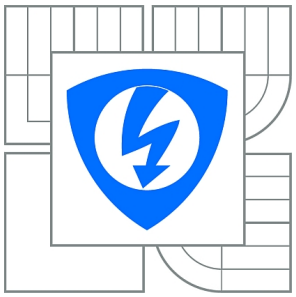
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

MODEL STEJNOSMĚRNÉHO MOTORU

MODEL OF DC MOTOR

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

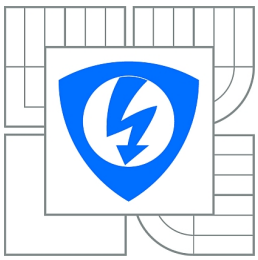
AUTOR PRÁCE — LUBOŠ TURANSKÝ

AUTHOR

VEDOUCÍ PRÁCE — Ing. LUDĚK CHOMÁT

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Luboš Turanský	***ID:***	106842
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Model stejnosměrného motoru

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s programovatelným automatem s vestavěným dotykovým panelem od firmy B&R. Prostudujte možné regulace pohonů jak v průmyslu, tak v domácnosti. Matematický model ověřte pomocí Matlab/Simulink. Vytvořte do mikročipu simulaci stejnosměrného motoru, který budete řídit programovatelným automatem. Pro model vytvořte jednoduchou vizualizaci, kde bude možné nastavovat parametry regulátoru. Výsledkem bakalářské práce je sestavení trenažéru technologického procesu, ověření funkčnosti, vytvoření manuálu a zadání pro model.

DOPORUČENÁ LITERATURA:

Matoušek, D.: Práce s mikrokontroléry ATMEL AVR - ATmega16. ISBN 80-7300-174-8, BEN, 2006.

B&R www.br-automation.com

Pavelka, J.: Elektrické pohony. Praha, ISBN 978 80 01 03588 7, ČVUT, 2007.

Termín zadání:	8.2.2010	***Termín odevzdání:***	31.5.2010

Vedoucí práce: Ing. Luděk Chomát

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

V práci je popsáno základní rozdělení elektrických motorů a možnosti jejich regulace. Podrobně je pak rozebráno získání matematického modelu DC motoru včetně simulace pro Matlab/Simulink. Dále je popsána tvorba hardwaru a firmwaru modelu, jehož základem je mikrokontrolér ATmega8 simulující chování popsaného DC motoru. Hotový model je propojen s PLC, pro které je vytvořena vizualizace umožňující monitorovat průběhy regulačních dějů a měnit parametry PSD regulátoru. Poslední částí práce je návod k modelu a zadání laboratorní úlohy pro model.

Abstract

In this work is described essential division of electric motors and possibilities of their regulation. In detail is shown how was obtained mathematical model of DC motor and its implementation to Matlab/Simulink. Subsequently is described how was created hardware and firmware of model which is simulating behavior of described DC motor. Final model is connected with PLC. For PLC is created visualization which allows monitoring of regulation process and changing PSD regulator parameters. Last part of the work is model manual and task laboratory exercise for the model.

Klíčová slova

AVR, mikrokontrolér, stejnosměrný motor, Simulink, PLC

Keywords

AVR, microcontroller, DC motor, Simulink, PLC

Bibliografická citace

TURANSKÝ, L. *Model stejnosměrného motoru.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 82 s. Vedoucí bakalářské práce Ing. Luděk Chomát.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma „Model stejnosměrného motoru“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **27. května 2010** ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Luďkovi Chomátovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **27. května 2010** ……
podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Jedním z cílů semestrálního projektu, na který tato práce navazuje, bylo vytvoření konceptu modelu technologického procesu pro procvičení návrhu regulátorů. Ve výsledku tak vznikl přípravek pro simulaci regulace rychlosti stejnosměrného motoru s cizím buzením permanentním magnetem. Tento technologický proces byl zvolen zejména z důvodu častého výskytu této aplikace číslicových regulátorů v praxi. Jeho charakter navíc umožňuje vytvoření názorného modelu, s designem obdobným již hotovým přípravkům v laboratoři inteligentních regulátorů UAMT FEKT VUT.

Hlavními cíly této práce je vytvoření matematického modelu DC motoru a jeho simulace v programu Matlab/Simulink. Dále pak vytvoření hardwaru modelu, jehož základem bude mikrokontrolér s potřebnými periferiemi. Hotový přípravek propojit s PLC, vytvořit vizualizaci umožňující nastavení parametrů regulátoru a ověřit funkčnost. Jako poslední část práce by měl být vytvořen návod k obsluze a zadání laboratorní úlohy pro model.

Přípravek by měl rozšířit paletu již používaných modelů technologických procesů a přispět tak ke zkvalitnění výuky ve výše zmíněné laboratoři. Přitom pořizovací cena toho modelu je výrazně nižší v porovnání s cenou laboratorního přípravku obdobných parametrů s reálným motorem.

2. REGULACE POHONŮ

2.1 ÚVOD

Jak se píše v [1], při realizaci kvalitního moderního pohonu, je nutné uplatnit znalosti o mechanických soustavách, o elektrických strojích, výkonové elektronice i regulaci. Tato kapitola se bude však věnovat podrobněji především části regulační, protože výsledný model bude použit právě k procvičení návrhu a nastavení regulátorů. V kapitole je uveden přehled hlavních metod, používaných k regulaci pohonů. Podrobnější informace o jednotlivých principech regulace, spolu s dalšími metodami je možné nalézt v literatuře [1], [2], [3], [4] a [5], ze které tento přehled vychází.

2.2 TYPY ELEKTRICKÝCH POHONŮ

Odpovídají-li vlastnosti samotného stroje požadavkům na elektrický pohon, může být stroj napájen přímo ze sítě. Většina poznatků vědního oboru elektrické pohony se však týká řízení a regulace elektrických strojů, umožňujících rozšířit škálu dosažitelných vlastností. Vzniká tak nová kategorie – regulovaný pohon všeobecného použití, tj. systém přizpůsobitelný mnoha požadavkům technologického zařízení. U pohonu je možné regulovat moment (nebo tah), rychlost a polohu. Z hlediska nejrozšířenější regulace otáčivé rychlosti lze rozlišit pohony několikarychlostní nebo s plynulou regulací rychlosti, přičemž se může vyžadovat práce při jednom nebo obou směrech otáčení. Důležitá je také schopnost elektrického brzdění, kdy se kinetická a potenciální energie soustavy přeměňuje v elektrickou, která se buď rekuperuje zpět do napájecí sítě, akumuluje, nebo se přeměňuje v teplo v elektrickém stroji či v odporech [1].

2.2.1 Rozdělení pohonů

Pohony je možné rozdělit do několika skupin, odlišujících se nároky na regulaci, typem a druhem napájení akčních členů, způsobem ovládání a nejdůležitějšími problémy řešenými při jejich použití [1]:

- Pohony pro malou automatizaci, kancelářskou a výpočetní techniku, které jsou nejčastěji řešeny na bázi krokových nebo stejnosměrných motorů.
- Pohony pro domácí spotřebiče, nářadí a podobné aplikace. Většinou jsou to univerzální komutátorové motorky, napájené přímo nebo z měniče napětí.
- Pohony všeobecného použití, pokrývající širokou škálu aplikací, realizovány většinou asynchronním strojem.
- Servopohony pro obráběcí stroje, manipulátory, roboty a další servomechanismy, které používají pro dosažení špičkových vlastností speciální synchronní stroje.
- Pohony mezních výkonů, které si mnohdy vynutily speciální konstrukci akčních členů a jsou často realizovány na bázi synchronních strojů, napájených z polovodičových měničů.
- Speciální pohony, vyráběné na míru konkrétním požadavkům, které zahrnují i stroje s jiným druhem pohybu než rotačním, např. lineárním.

Akční členy elektrických pohonů jsou tvořeny elektrickými stroji. Odpověď na otázku, jaký je nejrozšířenější elektrický stroj, není jednoduchá. Je-li třeba se soustředit na stroje, ve kterých je instalován největší výkon, je nutné si uvědomit, že téměř veškerá elektrická energie je vyrobena v elektrárnách v synchronních generátorech. Dalším hlediskem může být počet vyrobených kusů, kde pro potřeby automobilové, automatizační, kancelářské, spotřební a zábavní elektrotechniky jsou vyráběny stotisícové série malých stejnosměrných a DC brushless motorků. S počátkem rozvoje regulovaných elektrických pohonů jsou spojeny klasické stejnosměrné stroje s cizím buzením, které jsou stále nejvhodnější pro vysvětlení základních regulačních principů. Nejdůležitější jsou však v současné době asynchronní stroje, které se prosadily v široké třídě průmyslových neregulovaných i regulovaných pohonů všeobecného použití, kde splňují v podstatě všechny požadavky [1].

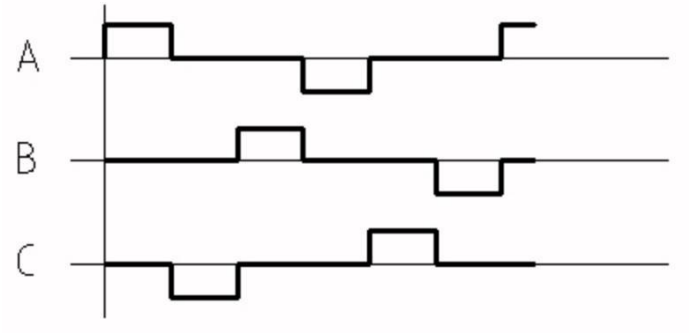
2.3 REGULACE OTÁČEK MOTORŮ

2.3.1 Regulace asynchronních motorů

- **Změna skluzu** se u asynchronních motorů s kroužkovým rotorem provádí rotorovým spouštěčem. Tím s však značná část výkonu ztrácí na rezistorech, a proto je tento způsob nehospodárný [2].
- **Změna kmitočtu napájecího proudu**. Tato metoda vyžaduje samostatný trojfázový zdroj s proměnným kmitočtem. Dříve používané rotační měniče jsou drahé a mají složitou údržbu, proto se v poslední době používají statické měniče kmitočtu. Výhodou použití frekvenčních měničů je možnost plynulého řízení otáček při rozběhu, za chodu i během doběhu [2].
- **Regulace napětí na statoru** pomocí tyristorových můstků. Řízením okamžiku spínání tyristorů lze měnit střední hodnotu velikosti přiváděného napětí, a tím i moment motoru. Čímž dojde při stejném zatížení motoru ke změně otáček [2].

2.3.2 Řízení krokových motorů

- **Řízení po celých krocích** – Jedná se o nejjednodušší způsob řízení krokových motorů, při kterém je buzení spínáno pro celé kroky. V tomto režimu je vždy napájeno jen vinutí jedné fáze statoru, a to jmenovitým proudem kladné nebo záporné polarity (viz. Obrázek 2.1) [2].



Obrázek 2.1 Spínání fází krokového motoru pro celý krok [2]

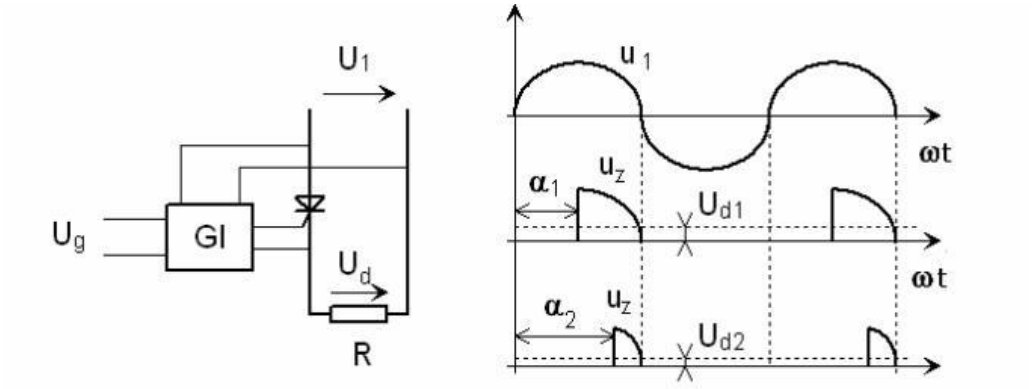
Tento způsob krokování je zastaralý a mezi jeho hlavní nevýhody patří výrazné přechodové děje, pulzující moment a nestabilita krokových motorů při různých budících frekvencích [2].

- **Mikrokrokování** - Užívá se v aplikacích se zvýšenými nároky na přesnost pohybu, přičemž tento princip také odstraňuje nežádoucí jevy řízení po celých krocích. Princip mikrokrokování je podrobně rozveden v [3] a zjednodušeně ho lze popsat následovně:

  Plná délka kroku krokového motoru může být rozdělena do menších inkrementů pohybu rotoru. Budeme-li v jedné fázi postupně snižovat hodnotu budícího proudu a v sousední fázi hodnotu proudu zvyšovat, bude se výsledný vektor magnetické indukce pohybovat mezi krajními polohami, danými vybuzením jednotlivých fází. Hodnota mezního vazebního momentu je konstantní a odpovídá velikosti momentu, získaného při buzení jedné fáze velikostí proudu I. Mikrokrokování sice nezvyšuje přesnost polohování, ale zvětšuje citlivost krokového motoru [3].

2.3.3 Regulace otáček motorů na stejnosměrné napětí

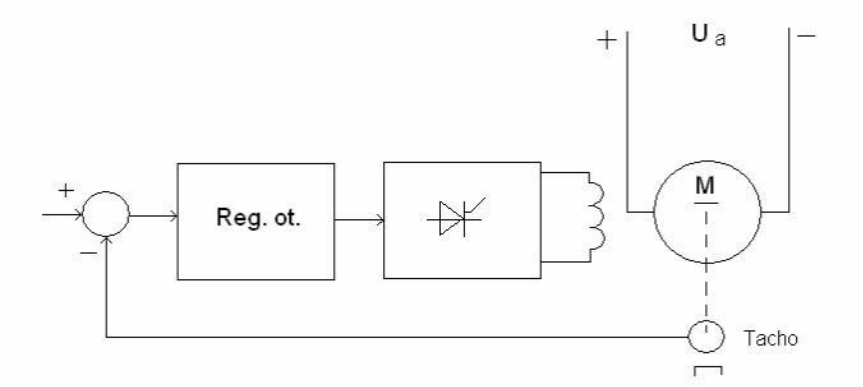
- **Změnou odporu ve spouštěcím obvodu** – Nejjednodušší řízení otáček lze realizovat přidáním proměnného odporu do budícího obvodu. Toto řešení sebou však přináší velký ztrátový výkon [2].
- **Tyristorové usměrňovače pro stejnosměrné pohony** – Princip je patrný z Obrázek 2.2, na kterém je nakreslen jednopulzní usměrňovač s jedním tyristorem a průběhy výstupních napětí na odporové zátěži pro dva různé řídící úhly α. Řídící úhel se v tomto případě měří od průchodu napětí nulou. Změnou řídícího úhlu je možné plynule řídit výstupní napětí usměrňovače [4].



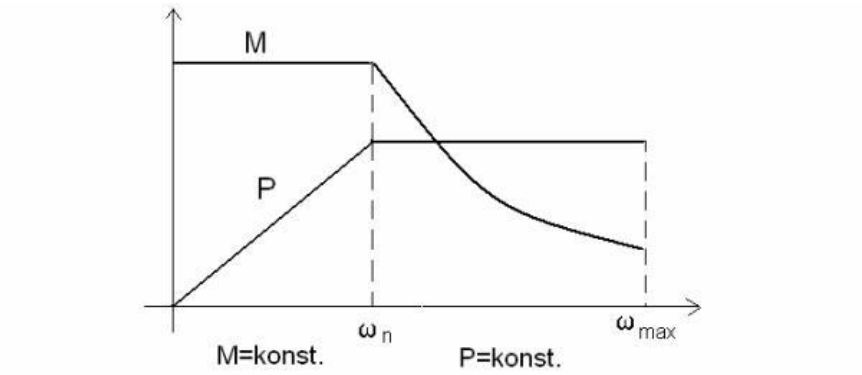
Obrázek 2.2 Princip fázového řízení tyristorového usměrňovače [4]

- **Regulace napětím kotvy** - Při této regulaci se motor chová v zásadě jako lineární prvek s konstantou úměrnosti danou konstrukčním uspořádáním motoru. S rostoucím napětím kotvy se zvětšuje moment vytvářený motorem. K regulaci je také možné použít budící napětí kotvy, tento způsob se však používá zřídka [5].

- **Regulace rychlosti magnetickým tokem** - Tento způsob lze rozdělit na regulaci s konstantním napětím kotvy a na regulaci kombinovanou (napětím kotvy a buzením). Při regulaci rychlosti magnetickým tokem s konstantním napětím kotvy je kotva stejnosměrného motoru připojena na zdroj konstantního napětí. V praktických aplikacích to bývá stejnosměrná sběrnice pro napájení vícemotorových pohonů kontinuálních linek, jejímž napětím se řídí základní rychlost linky. Struktura zpětnovazebního řízení otáček sestává z tyristorového usměrňovače pro napájení budícího vinutí, regulátoru otáček a tachodynama (viz. Obrázek 2.3). Při kombinované regulaci je zpravidla stejnosměrný motor řízen v rozsahu od nuly do jmenovité rychlosti napětím kotvy při plně nabuzeném motoru. V rozsahu od jmenovitých do maximálních otáček se pak motor řídí odbuzováním, při jmenovitém napětí kotvy. Motor je tedy stále zatěžován konstantním výkonem, ale moment s otáčkami klesá hyperbolicky (viz. Obrázek 2.4) [4].



Obrázek 2.3 Regulace otáček stejnosměrného motoru změnou budícího proudu [4]



Obrázek 2.4 Grafické znázornění pracovních oblastí při kombinované regulaci rychlosti [4]

3. PROGRAMÁTOR

3.1 ÚVOD

Jak již bylo zmíněno výše, hlavní částí modelu je mikrokontrolér ATmega8. K nahrání kódu do mikrokontroléru je kromě PC nutný programátor, který propojí USB port počítače s programovacími piny ATmega8, zařídí hodinový signál atd. K vývoji kódu a stavbě modelu bylo nezbytné, mít možnost nahrávat program do mikrokontroléru i mimo školní laboratoř, což znamená vlastnit programátor.

Byl jsem tedy postaven před možnost si programátor koupit již hotový s cenou okolo 700 Kč nebo se pustit do jeho postavení. Rozhodl jsem se pro druhou možnost, přičemž jsem jako návod použil internetovou stránku [6].

3.2 SOFTWARE PRO PC

Pro programátor je třeba nejprve nainstalovat USB ovladač. Ten je možné stáhnout z [6]. K vlastnímu nahrávání HEX souborů do mikrokontroléru je pak možné použít například program Khazama AVR Programmer v1.6.2, který je volně dostupný na internetových stránkách výrobce:

(*http://khazama.com/project/programmer/*).

4. MODEL MOTORU

4.1 VOLBA TYPU MOTORU

Stejnosměrné motory s cizím buzením se vyznačují velmi dobrými regulačními vlastnostmi a po dlouho dobu byly téměř jediným druhem motorů, se kterým bylo možné se v regulačních pohonech setkat. V současné době je již možné ve velkém množství aplikací stejnosměrný motor nahradit asynchronním.

Výhod stejnosměrného motoru je však stále celá řada a patří mezi ně například jednoduché řízení rychlosti změnou napětí kotvy, či budícího proudu, přičemž se otáčky mohou pohybovat v širokém rozsahu, který není nijak vázán na kmitočet sítě. Smysl otáčení lze snadno měnit změnou polarity kotvy nebo budícího proudu. Při řízení pouze napětím kotvy, což je případ navrženého modelu, se jedná v zásadě o lineární prvek.

Největší nevýhodou stejnosměrných motorů v praxi je napájení rotoru přes komutátor, což způsobuje nižší spolehlivost a větší nároky na údržbu oproti asynchronnímu motoru. Mezi další nevýhody patří nižší poměr hmotnosti motoru k výkonu a zpravidla vyšší cena oproti střídavým motorům obdobného výkonu.

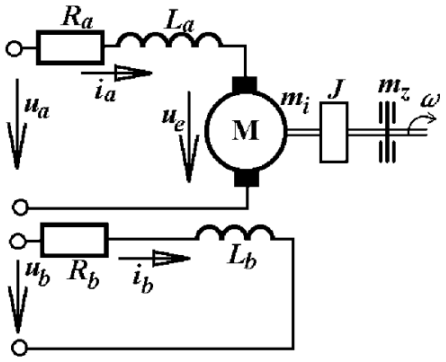
Pro tuto práci byl zvolen model stejnosměrného motoru s cizím buzením permanentním magnetem. A to i přes to, že se v poslední době v praxi čím dál více uplatňují motory asynchronní. Tyto akční členy (bez přidání řídící jednotky) jsou však nelineárními prvky s poměrně komplikovaným modelem a implementace takového modelu do mikrokontroléru by byla velmi složitá [5]. Navíc klasické stejnosměrné stroje s cizím buzením jsou stále pravděpodobně nejvhodnější pro vysvětlení základních regulačních principů [1].

4.2 MODEL PRO MATLAB / SIMULINK

Před samotnou implementací matematického modelu do mikrokontroléru byl nejprve vytvořen model motoru v programu Matlab / Simulink. Díky tomu bylo možné sledovat obvodové veličiny motoru a experimentálně nastavit parametry motoru pro optimální dobu přechodného děje. Dále byl simulinkovský model využit při ověření správnosti implementace modelu do mikrokontroléru (viz kapitola 8) a návrhu

regulátoru. Soubory s modelem pro program Matlab je možné nalézt na přiloženém DVD.

Při získávání modelu je třeba vyjít z náhradního schématu stejnosměrného motoru [5] (Obrázek 4.1). Elektrické parametry obvodu kotvy jsou charakterizovány celkovým odporem a indukčností vinutí kotvy i dalších vedení a vinutí, která jsou s nimi v sérii.



Obrázek 4.1: Náhradní schéma SS motoru s cizím buzením [5]

Symboly použité v Obrázek 4.1:

u_a - napájecí napětí kotvy

i_a - proud kotvy

Ra - odpor vynutí kotvy

La - indukčnost vynutí kotvy

u_e - napětí indukované v kotvě motoru při jejím otáčení v magnetickém poli

m_i - moment působící na hřídel vyvolaný napájecím napětím

J - setrvačný moment motoru a všech pohybujících se částí s pojených s hřídelí motoru

m_z - zatěžovací moment vyvolaný zátěží a pasivními odpory motoru

ω - úhlová rychlost hřídele motoru

u_b - napájecí napětí budícího obvodu

Rb - odpor vynutí budícího obvodu

i_b - proud budícím obvodem

Lb - indukčnost vynutí budícího obvodu

Obvod kotvy lze popsat následujícími rovnicemi:

$$u_a = R_a i_a + L_a \frac{di_a}{dt} + u_e; \quad u_e = k\phi\omega; \quad u_e = kf(i_b)\omega \tag{4.1}$$

Magnetizační charakteristika $\phi = f(i_b)$ je nelineární a vyjadřuje závislost magnetického toku ϕ na proudu budícího obvodu i_b. Ve schématu na Obrázek 4.1 jsou dvě vstupní veličiny: napájecí napětí kotvy u_a a budícího obvodu u_b. Obecně lze k regulaci použít obě. Matematický model je pak třeba doplnit o popis budícího obvodu. Výsledkem však bude, vzhledem k nelinearitě magnetizační charakteristiky, buď model nelineární, nebo linearizovaný, použitelný jen v úzkém okolí pracovního bodu.

Regulace změnou budícího napětí se však nepoužívá příliš často, přičemž motory malých a středních výkonů (20–40 kW), se kterými se v regulovaných pohonech setkáváme nejčastěji, jsou obvykle buzeny nikoliv elektromagnetem, ale permanentním magnetem a možnost měnit magnetický tok změnou budícího napětí tak zcela odpadá.

Magnetický tok je pak konstantní (zanedbáváme reakci kotvy, resp. předpokládáme, že je kompenzována). Nelineární funkci z (4.1) lze tak nahradit konstantou ξ a indukované napětí kotvy je pak přímo úměrné rychlosti otáčení a je možné psát:

$$\xi = kf(i_{b0}) \tag{4.2}$$

$$u_e = \xi\varpi \tag{4.3}$$

Závislost momentu vytvářeného kotvou motoru na proudu kotvy pak bude také přibližně lineární s konstantou úměrnost ξ a pro momentovou rovnováhu na hřídeli je možné psát.

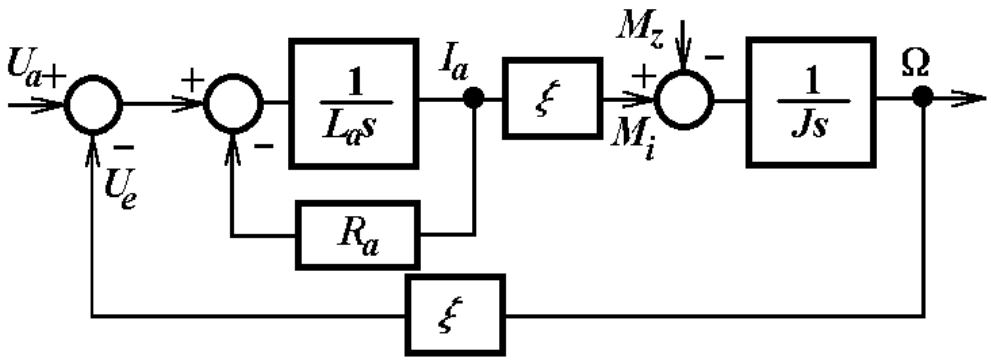
$$\mathrm{m_i} = \xi \mathrm{i_a} = \mathrm{J}\frac{\mathrm{d}\omega}{\mathrm{dt}} + \mathrm{m_z} \tag{4.4}$$

Pomocí Laplaceovy transformace je nyní možné celý model převést do přenosového vyjádření:

$$U_a(s) = R_a I_a(s) + L_a s I_a(s) + \xi\Omega(s) \quad \textbf{(4.5)}$$

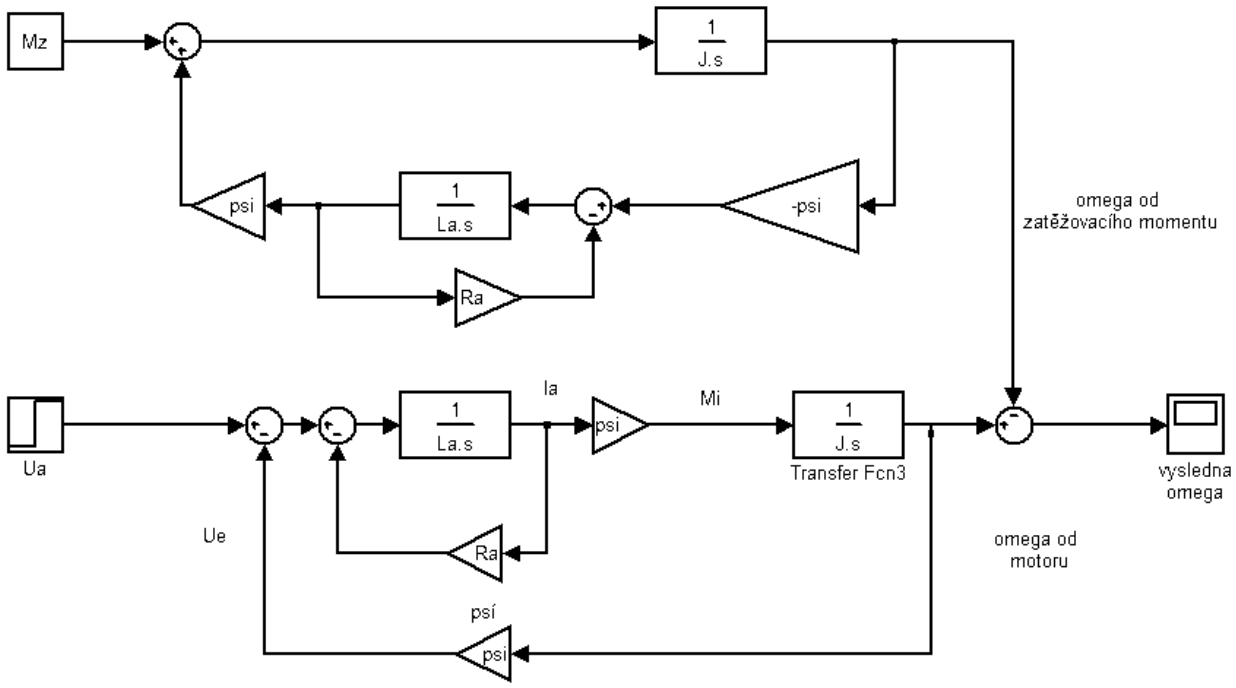
$$Js\Omega(s) = \xi I_a(s) - M_z(s) \quad \textbf{(4.6)}$$

Z těchto rovnic je možné nakreslit blokové schéma uvedené na Obrázek 4.2.



Obrázek 4.2 Blokové schéma stejnosměrného motoru řízeného napětím kotvy, verze 1 [5]

Pro větší názornost a snadnější odvození přenosů, je možné Obrázek 4.2 překreslit dle pravidel blokové algebry, jako dva samostatné systémy jejichž otáčky se budou odčítat. Výsledný model je na Obrázek 4.3.



Obrázek 4.3 Blokové schéma DC motoru řízeného napětím kotvy, verze 2

Symboly použité v Obrázek 4.2 a Obrázek 4.3 mají stejný význam jako na předchozích obrázcích (viz. popis pod Obrázek 4.1).

Dále pak:

ξ … (psi) je konstanta úměrnosti momentu vytvářeného kotvou na proudu kotvy

Ω … úhlová rychlost hřídele

4.3 FYZIKÁLNÍ PARAMETRY MOTORU

4.3.1 Úvod

Získat konkrétní fyzikální parametry motoru nebylo jednoduché, protože každá firma si tyto parametry tají jako své „Know-how“. Z toho důvodu se mi nikde v literatuře ani na internetu nepodařilo najít věrohodný zdroj těchto informací. Obrátil jsem se tedy na doc. Koláčného z UVEE FEKT, který se elektrickými pohony zabývá. Ten mi poskytl parametry modelu motoru použitého v semestrálním projektu jednoho ze svých studentů.

4.3.2 Volba konkrétních parametrů motoru

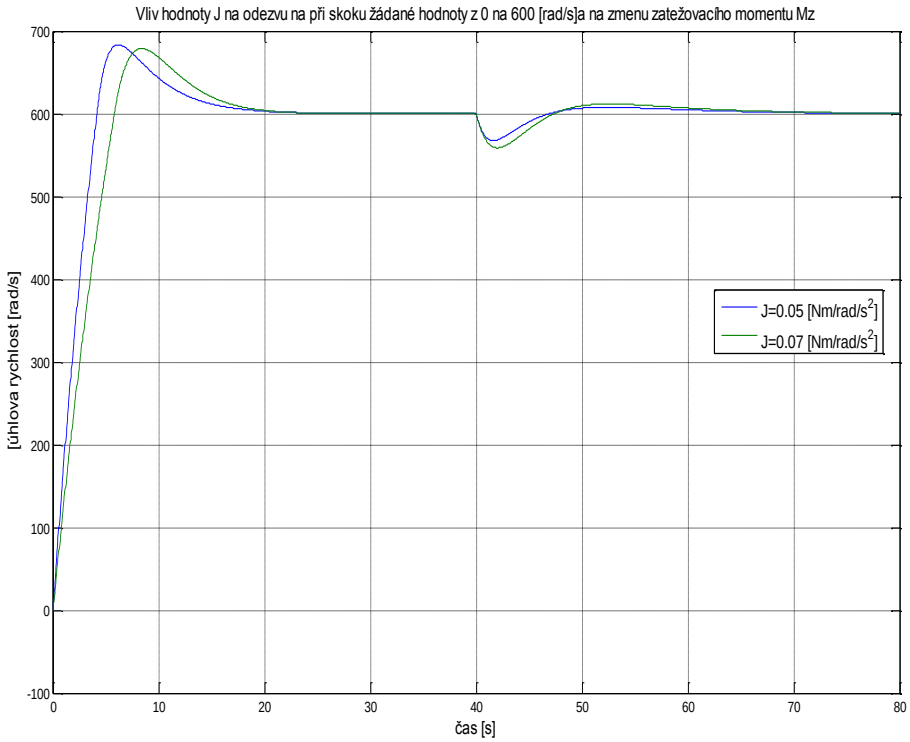
Dominantní časová elektromechanická konstanta původního modelu však byla příliš malá $\tau_m = 0.09s$ (výpočet (4.10)). Při takto malé časové konstantě by na displeji modelu přechodný děj téměř nebyl patrný a změny úhlové rychlosti by se jevily jako skokové.

Parametry původního fyzikálního modelu jsou tedy brány pouze jako výchozí a pomocí modelu v Simulinku byly konečné parametry nastaveny experimentálně tak, aby rychlost přechodných dějů byla optimální a průběh změn úhlové rychlosti pozorovatelný na numerickém displeji modelu. Porovnání parametrů původního a konečného modelu je v Tabulka 4.2.

V modelu je dále implementována možnost měnit zatěžovací moment Mz a setrvačný moment J.

Změna Mz (přepínač P1) simuluje skokovou změnu poruchy.

Druhý přepínač (P2) slouží pro změnu setrvačného momentu J. To má za následek změnu přenosu celé soustavy a je tak možné testovat robustnost navrhovaného regulátoru. Další možností je využití změny parametrů soustavy (J) při úlohách týkajících se návrhu adaptivních regulátorů. Na Obrázek 4.4 je graf, porovnávající vliv změny setrvačného momentu, na průběh regulačního děje modelu s regulátorem, popsaným v kapitole 7.1. Z grafu je patrné, že změna J má vliv jak na dobu přechodného děje, tak na překmit žádané hodnoty.



Obrázek 4.4 Vliv hodnoty zatěžovacího momentu J na průběh regulačního děje

Konkrétní hodnoty Mz a J jsou uvedeny v Tabulka 4.1.

Tabulka 4.1 Hodnoty Mz a J

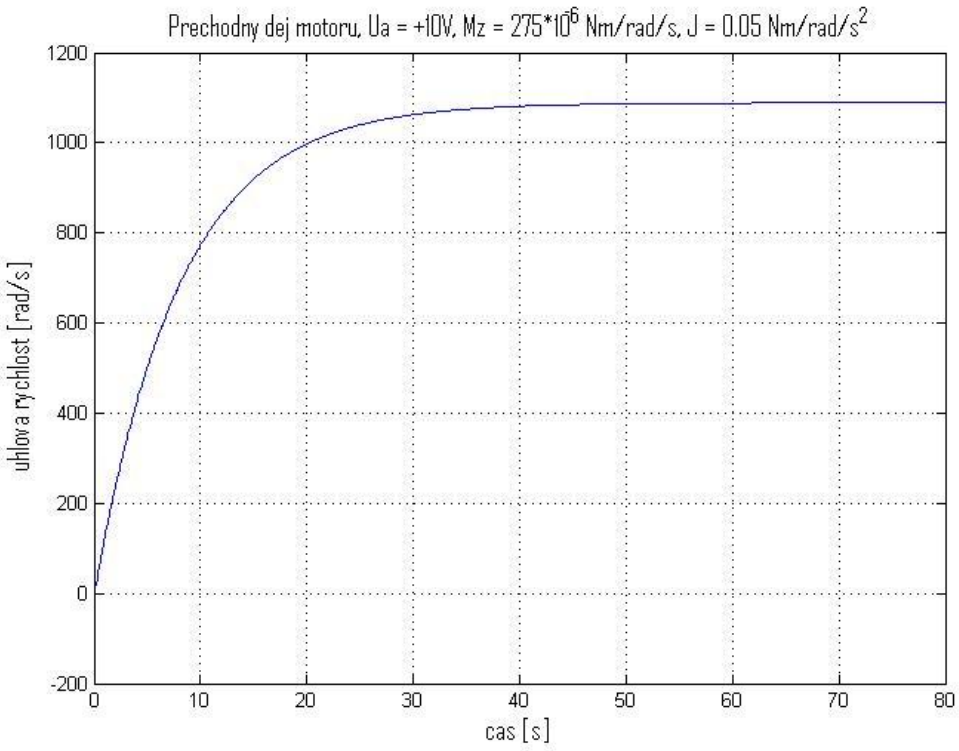
Mz (P1)	vyp.	$2{,}75*10^{-4}$	[Nm/rad/s]
	zap.	2.5	
J (P2)	vyp.	0.5	$Nm/rad/s^2$
	zap.	0.7	

Tabulka 4.2 Fyzikální parametry motoru

Parametr	Původní hodnoty	Upravené hodnoty	Jednotka
La	53,7	53,7	mH
Ra	2,9	2,9	Ω
K_E	0,1356	0,1356	V/rad/s
K_T	0.1324	0,1324	Nm/A
J	0,000557	0,5 - 0.7	$Nm/rad/s^2$
M_Z	0,000275	0,000275 – 2.5	Nm/rad/s

Protože parametry K_E a K_T jsou téměř totožné a vliv jejich rozdílu na chování motoru je nepatrný, je pro jednodušší výpočty a odvození užíván jejich průměr $\xi = 0{,}134$.

Po úpravě parametrů se rozběh i zastavování motoru výrazně zpomalilo a je tedy možné průběh regulace úhlové rychlosti motoru pozorovat i přímo na displeji modelu bez nutnosti záznamu měřených dat např. v Matlabu. Pro představu lze uvést odezvu motoru na skok řídícího napětí z 0V na maximální hodnotu (Ua = +10V). Při nastavených parametrech J = 0,5 $Nm/rad/s^2$ a Mz = 0,000275 Nm/rad/s. Doba, za kterou se úhlová rychlost motoru dostane na maximální reálně možnou hodnotu 1000 [rad/s], je rovna přibližně 20s (viz. Obrázek 4.5). Taková doba je již dostatečně dlouhá k pozorování přechodného děje i na displeji modelu.



Obrázek 4.5 Přechodný děj motoru

4.3.3 Omezení úhlové rychlosti

Každý reálný motor má určitá omezení, při jejichž překročení výrobce negarantuje bezchybný chod a může dojít k poškození motoru. Proto jsou i v tomto modelu podobná omezení implementována.

Pokud je překročena hodnota ω = 800 rad/s, rozsvítí se *ALARM LED* v levém horním rohu modelu, která svítí ještě další 3 vteřiny po poklesu úhlové rychlosti pod 800 rad/s.

Pokud je překročena hodnota ω = 999 rad/s, motor se zadře. To je signalizováno pomlčkami na displeji a v takové případě je nutné motor uvést do původního stavu stiskem tlačítka *RESET*.

4.3.4 Volba zobrazení rychlosti otáčení motoru

Údaj o rychlosti otáčení motoru je na modelu zobrazován pomocí třímístného displeje. Ve všech výpočtech se pracuje s úhlovou rychlostí v radiánech za sekundu a její přepočet na otáčky za minutu by bylo pouhé vynásobení konstantou, které by mělo za následek zvětšení rozsahu hodnot zobrazované veličiny na displeji. To by při

stejných parametrech motoru zapříčinilo nutnost použití relativně zbytečně displej s větším počtem zobrazovaných číslic. Proto je na displeji modelu zobrazována přímo úhlová rychlost v radiánech za sekundu, což sice není u motorů běžná veličina pro udávání rychlosti otáčení, nicméně pro použití v laboratoři tato skutečnost není podstatná.

4.4 DIFERENČNÍ ROVNICE MOTORU

Protože model bude implementován do mikrokontroléru, což znamená diskrétní zpracovávání, je nutné model diskretizovat a získat matematický popis, který je možné naprogramovat v programovacím jazyce C. Takovým tvarem je dvojice diferenčních rovnic, kdy jedna rovnice popisuje účinky řídícího napětí Ua a druhá zatěžovacího momentu Mz. Protože změna J mění celý přenos soustavy, je nutné vypočtení koeficientů pro každou hodnotu J zvlášť.

Z Obrázek 4.3 je možné získat dle pravidel blokové algebry přenos pro úhlovou rychlost samotného motoru F_S a úhlovou rychlost, kterou by vyvolal zatěžovací moment při nulovém napětí kotvy (F_D).

$$x = \frac{\xi}{(L_a s + R_a)J_s} \tag{4.7}$$

$$F_S = \frac{x}{1 + x\xi} = \frac{\xi}{(L_a s + R_a)J_s + \xi^2} \tag{4.8}$$

$$F_D = \frac{L_a s + R_a}{(L_a s + R_a)J_s + \xi^2} \tag{4.9}$$

Vztahy (4.8) a (4.9) je možné algebraicky upravit a získat tak přenosy (4.12) a (4.13), které jsou ve tvaru běžně užívaném v literatuře např. v [5], kde τ_m označuje elektromechanickou a τ_a elektromagnetickou časovou konstantu motoru.

$$\tau_m = \frac{R_a J}{\xi^2} \tag{4.10}$$

$$\tau_a = \frac{L_a}{R_a} \tag{4.11}$$

$G_s(s)$ zde označuje přenos regulované soustavy a $G_d(s)$ přenos poruchové veličiny.

$$G_s(s) = \frac{1}{\xi} * \frac{1}{\tau_a \tau_m s^2 + \tau_m s + 1} \tag{4.12}$$

$$G_d(s) = -\frac{R_a}{\xi^2} * \frac{\tau_a s + 1}{\tau_a \tau_m s^2 + \tau_m s + 1} \tag{4.13}$$

Diskretizace přenosů je provedena v programu Matlab příkazem c2d, kde byla perioda vzorkování zvolena 0.01s, což je doba rovnající se periodě, s níž rovnice počítá mikrokontrolér (viz. kapitola 6.5 Volba periody vzorkování) a diskretizační metodou ‘zoh‘ (zero-order hold on the inputs).

Perioda vzorkování je řádově menší, než dominantní časová konstanta motoru τ_m = 8,07s (pro J = 0.05 Nm/rad/s^2). Při návrhu regulátoru je tedy možné na diskretizovaný model pohlížet stále jako na model spojitý.

Konkrétní hodnoty parametrů dosazených do přenosů (4.12) a (4.13) vycházejí z Tabulka 4.2. Po diskretizaci jsou tvary přenosu soustavy (Gs) a poruchy (Gd) následující:

$$Gs(z) = \frac{a * z + b}{z^2 - c * z + d} \tag{4.14}$$

$$Gd(z) = \frac{e * z - f}{z^2 - c * z + d} \tag{4.15}$$

Hodnoty koeficientů se liší v závislosti na hodnotě J a jsou uvedené v Tabulka 4.3. Diferenční rovnice jsou vypočítávány s relativně krátkou periodou (každých 10ms). To způsobuje, že změny v každé iteraci jsou minimální. Je tedy nutné při výpočtu užívat maximální možný počet desetinných míst. V opačném

případě by docházelo vlivem zaokrouhlování k nezanedbatelné chybě a výsledky z Matlabu by se neshodovaly s výsledky modelu počítaného mikrokontrolérem.

Tabulka 4.3 Koeficienty diferenčních rovnic

Koeficient	J = 0.05 Nm/rad/s^2	J = 0.07 Nm/rad/s^2
a	0.002100680888099	0.001500509346833
b	0.001755287488907	0.001253797423895
c	1.582209849170108	1.582357471825350
d	0.582726548932627	0.582726548932627
e	0.199980420559562	0.142847153255547
f	0.116530358669116	0.083239021650231

Přenosy (4.14) a (4.15) byly převedeny na diferenční rovnice dle postupu uvedeného v [5].

$$Gs(z) = \frac{\mathrm{a} * z^{-2} + \mathrm{b} * z^{-1}}{1 - \mathrm{c} * z^{-1} + \mathrm{d} * z^{-2}} = \frac{\Omega_s(z)}{U_a(z)} \qquad \textbf{(4.16)}$$

$$\Omega_{s(k)} = \mathrm{a} * U_{a(k-1)} + \mathrm{b} * U_{a(k-2)} + \mathrm{c} * \Omega_{s(k-1)} - \mathrm{d} * \Omega_{s(k-2)} \qquad \textbf{(4.17)}$$

$$Gs(z) = \frac{\mathrm{e} * z^{-2} + \mathrm{f} * z^{-1}}{1 - \mathrm{c} * z^{-1} + \mathrm{d} * z^{-2}} = \frac{\Omega_s(z)}{M_z(z)} \qquad \textbf{(4.18)}$$

$$\Omega_{z(k)} = \mathrm{e} * M_{z(k-1)} - \mathrm{f} * M_{z(k-2)} + \mathrm{c} * \Omega_{z(k-1)} - \mathrm{d} * \Omega_{z(k-2)} \qquad \textbf{(4.19)}$$

Ω_s … úhlová rychlost hřídele bez působení zatěžovacího momentu Mz

Ω_z … úhlová rychlost hřídele bez působení napětí kotvy Ua

Odečtením rovnic (4.17) a (4.19) získáváme výslednou úhlovou rychlost motoru v dané iteraci.

$$\Omega_{(k)} = \Omega_{s(k)} - \Omega_{z(k)} \quad [\mathrm{rad/s}] \qquad \textbf{(4.20)}$$

Rovnice (4.17) a (4.19) jsou ve tvaru, který je možné naprogramovat v jazyce C a při aktualizování stavů s periodou rovnou periodě vzorkování zadané při

diskretizace (T_S = 0,01s) je jejich výsledek totožný s hodnotami získanými z modelu v programu Simulink popsaném v kapitole 4.2. Tyto rovnice jsou tedy použity i při výpočtech odezvy motoru mikrokontrolérem.

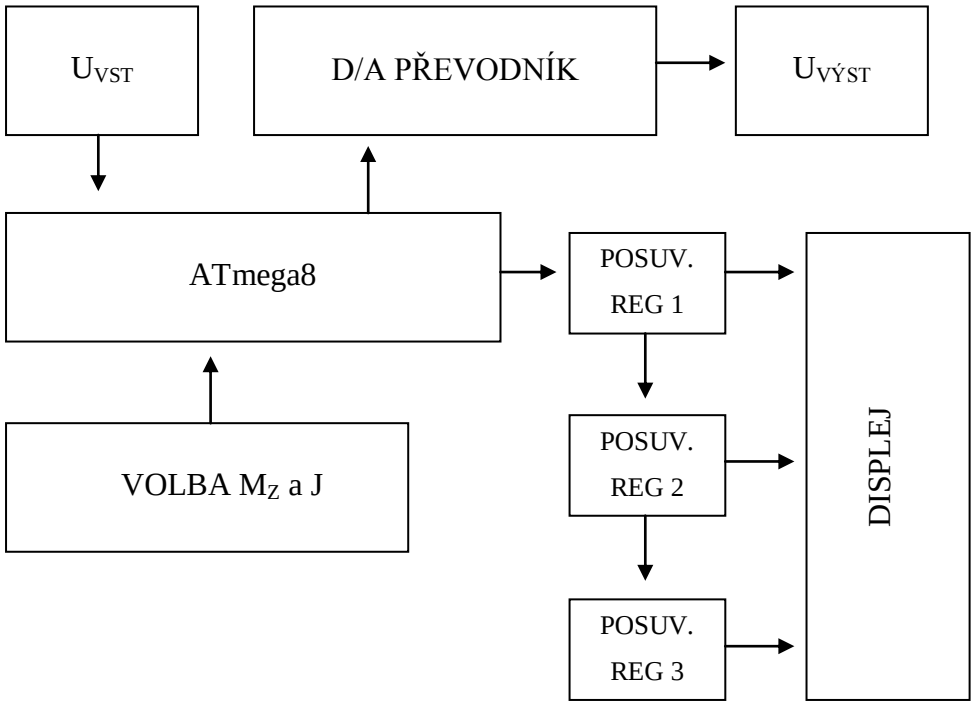
5. NÁVRH HARDWARU

V dnešní době má návrhář pro vývoj mikroprocesorem řízených aplikací široký výběr mikropočítačů. Od primitivních několika pinových zařízení pro velmi jednoduché aplikace, až po 32-bitové výkonné procesory. Na trhu je také široká škála výrobců jako např. Microchip, Atmel, Motorola atd. Pro tento model byl vybrán mikrokontrolér z rodiny AVR od firmy Atmel a to především pro širokou softwarovou podporu.

Při výběru konkrétního modelu byly výchozí požadavky na mikrokontrolér následující: alespoň 14 vstupně-výstupních bran, integrovaný časovač a analogově-digitální převodník. Všechny tyto požadavky splňuje mikrokontrolér ATmega8-16PU, který je k řízení modelu použit.

5.1 ARCHITEKTURA ZAPOJENÍ

Jak je uvedeno výše, srdcem modelu je mikrokontrolér ATmega8-16PU [7], ten přes SPI posílá data dvanáctibitovému D/A převodníku MCP4922 [8], který zajišťuje výstupní napětí modelu. Dále jsou na desce tři posuvné registry 74HCT59 [9]. První z nich je s mikrokontrolérem propojen přes SPI a do následujících dvou se data postupně „nasouvají“. Výstupy posuvných registrů jsou připojeny k třímístnému sedmi segmentovému displeji BT-A515RD [10] a rozsvěcují jednotlivé segmenty. Dále jsou na modelu dva spínače, sloužící pro změnu zatěžovacího momentu a změnu hodnoty momentu setrvačnosti motoru. Tlačítko RESET, které resetuje program v mikrokontroléru a model uvede do původního stavu, kdy jsou otáčky nulové. Na vrchní desce pod displejem je dále čtveřice LED diod indikujících aktuální stav zátěže a momentu setrvačnosti a jedna signalizační LED indikující překročení meze úhlové rychlosti.



Obrázek 5.1: Blokové schéma modelu

5.1.1 Napájení

Model je napájen přímo ze zdroje PLC +24V DC a do modelu je přiváděno přes plochý, v laboratoři standardní, dvacetižilový kabel.

K napájení modelu bylo původně zamýšleno použít DC/DC měnič z 24V na 5V (L7800). Ten se však vlivem výše proudu odebíraného modelem (200mA) i s připojeným chladičem velmi hřál a po uzavření celého modelu do montážní krabičky by pravděpodobně životnost součástek vlivem značně vysokých teplot nebyla dlouhá.

Ve výsledném modelu je proto použit spínaný zdroj TPS54231 [11], který mění vstupní napětí 24V na 5,21V, což je hodnota použitá k napájení všech součástek modelu. Tento konkrétní model spínaného zdroje sice není v ČR běžně dostupný, avšak firma Texas Instruments nabízí zasílání[1] vzorků svých součástek zdarma po celém světě a tak nebylo získání součástky problémem.

[1] Více informací o programu Sample & Buy je možné získat na www.ti.com.

Konečné parametry zdroje jsou v porovnání s původním DC/DC měničem výrazně lepší. Zdroj se nezahřívá, model je možné napájet napětím v rozmezí přibližně od 8V až do 28 V a maximální odebíraný proud udávaný výrobcem v [11] je až 2A, což je desetkrát větší hodnota proudu, než model skutečně odebírá.

5.1.2 Vstupní napětí

Použitá výstupní karta PLC B&R X20AO4632 dává výstupní napětí $V_{out} = \pm 10V$, proto je A/D převodník integrovaný na ATmega8 připojen přes odporový dělič (2x 1kΩ), díky kterému není možné přivést na vstup A/D převodníku napětí vyšší než 5V. Jako ochrana, proti nechtěnému otočení polarity vstupního napětí, je zapojena dioda 1N4148.

5.1.3 Výstupní napětí a komunikace s D/A převodníkem

Výstupní napětí je zajištěno integrovaným obvodem MCP4922 [8]. Mikrokontrolér předává D/A převodníku data pomocí SPI komunikace, to znamená, že jsou využívány tři vodiče. Logická nula na pinu /CS aktivuje D/A převodník pro komunikaci. Piny SCK a SDI slouží pro hodinový a datový signál. Na D/A převodníku je vždy, po vypočtení nové hodnoty úhlové rychlosti, nastavena nová hodnota napětí, úměrná této úhlové rychlosti. D/A převodníku je pokaždé posíláno šestnácti bitové slovo. První čtyři bity jsou konfigurační a nastavují režimy D/A převodníku, zbylých dvanáct bytů je datových.

Protože hodnota úhlové rychlosti se může pohybovat v rozmezí 0 až 999 rad/s a D/A převodník je dvanácti bitový, což dává možnost rozdělit rozsah převodníku na 4096 hodnot, je vždy hodnota úhlové rychlosti násobena čtyřmi. Tím je zajištěna lepší odolnost vůči případnému rušení a bezchybné měření na vstupní kartě PLC.

Rozsah převodníku je od 0 až V_{ref}, které je rovno napájecímu napětí 5,21V. Pro hodnotu maximální úhlové rychlosti (999 rad/s) je tedy převodníku odesílaná hodnota 999*4 = 3996, což je úměrné hodnotě výstupního napětí převodníku 5,083V.

$$V_{out} = \frac{V_{ref} * \Omega * 4}{2^n} = \frac{5{,}21 * 999 * 4}{2^{12}} = 5{,}083V \qquad \textbf{(5.1)}$$

V_{out} … Výstupní napětí převodníku [V]

V_{ref} … Referenční napětí [V]

Ω … Úhlová rychlost motoru [rad/s]

n … počet bitů D/A převodníku

5.1.4 Zobrazení úhlové rychlosti

Displej byl původně řízen multiplexně. Toto řešení však kvůli omezenému maximálnímu proudu, který je mikrokontrolér schopný dodávat, nebylo bez přidání dalších obvodů pro zesílení vyhovující a čitelnost displeje nebyla na přímém světle dobrá.

Proto je aktuální hodnota úhlové rychlosti zobrazována na třímístném displeji, který má pro každou číslici společnou anodu připojenou na +5V a každý jednotlivý segment displeje má pak vlastní katodu, připojenou přes odpor (470 Ω) k pinu jednoho ze tří posuvných registrů. Přes posuvné registry jsou jednotlivé katody buď uzemněny (svítí) nebo přivedeny na stejný potenciál jako anoda (nesvítí).

Odpor je připojen ke každému segmentu zvlášť, protože při připojení odporu pouze ke společné anodě, by se sice ušetřilo místo na DPS a zapojení by bylo jednodušší, avšak za cenu zhoršených zobrazovacích vlastností. Protože vlivem různého počtu rozsvícených segmentů jednotlivých číslic a tím i různého odběru proudu při zobrazení například číslice „1“ a „8“, by na společném odporu vznikal různý úbytek napětí, což by mělo za následek různé napětí na LED segmentech a jejich různou svítivost.

5.2 FYZICKÉ PROVEDENÍ MODELU

Celý model (viz. Obrázek 5.2 a fotografie v příloze E) je uzavřen v montážní krabičce WEB-B8. Skládá se ze dvou plošných spojů umístěných na sobě a vzájemně propojených dvěma dvouřadými dvaceti pinovými konektory. Na spodní desce je mikrokontrolér, tři posuvné registry, D/A převodník, spínaný zdroj s nezbytnými součástkami a konektor pro připojení PLC. Na horní desce je umístěn displej, přepínače pro volbu zatěžovacího a setrvačného momentu společně s pětici signalizačních diod.

Obrázek 5.2 Fyzické provedení modelu

5.2.1 Návrh a výroba DPS

Schéma i následný návrh desky byl vytvořen v programu Eagle 5.7.0. Většina pouzder použitých součástek byla součástí defaultních knihoven programu. Pouze pouzdro pro displej nebylo obsaženo ve standardní knihovně a ani na internetu a tak jej bylo nutné vytvořit, dle rozměrů uvedených v [10].

Vyleptání DPS proběhlo v laboratoři na UAMT FEKT. Vlastní vrtání s osazením bylo již provedeno v domácích podmínkách.

5.3 PROPOJENÍ MODELU S PLC

K propojení modelu s PLC slouží plochý dvacetižilový kabel, který odpovídá zapojení dle Tabulka 5.1. Toto zapojení je společné i pro ostatní obdobné modely v laboratoři a bude tak možné k jednomu PLC jednoduše připojit různé modely bez nutnosti nepohodlného přepojování jednotlivých vodičů přímo do karet PLC.

Model nevyužívá všechny vodiče konektoru a proto příslušné piny zůstaly nezapojeny (viz. zašedlé řádky Tabulka 5.1).

Piny 6 a 7 jsou každý připojen na jeden z výstupů D/A převodníku. Oba výstupy jsou však programovány tak aby dávaly stejné napětí a proto pin 7 ve

skutečnosti není PLC monitorován a slouží jen jako rezerva, například při nechtěném zkratování a zničení jednoho z kanálů převodníku.

Tabulka 5.1 Zapojení propojovacího kabelu k PLC

PIN	Znační	Popis
1	24V	24V z PLC z karty BR9300
2	24V	24V z PLC z karty BR9300
3	GND	GND z PLC z karty BR9300
4	GND	GND z PLC z karty BR9300
5	xxxxxxx	bez pinu
6	AI_1	Analogový vstup do PLC
7	AI_2	Analogový vstup do PLC
8	AI_3	Analogový vstup do PLC
9	AO_1	Analogový výstup z PLC
10	AO_2	Analogový výstup z PLC
11	AO_3	Analogový výstup z PLC
12	xxxxxxx	bez pinu
13	DI_1	Digitalní vstup do PLC
14	DI_2	Digitalní vstup do PLC
15	DI_3	Digitalní vstup do PLC
16	DI_4	Digitalní vstup do PLC
17	DO_1	Digitální výstup z PLC
18	DO_2	Digitální výstup z PLC
19	DO_3	Digitální výstup z PLC
20	DO_4	Digitální výstup z PLC

6. FIMWARE PRO ATMEGA8

6.1 ÚVOD

Vzhledem k minimálním předchozím zkušenostem s mikrokontroléry AVR bylo nejprve nutné seznámit se s architekturou čipu, jeho registry, možnostmi a způsobem programování. V tomto byly jedny z nejužitečnějších zdrojů internetové tutoriály [11] a [12]. Ani jeden z nich se sice přímo nevěnuje ATmega8, avšak mikrokontroléry AVR mají velmi podobnou základní strukturu a tak byly tyto informace velice přínosné. Největší zdrojem informací však byl datasheet mikrokontroléru ATmega8 [7], kde je celý mikrokontrolér podrobně popsán jak obvodově, tak z hlediska způsobu programování.

Celý fimware je psán v jazyce C a k jeho tvorbě a ladění byl použit program AVR Studio v4.16, který je volně dostupný ze stránek výrobce (*www.atmel.com*).

6.2 PŘEHRÁNÍ FIMWARE

Mikrokontrolér je v modelu vsazen do patice a v případě nutnosti je tedy možné mikrokontrolér vyjmout, přeprogramovat a opět vložit do modelu.

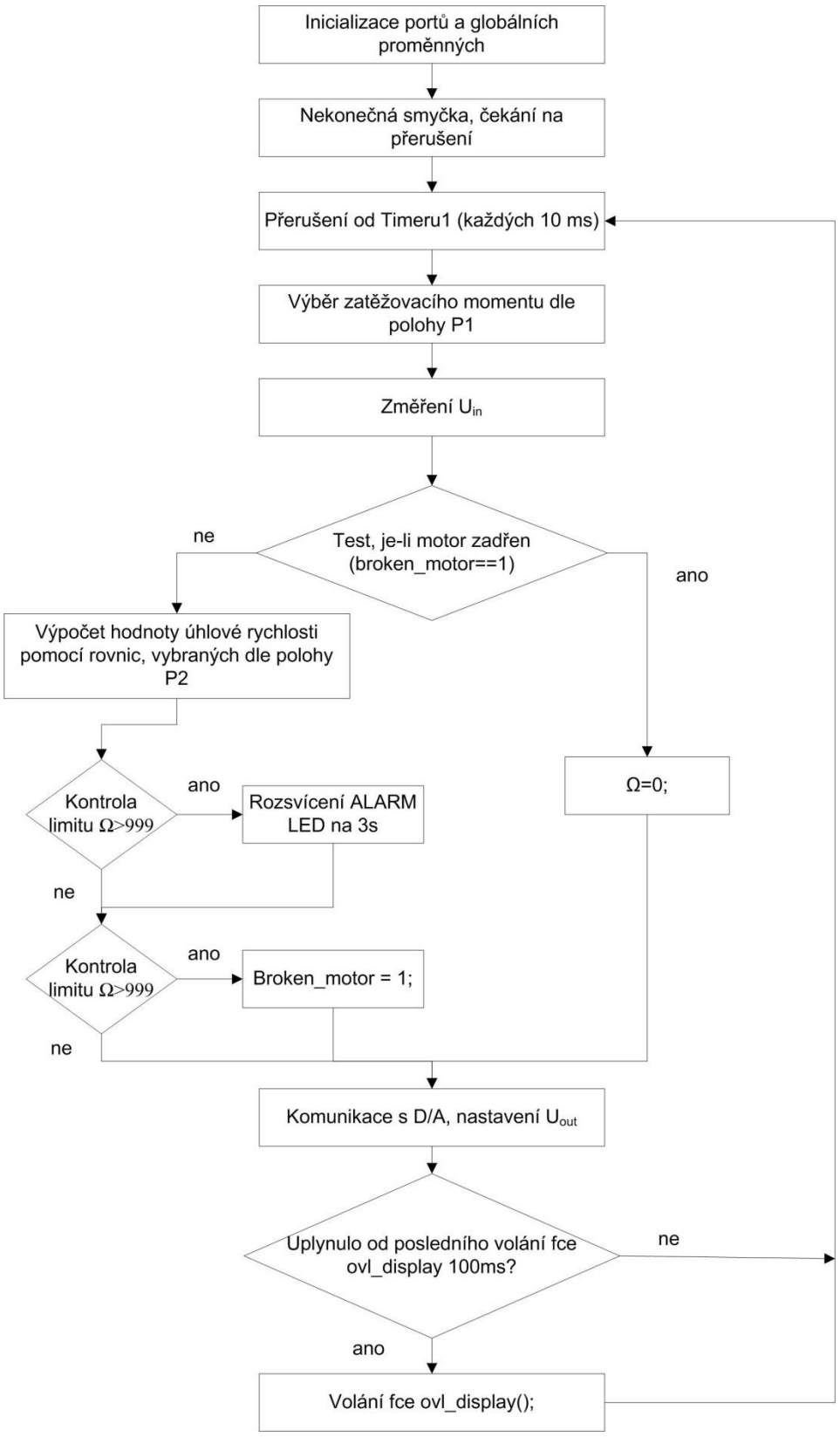
6.3 ARCHITEKTURA PROGRAMU

Po spuštění programu je mikrokontrolér nejprve inicializován (nastavení portů, časovačů, A/D převodu atd.) voláním funkce *inicialization()*. Poté program pracuje v nekonečné smyčce a čeká na přerušení od Timer1. To nastane každých 10ms.

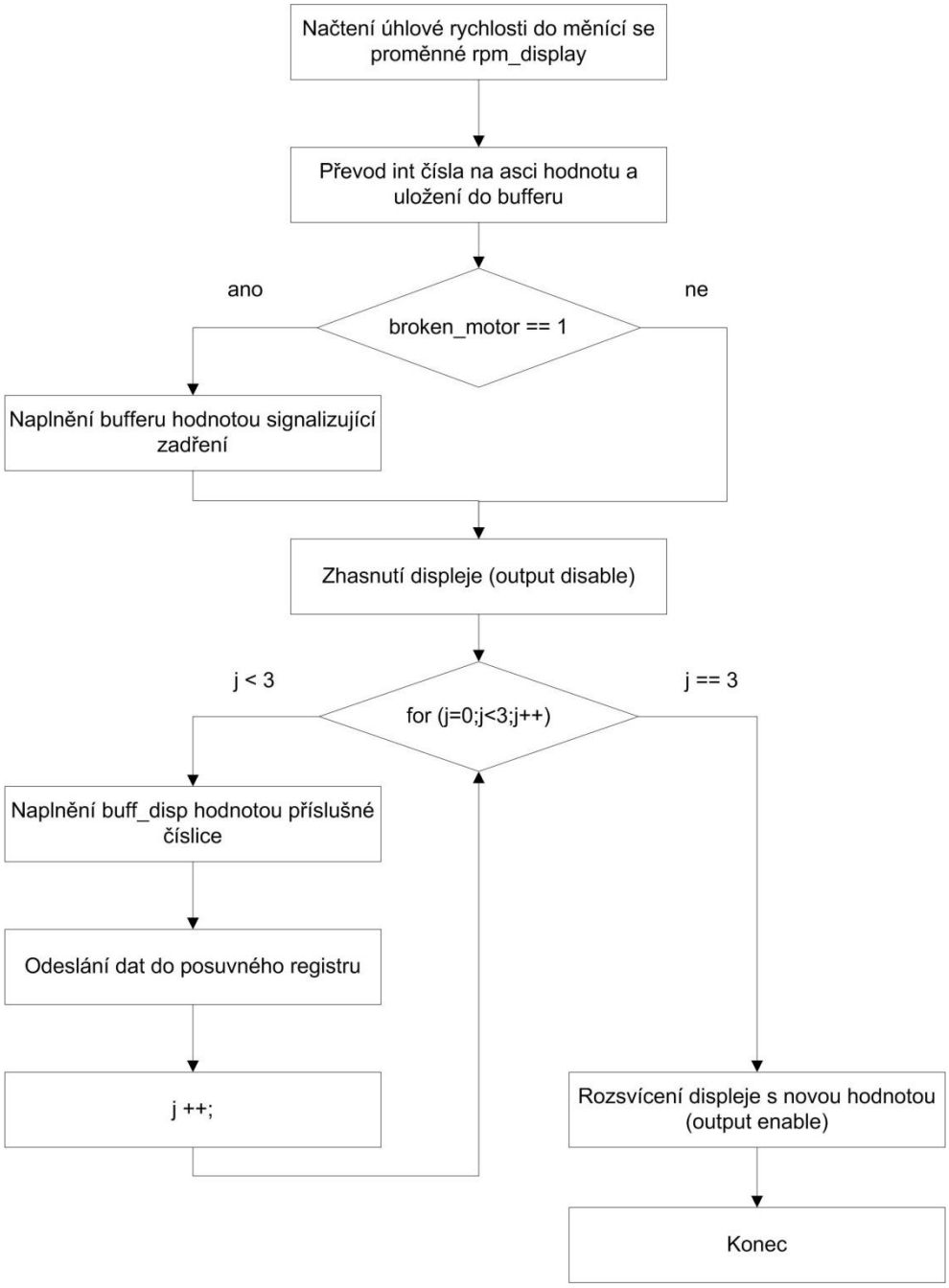
Při přerušení od Timer1 se nejprve provede změření vstupního napětí, aktualizuje se stav motoru výpočtem nových hodnot diferenčních rovnic, odešlou se data do D/A převodníků a jsou zkontrolovány limity pro ALARM a poškození motoru. Každé desáté přerušení je navíc volána funkce *ovl_display()*. Ta převede hodnotu úhlové rychlosti z int na asicii hodnotu, což umožní rozčlenění hodnoty na jednotlivé číslice. Data odpovídající jednotlivým číslicím jsou pak postupně „nasunuty“ pomocí SPI komunikace do tří sériově zapojených posuvných registrů, čímž je zajištěno zobrazení aktuální úhlové rychlosti na displeji.

Funkce *ovl_display()* není volána při každém přepočtu diferenčních rovnic (každých 10ms), ale každé desáté přerušení (jednou za 100ms) a to kvůli stabilnějšímu zobrazení. Změna hodnoty na displeji každých 10ms by totiž působila rušivě.

Vývojové diagramy, ze kterých je patrná struktura programu, jsou na Obrázek 6.1 a Obrázek 6.2.



Obrázek 6.1 Vývojový diagram hlavní funkce



Obrázek 6.2 Vývojový diagram funkce *ovl_display()*

6.4 PODROBNÝ POPIS JEDNOTLIVÝCH FUNKCÍ PROGRAMU

6.4.1 Funkce – *inicialization()*

Toto je první volaná funkce po spuštění programu (připojení mikrokontroléru k napájení). Zajišťuje inicializaci a správné nastavení všech vstupně výstupních portů, A/D převodníku, časovačů a povolení přerušení. Po provedení těchto operací program spadne do nekonečné smyčky a čeká na přerušení od časovače 1.

6.4.2 Přerušení od časovače 1

Vnitřní RC oscilátor mikrokontroléru je nastaven na 8MHz, před dělička tohoto časovače je nastavena na 64 a časovač pracuje v režimu TOP = OCR1A. V registru OCR1A je zapsána hodnota 1250. To znamená, že každých 10 ms je vyvoláno přerušen od toho časovače. Jakmile časovač dosáhne hodnoty zapsané v registru OCR1A, automaticky se vynuluje a začne čítat od nuly. Tím je zajištěna pevná perioda 10 ms. Podrobnější informace o funkci jednotlivých registrů, spojených s časovačem, jsou uvedeny v [7].

Jakmile přerušení od časovače 1nastane, aktualizuje se hodnota zatěžovacího momentu M_Z a rozsvítí se příslušná LED.

Poté se zahájí převod napětí na vstupu A/D převodníku. Po dokončení převodu se nastaví příznakový bit ADIF a výsledek převodu se uloží do proměnné *uak* typu int.

V dalším kroku je kontrolováno, není-li nastavena příznaková proměnná *broken_motor* a pokud ne, je dle polohy P2 vybrána dvojice diferenčních rovnic odpovídající vybranému momentu setrvačnosti J, ty jsou aktualizovány a je rozsvícena příslušná LED.

Dále jsou zkontrolovány limity $\Omega > 800$ rad/s a $\Omega > 999$ rad/s a pokud je některý z nich překročen nastaví se příslušné bity a je rozsvícena ALARM LED.

Hodnota výstupního napětí, odpovídajícího aktuální hodnotě úhlové rychlosti, je poté spolu s konfiguračními byty přes SPI odeslána do D/A převodníku, který tuto hodnotu ponechá nastavenou na svém výstupu, až do přijetí hodnoty nové.

Nakonec je inkrementována proměnná *slow_down_disp* a pokud je její hodnota rovna deseti, je vynulována a volá se funkce *ovl_display()*.

6.4.3 Funkce *ovl_display()*

Nejprve jsou deklarovány lokální proměnné *i*, *j* a *buff_disp*, které jsou užívány pouze v této funkci.

Funkcí *itoa(int __val,char * __s,int __radix)* je převedena int hodnota úhlové rychlosti na řetězec ascii číslic, který je uložen do pole znaků *buf*. Pokud je hodnota úhlové rychlosti menší než 100, respektive 10, je před řetězec s číslem připsána jedna, případně dvě nuly.

Je-li příznaková proměnná *broken_motor* nastavena na 1, je proměnná *buf* naplněna hodnotami 58, symbolizující zadřený motor.

Dále je na *output enable* piny posuvných registrů přivedena log. 1, čímž je displej odpojen od napájení. Tím se zajistí, že během „nasouvání“ dat do posuvných registrů je displej zhasnutý a namísto postupného měnění jednotlivých segmentů, což by působilo rušivě, displej na velmi krátkou dobu (0.4ms) pohasne, což lidské oko nezaznamená.

Poté následuje třikrát se opakující cyklus pro výběr a odeslání dat do posuvných registrů. V tomto cyklu je nejprve proměnná *buff_disp* naplněna daty pro rozsvícení segmentů odpovídajících dané číslici a obsah *buff_disp* je odeslán pomocí SPI komunikace do posuvného registru.

Po naplnění registrů daty odpovídajícími aktuální úhlové rychlosti, jsou *output enable* piny posuvných registrů uzemněny a výstupní piny posuv. registrů se nastaví na odpovídající logické úrovně.

6.5 VOLBA PERIODY VZORKOVÁNÍ MODELU

Vzhledem k tomu, že se jedná o model reálného DC motoru, který je ze své podstaty spojitý systém, je při jeho číslicovém (a tedy diskrétním) modelování nutné aby byly stavy modelu aktualizovány co nejrychleji. Jen tak je poté možné zanedbat rozdíl mezi spojitou a diskrétní verzí modelu.

Limitním parametrem, pro volbu co nejmenší periody, se kterou bude model aktualizován, byl výkon mikrokontroléru, ovlivňující počet desetinných míst, na které je mikrokontrolér schopný dostatečné rychle diferenční rovnice počítat. Se zmenšující se periodou vzorkování, se však zmenšuje i změna výsledku v jednom

kroku a tím se zvětšuje vliv zaokrouhlení. Například při počítání s koeficienty diferenčních rovnic zaokrouhlených na 5 desetinných míst a periodě vzorkování 10ms, se hodnota úhlové rychlosti po 20 sekundách od skoku akčního zásahu z 0 na 5V, liší od přesné hodnoty získané simulací v Matlabu přibližně o 12%. Hodnota chyby navíc s dobou simulace rostle. Pro koeficienty zadané s přesností na 15 desetinných míst je již rozdíl mezi hodnotami získanými simulací a přímo na modelu zanedbatelný (méně než 1%). Taková přesnost si však vyžádala počítání rovnic s proměnnými typu *double*.

Výpočet diferenčních rovnic s koeficienty typu *double*, spolu s dalšími částmi programu, jako ovládání displej, A/D převod, komunikace s D/A převodníkem atd., zabere mikrokontroléru přibližně 31 000 strojových cyklů (získáno pomocí debuggeru v programu AVR Studio). To je při frekvenci vnitřního oscilátoru 8 MHz rovné době přibližně 4 ms.

Vzhledem k velikosti dominantní časové konstanty motoru (τ_m = 8,07s pro J = 0.05 Nm/rad/s^2) je v programu ponechána rezerva a model je aktualizován každých 10 ms. Což je vzhledem ke zvyklosti volit periodu vzorkování regulátorů rovnu jedné třetině až třicetině největší časové konstanty, dostatečně jemné vzorkování.

7. PROGRAM PRO PLC

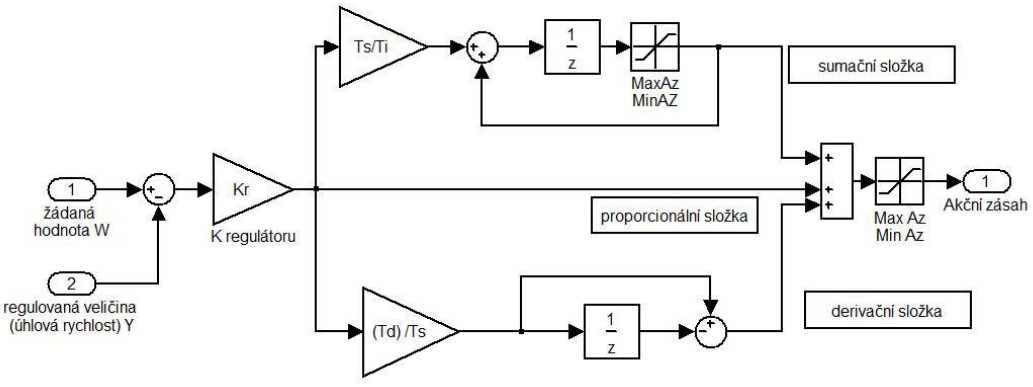
Součástí této práce je program s implementovaným algoritmem PSD regulátoru a vizualizací pro PLC B&R (typ 4PP480.1043-75).

Program byl vytvořen ve vývojovém prostředí Automation Studio v 3.071.10 firmy B&R a je postaven na šabloně obsahující v sobě hardwarové konfigurace všech PLC v laboratoři číslicové a řídící techniky UAMT FEKT VUT. Jednotlivé konfigurace se samozřejmě časem inovují, ale i přes to by po aktualizování změn v HW konfiguraci mělo být možné program přenášet mezi jednotlivými PLC v laboratoři.

Program s nastavením HW konfigurace pro PLC, v laboratoři označeném DA11, je na přiloženém DVD.

7.1 REGULÁTOR

V PLC je naprogramován algoritmus PSD regulátoru s omezením sumační složky, který je popsán například v [15] a jehož struktura je patrná z Obrázek 7.1.



Obrázek 7.1 Struktura PSD regulátor s omezením integrační složky

Regulátor pracuje s pevně nastavenou periodou vzorkování Ts = 100 ms. Parametry regulátoru, jako je zesílení Kr, integrační (Ts) a derivační (Td) časová konstanta, je možné měnit pomocí rozhraní na

dotykovém panelu PLC. Inicializační hodnoty těchto parametrů byly získány metodou optimálního tvaru frekvenční charakteristiky pro PID regulátor, který byl diskretizován a výsledné hodnoty parametrů dodatečně experimentálně upraveny pomocí simulace na modelu motoru pro Matlab, popsaného v kapitole 4.2. Přehled konkrétních hodnot všech parametrů regulátoru je uveden v Tabulka 7.1, kde jsou parametry editovatelné pomocí vizualizačního rozhraní podbarveny modře.

Tabulka 7.1 Inicializační hodnoty parametrů regulátoru

Kr	0,05
Ti	4,5
Td	0,0189
Ts	0,1
MaxAZ	10
MinAZ	0

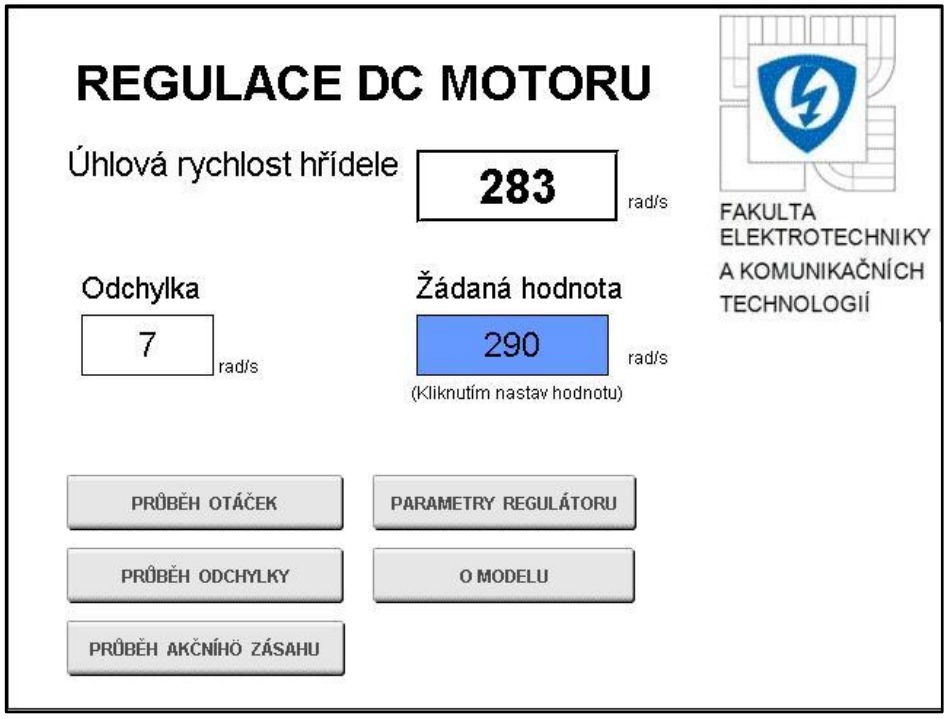
Kód, zajišťující funkci regulátoru je umístěn ve smyčce Cyclic#4, která probíhá každých 100ms (identické s periodou vzorkování regulátoru) a k nahlédnutí je v příloze G.

7.2 VIZUALIZACE

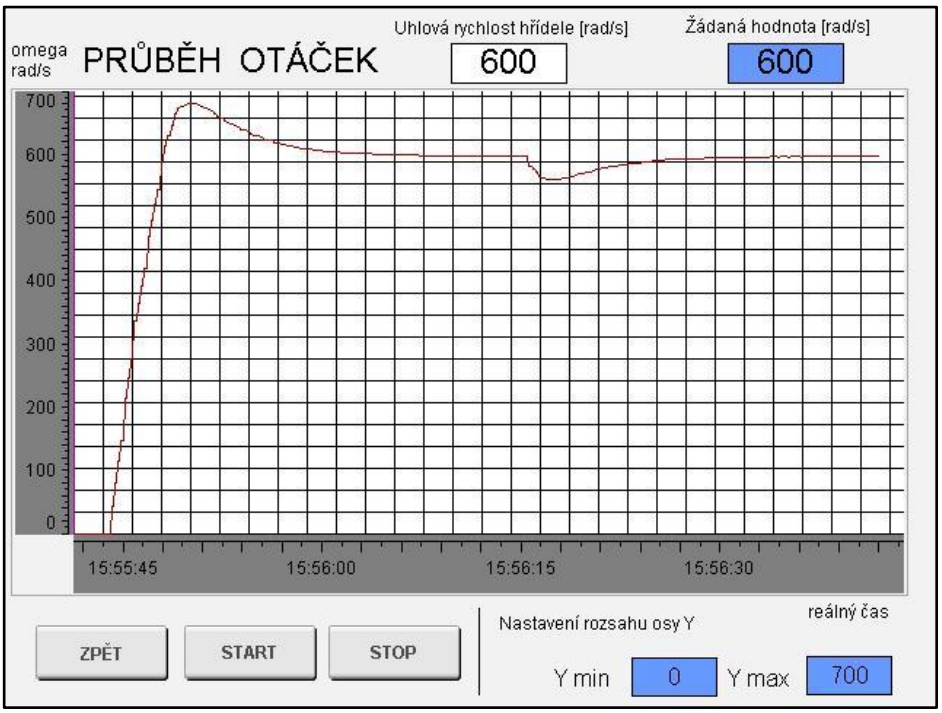
Pomocí vizualizace je možné nastavovat základní parametry PSD regulátoru (Kr, Ti, Td) a dále sledovat průběhy regulované veličiny, akčního zásahu a odchylky. Na úvodní stránce vizualizace (Obrázek 7.2) jsou zobrazeny základní údaje o stavu modelu a dále pětice tlačítek, sloužících k přepínání mezi jednotlivými stranami vizualizace.

Hodnoty, které jsou ve vizualizaci podbarvené modře, je možné editovat pomocí virtuální klávesnice, která se zobrazí na displeji po dotyku modré oblasti prstem. Tak je možné nastavovat například žádanou hodnotu, či rozsah osy Y na grafech (viz. Obrázek 7.3). Tlačítko *zpět* slouží pro návrat na hlavní stranu a je na všech stranách kromě úvodní. Tlačítko stop slouží k zastavení aktualizace hodnot

grafu a po jeho stisknutí zůstane graf neměnný. Stisknutím tlačítka *start* se starý průběh vymaže a začne nové vykreslování.



Obrázek 7.2 Úvodní strana vizualizace



Obrázek 7.3 Ukázka grafu vizualizace - průběh otáček

8. OVĚŘENÍ FUNKČNOSTI MODELU

8.1 POROVNÁNÍ SIMULACE A VÝSLEDNÉHO MODELU

Ověření funkčnosti modelu bylo provedeno porovnáním průběhů simulace v programu Matlab, popsané v kapitole 4.2 a průběhů modelu propojeného s PLC. Vybrané hodnoty jsou uvedeny v Tabulka 8.1, a vycházejí z průběhů uvedených v příloze D.

Tabulka 8.1 Porovnání hodnot získaných simulací v Matlabu a na modelu

	Simulace Matlab	Model	Poznámka
Odezva na skok akčního zásahu z 0V na 5V v čase 45s	341,5 [rad/s]	340 [rad/s]	Mz=2,5 [Nm/rad/s] J=0.05 [Nm/rad/s^2]
Hodnota překmitu při skoku žádané hodnoty W z 0 na 600 [rad/s]	83 [rad/s]	82 [rad/s]	Mz=2,75*10^{-4} [Nm/rad/s] J=0.05 [Nm/rad/s^2]
Maximální hodnota odchylky, při skoku zatěžovacího momentu Mz z 2,75*10^{-4} na 2,5 [Nm/rad/s]	37 [rad/s]	37 [rad/s]	W=600 [rad/s] J=0.05 [Nm/rad/s^2]
Ustálená hodnota akčního zásahu pro žádanou hodnotu W=600 [rad/s]	4 V	4 V	Mz=2,75*10^{-4} [Nm/rad/s] J=0.05 [Nm/rad/s^2]

8.2 CHYBA MODELU

Z Tabulka 8.1 je patrné, že výsledky simulace v Matlabu se s výsledky modelu téměř shodují a jejich rozdíl (menší než 1%) je minimální a tudíž zanedbatelný.

Nepřesnosti, způsobují především chyby, vznikající kvantováním během převodu vstupního napětí na digitální hodnotu a zpětného převodu hodnoty úhlové rychlosti na úměrné napětí.

Další nepatrná chyba je do měření vnesena při přepočtu výsledku převodu na hodnotu akčního zásahu, vlivem zaokrouhlené konstanty *k*, viz. (8.1), (8.2) a (8.3).

$$\mathrm{ADC} = \frac{\mathrm{V_{in}} * 1024}{\mathrm{V_{ref}}} = \frac{5 * 1024}{5{,}21} = 982{,}7255 \qquad \textbf{(8.1)}$$

$$k = \frac{200}{982{,}7255} \doteq 0{,}20352 \qquad \textbf{(8.2)}$$

$$U = k * ADC' = 0{,}20352 * 983 \doteq 200{,}06V \qquad \textbf{(8.3)}$$

ADC … teoretický (nezaokrouhlený) výsledek A/D převodu

ADC´ … skutečný výsledek A/D převodu

V_{in} … vstupní napětí

V_{ref} … referenční napětí převodníku (rovno napájecímu napětí)

k … konstanta pro úpravu výsledku převodu

U … virtuální napětí, které by bylo připojené na reálný motor

9. MANUÁL A ZADÁNÍ PRO MODEL

Kapitola obsahuje manuál, kde je stručně popsáno zapojení a ovládání modelu. Dále pak příklad možného zadání laboratorní úlohy pro studenty bakalářského studia. Pro studenty navazujícího magisterského studia se nabízejí další úkoly týkající se adaptivity a robustnosti navržených regulátorů.

9.1 MANUÁL

Jedná se o model naprogramovaný v mikrokontroléru, který je propojen s PLC a simuluje chování stejnosměrného motoru s cizím buzením permanentním magnetem.

S PLC se s model propojuje přiloženým plochým kabelem. Na pin 9 propojovacího konektoru je přiváděno řídící napětí kotvy (0 až 10 V). Na piny 6 a 7 je přiváděn identický výstupní signál 0 až 5,08V, který je úměrný úhlové rychlosti hřídele motoru, v rozsahu 0-999 [rad/s]. Aktuální hodnota úhlové rychlosti je zobrazována na třímístném displeji. Přepínačem P1 (Mz) se volí mezi dvěma hodnotami zatěžovacího momentu působícího na hřídel. Přepínač P2 (J) slouží ke změně setrvačnosti připojené zátěže (změna setrvačného momentu působícího na hřídel motoru).

Pokud úhlová rychlost hřídele (Ω) překročí hodnotu 800 [rad/s], rozsvítí se ALARM LED, která svítí další 3s po poklesu Ω pod 800 [rad/s]. Pokud Ω překročí 999 [rad/s], motor se zadře, což je signalizováno pomlčkami na displeji. V takovém případě je možné motor uvést znovu do chodu pouze stiskem tlačítka *Reset*.

Obrázek 9.1 Model DC motoru

9.2 ZADÁNÍ

a) Seznamte se s principem činnosti a regulace stejnosměrného motoru s cizím buzením permanentním magnetem.

b) Propojte model s PLC a v programu Automation Studio vytvořte projekt, který vám umožní měnit akční zásah a sledovat průběh úhlové rychlosti.

c) Identifikujte soustavu, například pomocí přechodové charakteristiky.

d) Navrhněte regulátor pro identifikovanou soustavu a ten implementujte do PLC. Ověřte jeho funkčnost.

e) Pokuste se navrhnout regulátor se strukturou omezující hodnotu překmitu. Například některý z popsaných v [14]. Pro ověření můžete použít signalizační ALARM LED.

10. ZÁVĚR

Výsledkem práce je funkční přípravek, simulující činnost stejnosměrného motoru s cizím buzením permanentním magnetem implementovaný v mikrokontroléru. Ten je propojen s PLC pro které je vytvořen program s regulátorem a vizualizací. Je tak sestavena laboratorní úloha sloužící k procvičení návrhu a nastavení parametrů číslicových regulátorů.

V první kapitole práce je vytvořen přehled základních možností a principů regulace elektrických pohonů. V dalších kapitolách je popsán návrh hardwaru a firmwaru modelu, jehož hlavní částí je mikrokontrolér, ve kterém je implementován matematický model motoru.

Přípravek za chodu umožňuje změnu zatěžovacího momentu působícího na hřídel a tím simulaci změny poruchové veličiny. Dále je možná simulace změny setrvačného momentu motoru, což mění přenos celé soustavy a model je tak možné použít i jako trenažér pro návrh adaptivních či robustních regulátorů.

Součástí práce je také model v programu Matlab/Simulink, pomocí kterého byla ověřena správnost diskretizace a následné implementace matematického modelu motoru do mikrokontroléru.

V poslední kapitole je vytvořen manuál k přípravku a zadání laboratorní úlohy.

SEZNAM POUŽITÉ LITERATURY

[1] JAVŮREK, Jiří. Regulované elektrické pohony. *Automa* [online]. 2004, 03, [cit. 2010-05-18]. Dostupný z WWW: <http://automa.mpresent.cz/index.php?id_document=32220>.

[2] KRŮŽELA, Miroslav. *Regulace otáček elektromotoru* [online]. [s.l.], 2006. 49 s. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně, Fakulta Aplikované Informatiky. Dostupné z WWW: <https://www.stag.utb.cz/apps/stag/dipfile/index.php?download_this_unauthorized=3376>.

[3] Pohony s krokovými motorky. In *Pohony s krokovými motorky : Laboratorní návod*. [s.l.] : [s.n.], [2006], 2006-03-29 [cit. 2010-05-18]. Dostupné z WWW: <http://fei1.vsb.cz/kat453/www453/soubory/texty/ucebni_texty/se/cast_C_el_pohony/se_eph_c1_krokac_02_teorie.pdf>.

[4] SKALICKÝ, Jiří. *Elektrické regulované pohony* [online]. Brno : [s.n.], 2007 [cit. 2010-05-19]. Dostupné z WWW: <https://www.vutbr.cz/www_base/priloha.php?dpid=18964>.

[5] HLAVA, Jaroslav. *Prostředky automatického řízení II* [online]. [s.l.] : [s.n.], 2000 [cit. 2010-05-18]. Regulační vlastnosti elektrických pohonů a výkonových členů, s. . Dostupné z WWW: <http://www.fm.vslib.cz/~krtsub/fm/par/Skripta_PAR.pdf>.

[6] FISCHL, Thomas. *USB programmer for Atmel AVR controllers* [online]. 1998 [cit. 2010-05-18]. Fischl.de. Dostupné z WWW: <http://www.fischl.de/usbasp/>.

[7] ATMEL. Datasheet ATmega8 [online], [cit. 2010-05-19]. Dostupné z URL: <http://www.datasheetarchive.com/pdf-datasheets/Datasheets-6/DSA-108020.pdf>

[8] MICROCHIP. Datasheet, 12-Bit DAC with SPI Interface MCP 4921/4922. [online], [cit. 2010-05-19]. Dostupné z WWW: <http://www.datasheetarchive.com/pdf-datasheets/Datasheets-18/DSA-358015.pdf>

[9] PHILIPS. Datasheet 74HC595; 74HCT595. [online], [cit. 2010-05-19]. Dostupné z WWW: <http://www.datasheetarchive.com/pdf-datasheets/Datasheets-308/11080.pdf>

[10] BRIGHT LED ELECTRONIC CORP. Datasheet k BT-A515RD, Three digit LED displays [online], [cit. 2010-05-19]. Dostupné z WWW: < http://www.datasheetarchive.com/pdf-datasheets/Datasheets-310/104101.pdf >

[11] Texas Instruments. *2A, 28V INPUT, STEP DOWN SWIFT™ DC/DC CONVERTER WITH ECO-MODE* [online]. [s.l.] : [s.n.], 2008 [cit. 2010-05-19]. Dostupné z WWW: <http://focus.ti.com/lit/ds/symlink/tps54231.pdf>.

[12] KORHONEN, Aki. *Metku.net* [online]. 26.01.2009 [cit. 2010-05-19]. How to get started with microcontrollers. Dostupné z WWW: <http://metku.net/index.html?path=articles/microcontroller-part-1/index_eng>.

[13] KARAS, Ondřej. *Programujte.com* [online]. 15.06.2006, 20.04.2008 [cit. 2010-05-19]. AVR - úvod. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2006061202-avr-uvod>

[14] VELEBA, Václav. *Číslicová řídící technika - Počítačová cvičení – kapitola Diskrétní PSD regulatory,* skripta FEKT VUT. Brno 1.10.2005.

[15] PIVOŇKA, P. *Číslicová řídicí technika, 151 s.* Brno, 2003: 2003. s. 1-150. ISBN: AMT101

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

rad/s - radiány za sekundu – jednotka úhlové rychlosti

DPS - deska plošného spoje

U_a - napájecí napětí kotvy

i_a - proud kotvy

Ra - odpor vynutí kotvy

La - indukčnost vynutí kotvy

J - setrvačný moment motoru a všech pohybujících se částí s pojených s hřídelí motoru

m_z - zatěžovací moment vyvolaný zátěží a pasivními odpory motoru

M_z - zatěžovací moment vyvolaný zátěží a pasivními odpory motoru

ω - úhlová rychlost hřídele motoru

Ω - úhlová rychlost hřídele

ξ - konstanta úměrnosti momentu vytvářeného kotvou na proudu kotvy

τ_m - elektromechanická časová konstanta motoru

τ_a - elektromagnetická časová konstanta motoru

ξ - motorová konstanta

U_{out} - výstupní napětí

U_{ref} - referenční napětí

DC - direct current (stejnosměrný proud)

α - řídící úhel

P - výkon

M - moment

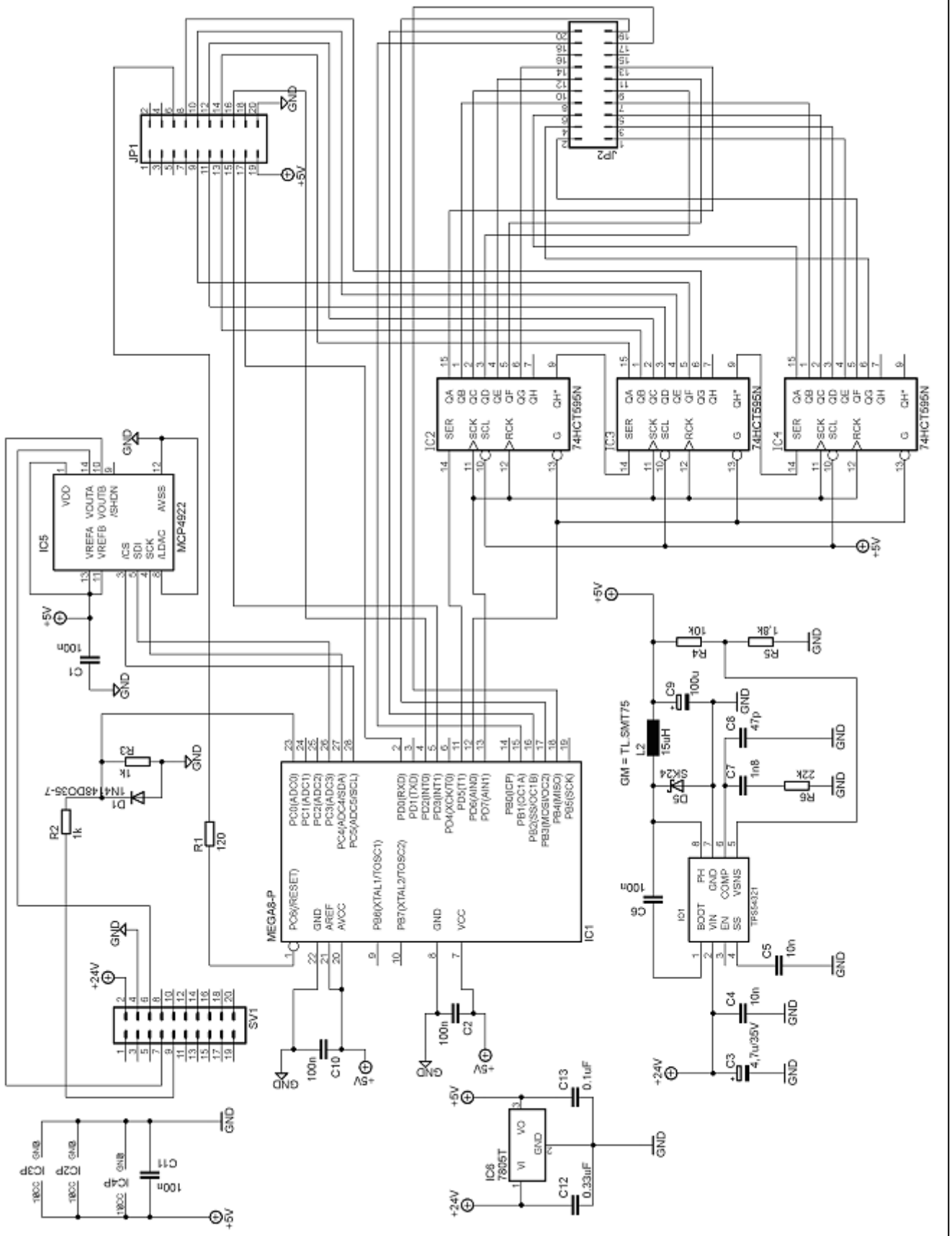
HEX - zkomprimovaný soubor, který je možné nahrát přímo do mirkokontroléru

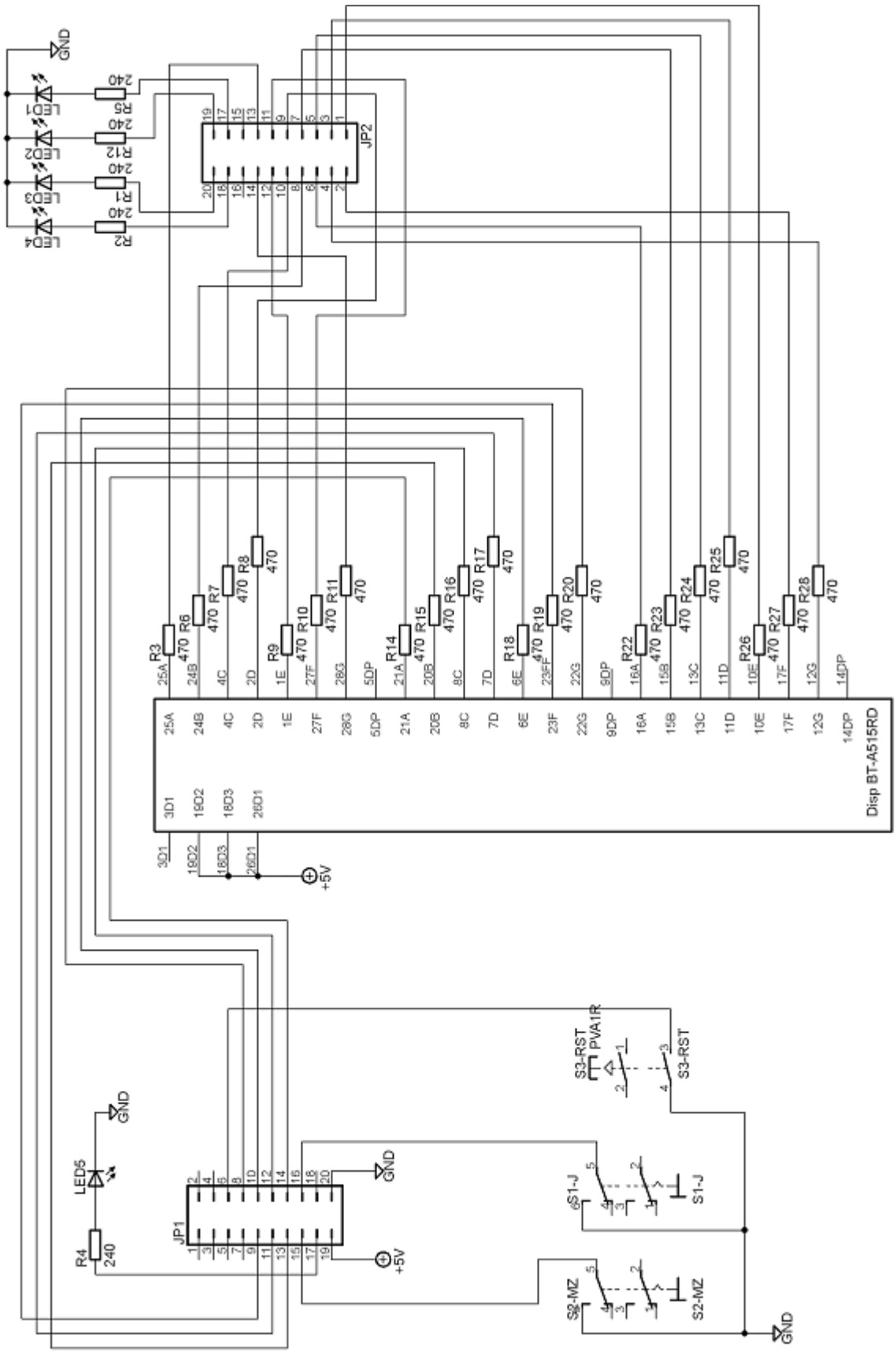
SPI - sériovo-paralelní komunikace

ADC - výsledek A/D převodu

SEZNAM PŘÍLOH

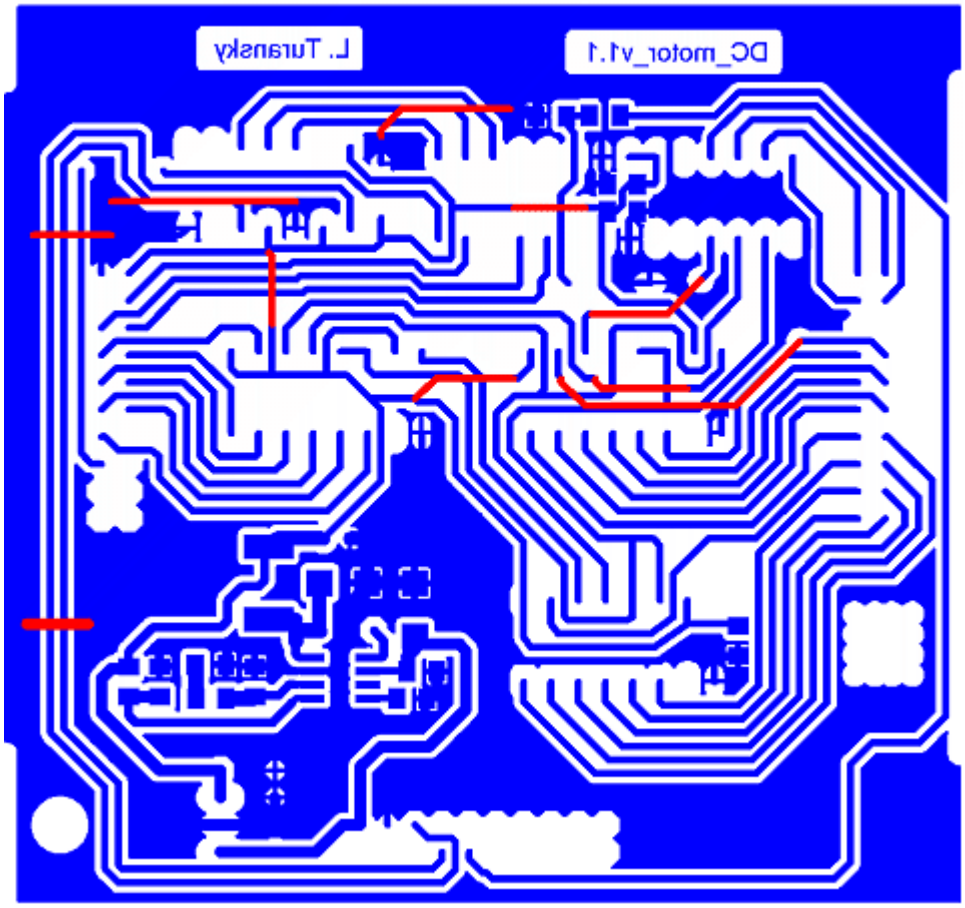
A. PRVNÍ PŘÍLOHA – SCHÉMA MODELU


JP1
JP2
IC2
IC3
IC4
74HCT595N
IC5
MCP4922
IC1
MEGA8-P
IC6
7805T
IC01
TPS54321
SV1
GM = TL_SMT75
L2
15uH
D5
SK24
D1
1N4148DO35-7
R1
120
R2
1k
R3
1k
R4
10k
R5
1,8k
R6
2k2
R7
2k2
C1
100n
C2
100n
C3
4,7u/35V
C4
10n
C5
10n
C6
100n
C7
1n8
C8
47p
C9
100u
C10
100n
C11
100n
C12
0,33uF
C13
0,1uF
+5V
+24V
GND

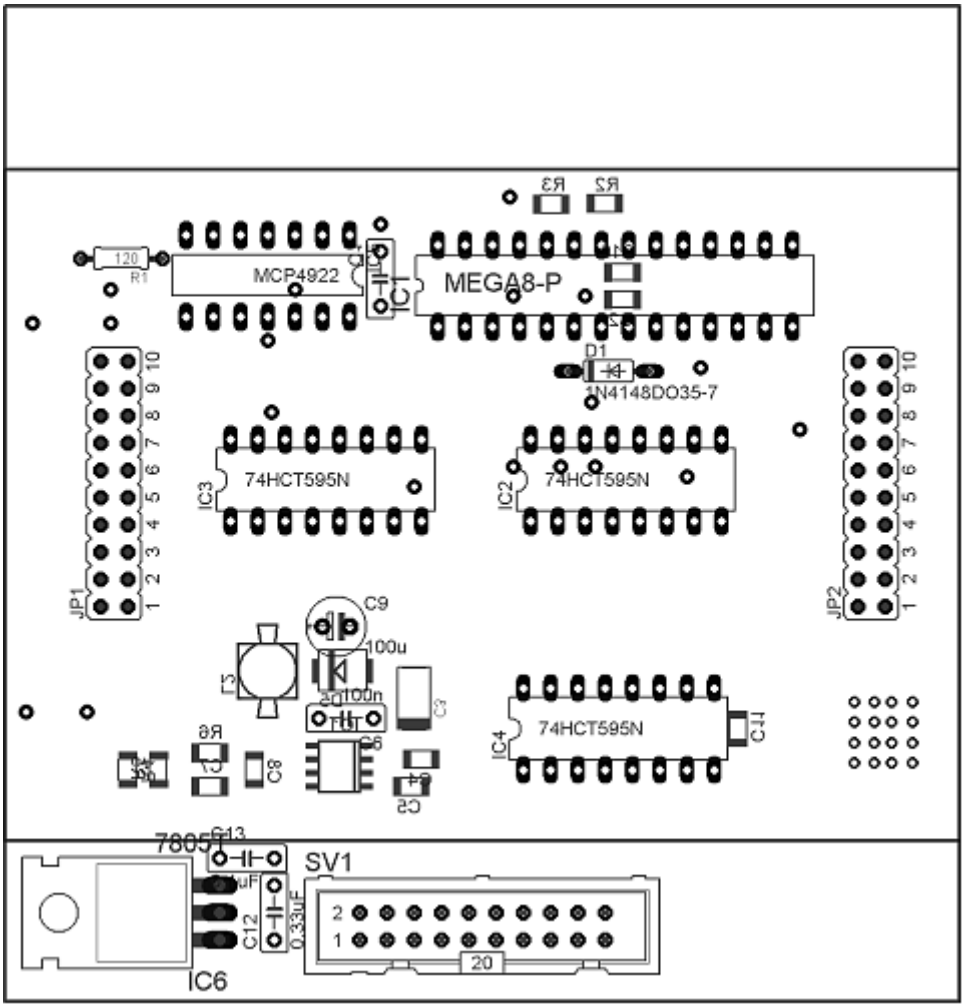

GND
LED1
LED2
LED3
LED4
R5 240
R12 240
R1 240
R2 240
JP2
R3 470
R6 470
R7 470
R8 470
R9 470
R10 470
R11 470
R14 470
R15 470
R16 470
R17 470
R18 470
R19 470
R20 470
R22 470
R23 470
R24 470
R25 470
R26 470
R27 470
R28 470
Disp BT-A515RD
+5V
LED5
R4 240
JP1
S3-RST
PVA1R
S1-J
S2-MZ

B. DRUHÁ PŘÍLOHA - DPS

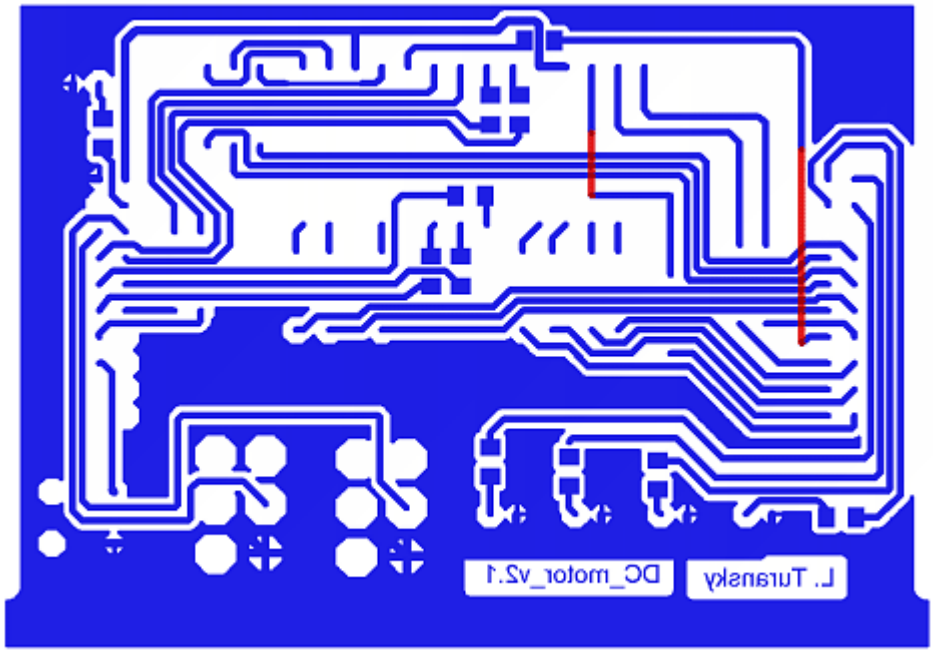
B.1. DPS – spodní deska – strana Top a Bottom



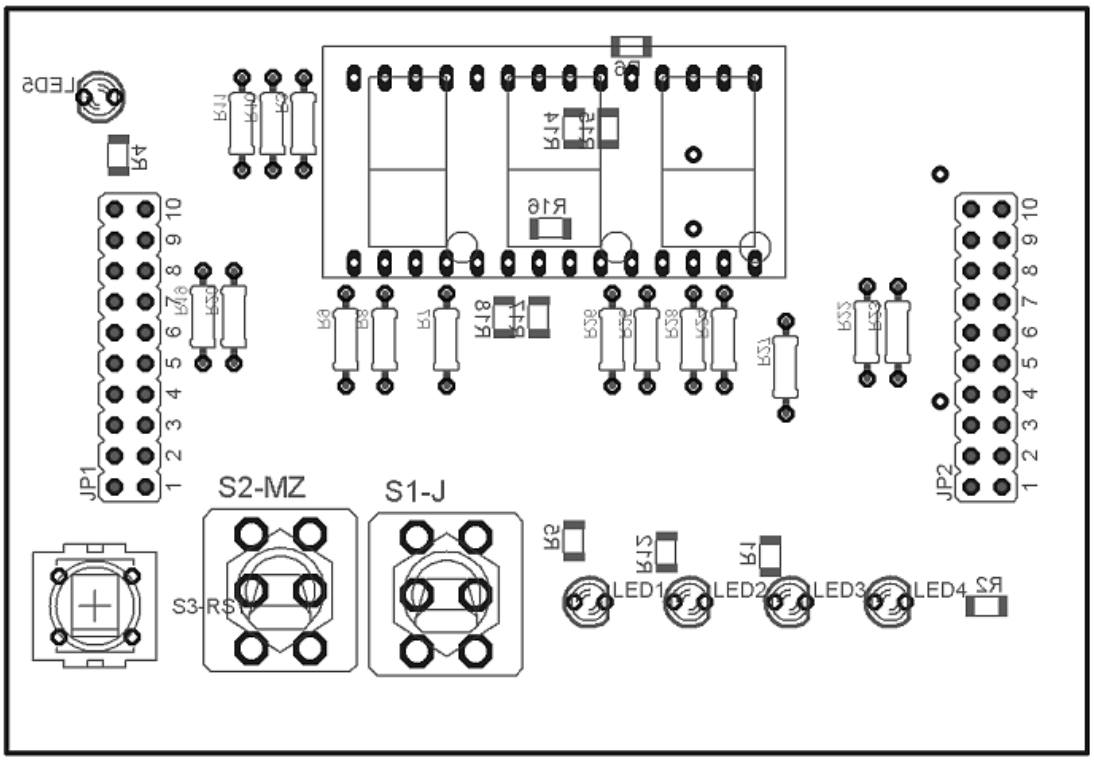
B.2. DPS – spodní deska – osazovací plán



B.3. DPS – vrchní deska – strana Top a Bottom

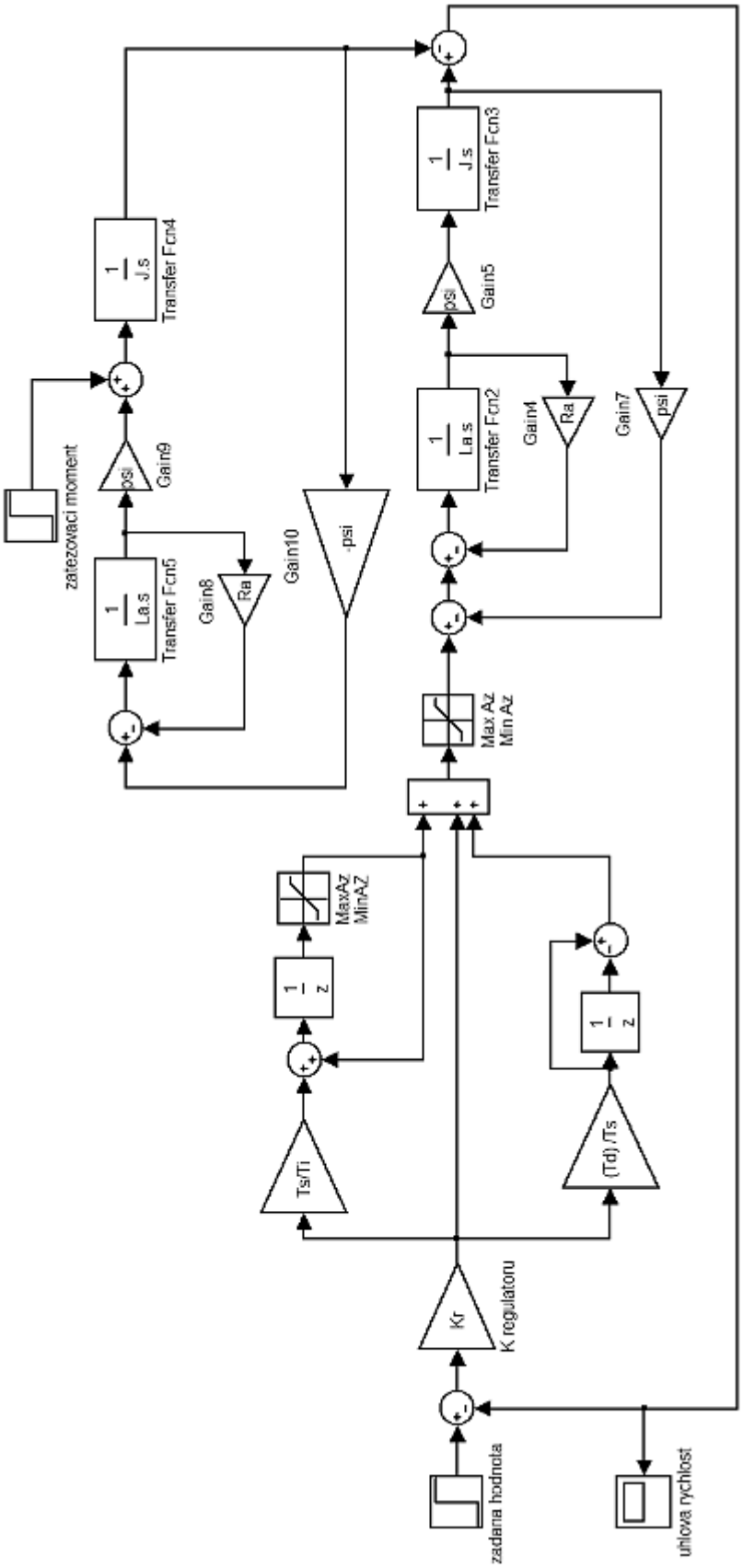


B.4. DPS – vrchní deska – osazovací plán



C. TŘETÍ PŘÍLOHA – SCHÉMA MATLAB

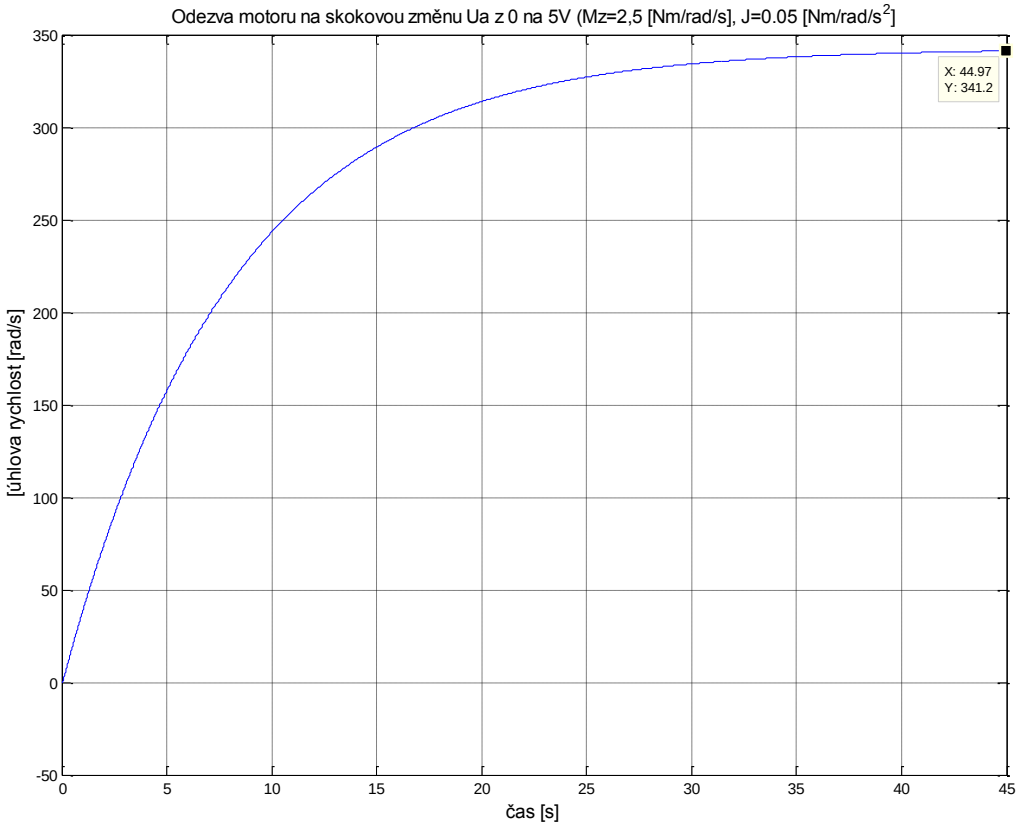
C.1. Blokové schéma regulátoru se soustavou pro model v Matlab Simulink


1/J.s
Transfer Fcn3
psi
Gain5
1/La.s
Transfer Fcn2
Gain4
Ra
Gain7
psi
Max Az
Min Az
MaxAz
MinAz
1/z
Ts/Ti
1/z
(Td) /Ts
Kr
K regulatoru
zadana hodnota
uhlova rychlost
1/J.s
Transfer Fcn4
psi
Gain9
zatezovaci moment
1/La.s
Transfer Fcn5
Gain8
Ra
Gain10
-psi

D. ČTVRTÁ PŘÍLOHA - GRAFY

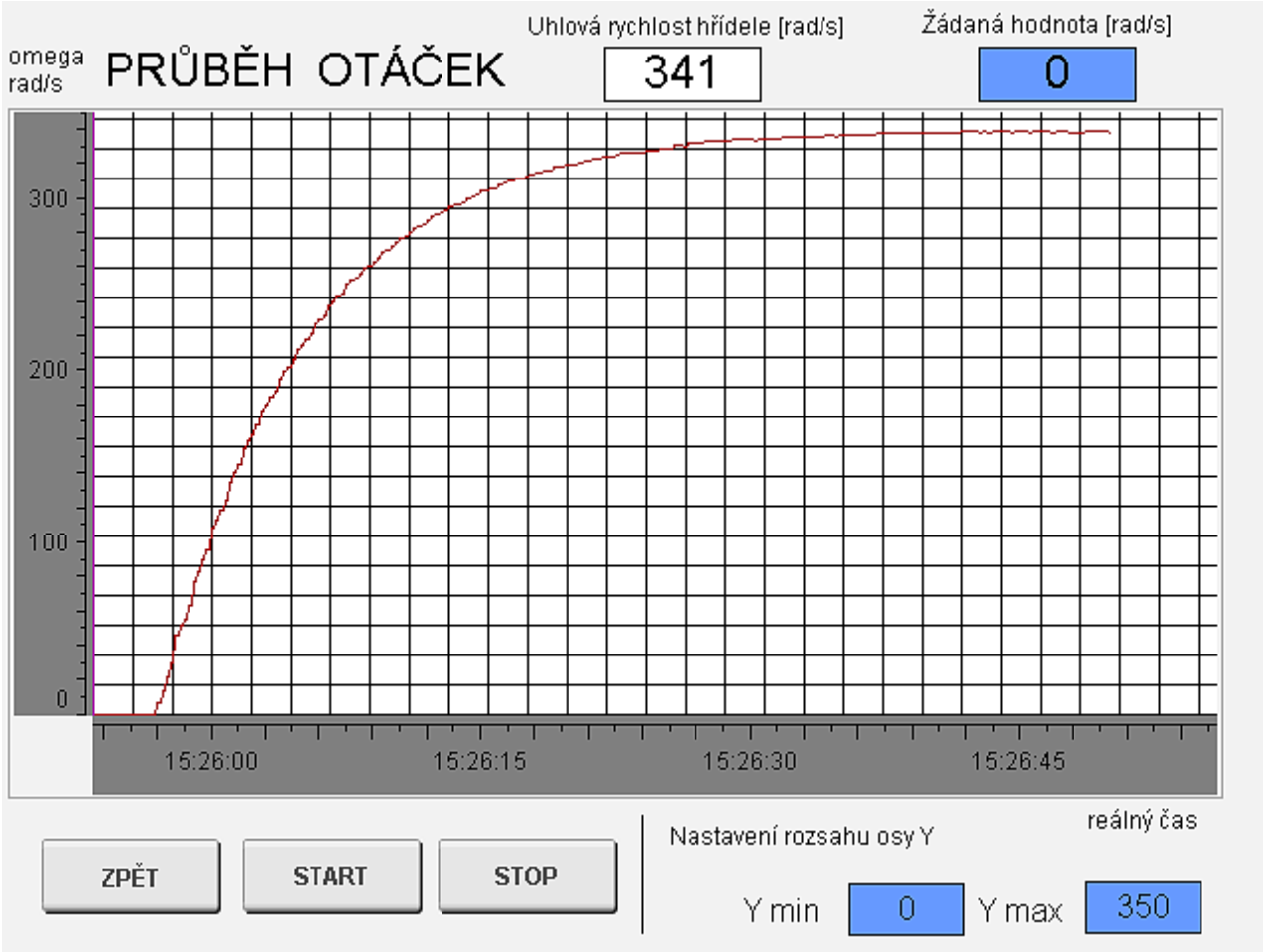
D.1. Odezva soustavy na skok akčního zásahu – simulace v Matlabu

Změna Ua z 0 na 5 V, při Mz=2,5 [Nm/rad/s], J=0.05 [Nm/rad/s^2]



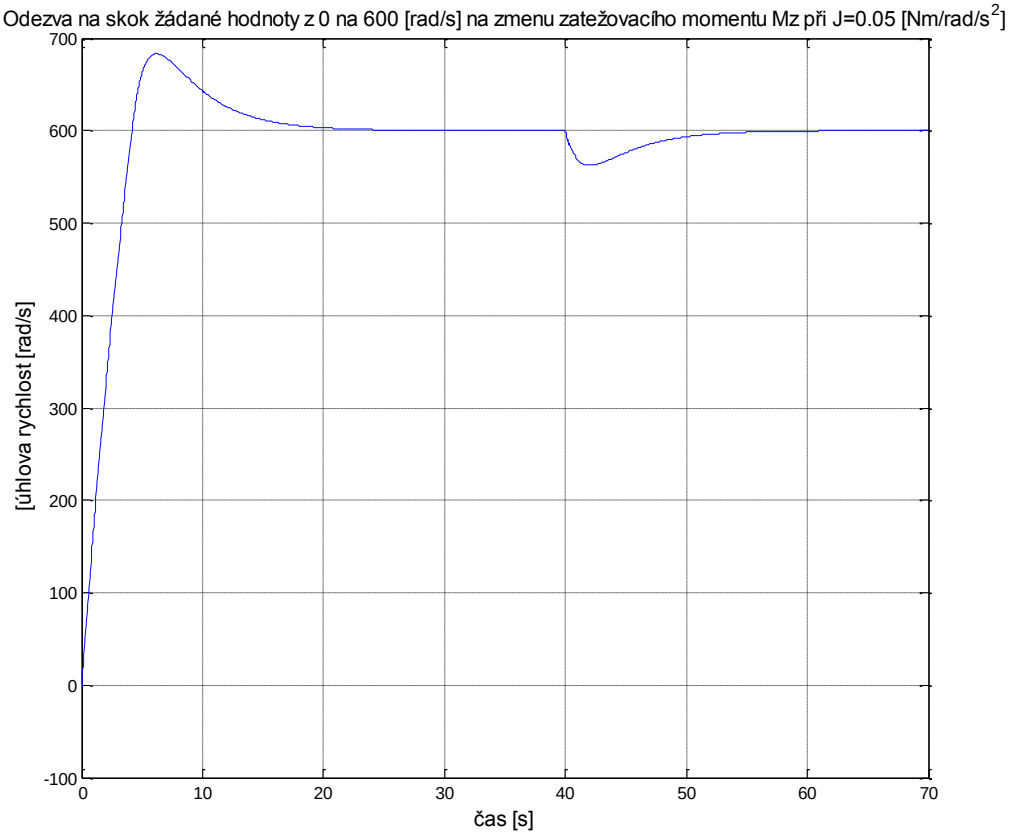
D.2. Odezva soustavy na skok akčního zásahu – Model

Změna Ua z 0 na 5 V, při Mz=2,5 [Nm/rad/s], J=0.05 [Nm/rad/s^2]



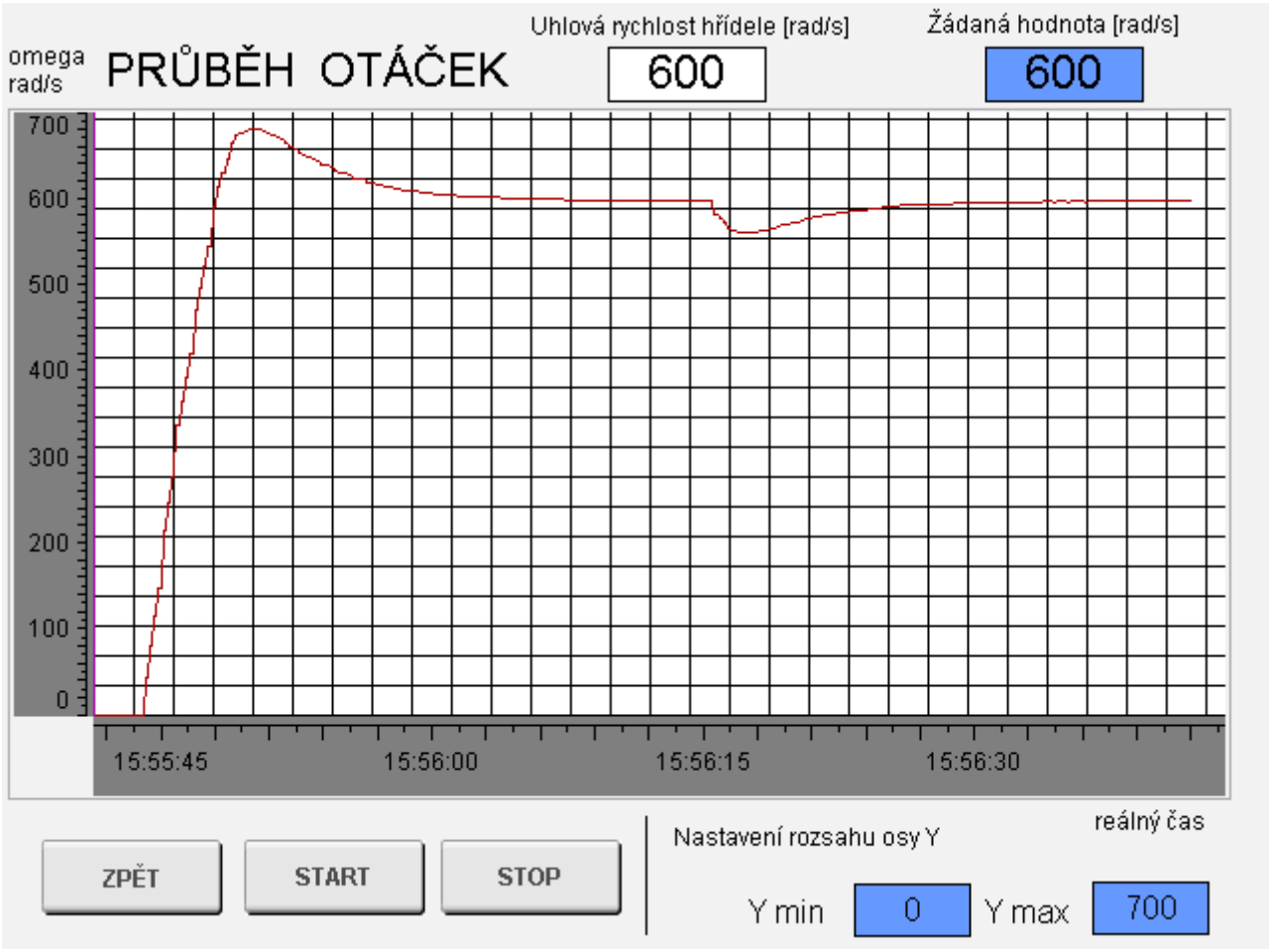
D.3. Odezva regulované soustavy na skok žádané hodnoty W a změnu zatěžovacího momentu Mz - simulace v Matlabu

W se mění z 0 na 600 [rad/s] v čase 0s, Mz z $2.57*10^{-4}$ na 2,5 [Nm/rad/s] v čase 40s

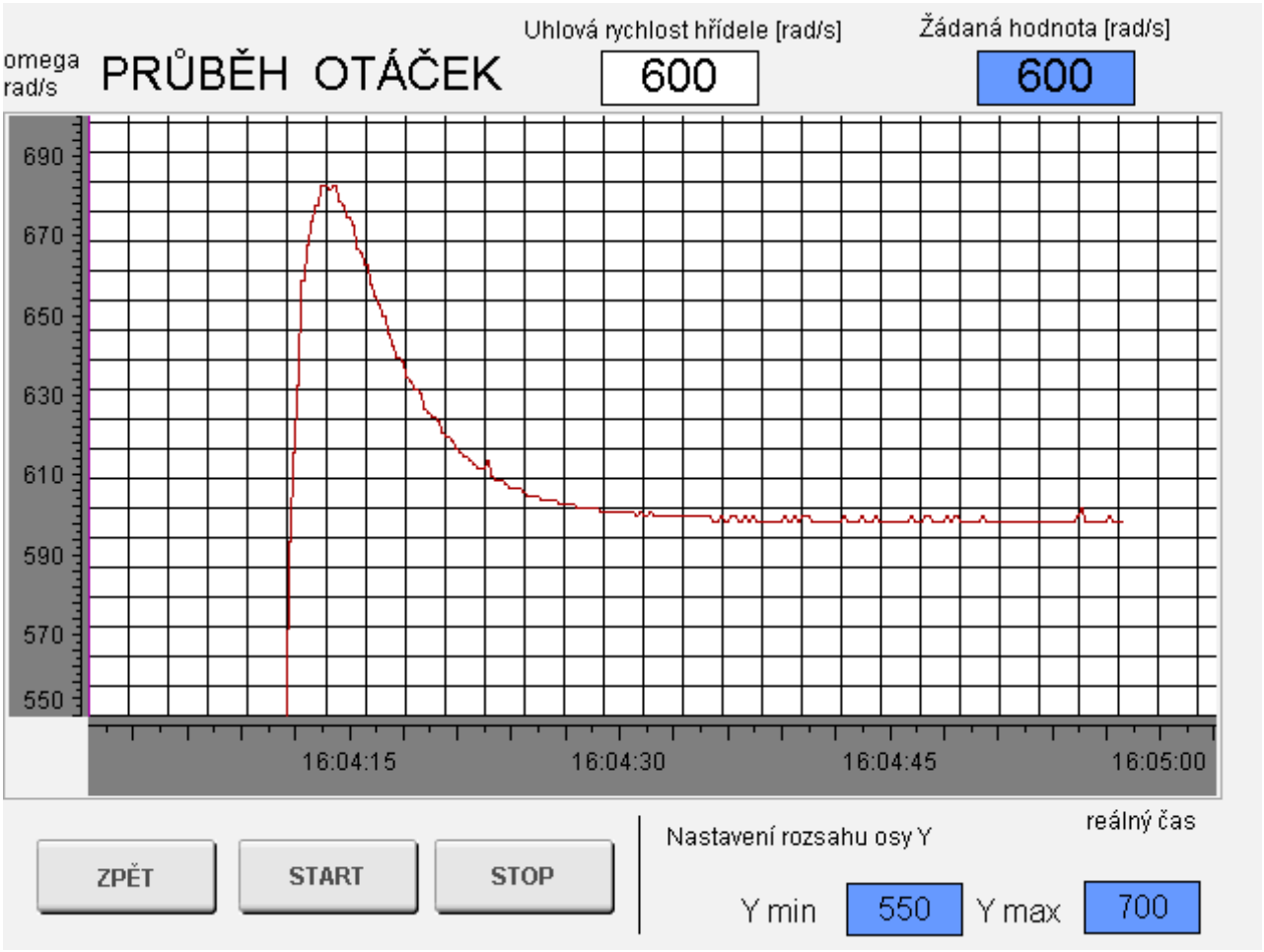


D.4. Odezva regulované soustavy na skok žádané hodnoty W a změnu zatěžovacího momentu Mz – Model

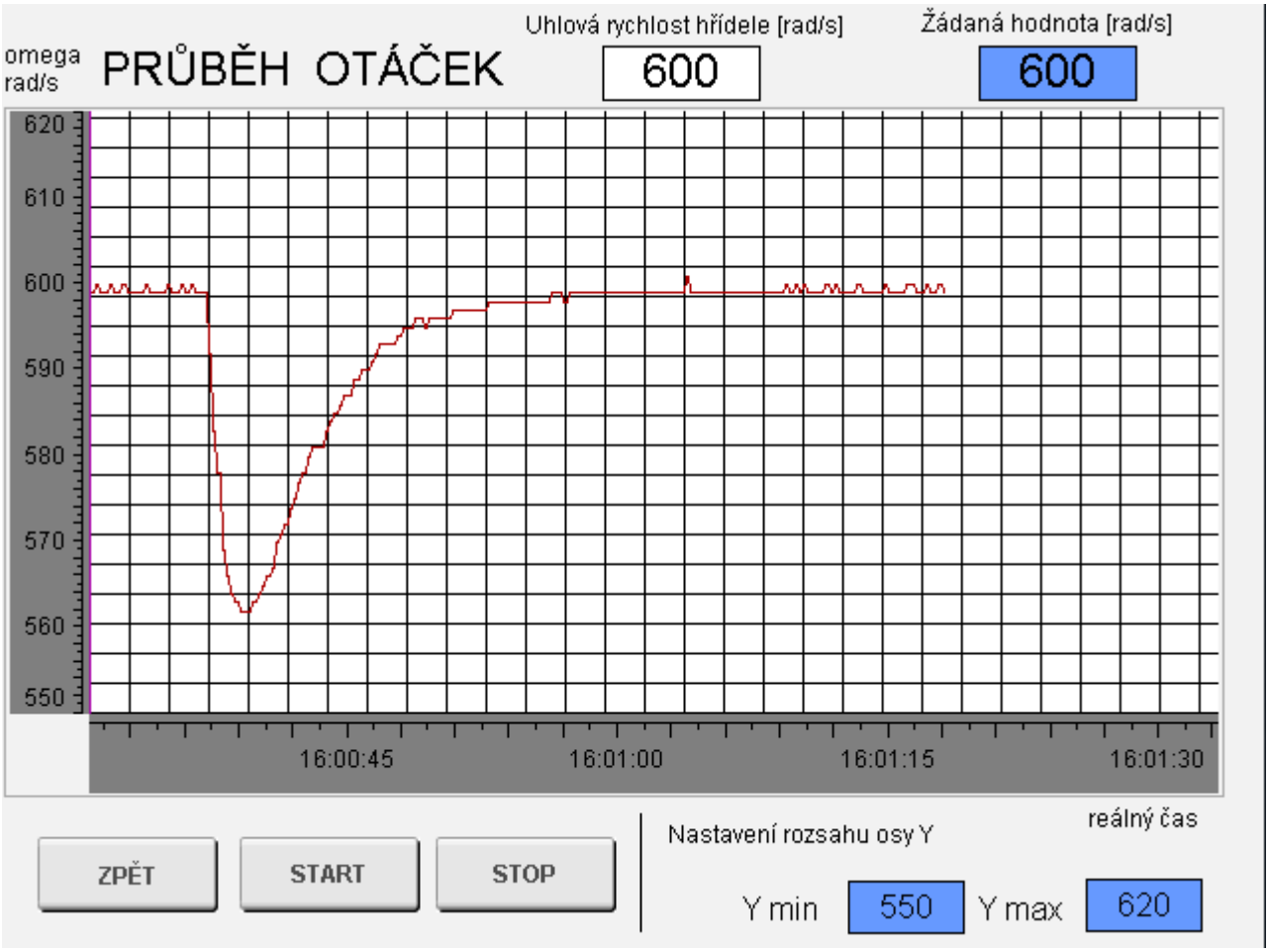
W se mění z 0 na 600 [rad/s] v čase 0s, Mz z $2.57*10^{-4}$ na 2,5 [Nm/rad/s] v čase 40s



D.5. Detail skoku žádané hodnoty z D.4



D.6. Detail skoku zatěžovacího momentu z D.4

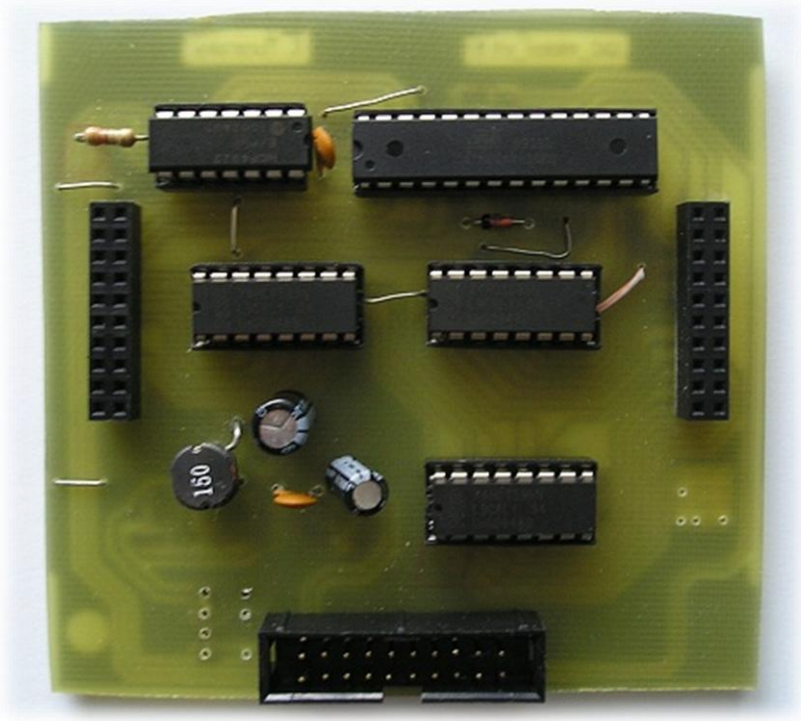


E. PÁTÁ PŘÍLOHA - FOTOGRAFIE MODELU

E.1. Kompletní model

E.2. Odkrytovaný model

E.3. Spodní deska modelu



F. ZDROJOVÝ KÓD PRO ATMEGA8

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
#include <compat/deprecated.h>
#define D1 4
#define D2 3
#define D3 2
#define D4 1
#define D_alarm 0
#define SS 2
// ------- spi for D/A
#define SCK 4 // clk spi
#define CS 5        // /CS spi
#define SDI 3       // MOSI spi
// ------- spi handling display data
#define SCK_DISP 7
#define DS_DISP 5
#define OE_DISP 6
#define MR_DISP 0
//----- Mz signalizations diods
#define RED_D 4
#define YELOW_D 3
#define RANDOM_D 1
#define GREEN_D 2
//--------------------
#define prepinac1 3
#define prepinac2 2
char buf[3];
int segment = 0,uak_pom = 0, Mz_zpomalovac = 0;
int slow_down_disp = 0, wait_alarm = 0, alarm = 0, broken_motor = 0;
```

```
unsigned short int u_out;
int rpm = 0,rpm_display = 0,x=0, J = 0;
// koeficients for diferential equations
double a1= 0.002100680888099, b1= 0.001755287488907, c1=1.582209849170108;
double d1=0.582726548932627, e1=0.199980420559562, f1=0.116530358669116;// J = 0.05
double a2=0.001500509346833, b2=0.001253797423895, c2=1.582357471825350;
double d2=0.582726548932627,e2=0.142847153255547,f2=0.083239021650231; // J = 0.07
double rpm_float = 0;
double omkz = 0, omkz1 = 0, omkz2 = 0, mzk1 = 0, mzk2 = 0, mzk = 0;
double omku = 0, omku1 = 0, omku2 = 0, uak1 = 0, uak2 = 0, uak =0;
```

//----------Timer1 Ineterupt----------

```
SIGNAL(SIG_OUTPUT_COMPARE1A)          // handling omega
{
// ------------------------select of Mz level
if(bit_is_clear(PIND, prepinac1))    // Mz high
        {
                mzk = 2.5;
                PORTB |= _BV(D1);
                 PORTB &= ~(_BV(D2));}
else                                          //Mz low
        {
                mzk = 0.000275;
                PORTB |= _BV(D2);
                PORTB &= ~(_BV(D1));
        }


//----------------------------- analog to digital conversion
        ADCSRA |= _BV(ADSC); // start
        while  ((bit_is_clear(ADCSRA, ADIF)));

        unsigned char adlow,adhigh;
        adlow=inp(ADCL); // read low first, two lines. Do not combine
                         //   the two lines into one C statement
        adhigh=inp(ADCH);
        uak_pom =(adhigh<<8)|(adlow & 0xFF); // result of the A/D conversion
                                             //1024 is equal to 5V
        uak = uak_pom;
        uak = uak*0.20352;     // conversion to make max. value from 1024 to 200
        ADCSRA |= _BV(ADIF); // clear A/D interupt flag bit

// ------------------------select of J level
if(bit_is_clear(PIND, prepinac2) && broken_motor == 0)        // J = 0.07
        {
                PORTB |= _BV(D3);
```

```
                PORTB &= ~(_BV(D4));
        //---------------------- actualization of model states ----------------
                omku = a2*uak1 + b2*uak2 + c2*omku1 - d2*omku2;
                omkz = e2*mzk1-f2*mzk2+c2*omkz1-d2*omkz2;
                omku2 = omku1;
                omku1 = omku;
                omkz2 = omkz1;
                omkz1 = omkz;
                uak2 = uak1;
                uak1 = uak;
                mzk2 = mzk1;
                mzk1 = mzk;
                rpm_float = omku-omkz;
                if (rpm_float <0)
                        {rpm_float =0; omku=0; omkz = 0;}
                rpm = (int)rpm_float;
        }

        //---------------------------------------------
else if (broken_motor == 0)                                     //J = 0.05
        {
                PORTB |= _BV(D4);
                PORTB &= ~(_BV(D3));

                //---------------------- actualization of model states ----------------
                omku = a1*uak1 + b1*uak2 + c1*omku1 - d1*omku2;
                omkz = e1*mzk1-f1*mzk2+c1*omkz1-d1*omkz2;
                omku2 = omku1;
                omku1 = omku;
                omkz2 = omkz1;
                omkz1 = omkz;
                uak2 = uak1;
                uak1 = uak;
                mzk2 = mzk1;
                mzk1 = mzk;
                rpm_float = omku-omkz;
                if (rpm_float <0)
                        {rpm_float =0; omku=0; omkz = 0;}
                rpm = (int)rpm_float;
                }

//------------- watching rpm > 800 limit and conrol alarm LED -------------------
        if (rpm>800)
        {
                PORTD |= _BV(D_alarm);
                alarm = 1;
                wait_alarm = 0;
        }
        if (alarm == 1)
                wait_alarm ++;
        if (wait_alarm == 300)
        {
```

```
                alarm = 0;
                wait_alarm = 0;
                PORTD &= ~(_BV(D_alarm));
        }

        // watching rpm > 999 limit and simulation of broken motor
        if (rpm > 999)
                {
                        rpm = 0;
                        broken_motor = 1;
                }



//------------------------- set value to D/A converter
        int i = 0;
        u_out = (unsigned short int)(rpm*4);

        // chanel A --------------------
                u_out |= 0b0111000000000000;
                PORTC &= ~(_BV(CS)); // /CS to low
                for (i = 0; i < 16 ; i++)
                        {
                                if( (u_out & (1<<(15-i)))  && (1<<(15-i)) )
                                {
                                        PORTC |= _BV(SDI);
                                }
                                else
                                {
                                        PORTC &= ~(_BV(SDI));
                                }
                                PORTC |= _BV(SCK);
                                PORTC &= ~(_BV(SCK));
                                 }
                 PORTC |= _BV(CS); // /CS high

                // chanel B ----------------------
                u_out |= 0b1111000000000000;
                PORTC &= ~(_BV(CS)); // /CS to low
                for (i = 0; i < 16 ; i++)
                        {
                                if( (u_out & (1<<(15-i)))  && (1<<(15-i)) )
                                {
                                        PORTC |= _BV(SDI);
                                }
                                else
                                {
                                        PORTC &= ~(_BV(SDI));
                                }
                                PORTC |= _BV(SCK);
                                PORTC &= ~(_BV(SCK));
                        }
```

```
                PORTC |= _BV(CS); // /CS high

//------------ actualization data on display every 100 ms
        if (slow_down_disp == 1)
        {
                ovl_display();
                slow_down_disp = 0;
        }
        slow_down_disp++;
}
//------- end of Timer1 interupt ----------
```

```
void ovl_display(void)          // function for handling display
{
        int i = 0,j=0;
        int buff_disp;

        rpm_display = rpm;
        itoa(rpm_display, buf, 10);// convert rotation to string [buf]

        if (rpm_display <10)
                {buf [2] = buf [0];  buf[1] = '0';        buf[0] = '0'; }
                // for example change from 1 to 001
        if (rpm_display <100 && rpm_display > 9)
                {buf [2] = buf [1]; buf[1] = buf[0]; buf[0] = '0';}
                 // for example change from 10 to 010
        if (broken_motor == 1)
                {buf [2] = 58; buf[1] = 58; buf[0] = 58;}

        PORTB &= ~(_BV(MR_DISP));       // reset pulse
        PORTB |= _BV(MR_DISP);
        PORTD |= _BV(OE_DISP);          // output disable

        PORTD |= _BV(DS_DISP);
        PORTD |= _BV(SCK_DISP);         // additional clock pulse to ensure shift one more bit
        PORTD &= ~(_BV(SCK_DISP));      // in register

        for (j=0;j<3;j++)
        {
                switch (buf[2-j]-48)  //set the current segments of current number on outputs pins
                {       case 0 :        buff_disp = 0b01000000;         break;
                        case 1 :        buff_disp = 0b01111001;  break;
                        case 2 :        buff_disp = 0b00100100;         break;
                        case 3 :        buff_disp = 0b00110000;         break;
                        case 4 :        buff_disp = 0b00011001;  break;
                        case 5 :        buff_disp = 0b00010010;         break;
                        case 6 :        buff_disp = 0b00000010;  break;
                        case 7 :        buff_disp = 0b01111000;         break;
                        case 8 :        buff_disp = 0b00000000;         break;
                        case 9 :        buff_disp = 0b00010000;  break;
                        case 10:        buff_disp = 0b00111111;         break;
                }

                for (i = 0; i < 8; i++)
                 {
                        if( (buff_disp & (1<<(7-i)))  && (1<<(7-i)) )
                                PORTD |= _BV(DS_DISP);
                        else
                                PORTD &= ~(_BV(DS_DISP));

                        PORTD |= _BV(SCK_DISP);         //      clock pulse
                        PORTD &= ~(_BV(SCK_DISP));
                }
        }
```

```
        PORTD |= _BV(DS_DISP);
        PORTD |= _BV(SCK_DISP);          // additional clock pulse to ensure shift one more bit
        PORTD &= ~(_BV(SCK_DISP));       // in register
        PORTD &= ~(_BV(OE_DISP));        // output enable - data will appear on the display

}
//--------- end of ovl_display() fiction -------------------
```

int initialization(void) // inicialization function

```
{
        DDRB = 0xFF;      // whole PORTB as output
        DDRD = 0xF3;
        PORTD = 0b0001100;

        // settings of Timer1
        TIMSK = _BV(OCIE1A) ;            // Timner 1 Output Compare A Match
        TCCR1B = _BV(CS11) | _BV(CS10) |// 64 prescale for Timer1
         _BV(WGM12);                     // CTC mode, TOP = OCR1A timer1
        OCR1A =1250;                     // count up to TOP (equal to 10ms)

        // settings of AD conversion
        ADMUX = 0;
        ADCSRA |= _BV(ADEN) | _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0)| _BV(ADIE);

        // settings for SPI D/A
        DDRC |= 0b111000;        // PB 5-3 as outputs
        PORTC = _BV(CS);         // /CS to high state

        // inicialization of equition variables
        omkz = 0;
        omkz1 = 0;
        omkz2 = 0;
        mzk1 = 0;
        mzk2 = 0;
        mzk = 0;
        omku = 0;
        omku1 = 0;
        omku2 = 0;
        uak1 = 0;
        uak2 = 0;
        uak =0;
        broken_motor = 0;
        alarm = 0;
        sei();              // global interupts enabled
        return 1;
}

        //--------- end of program ------------------
```

G. ALGORITMUS REGULÁTORU PRO PLC

```
#include <bur/plctypes.h>

#ifdef _DEFAULT_INCLUDES
#include <AsDefault.h>
#endif

void _CYCLIC ZakladniCyclic( void )
{
	#define MaxAZ 10
	#define MinAZ 0
	reg_velicina=(REAL)(iAm/16.671875);
	E = W - reg_velicina;
	E_int = E;
	uhlova_rychlost = reg_velicina;

	U = K*E+sum+(K*E*(Td/Ts)-ds);
	ds = E*K*(Td/Ts);
	sum=K*E*(Ts/Ti) + sum;

	if (sum>MaxAZ) sum = MaxAZ;
	if (sum < MinAZ) sum = MinAZ;

	if (U > 10) U = 10;
	if (U < 0) U = 0;
	oAm = (INT)(U*0x7fff/10);
	vstA = iAm;
	vstB = iAm2;
}
```

H. OBSAH PŘILOŽENÉHO DVD

Součástí této práce je přiložené DVD s následujícími adresáři:

- **Text** - Elektronická verze dokumentu BP ve formátu PDF
- **PLC** - Program pro PLC vytvořený v programu Automation Studio verze 3.071.10
- **Simulace** – Model motoru pro Matlab/Simulink
- **Firmware ATmega8** – Projekt vytvořený v programu AVR Studio v4.16 a zkompilovaný HEX soubor k nahrání do ATmega8
- **DPS** – návrh desky plošných spojů v programu Eagle v 5.7.0