESILOVACE __ *)
        IF NOT xActualValue
                THEN
                        awIndex[1]:=WORD#16#6000;
                        awIndex[2]:=WORD#16#06;
                        PCP_Writing_2(xReq:=TRUE,
                                        iID:=iIDPCP,
                                        iCount:=2,
                                        awIndexSubindex:=awIndex,
                                        awWriteData[1]:=wActualValueValue);

                        xActualValue:=PCP_Writing_2.xWritingDone;
        END_IF;

        (* __ NASTAVENI POZADOVANE HODNOTY SERVOZESILOVACE __*)
        IF NOT xSetPoint
                THEN
                        awIndex[1]:=WORD#16#6001;
                        awIndex[2]:=WORD#16#06;
                        PCP_Writing_3(xReq:=TRUE,
                                        iID:=iIDPCP,
                                        iCount:=2,
                                        awIndexSubindex:=awIndex,
                                        awWriteData[1]:=wSetPointValue);

                        xSetPoint:=PCP_Writing_3.xWritingDone;
        END_IF;
END_IF;
xServoMode := xSetMode & xActualValue & xSetPoint;
```

- **PROGRAMOVÁ SMYČKA MAIN**

```
(* __ INICIALIZACE__ *)
IF      NOT xInicializationDone
        THEN  Inicialization_1();
                        xInicializationDone := Inicialization_1.xInicialization;
END_IF;


(* __ NAVAZANI PCP KOMUNIKACE (PCP CONNECT) __ *)
IF xInicializationDone
        THEN  PCP_Connection_1(sPartner:=sPartnerPCP);
                        iIDPCP:=PCP_Connection_1.iID;
                        xPCPConnectionDone:=PCP_Connection_1.xValid;
END_IF;


(* __ NASTAVENI PARAMETRU SERVO ZESILOVACE __ *)
IF  (wStatusWord.X0  =  FALSE)&(wStatusWord.X1  =  FALSE)&(wStatusWord.X2  =
FALSE)&(wStatusWord.X3 = FALSE)&(wStatusWord.X6 = TRUE)
        THEN
                IF xPCPConnectionDone & (NOT xServoAmplifierSettingDone)
                        THEN  Setting_1();
                                xServoAmplifierSettingDone:=Setting_1.xSettingDone;
                END_IF;
END_IF;


(* ___ SERVOAMPLIFIER -> OPERATION ENABLED  ___ *)
(* ___ (PRUCHOD VSECH STAVU ZESILOVACE) ___ *)
IF      NOT xDeviceStatesDone &  xServoAmplifierSettingDone
        THEN  DeviceStates_1();
                xDeviceStatesDone := DeviceStates_1.xOperationEnabled;
END_IF;


(* ___ ETH KOMUNIKACE  ___ *)
(* ___ NAVAZANI ETH KOMUNIKACE  ___ *)
IP_CONNECT_1(EN_C:=TRUE,PARTNER:=sPartnerETH);
xETHConnectionDone :=IP_CONNECT_1.VALID;
iIDETH :=IP_CONNECT_1.ID;
```

- **PROGRAMOVÁ SMYČKA ETHSETTING**

```
(* _____ CTENI Z ETH _____ *)
IF xETHConnectionDone THEN
        IP_URCV_1(EN_R:=TRUE,ID:=iIDETH,RD_1:=stETHReceivedData);
                xETHDataReceived:=IP_URCV_1.NDR;
                stETHReceivedData:=IP_URCV_1.RD_1;
        wMode                   := stETHReceivedData.Mode;
        diAngleRelease:= DWORD_TO_DINT(stETHReceivedData.AngleRelease);
        iSpeedRelease  := DWORD_TO_INT(stETHReceivedData.SpeedRelease);
(* ____________________________________________________________ *)
```

```
		stETHSentData.Mode := wMode;
		stETHSentData.AngleRelease := DINT_TO_DWORD(diAngleRelease);
		stETHSentData.SpeedRelease := INT_TO_DWORD(iSpeedRelease);
		IF diAngleRelease = DINT#0 & iSpeedRelease = 0
			THEN xDataNeprazdna := FALSE;
			ELSE xDataNeprazdna := TRUE;
		END_IF;
		xReq := FALSE;

		IF xETHDataReceived THEN
		(* ___ NASTAVOVACI OBRAZOVKA ___ *)
			IF wMode.X4 & wMode.X5 & wMode.X6 THEN
			(* PO STISKU TLACITKA SET INTELIGENT PARAMETERS *)
				IF wMode.X0 & wMode.X1 & wMode.X2 & NOT wMode.X3 &
xDataNeprazdna THEN
					iInteligentHomeAngle := iHomeAngleABS;
					iInteligentSpeed := iSpeedRelease * -1;
					diInteligentReleaseAngle := (INT_TO_DINT(-1) *
diAngleRelease) + REAL_TO_DINT(INT_TO_REAL(iSpeedRelease)*rgPreBrakingAngle);
					iInteligentReverseSpeed := iReverseSpeedABS;
					xInteligentDirection := TRUE;
				END_IF;
			END_IF;

		(* ___ PROVEDENI JEDNOHO TRIKU - HOD ___ *)
			IF wMode.X4 & NOT wMode.X5 & NOT wMode.X6 THEN
			(* PO STISKU TLACITKA SET *)
				IF wMode.X0 & NOT wMode.X1 & NOT wMode.X2 & NOT
wMode.X3 & xDataNeprazdna THEN
					xDirection := wMode.X7;
					IF xDirection THEN
						iHomeAngle := iHomeAngleABS;
						iSpeed := iSpeedRelease * -1;
						diReleaseAngle := (INT_TO_DINT(-1) *
diAngleRelease) + REAL_TO_DINT(INT_TO_REAL(iSpeedRelease)*rgPreBrakingAngle);
						iReverseSpeed := iReverseSpeedABS;
					ELSE
						iHomeAngle := iHomeAngleABS * -1;
						iSpeed := iSpeedRelease;
						diReleaseAngle := diAngleRelease -
REAL_TO_DINT(INT_TO_REAL(iSpeedRelease)*rgPreBrakingAngle);
						iReverseSpeed := iReverseSpeedABS * -1;
					END_IF;
					xReq := TRUE;
				END_IF;

				(* PO STISKU TLACITKA JUGGLE *)
				IF NOT wMode.X0 & wMode.X1 & NOT wMode.X2 & NOT
wMode.X3 & xDataNeprazdna THEN
					xTrick := TRUE;
					xReq := TRUE;
				END_IF;
			END_IF;
```

```
(* ___ INTELIGENTNI ZONGLOVANI ___ *)
    IF NOT wMode.X4 & wMode.X5 & NOT wMode.X6 THEN
        (* PO STISKU TLACITKA JUGGLE *)
        IF NOT wMode.X0 & wMode.X1 & NOT wMode.X2 & NOT wMode.X3 & xDataNeprazdna THEN
            iHomeAngle := iInteligentHomeAngle;
            diReleaseAngle := diInteligentReleaseAngle;
            iSpeed := iInteligentSpeed;
            iReverseSpeed := iInteligentReverseSpeed;
            xDirection := xInteligentDirection;
            xTrick := TRUE;
            xJuggling := TRUE;
            xReq := TRUE;
            xStep := 0;
        END_IF;
    END_IF;

(* ___ INTELIGENTNI ZONGLOVANI S KROKOVANIM ____ *)
    IF wMode.X4 & wMode.X5 & NOT wMode.X6 THEN
        (* PO STISKU TLACITKA SET *)
        IF wMode.X0 & NOT wMode.X1 & NOT wMode.X2 & NOT wMode.X3 & xDataNeprazdna THEN
            iHomeAngle := iInteligentHomeAngle;
            diReleaseAngle := diInteligentReleaseAngle;
            iSpeed := iInteligentSpeed;
            iReverseSpeed := iInteligentReverseSpeed;
            xDirection := xInteligentDirection;
            xTrick := TRUE;
            xJuggling := TRUE;
            xReq := TRUE;
            xStep := 1;
        END_IF;

        (* PO STISKU TLACITKA JUGGLE *)
        IF NOT wMode.X0 & wMode.X1 & NOT wMode.X2 & NOT wMode.X3 & xDataNeprazdna THEN
            xStep := 2;
            IF wMode.X8 & NOT wMode.X9 THEN
                xStep := 1;
            END_IF;
        END_IF;
    END_IF;

(* ___ STOP ___ *)
    IF  wMode.X0 & wMode.X1 & NOT wMode.X2 & NOT wMode.X3 & xDataNeprazdna THEN
        xTrick := FALSE;
        xJuggling := FALSE;
        xStop := TRUE;
        xReq := TRUE;
    END_IF;
```

```
        (* ___ TOTAL STOP ___ *)
            IF  wMode.X0 & wMode.X1 & wMode.X2 & wMode.X3 THEN
                xTrick := FALSE;
                xJuggling := FALSE;
                IF wStatusWord.X2 THEN
                    wControlWord.X0 := TRUE;
                    wControlWord.X1 := TRUE;
                    wControlWord.X2 := TRUE;
                    wControlWord.X3 := FALSE;
                    wControlWord.X5 := FALSE;
                    wControlWord.X6 := FALSE;
                ELSE
                    wControlWord.X1 := TRUE;
                    wControlWord.X2 := FALSE;
                END_IF;
            END_IF;

        (* ___ RESET TLACITKA TOTAL STOP ___ *)
            IF  NOT wMode.X0 & wMode.X1 & wMode.X2 & wMode.X3 THEN
                xDeviceStatesDone := FALSE;
                xTrick := FALSE;
                xTotalStop := TRUE;
            END_IF;
        END_IF;
        (* ___ ODESLANI POTVRZENI PO ETH ___ *)
        IP_USEND_1(REQ:=xReq,ID:=iIDETH,SD_1:=stETHSentData);
            xSendDone:=IP_USEND_1.DONE;
            xErr:=IP_USEND_1.ERROR;
            stETHSentData:=IP_USEND_1.SD_1;
END_IF;
```

- **PROGRAMOVÁ SMYČKA JUGGLING**

```
IF xDeviceStatesDone THEN
(* ____ PROVEDENI TRIKU ____*)
IF xTrick THEN
    iPocet := iPocet + 1;
    CASE iPocet OF
500:        dwpgSetPoint := DINT_TO_DWORD(INT_TO_DINT(iHomeAngle));
            wControlWord.X3 := TRUE;
            wControlWord.X4 := TRUE;
1100:       wControlWord.X3 := FALSE;
            wControlWord.X4 := FALSE;
1600..3000: ServoMode_1(iModeSelectionCode:=2,iReset:=iPocet MOD 1600);
            IF ServoMode_1.xServoMode THEN iPocet := 3000;END_IF;
3002:       dwpgSetPoint.W1 := INT_TO_WORD(iSpeed);
            wControlWord.X3 := TRUE;
3003..4000: IF xDirection THEN
                IF  DWORD_TO_DINT(dwpgActualValue) <=  diReleaseAngle
                    THEN
                    dwpgSetPoint.W1 := INT_TO_WORD(iReverseSpeed);
                    iPocet := 4000;
                END_IF;
            ELSE
```

```
                        IF  DWORD_TO_DINT(dwpgActualValue)  >=  diReleaseAngle
                            THEN
                            dwpgSetPoint.W1 := INT_TO_WORD(iReverseSpeed);
                            iPocet := 4000;
                        END_IF;
                END_IF;
4030:           dwpgSetPoint := DWORD#0;
4060..6000:     ServoMode_2(iModeSelectionCode:=1,iReset:=iPocet MOD 4060);
                IF ServoMode_2.xServoMode THEN iPocet := 6000;END_IF;
6001:           dwpgSetPoint := DWORD#0;
                wControlWord.X3 := TRUE;
                wControlWord.X4 := TRUE;
7000:           xTrick := FALSE;
                iPocet := 0;
        END_CASE;
END_IF;
(* ____________________________________________________________________ *)
(* Zmena hodnot pro opacny smer *)
        IF xJuggling & iPocet=6999 THEN
                iPocet                  := 0;
                iHomeAngle              := iHomeAngle * -1;
                diReleaseAngle:= diReleaseAngle * INT_TO_DINT(-1);
                iSpeed                  := iSpeed * -1;
                iReverseSpeed  := iReverseSpeed * -1;
                xDirection := NOT xDirection;
        END_IF;
(* Zastaveni triku v prvnim kroce *)
        IF xJuggling & xStep = 1 & iPocet = 2 THEN
                iPocet                  := 1;
        END_IF;
(* Zastaveni triku v druhem kroce *)
        IF xJuggling & xStep = 2 & iPocet = 3001 THEN
                iPocet                  := 3000;
        END_IF;
(* Polohova regulace na nulovou hodnotu pri zastaveni *)
        IF xStop THEN
        IF NOT xPocetReset THEN iPocet := 0; xPocetReset := TRUE; END_IF;
        iPocet := iPocet + 1;
        CASE iPocet OF
1:              wControlWord.X3 := FALSE;
                wControlWord.X4 := FALSE;
100:            ServoMode_3(iModeSelectionCode:=1,iReset:=0);
101..1000:      ServoMode_3(iModeSelectionCode:=1,iReset:=1);
                IF ServoMode_3.xServoMode THEN iPocet := 1000; END_IF;
1001:           dwpgSetPoint := DWORD#0;
                wControlWord.X3 := TRUE;
                wControlWord.X4 := TRUE;
2500:           wControlWord.X3 := FALSE;
                wControlWord.X4 := FALSE;
                xStop := FALSE;
                iPocet := 0;
                xPocetReset := FALSE;
        END_CASE;
        END_IF;
```

```
(* Po navratu z total stopu nastaveni zadane hodnoty na aktualni("vynulovani") *)
        IF xTotalStop THEN
                dwpgSetPoint := dwpgActualValue;
                xTotalStop := FALSE;
        END_IF;
(* Aktualni poloha *)
        ihodnotaActual := DWORD_TO_DINT(dwpgActualValue);
(* vypnuti pohodnu v pripade neprovadeni triku *)
        IF iPocet=0 THEN
                wControlWord.X3 := FALSE;
                wControlWord.X4 := FALSE;
        END_IF;
END_IF;
```

Celý zdrojový kód v podobě funkčního programu je nahrán na CD, které představuje přílohu 3. Program je vytvořen v softwaru PC WORX 6.00.25 SP 3.73 na počítačové stanici s operačním softwarem Windows 7. Pro spuštění programu je nutné mít k dispozici počítač se softwarem PC WORX stejné verze, programovatelný automat ILC 350ETH od firmy Phoenix Contact a servozesilovač IB IL EC AR 48/10A taktéž od firmy Phoenix Contact.