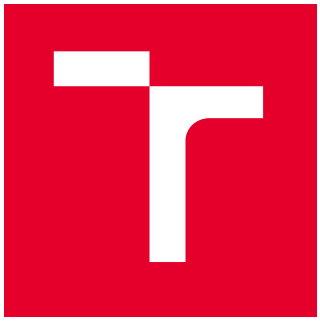


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍZENÍ REÁLNÉHO SYSTÉMU POMOCÍ PLC S VYUŽITÍM AUTOMATICKY GENEROVANÉHO KÓDU

A SYSTEM CONTROL USING AUTOMATICALLY GENERATED CODE FOR PLC

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Stanislav Pacal**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Miroslav Jirgl, Ph.D.**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Stanislav Pacal ***ID:*** 195405

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Řízení reálného systému pomocí PLC s využitím automaticky generovaného kódu

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je prozkoumat možnosti automatického generování kódu pro PLC z prostředí MATLAB/Simulink a následně aplikovat příslušné nástroje na úlohu řešící regulaci jednoduchého reálného systému. Součástí práce by měl být i návrh a realizace nové verze platformy vybavené analogovými obvody s nastavitelnými parametry simulující různé dynamiky připojitelné na vstupy/výstupy PLC.

1. Analyzujte koncepci stávající platformy a navrhněte vhodné úpravy. Ty následně realizujte (upravte schéma DPS, realizujte DPS, osaďte a oživte).
2. Připojte realizovanou platformu k PLC a otestujte její funkčnost a vlastnosti.
3. Seznamte se s nástroji MATLAB PLC Coder a System Identification Toolbox, popište je a prozkoumejte jejich možnosti.
4. Navrhněte testovací úlohu s PID regulátorem a realizujte ji v prostředí MATLAB/Simulink.
5. Vygenerujte kód pro PLC a otestujte požadovanou funkčnost s využitím realizované platformy.
6. Porovnejte dosažené výsledky s výsledky získanými využitím vestavných funkcí PLC.
7. Zhodnoťte vlastnosti vygenerovaného kódu a omezení celé metody.

DOPORUČENÁ LITERATURA:

Simulink PLC Coder [online]. The MathWorks, © 1994-2021 [cit. 2021-9-7]. Dostupné z: https://www.mathworks.com/help/plccoder/

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 18.5.2022

Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem této práce je seznámení se s možnostmi automaticky generovaného kódu pro PLC z prostředí *MATLAB/Simulink*, návrh nové verze laboratorní platformy pro simulaci jednoduchých technologických procesů a její realizace. Součástí práce je i návrh demonstrační úlohy. Výsledkem práce je platforma připojitelná na vstupy/výstupy PLC vybavená analogovými obvody s nastavitelnými parametry simulující různé dynamiky a dále demonstrační úloha znázorňující možnosti automaticky generovaného kódu.

KLÍČOVÁ SLOVA

Systém, model, řízení, PLC, Simulink PLC Coder

ABSTRACT

The subject of this thesis is to get acquainted with the possibilities of automatically generated code for PLC from the environment *MATLAB/Simulink*, design of a new version of laboratory platform for simulation of simple technological processes and its implementation. Part of the work is also the design of a demonstration task. The result of the work is a platform connectable to PLC inputs / outputs equipped with analog circuits with adjustable parameters simulating various dynamics and a demonstration task showing the possibilities of automatically generated code.

KEYWORDS

System, model, control, PLC, Simulink PLC Coder

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

PACAL, Stanislav. *Řízení reálného systému pomocí PLC s Využitím automaticky generovaného kódu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 88 s. Diplomová práce. Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Stanislav Pacal
VUT ID autora:	195405
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Řízení reálného systému pomocí PLC s Využitím automaticky generovaného kódu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Miroslavu Jirglovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cílem této práce je navrhnout novou verzi laboratorní platformy vybavenou analogovými obvody s nastavitelnými parametry simulující různé dynamiky nejčastěji se v praxi vyskytujících systémů připojitelnou na vstupy/výstupy PLC. Dalším cílem je prozkoumat možnosti automatického generování kódu pro PLC z prostředí *MATLAB/Simulink*, seznámit se s možnostmi identifikace pomocí nástroje *System Identification Toolbox* a následně aplikovat tyto nástroje při řešení demonstrační úlohy.

První část této diplomové práce obsahuje stručný teoretický úvod do oblasti modelování a identifikace a seznámení se s možnostmi identifikace pomocí nástroje *System Identification Toolbox*. Dále první kapitola rozebírá možnosti generování kódu strukturovaného textu pomocí nástroje *Simulink PLC Coder* pro simulinkové modely a stavové diagramy. V této části první kapitoly je popsána příprava simulinkového model pro generování kódu, implementace kódu do použitého IDE *TIA Portal*, možnost optimalizace kódu a možnosti dalších konfigurovatelných parametrů pro generování kódu.

Další částí je analýza předešlé verze platformy a návrh verze nové. V této kapitole je uveden rozbor první verze platformy a zhodnocení jejích nedostatků. Dále také návrh řešení nedostatků, popis jednotlivých bloků, ze kterých je platforma sestavena, doplnění o digitální část a návrh obvodového řešení. V závěru této části je uveden detailní popis způsobu ovládání této platformy a výsledky měření na realizované platformě v porovnání s teoretickými předpoklady.

Poslední kapitola této práce se zabývá návrhem demonstrační úlohy demonstrující možnosti generovaného kódu a funkčnosti realizované platformy. V této kapitole je uveden návrh demonstrační úlohy s PID regulátorem v prostředí *MATLAB/-Simulink*, pro který je automaticky generován kód pomocí nástroje *Simulink PLC Coder*. Dále jsou zde porovnány dosažené výsledky s výsledky získanými využitím vestavných funkcí PLC.

V závěru poslední kapitoly je uvedeno zhodnocení generování kódu pomocí nástroje *Simulink PLC Coder*.

1 Základní pojmy v oblasti modelování a identifikace systémů

Při identifikaci a modelování systémů jsou hlavními základními pojmy zkoumaný systém a jeho model. Znalost zkoumaného systému je v oblasti automatického řízení stěžejní. Bez znalosti vlastností zkoumaného systému by nebylo možné exaktně analyzovat chování daného systému. [1]

Cílem identifikace a modelování je získat takový model zkoumaného systému, který bude co nejpřesněji popisovat vnitřní funkční závislosti mezi vstupními a výstupními veličinami daného zkoumaného systému. [2]

1.1 Systém

Systém lze definovat jako objekt nebo skupinu objektů, jejichž proměnné veličiny různého typu navzájem reagují a produkují pozorovatelné signály. Takové pozorovatelné signály, které jsou pro nás zajímavé nazýváme výstupy. Takovýto systém může interagovat se svým okolím, tudíž mohou na něj působit vnější vlivy. Vnější vlivy, které může pozorovatel ovlivnit, jsou nazývány vstupy, naopak ty, které pozorovatel ovlivnit nemůže, jsou nazývány poruchy. Poruchy se dále dělí na ty, které můžeme měřit přímo a ty, které jsou pozorovatelné pouze skrze ovlivněný výstupní signál. [3]

Na obrázku 1.1 a) je zobrazeno schéma systému, kde u je ovlivnitelný vstupní signál, w je přímo měřitelná porucha, v je neměřitelná porucha a y je výstupní signál.

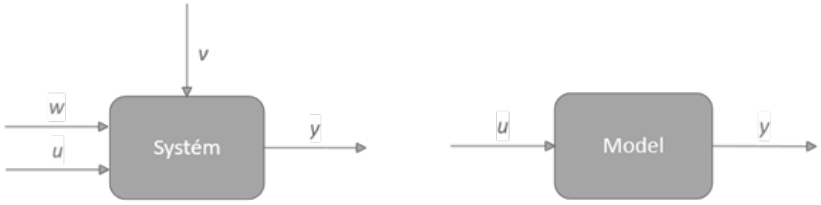
1.2 Model

Pro bližší pochopení složitých systémů ve vědě a technice využíváme jejich modely. Model je umělý systém, který zjednodušeně popisuje reálný zkoumaný systém a zachycuje ty stránky daného systému, které jsou pro nás podstatné. Naopak nepodstatné stránky zanedbává. [2]

Proces tvorby modelu se nazývá modelování. Reálný systém je obklopen prostředím, se kterým neustále interaguje. Transformací systému na model dochází k aproximaci. Příliš komplexní popis systému, který by pouze zapříčinil složitost modelu a těžkopádnost analýzy, není při modelování žádoucí. [4]

Na obrázku 1.1 b) je znázorněno schéma modelu se vstupním signálem u a výstupním signálem y.

Úkolem modelu je v podstatě co nejvěrohodněji reprodukovat čí predikovat chování zkoumaného systému. [3]



(a) Systém (b) Model

Obr. 1.1: Schéma systému a modelu

1.3 Identifikace

Identifikace je proces vyšetřující dynamické vlastnosti reálného systému a určování matematického popisu (modelu). Při této činnosti je určena struktura a parametry modelu. [5]

Model musí být postaven na základě pozorovaných dat. K získání matematického modelu, který je využíván v automatizační technice vedou v zásadě tři cesty, a to je analytický způsob identifikace *(white box)*, experimentální způsob identifikace *(black box)* nebo kombinace těchto dvou přístupů *(grey box)*. [6]

1.3.1 Analytický způsob identifikace

Při analytickém způsobu identifikace je zkoumaný systém rozdělen, obrazně řečeno, na subsystémy, jejichž vlastnosti jsou již dobře známy na základě empirických znalostí. Matematickým spojením těchto subsystémů je získán matematický model. Tento způsob získání modelu je známý jako modelování a nemusí zahrnovat žádné experimentování s reálným zkoumaným systémem. [6]

Základní technikou modelování je rozdělení procesu do blokového diagramu s bloky obsahujícími základní prvky. Rekonstrukce systému z takovýchto bloků je většinou prováděna počítačem, výsledkem je tedy spíše softwarový model než matematický model. Takto získaný model popisuje vnitřní chování daného systému ve větším rozsahu a jeho parametry mají fyzikální smysl. Výsledkem modelování je schéma zkoumaného systému, které je v závislosti na své složitosti popsáno vhodným matematickým aparátem. [6] [5]

1.3.2 Experimentální způsob identifikace

Druhou možností identifikace je experimentální způsob identifikace, kdy je matematický popis získán vyhodnocením odezvy systému na známý vstupní signál. [6]

Při experimentálním způsobu identifikace představuje zkoumaný systém černou schránku, jedná se tedy o vnější popis systému, kdy parametry takto získaného

modelu často nemají fyzikální smysl. Tato cesta získání modelu se nazývá systémová identifikace. [6]

Shrnutí vlastností analytické a experimentální identifikace je popsáno v tabulce 1.1.

1.4 Metody identifikace

Metody identifikace lze dělit podle několika kritérií. Podle použitého vstupního signálu, podle matematického modelu nebo podle vyhodnocení kvality modelu. Dále lze metody dělit na základě způsobu měření, vyhodnocení a zpracování dat a použitých algoritmů. [4]

1.4.1 Typy matematických modelů

Matematické modely je možné rozdělit do dvou následujících typů

- Parametrické modely
- Neparametrické modely

Parametrické modely mají přesně definovanou strukturu a mohou být popsány pomocí diferenciálních či diferenčních rovnic nebo pomocí přenosů. Tyto rovnice explicitně obsahují parametry modelu. [4]

Neparametrické modely jsou nejčastěji znázorněny graficky nebo jako tabulka hodnot vyjadřující funkční závislost mezi vstupním signálem a odpovídajícím výstupním signálem. Neparametrické modely také obsahují parametry modelu, oproti parametrickým modelům ale pouze implicitně. Parametry modelu je pak možné získat vhodným vyhodnocením funkčních závislostí neparametrických modelů. [4]

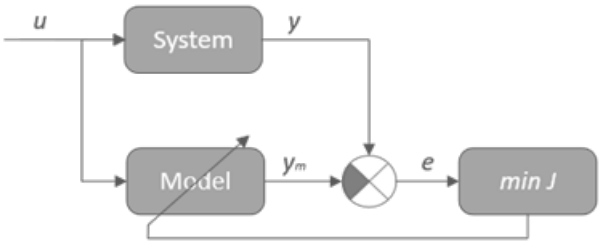
1.4.2 Vyhodnocení kvality modelu

Jak již bylo zmíněno výše, model je vždy nějaká aproximace reálného systému. Z tohoto důvodu výstup systému a zjednodušeného modelu nikdy nebude úplně totožný. Proto jsou pro validaci modelu využívána různá kritéria, která nám dávají představu o kvalitě modelu.[4] [6]

Pro vyhodnocení kvality modelu lze využít například kvadratická kritéria. Úloha vyhodnocení kvality modelu pak může být formulována jako optimalizační úloha minimalizace součtu kvadrátů odchylek například výstupu modelovaného a reálného systému na stejný vstupní signál, tak jak je znázorněno ve schématu 1.2. [4] [6]

Tab. 1.1: Vlastnosti analytické a experimentální identifikace [4]

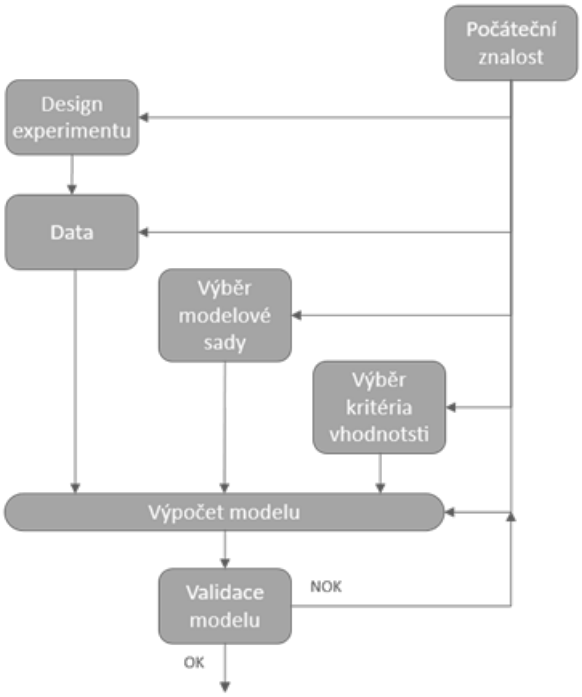
Vlastnost modelu	**Analytická identifikace**	**Experimentální identifikace**
Struktura modelu	Struktura vyplývá z přírodních zákonů	Struktura musí být zvolena
Parametry modelu	Parametry modelu jsou funkcemi systémových veličin, mají fyzikální význam	Parametry modelu jsou analytické proměnné, které umožňují většinou nalézt souvislost s fyzikálními systémovými proměnnými
Platnost modelu	Model platí pro celou třídu typů procesu a pro různé provozní stavy	Model platí pouze pro zkoumaný proces a konkrétní provozní stav. Proto lze chování popsat relativně přesně
Existence originálního systému	Model může být vytvořen i pro neexistující systém	Model může být identifikován pouze pro existující systém
Znalost vnitřní struktury	Důležité vnitřní procesy systému musí být známé a matematicky popsatelné	Vnitřní procesy nemusí být známé
Opakované použití metody	Každá tvorba modelu představuje opakovanou aplikaci fyzikálních zákonů	Metody nezávislé na jednotlivých systémech, vytvořené programové vybavení může být opakovaně použito pro identifikaci různých systémů
Časová náročnost	Tvorba modelu vyžaduje větší časové nároky	Menší časové nároky



Obr. 1.2: Vyhodnocení kvality modelu [5]

1.4.3 Proces systémové identifikace

V následujícím schématu je zobrazen postup při konstrukci modelu zkoumaného systému.



Obr. 1.3: Postup pří identifikaci systému [6]

Experimentátor na základě svých zkušeností navrhne způsob identifikace systému. Rozhodne, jaká data budou měřena, za jakých vstupních signálů a v jakých časových okamžicích, aby bylo dosaženo maximální informační hodnoty měřených dat. [6]

Dále experimentátor volí, jaký typ modelu je nejefektivnější pro modelování daného zkoumaného systému. Záleží na vlastnostech zkoumaného systému a také

na znalostech, zkušenostech experimentátora. Vybrat vhodný model je velmi důležitou částí procesu identifikace.[6]

Posledním krokem po obdržení modelu na základě vhodně zvoleného kritéria je jeho validace. Posouzení kvality modelu je typicky postaveno na tom, jak přesně model dokáže reprodukovat data naměřená na zkoumaném systému. Podle zvoleného kritéria vyhodnotíme, je-li model opravdu „dost dobrý“. Výsledný model se nikdy nebude chovat přesně jako skutečný zkoumaný systém, ale pro účely identifikace stačí dostatečně přesná aproximace.[6]

1.5 Matlab System Identification Toolbox

Tato kapitola se věnuje možnostem identifikace, které nabízí *System Identification Toolbox* v prostředí *MATLAB/Simulink*.

System Identification Toolbox umožňuje navrhovat matematické modely dynamických systémů z naměřených vstupních a výstupních dat. Tento přístup pomáhá modelovat systémy, které není snadné popsat pomocí základních fyzikálních zákonů. Toolbox nabízí grafické rozhraní, které usnadňuje práci při organizaci dat a modelů. Vytvořené modely lze importovat do prostředí *Workspace* nebo *Simulink* a dále provádět analýzu systémů a návrh řízení. [7]

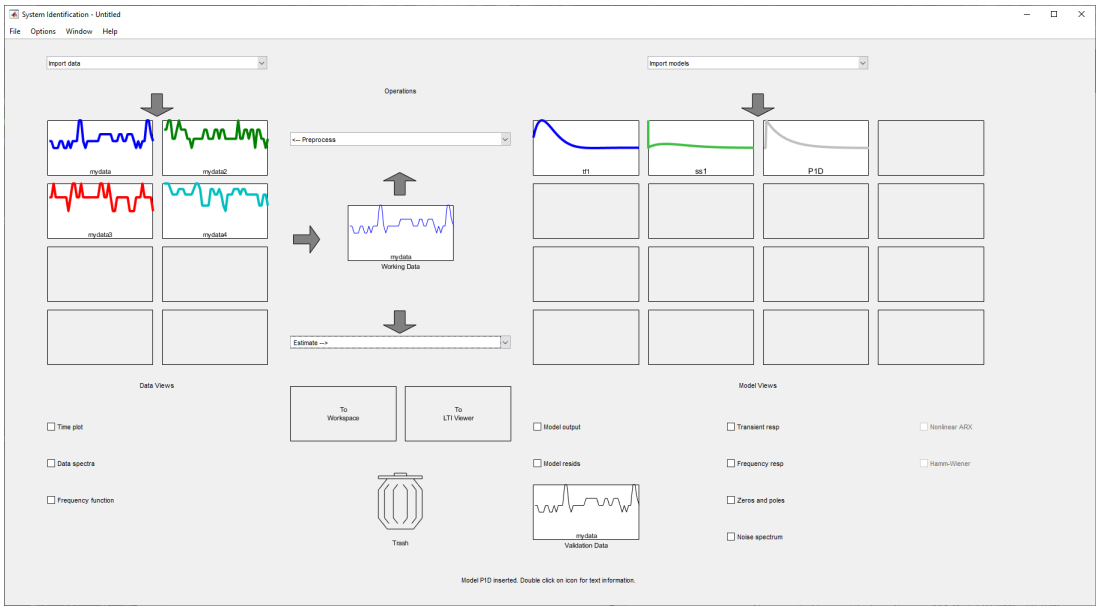
Vstupní data použita k identifikaci parametrů modelu mohou být vkládána v časové nebo frekvenční oblasti. Modely vytvořené pomocí tohoto toolboxu mohou být ve formě diskrétních či spojitých přenosových funkcí, popisu ve stavovém prostoru nebo ve formě nelineárních modelů. [7]

1.5.1 Seznámení se s prostředím toolboxu

Tato podkapitola obsahuje stručné seznámení se s grafickým prostředím toolboxu, postupem při zpracování signálu a vytvořením modelu.

Na obrázku 1.4 je zobrazeno grafické rozhraní toolboxu. V levém horním rohu se nachází rozbalovací seznam pro vkládání naměřených dat *Import data*. Vložená data se zobrazí v jednom z osmi políček v levé části okna. Data je možné si prohlédnout kliknutím na políčka *Time plot*, *Data spektra* nebo *Frequency function*. Pokud data nejsou vhodná pro identifikaci systému, obsahuje tento toolbox několik funkcí pro předzpracování dat dostupných v rozbalovacím seznamu *Preprocess*. [7]

Pro vytvoření modelu je nutné nejdříve pracovní data vložit na políčko *Working Data* a validační data na políčko *Validation Data*. Následně pomocí rozbalovacího seznamu *Estimate* vybrat vhodný model a zvolit jeho strukturu. Vytvořený model je importován do jednoho z šestnácti políček v pravé části okna, kde je možné porovnávat ho s ostatními modely a vyhodnotit jeho kvalitu. Po vytvoření dostatečně



Obr. 1.4: Hlavní okno *System identification toolboxu [7]*

přesného modelu lze daný model exportovat do *LTI Viewer* nebo do *Workspace* a následně provádět další experimenty. [7]

Nerelevantní vstupní data či vytvořené modely je možné jednoduše odstranit přesunutím těchto dat na ikonu popelnice. [7]

1.5.2 Vložení a předzpracování dat

Jako vstupní mohou být použita data naměřená v diskrétním čase (*time domain data*), data ve frekvenční oblasti (*freq. domain data*) nebo mohou být data vložena jako datový objekt (*data object*), který je nutné nejdříve vytvořit v *MATLAB Workspace* pomocí funkce *iddata*. V rámci vypracování demonstrační úlohy v této diplomové práci, byla naměřená data na reálné platformě vkládána do toolboxu ve formátu(*time domain data*). [7]

Získaná data je vhodné prohlédnout a případně provést předzpracování, pokud data vykazují některé z následujících nedostatků.

- Chybějící nebo chybná data (tzv. *outliers*)
- Offsety a drifty (nízkofrekvenční poruchy)
- Zašuměná data (vysokofrekvenční poruchy)

Rozbalovací seznam *Preprocess*, viz obrázek 1.4, nabízí možnosti, jak tyto nedostatky vstupních dat zpracovat ještě před samotnou identifikací, například: [7]

- Merge experiments
  - Pokud jsou k dispozici pouze segmenty naměřených dat, je možné je po nahrání těchto dat do toolboxu pomocí tohoto příkazu preprocessingu spojit.
- Select range
  - Výběr rozsahu dat, se kterými bude identifikace probíhat.
- Remove means
  - Odstranění střední hodnoty z naměřených dat.
- Filter
  - Slouží k odstranění šumu a driftu způsobenými vysokými a nízkými frekvencemi.

1.5.3 Volba struktury modelu

Kvalita výsledného modelu je kriticky závislá na jeho struktuře. Struktura modelu je volena v závislosti na apriorních znalostech o zkoumaném systému nebo ji experimentátor volí na základě svých empirických znalostí. Před zahájením procesu modelování, je tedy vhodné zobrazit si přechodové a frekvenční charakteristiky pro získání představy o základních vlastnostech zkoumaného systému. [7] [6]

System Identification Toolbox nabízí velké množství variant modelů. K volbě typu modelu a jeho struktury slouží rozbalovací seznam *Estimate* nabízející metody k odhadu lineárních a nelineárních modelů. Tyto modely je možné v identification toolboxu rozdělit do tří skupin. [7]

1. Lineární parametrické modely
   - Modely vstup-výstup (přenosové funkce)
   - Modely ve stavovém prostoru
   - Procesní modely
2. Lineární neparametrické modely
   - Jsou to přímé modely dané odezvy ať už v časové, čí frekvenční oblasti
   - Model impulsní odezvy
   - Frekvenční odezvy
3. Nelineární modely
   - ARX model
   - Hammerstein-Wiener model

1.5.4 Nalezení nejlepšího modelu zvolené struktury

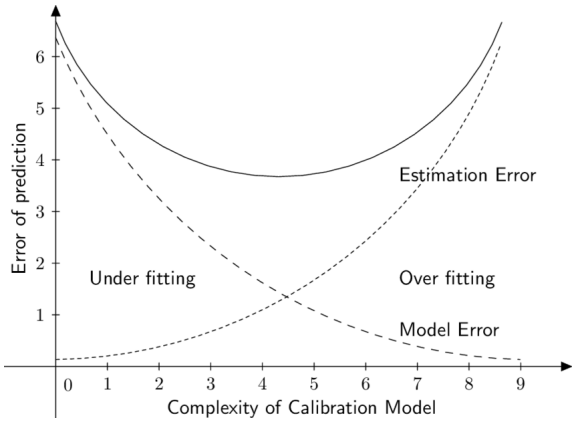
Nalezení nejlepšího modelu zvolené struktury je hlavní úlohou tohoto toolboxu. Díky mnoha funkcím toolbox umožňuje rychle a jednoduše vytvářet množství modelů,

které lze snadno porovnávat jak se skutečnými naměřenými daty, tak mezi sebou. [7]

Vhodným postupem při vytváření modelu je začít s odhadem parametrů modelu s jednoduchou strukturou. Jestliže je odezva takového modelu nevyhovující, je nutné zvyšovat komplexnost struktury až do bodu, kdy je dosaženo požadované přesnosti modelu. [8]

Při zvyšování komplexity, tedy zvyšování počtu parametrů modelu, je možné dostat se do stavu, kdy odchylka na trénovacích datech neustále klesá a model se s přibývajícím počtem parametrů zdá stále přesnější. Ve skutečnosti se model naučil takové závislosti na trénovacích datech, které se ve zkoumaném systému nevyskytují, tzv. *"over fitting"*. Při použití testovacích dat je poté vidět, že s přibývajícím počtem parametrů přesnost modelu klesá. [7] [8]

Nakonec je vhodné vybrat nejjednodušší model nejlépe popisující dynamiku zkoumaného systému. [8]

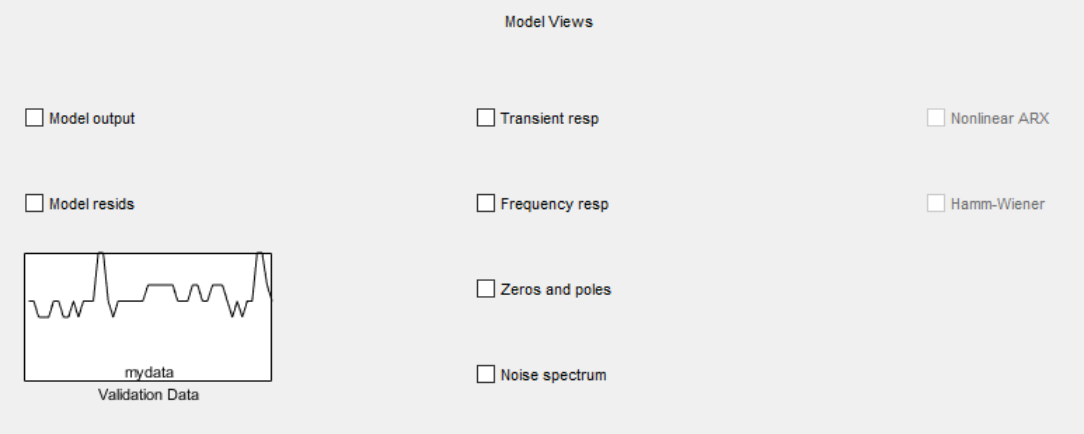


Obr. 1.5: Chyba modelu v závislosti na jeho komplexnosti [9]

1.5.5 Práce s již identifikovanými modely

Modely je možné prohlížet, přeskupovat, odstranit nebo je exportovat do *MATLAB Workspace* či *MATLAB Simulink*. Toolbox poskytuje funkce pro zobrazování a porovnávání jednotlivých modelů. [7]

Pro porovnání výstupu modelu a výstupu zkoumaného systému slouží funkce *Model output*. Ke zobrazení frekvenční/impulsové charakteristiky vytvořeného modelu a jeho frekvenční odezvy slouží funkce *Transient response* a *Frequency response*. Další vhodnou funkcí pro zkoumání kvality výsledného modelu je analýza odchylky mezi výstupem validačních dat modelovaného systému a predikovaným výstupem



Obr. 1.6: Funkce pro práci s modely v System identification toolboxu [7]

modelu, tzv. analýza rezidua. Pro zobrazení těchto testů v toolboxu zaškrtneme políčko *Model resids*. [7]

1.6 Simulink PLC coder

Tato kapitola se věnuje možnostem generování kódu pro PLC s využitím nástroje *Simulink PLC Coder* v nadstavbě programu *MATLAB/Simulinku*.

Simulink PLC Coder slouží pro generování strukturovaného textu a ladder diagramu dle normy IEC 61131-3 pro programovatelné automaty ze simulinkových modelů, stateflow diagramů a algoritmů v jazyce *MATLAB*. Strukturovaný text je generovaný v *PLCopen XML* a dalších souborových formátech podporovaných integrovanými vývojovými prostředími (IDE) zahrnujícími *Siemens TIA Portal, Rockwell Automation Studio 5000, Omron Sysmac Studio* a další. Ladder diagram je generovaný v souborovém formátu podporovaným IDE *Rockwell Automation Studio 5000*. [10]

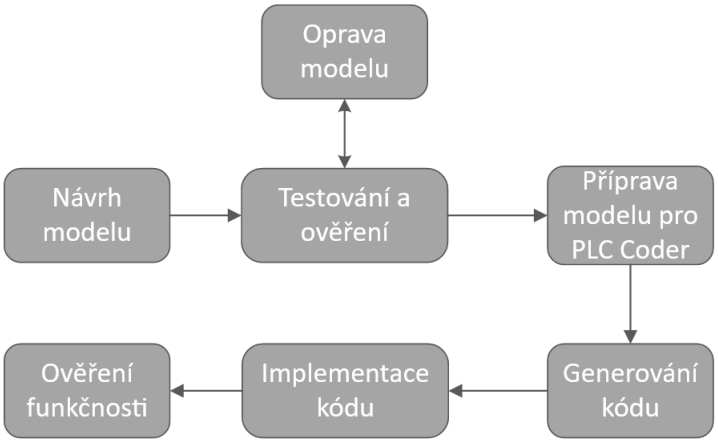
Pomocí *Simulink PLC Coderu* lze také vytvořit doplňkové testovací soubory, díky kterým lze výsledky získané pomocí strukturovaného textu spuštěného v IDE porovnat s výsledky ze simulací. [10]

Jelikož je demonstrační úloha uvedená v této práci implementována pomocí vývojového prostředí *Tia Portal V15* na zařízení *SIMATIC S7-1500 PLC* společnosti *Siemens*, bude se tato práce dále zabývat pouze možností generování strukturovaného textu.

1.6.1 Příprava simulinkového modelu pro generování kódu

Nejdříve musí být daný model a samotná simulace nakonfigurována tak, aby byly splněny všechny podmínky PLC coderu pro generování strukturovaného textu.

postup při návrhu modelu pro generování kódu a jeho následná implementace by se dal shrnout následujícím blokovým schématem.



Obr. 1.7: Postup při návrhu modelu pro generování kódu

Tasking mode

Jako první je potřeba si uvědomit, jestli všechny části vytvořeného modelu, respektive subsystému, pro který má být vygenerován kód, pracují se stejnou periodou vzorkování. Pokud ano, jedná se o tzv. *single-tasking* subsystém a je možné pokračovat v nastavení solveru. Pokud ne, jde o tzv. *multirate* subsystém a jelikož PLC coder pracuje pouze se *single-tasking* subsystémy, je nutné před výběrem solveru provést explicitní nastavení *Tasking modu* na *Single-tasking*. V okně *Configuration parameters* v záložce *Solver* je nutné zrušit zaškrtnutí políčka *Treat each discrete rate as a separate task* viz obrázek 1.11. [10]

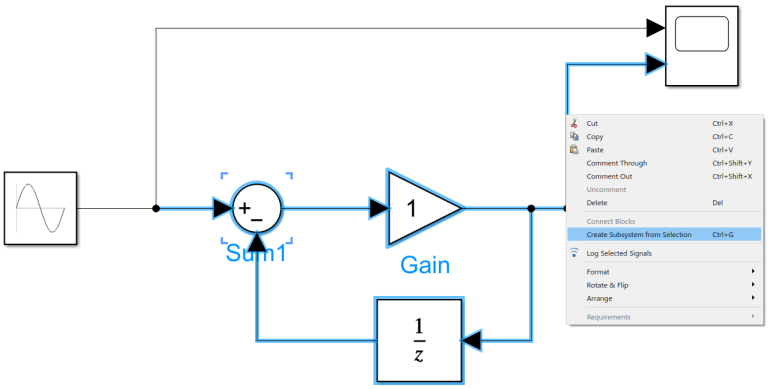
Vytvoření subsystému

Dalším krokem v přípravě modelu je vytvoření subsystému z bloků, které mají být součástí generovaného kódu. Subsystém je blok skládající se z podmnožiny bloků v rámci modelu nebo systému. Blok subsystému může představovat virtuální subsystém nebo nevirtuální subsystém. [10]

Simulink PLC Coder umožňuje generování kódu pouze pro nevirtuální subsystémy tzv. *atomic subsystems*. Tyto nevirtuální subsystémy se oproti virtuálním subsystémům chovají jako jeden celek, tudíž při vykonávání simulace se nejdříve provedou všechny bloky v daném nevirtuálním subsystému, než simulace přejde

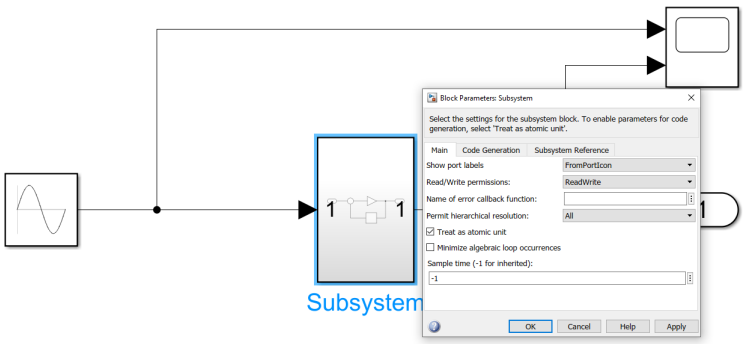
k vykonávání dalších bloků mimo subsystém. Označení subsystému jako nevirtuálního subsystému tedy zajistí, že nedojde k proložení vykonávání jednotlivých bloků v rámci subsystému s blokem nacházejícím se mimo tento subsystém.[10]

Subsystém lze vytvořit označením bloků, pro které má být generován kód, dále kliknutím na libovolný z bloků pravým tlačítkem a výběrem možnosti *Create Subsystem from Selection*.



Obr. 1.8: Vytvoření subsystému ze zvolených bloků [10]

Po vytvoření subsystému už zbývá pouze jeho konfigurace na nevirtuální subsystém. To lze provést opět kliknutím pravým tlačítkem, nyní už na vytvořený subsystém, a výběrem *Block Parameters*. V zobrazeném dialogovém okně stačí zakliknout možnost *Treat as atomic unit*. [10]

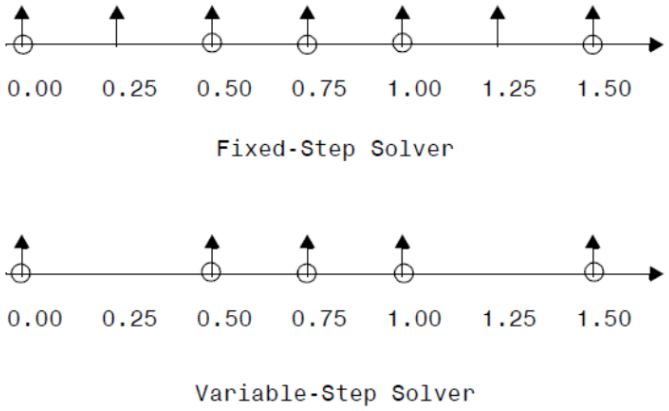


Obr. 1.9: Konfigurace subsystému na nevirtuální subsystém [10]

Výběr numerické metody výpočtu a kroku výpočtu

Dalšími důležitými parametry, které je nutné nastavit před generováním kódu jsou numerické metody výpočtu a krok výpočtu.

Pro nastavení kroku výpočtu jsou dostupné dvě možnosti. Řešení modelu s fixním krokem *(Fixed-step)*, které řeší model ve fixních časových intervalech, a řešení s proměnným krokem *(Variable-step)*, které optimalizuje velikost kroku podle toho, jakým způsobem se mění proměnné v simulaci. Pokud se proměnné mění velmi rychle, velikost kroku se zmenšuje a naopak, pokud se mění velmi pomalu, velikost kroku se zvětšuje. Tato metoda řešení modelu umožňuje vyhnout se zbytečnému vzorkování a také zlepšit přesnost simulace a její výpočetní efektivitu. [10] [11]



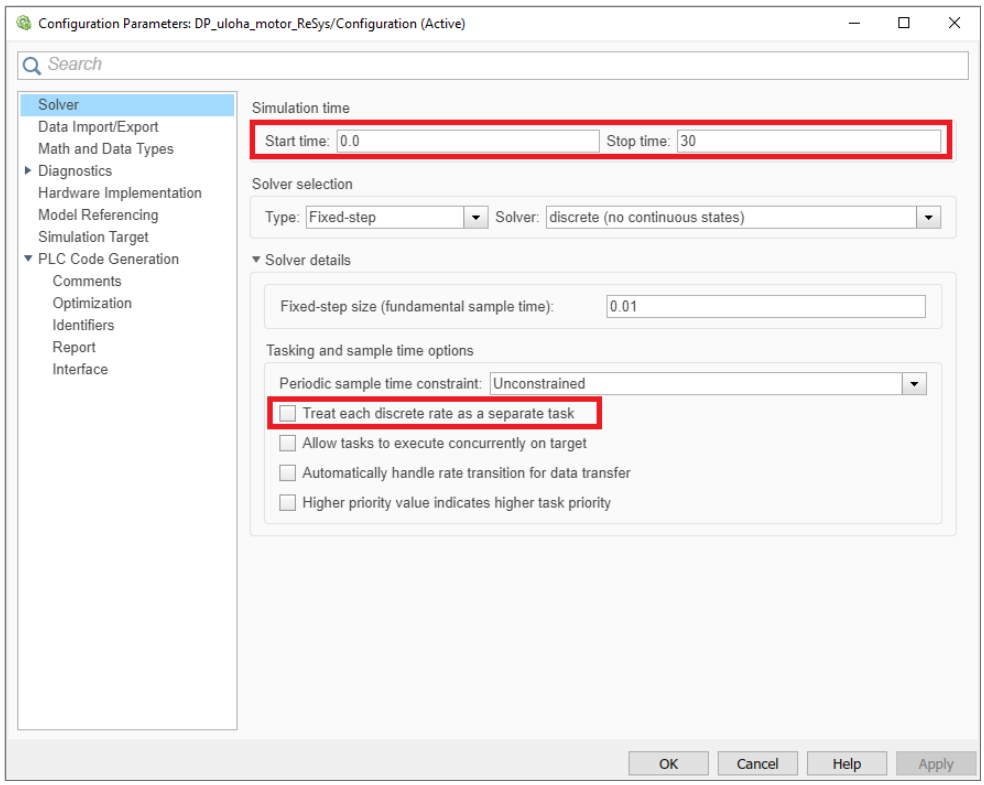
Obr. 1.10: Příklad fixních kroků a proměnných kroků *Solveru* [11]

Pro spojité modely je možné využít numerické metody výpočtu, které k získání spojitých stavů modelu v aktuálním časovém kroku využívají stavů v předchozích časových krocích a stavových derivacích. Matematici vyvinuli širokou škálu technik pro řešení obyčejných diferenciálních rovnic (ODR), které představují spojité stavy dynamických systémů. Simulink poskytuje rozsáhlou sadu numerických metod s fixním a proměnným krokem simulace, z nichž každá implementuje specifickou metodu řešení ODR. [10] [11]

Pro řešení diskrétních modelů se využívá metody s fixním krokem *Discrete (no continuous states)*. Pokud se tedy model skládá pouze z diskrétních bloků, lze zvolit typ solveru s fixním krokem *(Fixed-step)* a jeho nastavení na *Discrete (no continuous states)*. [11]

V případě, kdy se vytvořený model skládá ze spojitých i diskrétních částí, je nutné použít typ solveru s proměnným krokem *(Variable-step)* a vybrat vhodnou numerickou metodu řešení modelu. Poté je ale nezbytné definovat periodu vzorkování subsystému, pro který má být generován kód. Takový subsystém, musí být složen pouze z diskrétních bloků podporovaných PLC coderem, které lze prohlédnout v

knihovně *plclib* zadáním příkazu *plclib* do *Matlab Workspace*. Pro spojité bloky nelze generovat kód. [10] [11]



Obr. 1.11: Nastavení typu *Solveru* a *Tasking modu*

Nastavení periody vzorkování subsystému je možné kliknutím pravým tlačítkem na subsystém, následným výběrem možnosti *Block Parameters* dojde k zobrazení dialogového okna, ve kterém v kolonce *Sample time* lze definovat perioda vzorkování daného subsystému.

1.6.2 Generování a implementace kódu

Generování kódu

Po přípravě modelu a nakonfigurování simulace je možné přistoupit k samotnému generování kódu. Jako první je nutné v dialogovém okně *Configuration Parameters* v rozbalovacím seznamu *Target IDE* vybrat požadované IDE, které podporuje používané PLC. V rámci této práce je použito IDE od společnosti *Siemens*. Vybrat lze ze tři možností. [10]

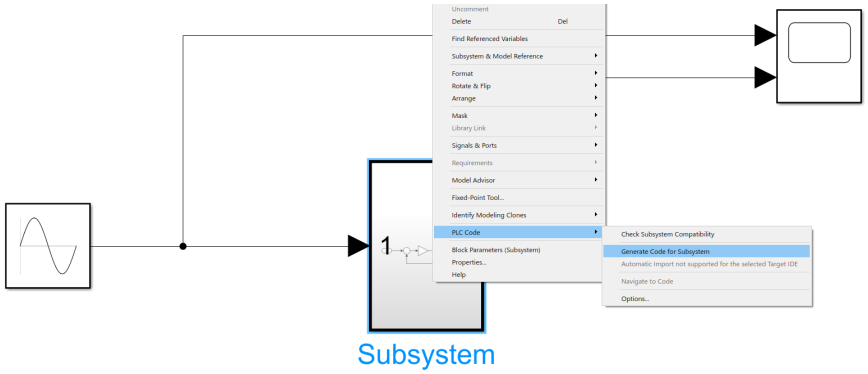
- Siemens SIMATIC Step 7
  - Generuje strukturovaný text pro Siemens SIMATIC Step 7.
- Siemens TIA Portal
  - Generuje strukturovaný text pro Siemens TIA Portal S7-300/400 CPU.
- Siemens TIA Portal: Double precision
  - Generuje strukturovaný text pro Siemens TIA Portal S7-1200 a S7-1500 CPU.

V této práci je využito dostupné *PLC SIMATIC S7-1500* tudíž zvolena je možnost *Siemens TIA Portal: Double precision*, jak lze vidět v dialogovém okně na obrázku 1.13.

Dále ve stejném dialogovém okně v okénku *Code Output Directory* je možné zvolit název složky do které budou uloženy generované soubory, defaultní název je *plcsrc*. Po vygenerování kódu se tato složka zobrazí ve stejném adresáři ve kterém se nachází vytvořený projekt. [10]

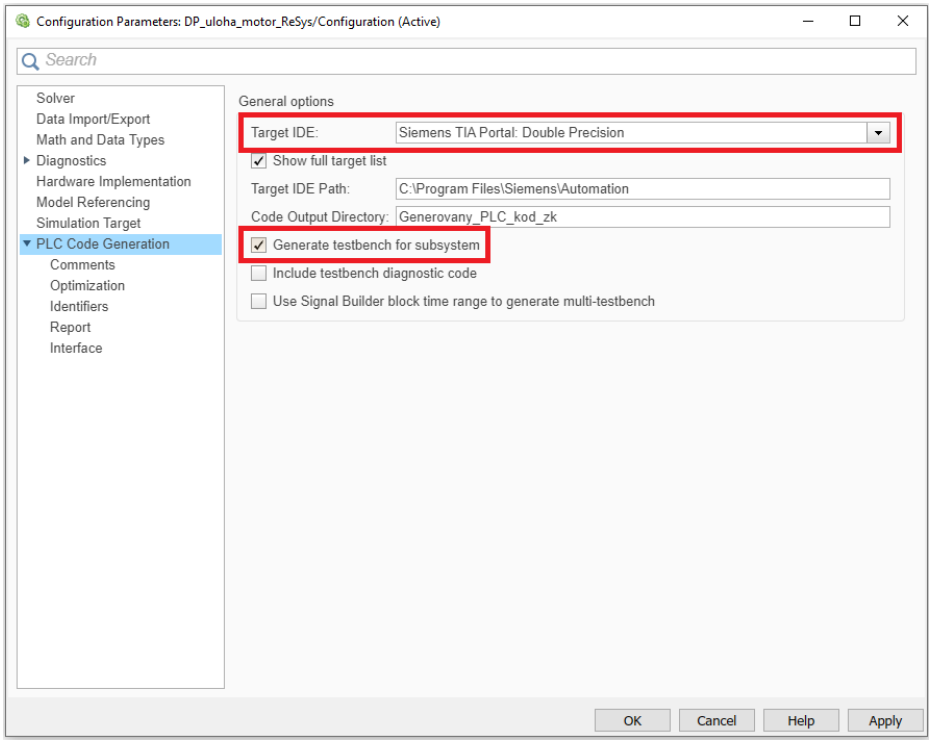
Jako poslední před generováním kódu je nutné zkontrolovat kompatibilitu subsystému, splňuje-li kritéria PLC coderu. Po ověření se otevře okno *Diagnostic Viewer* s potvrzením kompatibility nebo případnými nedostatky. [10]

Kód lze vygenerovat kliknutím pravým tlačítkem na zvolený subsystém a v záložce *PLC Code* výběrem *Generate Code for Subsystem* viz 1.12.



Obr. 1.12: Generování kódu pomocí *Simulink PLC Coderu* [10]

Bezprostředně po vytvoření kódu se otevře okno *Code Generation Report*, ve kterém je možné prohlédnout si vygenerovaný kód, seznam eliminovaných a použitých bloků a dále také statistické informace o vygenerovaném kódu, jako jsou například počet vygenerovaných souborů, počet řádků kódu bez komentářů, celkový počet řádků kódu a velikosti jednotlivých generovaných bloků.

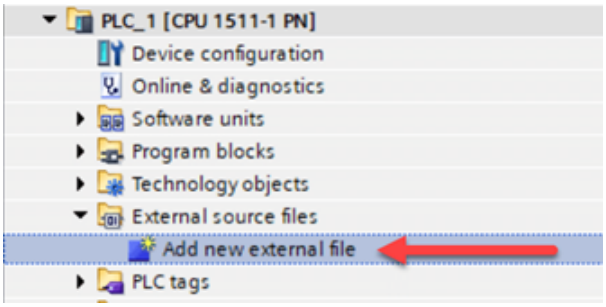


Obr. 1.13: Výběr cílového IDE

Implementace kódu

Jak již bylo napsáno výše, v této práci je využíváno PLC společnosti *Siemens*, a tudíž vývojové prostředí *TIA Portal*. Po vytvoření projektu v *TIA Portalu* a jeho hardwarové konfiguraci stačí k implementaci generovaného kódu vložit ve stromové struktuře projektu v záložce *External source files* vygenerovaný soubor s příponou SCL. [10]

Po přidání vygenerovaného souboru s příponou SCL, zbývá pouze vygenerovat blok obsahující generovaný kód. Tento blok lze vygenerovat kliknutím pravým tlačítkem na přidaný soubor a výběrem možnosti *Generate source from blocks*. Ve stromové struktuře programu v záložce *Program blocks* je poté vytvořen funkční blok s generovaným kódem a názvem odpovídajícím subsystému v simulinku. [12]



Obr. 1.14: Implementování externích souborů do projektu v IDE [12]

1.6.3 Optimalizace kódu

Simulink PLC Coder nabízí optimalizace pro redukci velikosti paměti a zvýšení rychlosti vykonávání programu generovaného strukturovaného textu. Tyto optimalizace zahrnují: [10]

- Dead-code elimination
  - Optimalizace, která odebere části kódu, které neovlivňují výsledek programu.
- Expresion folding
  - Výpočet jednotlivých bloků je sbalen do jednoho optimalizovaného výrazu, místo toho, aby každému jednomu bloku byl dedikován jeden řádek kódu a deklarováno místo v paměti pro mezivýsledek.
- For-loop fusion
  - Pokud je to možné, kombinuje více bloků do stejné for smyčky, čímž zvyšuje čitelnost a efektivitu kódu.

Některé možnosti optimalizace kódu lze nastavit v okně *Configuration Parameters* v záložce *PLC Code Generation* > *Optimization*. [10]

- Default parameter behavior

Inlined	Generovaný kód nealokuje paměť k reprezentaci číselných bloků, jako je například parametr Gain bloku Gain. Místo toho je do kódu přímo vložena číselná hodnota bloku.
Tunable	Umožňuje laditelnost parametrů numerických bloků ve vygenerovaném kódu.
Configuration	Umožňuje nastavit jednotlivé parametry použité v modelu jako globální.

- Signal storage reuse

On	Dojde ke znovupoužití paměti alokované k ukládání vstupních a výstupních signálů bloku, čímž snižuje požadavky na paměť programu.
Off	Přidělí samostatnou paměť pro výstupy každého bloku. díky této alokaci jsou výstupy každého bloku globální, čímž ovšem zvýšíme využití paměti.

- Remove code from floating-point to integer conversions that wraps out-of-range values

On	Z kódu, který převádí čísla s plovoucí desetinnou čárkou na celá čísla, odebere tu část, která se stará o zpracování hodnot mimo rozsah. Odstranění tohoto kódu zmenší velikost a zvýší rychlost generovaného kódu, ale může dojít k nesouladu s výsledky simulace.
Off	Neodstraní kód z převodů s plovoucí desetinu čárkou na celá čísla. Vygenerovaný kód je větší, ale výsledky odpovídají výsledkům simulace i pro hodnoty mimo rozsah.

- Generate reusable code

On	Pokud model obsahuje více identických subsystémů, které se liší pouze svými vstupy, výběrem této optimalizace dojde k vygenerování pouze jednoho funkčního bloku, který lze použít opakovaně.
Off	Namísto jednoho znovu použitelného funkčního bloku software vygeneruje stejné funkční bloky pro každý jeden subsystém a to kvůli jistým rozdílům v jejich vstupech.

- Inline named constants

On	Nahradí *named constants* jejich celočíselnou hodnotou.
Off	V generovaném kódu jsou použity *named constants*.

- Reuse MATLAB Function block variables

On	Na vhodných místech dojde k opětovnému použití proměnných.
Off	V generovaném kódu nedochází k opětovnému použití proměnných.

Potřeba optimalizace kódu se může měnit s fází práce na projektu. Například během vývoje projektu je výhodné umístit parametry do globální paměti pro ladění nebo kalibraci. Během následného produkčního sestavení je poté možné generovat parametry s jejich doslovnými hodnotami za použití možnosti *Inline parameters*, pro vytvoření optimálnějšího kódu. [10]

1.6.4 Čitelnost kódu a ověření jeho funkčnosti

Simulink PLC Coder umožňuje vkládat komentáře a uživatelem specifikované popisky jednotlivých bloků do strukturovaného textu, což zlepšuje orientaci v generovaném kódu. PLC Coder umožňuje vytvářet jedinečné identifikátory, které zachovávají názvy objektů a názvy signálů v modelu. Tyto funkce umožňují efektivnější kontroly kódu a přehled toho, jak byl model implementován. [10]

Pomocí PLC coderu je možné vytvořit doplňkový testovací soubor, který slouží k ověření, že výsledky simulace a výsledky získané pomocí generovaného kódu spuštěného v IDE se shodují v přijatelné toleranci. [10]

Pro vygenerování testovacího kódu stačí v dialogovém okně *Configuration Parameters* v záložce *PLC Code Generation* vybrat možnost *Generate testbench for subsystem* viz obrázek 1.13. [10]

Vytvoření testovacího kódu probíhá tím způsobem, že nejdříve proběhne simulace vytvořeného modelu a dojde k zaznamenání vstupních a výstupních signálů subsystému, pro který je generován kód. Takto zaznamenaná data jsou testovací data. [10]

Samotné testování v cílovém IDE probíhá následovně. Vygenerovaný testovací soubor je ve formě funkčního bloku, který distribuuje vstupní signál pro testovaný kód a čte jeho výstup, který poté porovnává se zaznamenanými hodnotami ze simulace, jsou-li jejich případné rozdíly v dané toleranci. [10]

Tab. 1.2: Tolerance chyby různých datových typů při ověřování generovaného kódu [10]

Datový typ	**Způsob porovnání**	**Tolerance**
integer	absolutní	0
boolean	absolutní	0
single	relativní	0.0001
double	relativní	0.00001

Výstupem testovacího bloku je parametr *testVerify*, který je datového typu *BOOL*. Pokud jsou si výsledky obou simulací dostatečně podobné, jeho hodnota je *TRUE*.

Dalším výstupem je *testCycleNum*, datového typu *DINT*, díky kterému lze případně zjistit, v jakém okamžiku testování selhalo. [10]

1.6.5 Generování kódu pomocí stateflow diagramů

Při generování kódu ze stateflow diagramů není nutné tyto diagramy vkládat do subsystémů a nastavovat je jako nevirtuální. Je nezbytné dát si pozor pouze na dodržení všech omezení, které jsou dány PLC coderem. [10]

Ostatní pravidla pro nastavení simulace, optimalizace a generování kódu popsané výše v této kapitole platí stejným způsobem i pro generování kódu pomocí stateflow diagramů.

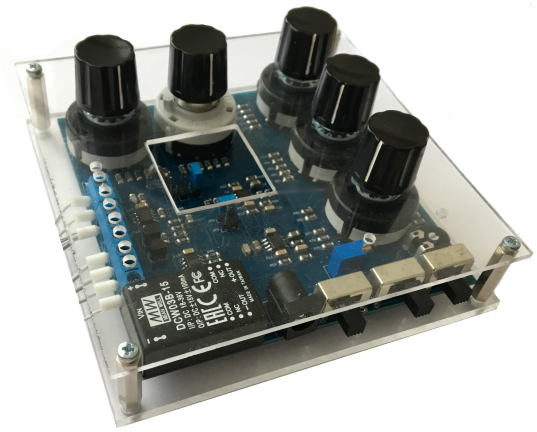
2 ANALÝZA PLATFORMY A NÁVRH NOVÉ VERZE

Tato kapitola se zabývá koncepcí návrhu a způsobem ovládání první verze platformy, zhodnocením nedostatků, návrhem jejich řešení a konceptem a realizací nové verze laboratorní platformy.

2.1 Rozbor první verze platformy

Tato platforma simuluje jednoduché technologické procesy, je vybavena analogovými obvody s nastavitelnými parametry simulující různé dynamiky nejčastěji se vyskytujících systémů z průmyslové praxe. [13]

První verze platformy je složena ze sedmi bloků, znázorněných v blokovém schématu na obrázku 2.3. Bližší popis jednotlivých bloků je uveden níže.



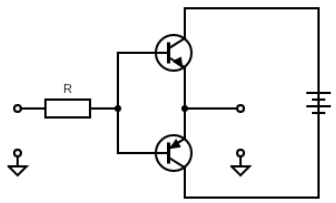
Obr. 2.1: První verze platformy [13]

Setrvačný člen prvního řádu

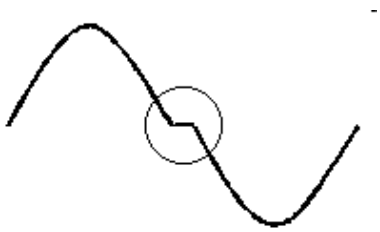
Tento člen je vytvořen pomocí RC článku, může představovat dynamiku například měřicího čidla, kterou je ve srovnání s ostatními dynamikami zkoumaného systému možné zanedbat. [13]

Pásmo necitlivosti

Dalším členem je pásmo necitlivosti tvořené komplementárními tranzistory zapojenými jako dva proti sobě pracující sledovače napětí. Při napětích blízkých nule zavádí do obvodu přechodové zkreslení. Tento člen může simulovat například vůli v převodech. [13]



(a) Komplementární sledovač [14]



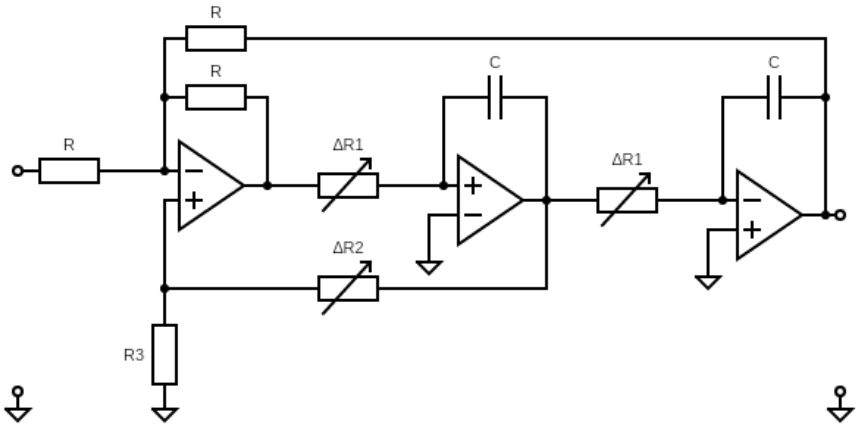
(b) Přechodové zkreslení [14]

Setrvačný člen prvního řádu s nastavitelnou časovou konstantou

Oproti prvnímu členu má tento setrvačný článek nastavitelnou časovou konstantu na hodnoty od 5,6 ms do 5,65 s. Dále je k rezistoru v RC článku paralelně připojitelný termistor nebo trimer, což má simulovat působení různých nelinearit, viz zapojení 2.4. [13]

Setrvačný člen druhého řádu

Tento člen je tvořen třemi operačními zesilovači v zapojení odpovídajícím přenosové rovnici druhého řádu 2.2. Změnou odporů R_1 a R_2 pomocí dvou rotačních přepínačů lze nastavovat tlumení a zesílení tohoto členu. [13]



Obr. 2.2: Setrvačný člen druhého řádu [15]

Rozdílový člen

Pátým blokem je rozdílový člen, který je vytvořen pomocí otáčkového potenciometru. Tento člen simuluje poruchu působící na vstupu/výstupu systému. [13]

Proporcionální člen

Tento člen simuluje proporcionální systémy. Je tvořen operačním zesilovačem v invertujícím zapojení se zesílením nastavitelným pomocí změny odporu připojeného na invertující vstup. [13]

Integrační člen

Posledním blokem v první verzi platformy je integrační člen s nastavitelnou rychlostí integrace změnou velikosti odporu. [13]

2.2 Ovládání platformy

Platforma je navržena tak, že jednotlivé bloky simulující různé dynamiky mohou být buďto přemostěny nebo úplně vynechány z cesty procházejícího vstupního signálu. Tímto způsobem lze jednotlivé dynamiky kombinovat a získat tak na výstupu platformy mnoho zajímavých průběhů.

Jak lze vidět v blokovém diagramu na obrázku 2.3, první dva členy, setrvačný člen prvního řádu a pásmo necitlivosti, je možné přemostit pomocí páčkových přepínačů *(SW)*. Dále za třetím až šestým členem jsou umístěny jumpery *(JP)*, které umožňují modulární propojení těchto členů. Paralelní připojení potenciometru a termistoru k druhému setrvačnému článku prvního řádu lze také pomocí jumperů.

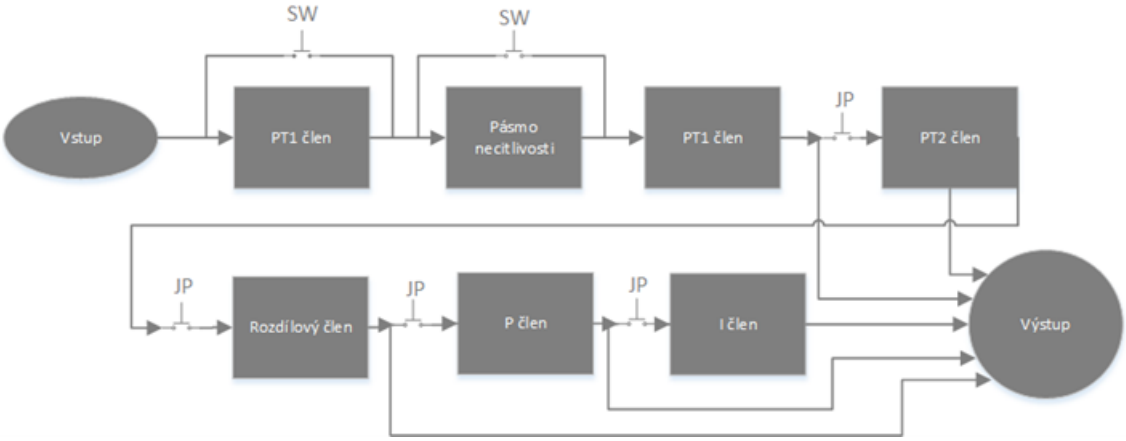
Rozdílový člen je tvořen otáčkovým potenciometrem, jehož otáčením je měněn poměr děliče a tím pádem velikost výstupního napětí tohoto bloku. U členů s nastavitelnými časovými konstantami se nastavení provádí dvanácti polohovými rotačními přepínači.

2.3 Zhodnocení nedostatků

V této kapitole jsou rozebrány nedostatky první verze platformy, co se týče její koncepce, použitých součástek a zapojení.

2.3.1 Koncepce platformy

Jedním z nedostatků první verze platformy je, že není dostatečně uživatelsky přívětivá. Uživatel se nejdříve musí seznámit se způsobem ovládání platformy, které není příliš intuitivní, a také s jejím zapojením.



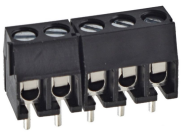
Obr. 2.3: Blokové schéma první verze platformy [13]

Dalším nedostatek umožňujícím vylepšení platformy je její samotná koncepce. Platforma je vytvořena z bloků, z nichž některé je možné přemostit pomocí jumperů *(JP)* nebo přepínačů *(SW)*, jak bylo popsáno v kapitole 2.2. Nicméně, vstupní signál musí vždy procházet přes setrvačný člen prvního řádu, který jej ovlivní svou dynamikou.

2.3.2 Volba vhodnějších součástek

PT1, *PT2*, *P* a *I* člen platformy mají nastavitelné parametry zesílení či časové konstanty. Pro přepínání těchto parametrů jsou použity dvanácti polohové rotační přepínače přepínající odpory různé velikosti. Tento typ přepínače není pro tento přípravek nejvhodnější volbou, jelikož přepínače na sobě ani na desce nemají žádné značení a není tudíž zcela jasné, jaká hodnota je nastavena. Další nevýhodou těchto přepínačů je také jejich vysoká cena.

(a) Rotační přepínač



(b) Svorkovnice

Platforma je připojitelná k PLC přes plochý pětadvaceti žilový kabel. Jelikož je použit pouze jeden analogový výstup a čtyři analogové vstupy, je využito pouze

deset žil, které jsou připojeny do svorkovnice. Vhodnějším řešením by bylo použití konektoru, který by urychlil připojení platformy k PLC a znemožnil nesprávné zapojení.

2.3.3 Filtrace napájecího napětí

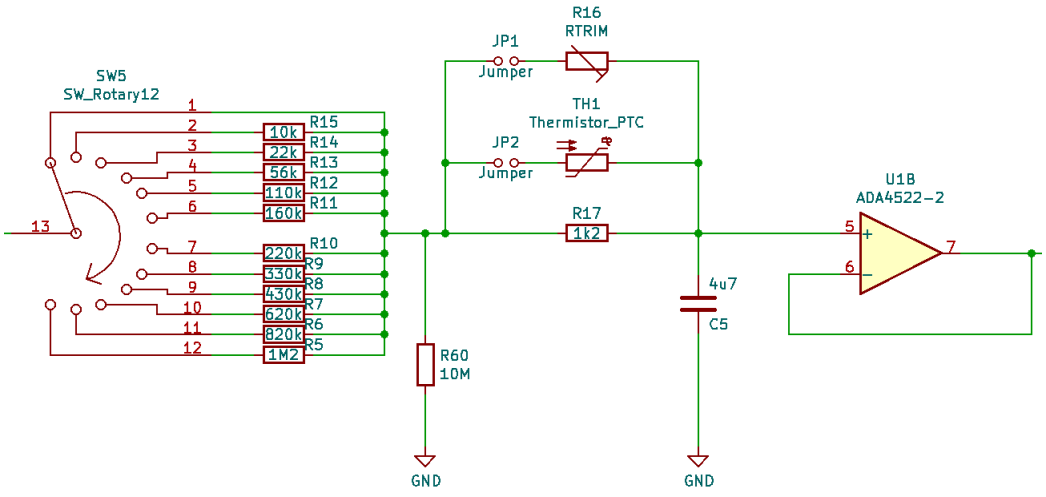
Tato platforma je postavena na operačních zesilovačích. Operační zesilovače jsou citlivé na změnu napájecího napětí. Při rychlých změnách vstupního signálu může šum napájecího napětí operačních zesilovačů ovlivnit jejich výstup. Většina operačních zesilovačů je schopna tento šum do určitých frekvencí potlačit. Tato schopnost se nazývá Power Supply Rejection Ratio *(PSRR)*.

Prevencí proti šumu je filtrace napájecího napětí jak u zdroje, tak i u operačních zesilovačů umístěním blokovacích kondenzátorů co nejblíže k součástce.

2.3.4 Setrvačný člen prvního řádu

V následujícím schématu 2.4 je zobrazeno zapojení setrvačného členu prvního řádu s možností paralelního připojení nastavitelného odporu a termistoru. Možnost paralelního připojení má představovat působení různých nelinearit, které ovlivní časovou konstantu RC článku.

Hodnoty odporů termistoru a potenciometru jsou řádově vyšší než odpor rezistoru *R17*, tudíž dojde-li k paralelnímu připojení těchto odporů, výsledný odpor se téměř nezmění, a tudíž nedojde k ovlivnění dynamiky.



Obr. 2.4: Nedostatky zapojení se setrvačným členem prvního řádu [15]

2.3.5 Saturace integračního členu

Při použití integračního členu dochází po přivedení signálu na vstup k nárůstu výstupního napětí. Problém přichází v okamžiku, kdy se vstupní signál vrátí na nulovou hodnotu a uživatel chce pokračovat v dalším měření. Integrační člen ovšem drží svoji hodnotu. Pro opětovné měření je nutné odpojit napájení a počkat na vybití kondenzátoru.

2.4 Návrh druhé verze laboratorní platformy

Tato kapitola se zabývá návrhem řešení nedostatků první verze platformy, jejím doplněním o digitální část, návrhem obvodového řešení a jeho realizací.

Koncepce druhé verze platformy je shodná s její první verzí. Jedná se tedy o laboratorní přípravek, který umožňuje simulovat jednoduché technologické procesy za využití analogových obvodů s nastavitelnými parametry simulující různé dynamiky. Nová verze je doplněna o digitální část, která má umožňovat měření výstupů jednotlivých bloků a nastavování některých dalších parametrů popsaných v 2.5.

Druhou verzi platformy je tedy možné rozdělit na analogovou a digitální část.

2.4.1 Analogová část

Analogová část je obdobná jako u první verze platformy doplněná o řešení nedostatků, které jsou uvedeny v předchozí kapitole.

Oproti první verzi je každý blok brán jako samostatný modul, který do obvodu nemusí nebo může být připojen. Celkem je použito pět bloků, které jsou navzájem propojitelné v libovolném pořadí, jak je znázorněno v blokovém schématu, obrázek 2.5.

- PT1 člen – simulující dynamiky systémů prvních řádů s nastavitelnou časovou konstantou
- PT2 člen – simulující dynamiky systémů druhých řádů s nastavitelnými parametry zesílení a tlumení
- Rozdílový člen – simulace poruchy působící na výstupu systému
- P člen – proporcionální člen s nastavitelným zesílením
- I člen – simulace systému s astatismem a nastavitelnou rychlostí integrace

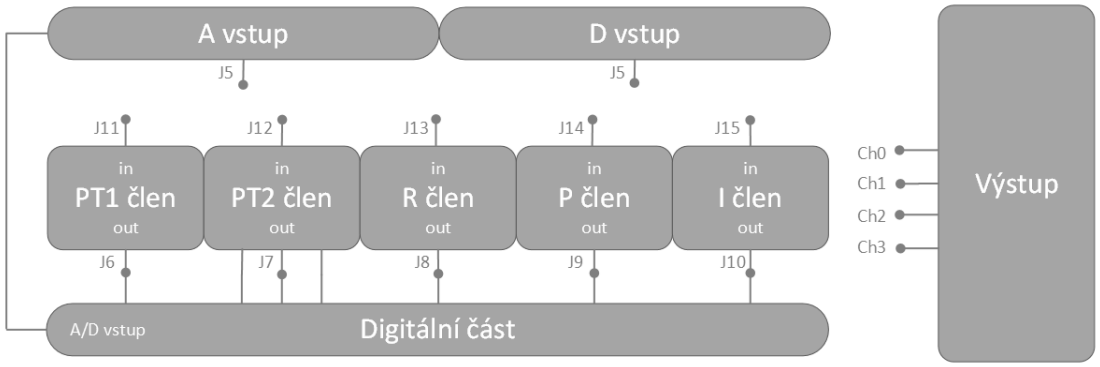
2.4.2 Digitální část

Nová verze platformy je doplněna o digitální část, pomocí které je možné vytvořit libovolný vstupní signál, měřit výstupní hodnoty a díky dvěma číslicovým potenciometrům nastavovat podíl rozdílového členu a ovlivňovat dynamiku setrvačného

členu prvního řádu.

V budoucnu je zamýšlen vývoj aplikace pro ovládání platformy pomocí digitální části.

Podrobnější popis digitální části je uveden v kapitole 2.5.



Obr. 2.5: Blokové schéma druhé verze platformy

2.5 Návrh obvodového řešení

Tato kapitola se zabývá návrhem obvodového řešení analogové a digitální části platformy doplněné o nedostatky uvedené v kapitole 2.3.

2.5.1 Analogová část

V této kapitole bude blíže popsán způsob obvodového návrhu bloků v analogové části platformy.

Setrvačný člen prvního řádu

Prvním blokem je setrvačný člen prvního řádu sestavený jako RC článek s možností změny časové konstanty přepínáním velikosti odporu.

Pro přepínání odporu je namísto dvanácti polohového rotačního přepínače použit šestimístný DIP přepínač. Výhodou použití tohoto přepínače je, že po přepnutí je jasně vidět, jaký odpor je nastaven. Další výhodou jsou jeho menší rozměry a možnost aktivovat více paralelních větví s odpory a tím získat více volitelných odporů za použití menšího množství součástek. V neposlední řadě jsou DIP přepínače o poznání levnější.

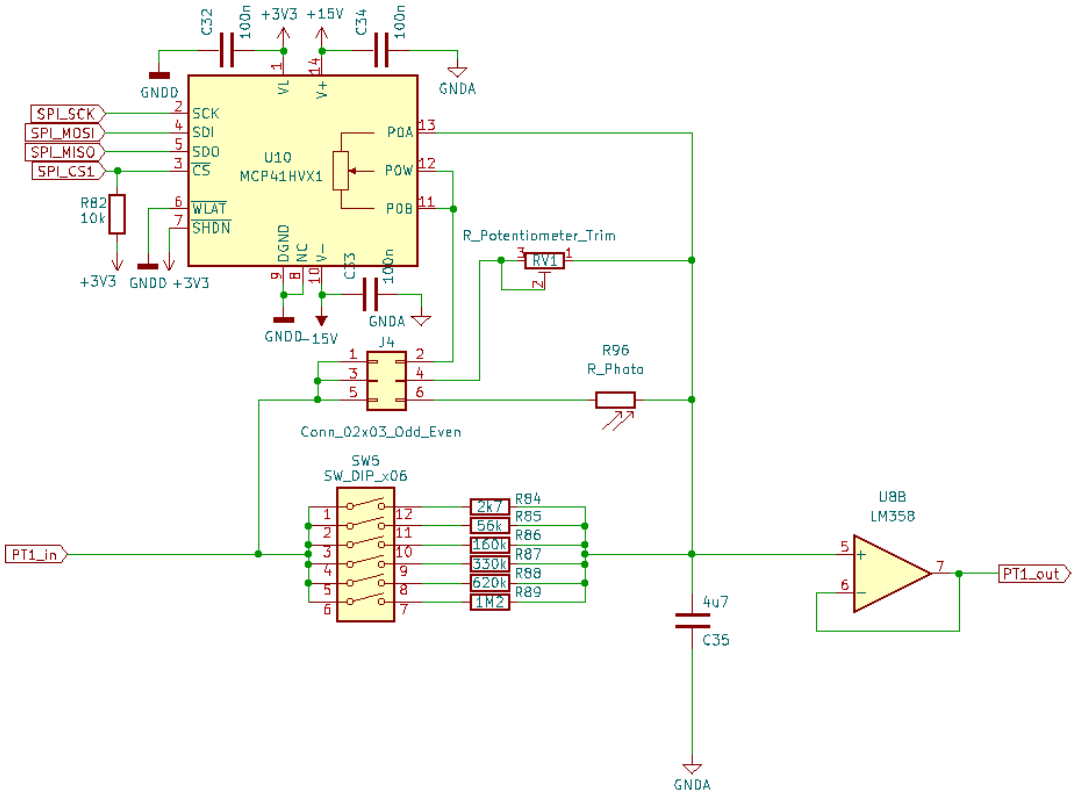
Přenos tohoto členu je

$$F(p) = \frac{1}{1 + \Delta RCp}, \tag{2.1}$$

kde ΔR je nastavitelný odpor pomocí DIP přepínače *SW5*.

Za tímto blokem, stejně jako za každým dalším blokem, je posazen operační zesilovač jako oddělovač, aby nedošlo k zatížení použité měřicí karty.

Ke zvolenému odporu pomocí DIP přepínače lze paralelně připojit buďto fotorezistor, otáčkový potenciometr nebo číslicový potenciometr. Tyto paralelně připojitelné proměnné odpory mají simulovat působení poruchy/nelinearity.



Obr. 2.6: Setrvačný člen prvního řádu

Setrvačný člen druhého řádu

Tento blok je tvořen zapojením se třemi operačními zesilovači pro realizaci přenosových funkcí druhého řádu s nastavitelnými parametry zesílení a tlumení. Odpovídající přenosová rovnice tohoto členu je 2.2. [15]

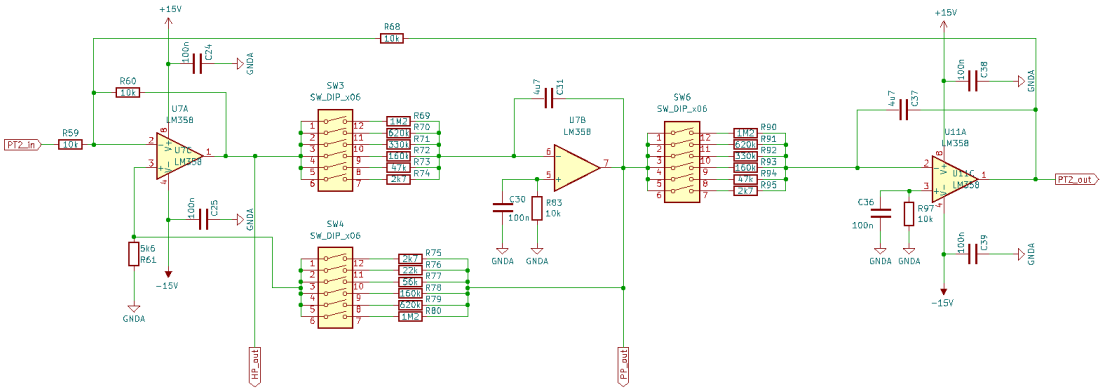
$$F(p) = -\frac{w^2}{p^2 + pw(2+m)d + w^2}, \tag{2.2}$$

kde $m = 1$ a

$$d = \frac{R_2}{R_2 + \Delta R_5}, \tag{2.3}$$

$$w^2 = \frac{1}{C \cdot \Delta R_4}, \tag{2.4}$$

kde ΔR je odpor nastavitelný pomocí DIP přepínačů. ΔR_5 je nastavován pomocí *SW4* ΔR_4 je nastavován pomocí *SW3* a *SW6*. Aby přenosová rovnice odpovídala tvaru 2.2 je nutné přepínače *SW3* a *SW6* nastavit na stejnou hodnotu.

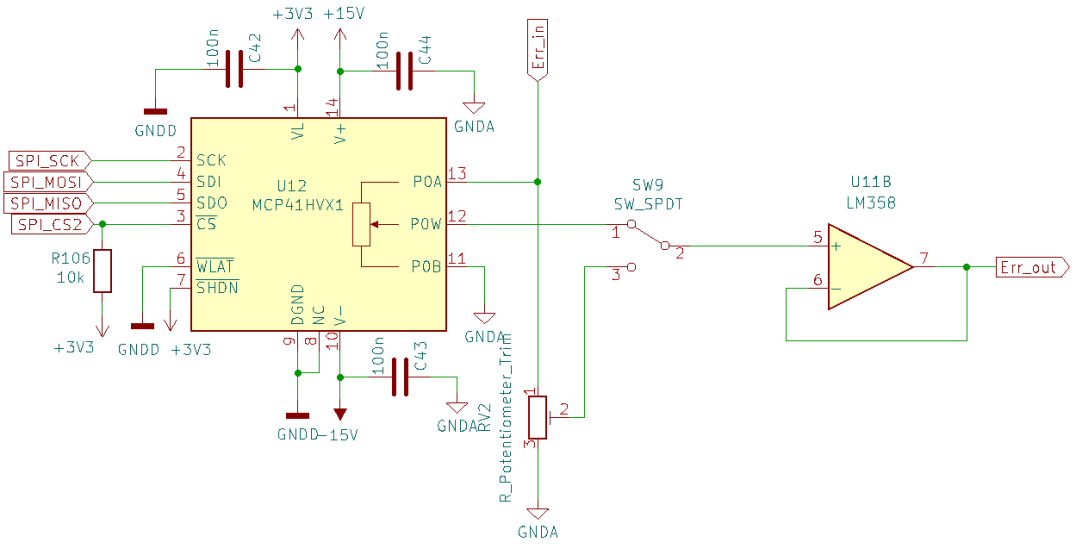


Obr. 2.7: Setrvačný člen druhého řádu [15]

Rozdílový člen

Tento člen simuluje poruchu působící na systém. Je tvořen jednoduše jako dělič za použití otáčecího potenciometru. Otáčením potenciometru je měněn dělicí poměr, a tak snižováno výstupní napětí.

Pomocí *SW9* je možné přepínat mezi manuálním potenciometrem a číslicovým potenciometrem jehož dělicí poměr je možné nastavit za pomoci mikroprocesorem.



Obr. 2.8: Rozdílový člen

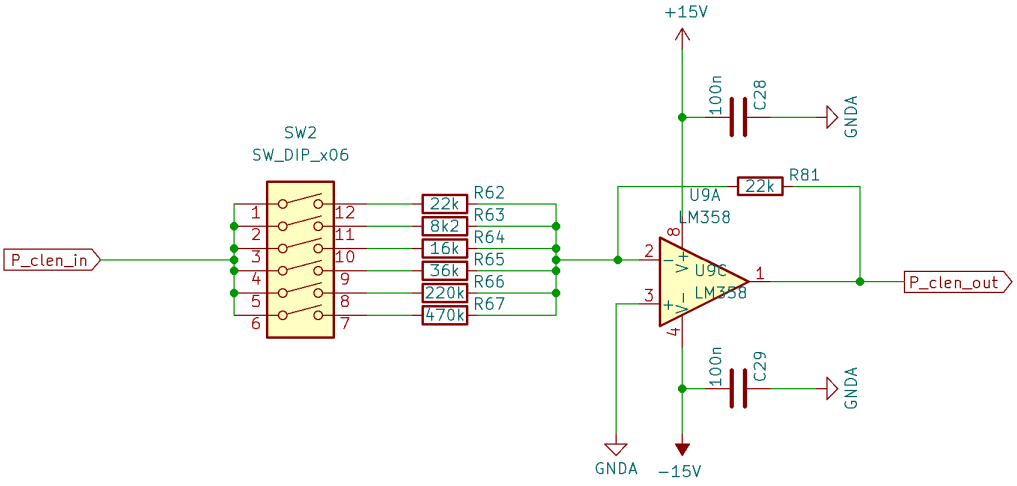
Proporcionální člen

Simuluje proporcionální systémy. Je tvořen pomocí invertujícího zapojení operačního zesilovače s nastavitelnými parametry. Zesílení může být nastaveno jako $K \doteq 1$, tudíž tento člen je možné použít i jako invertor signálu.

Odpovídající přenosová rovnice tohoto členu je

$$F = -\frac{R_2}{\Delta R_1}, \tag{2.5}$$

kde ΔR_1 je nastavován pomocí přepínače *SW2*



Obr. 2.9: Proporcionální člen

Integrační člen

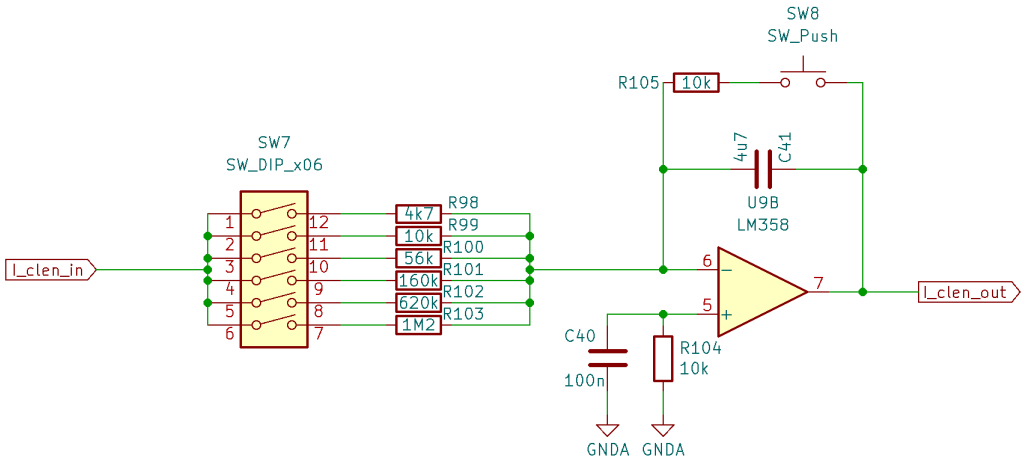
Posledním členem je integrátor s nastavitelnou rychlostí integrace. Pomocí tohoto členu je možné simulovat soustavy s astatismem.

Odpovídající přenosová rovnice tohoto členu je

$$F = \frac{1}{\Delta RCp}, \tag{2.6}$$

kde ΔR je nastavován pomocí přepínače $SW7$

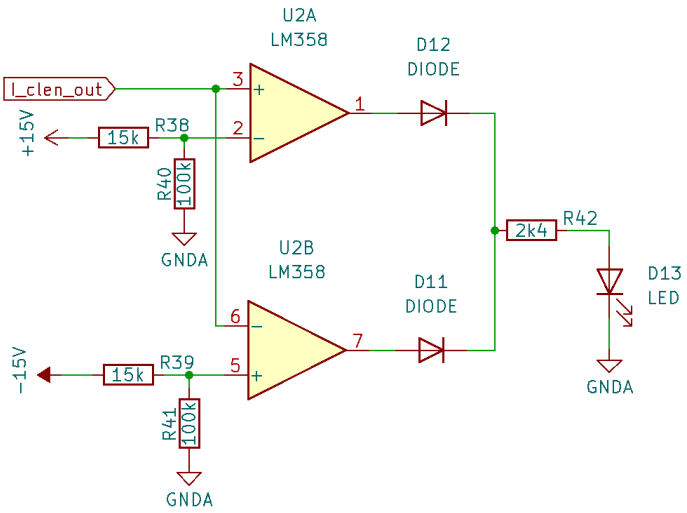
V kapitole 2.3 je uvedeno, že tento obvod po přivedení vstupního signálu drží na výstupu hodnotu naintegrovaného napětí, což komplikuje práci s platformou. Řešením tohoto problému je přidání vybíjecího obvodu do zpětné vazby operačního zesilovače. Tento obvod je tvořen tlačítkem a rezistorem o velikosti 10kΩ. Při použití kondenzátoru o kapacitě 4,7μF je doba pro dostatečné vybití kondenzátoru, zhruba 5τ, přibližně 0,25s.



Obr. 2.10: Integrační člen

V případě, že se obvod dostane do saturace, rozsvítí se led dioda umístěná na horní straně desky. Tato signalizace je řešena pomocí operačních zesilovačů v komparátorovém zapojení, které porovnávají napětí na výstupu integračního členu s napětím $\pm 13\,V$. Překročí-li napětí na výstupu integračního členu $U_{IclenOut} = \pm 13V$, dojde k překlopení komparátoru a k aktivaci led diody signalizující nasycení.

Nasycení může nastat také u členu simulujícího systémy druhého řádu, které signalizuje další led dioda.

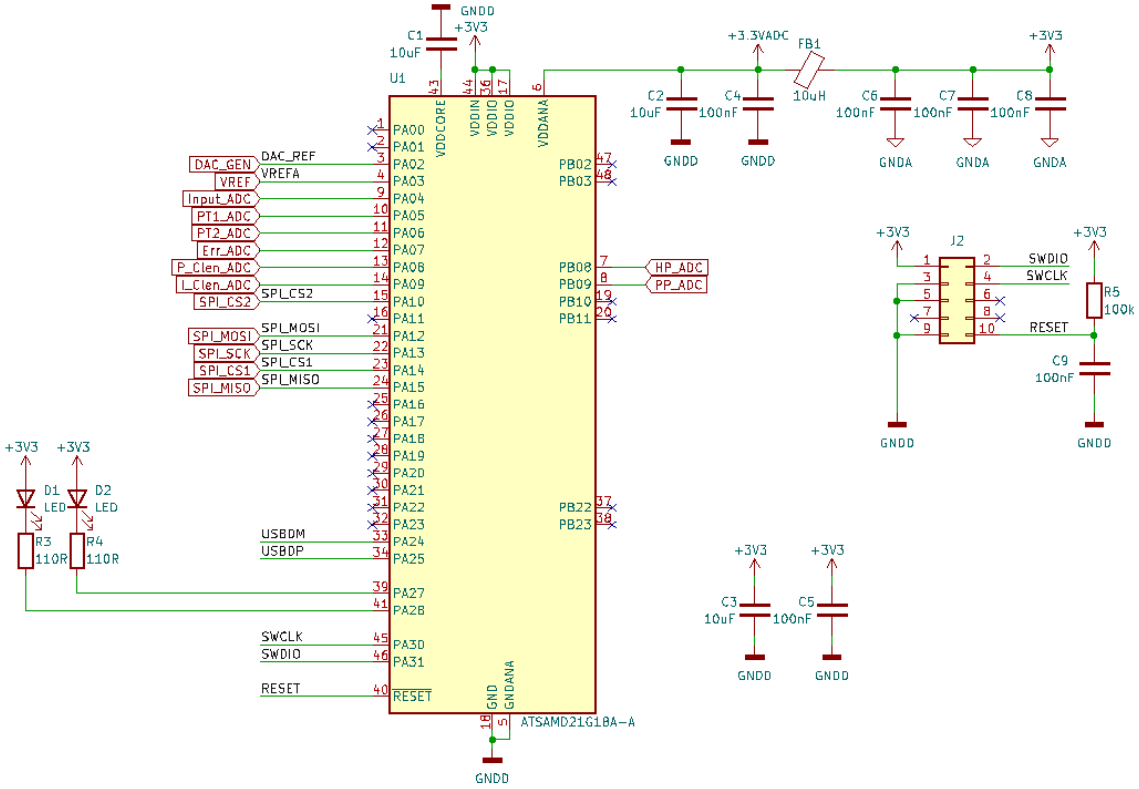


Obr. 2.11: Signalizace saturace

2.5.2 Digitální část

Použitím mikroprocesoru *ATSAMD21G18A*, schéma 2.12, je možné ovládat platformu částečně digitálně. Z mikroprocesoru je využit jeho vnitřní ADC s osmi analogovými vstupy pro měření výstupů za jednotlivými bloky, DAC pro generování vstupního signálu do analogové části a SPI pro komunikaci s číslicovými potenciometry.

Pro komunikaci s mikroprocesorem je využit USB-C a kolíkový programovací konektor.



Obr. 2.12: Zapojení mikroprocesoru

AD převodník

Mikroprocesor obsahuje 10bitový AD převodník s osmi vstupy. Maximální napětí na vstupu tohoto převodníku je $U_{DACmax} = 2,7V$. Jelikož výstupní napětí za jednotlivými bloky může dosahovat až $\pm 13{,}5\,V$, je toto napětí před vstupem do ADC převedeno na $0 \div 2V$. Tento převod je realizován pomocí zapojení 2.13.

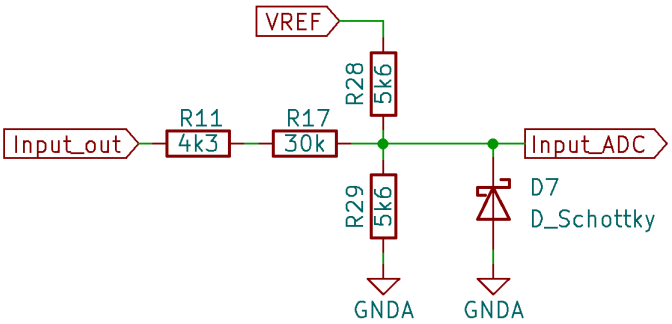
Toto zapojení je v podstatě sčítací obvod dvou přivedených napětí rozdělených v určitém poměru daném velikostí použitých rezistorů. Pro dělicí poměr vstupujícího referenčního napětí o hodnotě $U_{REF} = 2V$ platí

$$U_{inputADC} = U_{ref} \cdot \frac{4k8}{4k8 + 5k6} \doteq 1V. \tag{2.7}$$

Pro druhé napětí vstupující do tohoto zapojení, které dosahuje hodnot v rozsahu $\pm 13,5V$ je dělicí poměr dán

$$U_{inputADC} = U_{inputout} \cdot \frac{2k8}{2k8 + 34k3} = \pm 13,5 \cdot \frac{2k8}{37k1} \doteq \pm 1V. \tag{2.8}$$

Sečtením podělených napětí na $U_{inputout}$ a U_{REF} je výsledné napětí na výstupu v rozsahu přibližně $0 \div 2V$.



Obr. 2.13: Převod napětí pro vstup do ADC

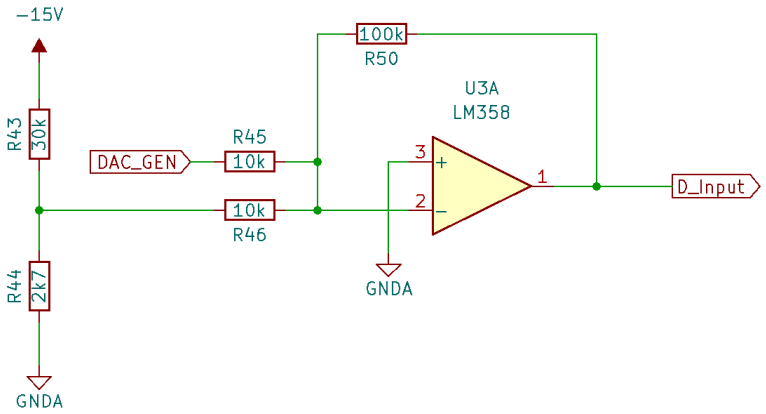
Hodnoty odporů v zapojení 2.13 jsou zvoleny tak, aby nedošlo k zatížení výstupu operačního zesilovače a zároveň nebyla vstupní impedance AD převodníku příliš vysoká. Celkový odpor tohoto zapojení je přibližně $2,6k\Omega$.

Při moc malém zatížení operačního zesilovače nedokáže výstupní napětí dosáhnout příliš vysoké hodnoty. Operační zesilovače použité pro realizaci této platformy mají při zatížení okolo $2k\Omega$ a použitém symetrickém napájení $\pm 15V$ rozkmit výstupního napětí přibližně $\pm 13,5V$.

Maximální vstupní impedance ADC je podle dokumentace mikroprocesoru $3,5k\Omega$. V případě, že je vstupní impedance příliš vysoká, je potřeba delší doba k nabití kondenzátoru na vstupu ADC a tím se snižuje schopnost měřit rychlé průběhy.

DA převodník

Mikroprocesor obsahuje také integrovaný desetibitový DA převodník, pomocí kterého lze vytvořit vstupní signál pro analogovou část. Napětí, které je možné pomocí převodníku generovat je v rozsahu $0 \div 2V$. Pro zvětšení tohoto rozsahu až na $\pm 10V$ je využito zapojení 2.14.



Obr. 2.14: Převod napětí pro vstup do analogové části platformy [15]

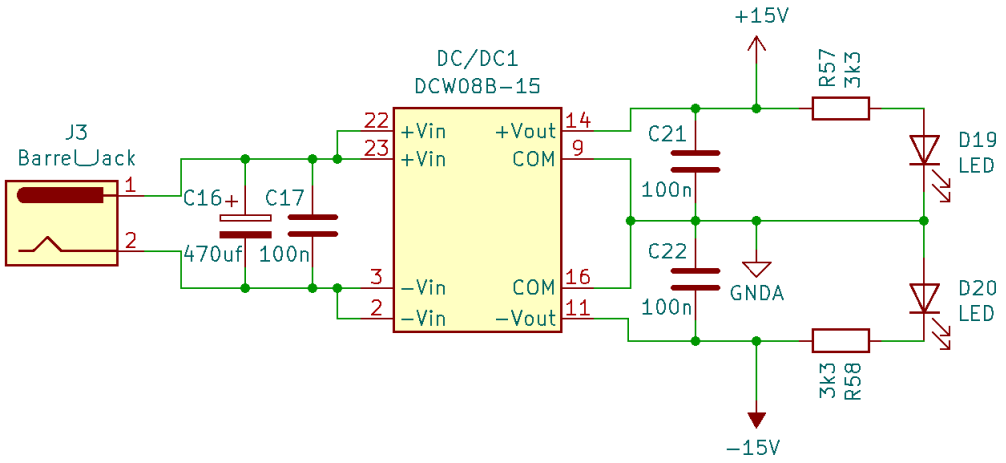
Signál generovaný DA převodníkem je zesílen pomocí součtového zesilovače tak, jak popisuje následující rovnice.

$$U_{Dinput} = \frac{R_{50}}{R_{45}} \left(U_{DACgen} + U_{REF} \right), \tag{2.9}$$

kde U_{Dinput} je napětí vstupující do analogové části, U_{DACGEN} je napětí generované pomocí mikroprocesoru a U_{REF} je referenční napětí odvozené z $-15V (U_{REF} = -1V)$.

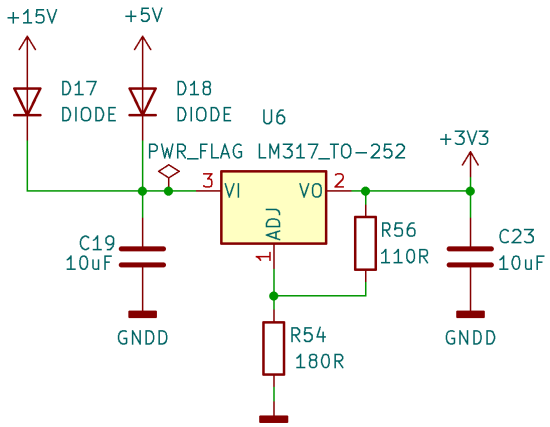
Napájení

Pro symetrické napájení operačních zesilovačů je použit stepdown stejnosměrný měnič *DCW08B-15*, který převádí vstupní napětí v rozsahu $18V \div 35V$ na výstupních $\pm 15V$ pro symetrické napájení operačních zesilovačů.



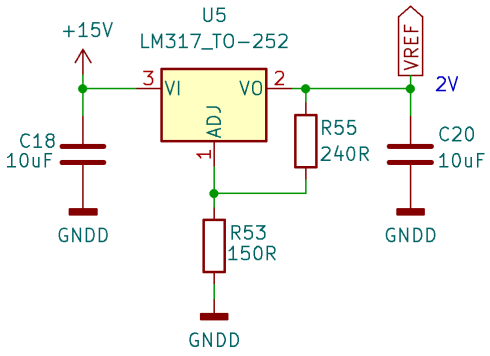
Obr. 2.15: Zapojení DC/DC měniče

Napájení digitální části zajišťuje stepdown měnič *LM317* převádějící +5 V přivedených přes *USB-C* nebo +15 V přivedených z *DCW08B-15*. Při takovémto zapojení je možné komunikovat s mikroprocesorem i když je odpojené napájení analogové části.



Obr. 2.16: Napájení digitální části

Integrovaný DA a AD převodník pro převod napětí využívají externí napěťovou referenci o velikosti 2 V vytvořenou pomocí LM317.



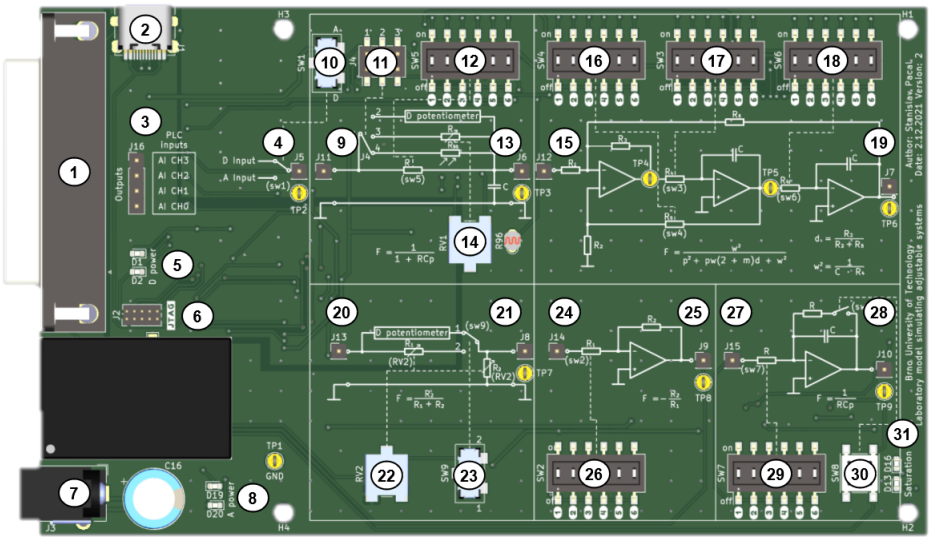
Obr. 2.17: Napěťová reference pro AD/DA převodník

2.6 Ovládání platformy

Jak lze vidět na obrázku 2.18, analogová část je rozdělena do bloků, které jsou graficky znázorněny na horní straně desky. V každém bloku je naznačeno jeho zapojení a odpovídající operátorový přenos.

Jednotlivé bloky je možné propojit přes kolíkové piny pomocí drátků. Tímto způsobem je možné kombinovat bloky v různém pořadí. Každý blok má na levé straně vstupní kolík a na pravé straně výstupní kolík.

V každém z bloků jsou umístěny mechanické součástky, pomocí kterých je možné nastavit proměnné parametry popsané v kapitole 2.5. Popis rozmístění těchto součástek na desce je znázorněn na obrázku 2.18 a jejich funkce blíže popsány v tabulce 2.1.



Obr. 2.18: Druhá verze platformy

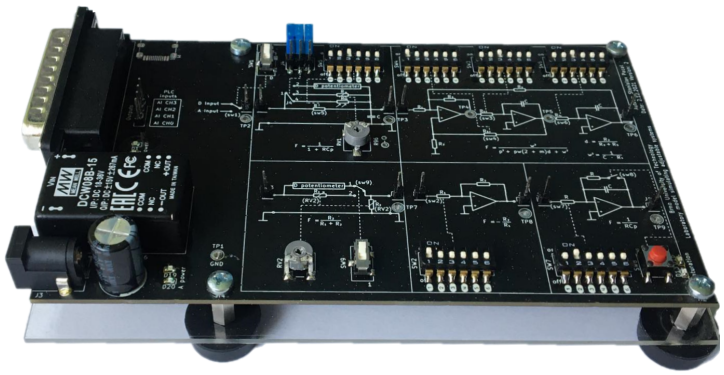
Tab. 2.1: Popis ovládacích prvků platformy

I/O část platformy	
1	D-SUB konektor sloužíci pro připojení platformy k měřicí kartě PLC
2	USB-C sloužící pro kumunikaci s mikroprocesorem
3	PIN header *J16* pro připojení výstupních signálů na napěťové měřicí kanály analogové karty *Ch0 - Ch3*
4	PIN *J5*, vstupní signál
5	LED signalizace od mikroprocesoru
6	Programovací konektor mikroprocesoru
7	Konektor pro připojení napájení v rozsahu $18V \div 35V$
8	LED signalizace připojení napájení
Setrvačný člen prvního řádu	
9	Vstupní PIN *J11* setrvčného členu prvního řádu
10	Switch *SW1* pro přepínání mezi vstupním signálem generovaným mikroprocesorem *D input* nebo generovaným pomocí PLC *A input*
11	PIN header *J4* pro možnost připojení paralelních komponentů termistor, otáčkový potenciometr nebo číslicový potenciometr
12	DIP switch *SW5* pro nastavení odporu v RC článku, pro bližší popis viz 2.6
13	Výstupní PIN *J6* setrvčného členu prvního řádu
14	Otáčkový potenciometr *RV1* připojitelný paralelně k odporu v RC článku pomocí konektoru *J4*
Setrvačný člen druhého řádu	
15	Vstupní PIN *J12* setrvčného členu druhého řádu
16	DIP switch *SW4* pro nastavení odporu R_5, pro bližší popis viz 2.2
17/18	DIP switch *SW3* a *SW6* pro nastavení odporu R_4, pro bližší popis viz 2.2
19	Výstupní PIN *J17* setrvčného členu druhého řádu
Rozdílový člen	
20	Vstupní PIN *J13* rozdílového členu
21	Výstupní PIN *J8* rozdílového členu

22	Otáčkový potenciometr *RV2* pro nastavení dělicího poměru
23	Switch *SW9* pro přepnutí mezi otáčkovým potenciometrem nebo číslicovým potenciometrem pro nastavení dělicího poměru
Proporcionální člen	
24	Vstupní PIN *J14* proporcionálního členu
25	Výstupní PIN *J9* proporcionálního členu
26	DIP switch *SW2* pro nastavení odporu R_1, pro bližší popis viz 2.5
Integrační člen	
27	Vstupní PIN *J15* Integračního členu
28	Vstupní PIN *J110* Integračního členu
29	DIP switch *SW7* pro nastavení odporu R, pro bližší popis viz 2.6
30	Tlačítko *SW8* pro vybití saturace integračního členu
31	LED signalizace dosažení saturace integračního členu LED *D13* a setrvačného členu druhého řádu LED *D16*

2.7 Ověření funkčnosti platformy

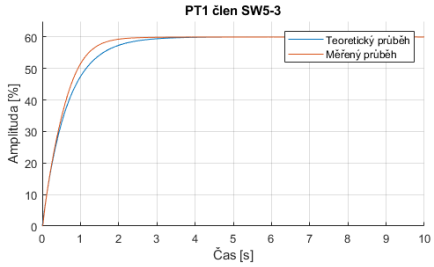
V této kapitole jsou zobrazeny některé naměřené průběhy na realizované platformě v porovnání s teoretickými průběhy vypočítanými podle uvedených přenosových rovnic porovnávaných bloků. Další průběhy měřené na realizované platformě zobrazené na obrázku 2.19 jsou uvedeny v příloze C.



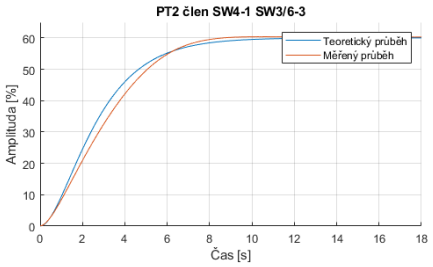
Obr. 2.19: Fotografie druhé verze platformy

V grafu 2.20 lze vidět naměřené průběhy na modulech setrvačného členu prvního a druhého řádu v porovnání s teoretickými průběhy vypočítanými podle hodnot

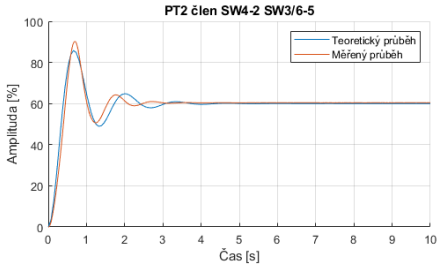
nastavených odporů. V popiscích grafů jsou uvedeny jednotlivé nastavení odporů pomocí DIP přepínačů. Tabulka s hodnotami odporů odpovídajícím zvolené pozici je vložena v příloze A.1.



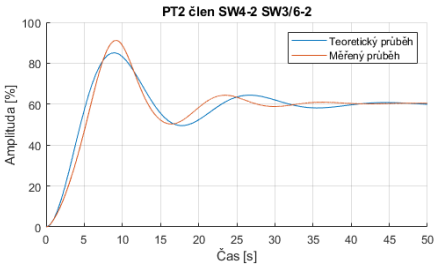
(a) PT1 člen SW5-3



(b) PT2 člen SW4-1 SW3/6-3



(c) PT2 člen SW4-2 SW3/6-5



(d) PT2 člen SW4-2 SW3/6-2

Obr. 2.20: Porovnání měřených vs. teoretických průběhů na realizované platformě

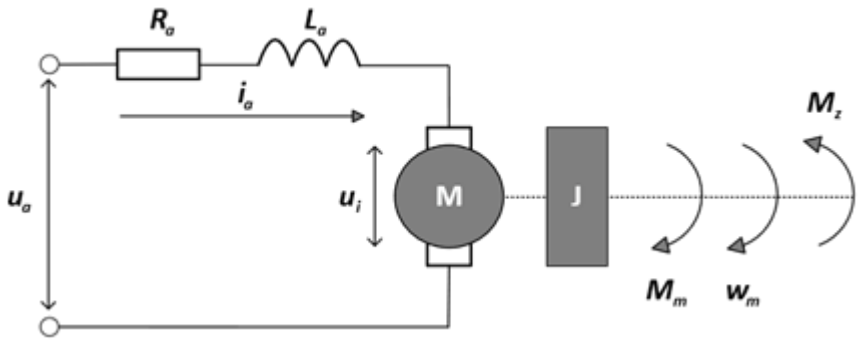
3 Návrh demonstrační úlohy

Cílem demonstrační úlohy je navrhnout regulátor pro řízení otáček fiktivního stejnosměrného motoru s permanentními magnety popsaného v následující podkapitole.

Dalším cílem návrhu demonstrační úlohy je ukázka funkčnosti navržené platformy a demonstrace automatického generování kódu pomocí PLC coderu.

3.1 Popis regulované soustavy

Předpokládejme, že máme stejnosměrný motor s permanentními magnety, jehož náhradní schéma je zobrazen na obrázku 3.1. Náhradní schéma se skládá z napětí kotvy motoru u_a, proudu kotvy motoru i_a, odporu vinutí kotvy R_a, indukčnosti vinutí kotvy L_a, generovaného napětí u_i, setrvačnosti motoru J, momentu působícího na hřídel vyvolaného napájecím napětím a zatěžovacího momentu M_m a M_z a nakonec úhlové rychlosti hřídele motoru ω_m. [16]



Obr. 3.1: Náhradní schéma stejnosměrného motoru [16]

Jako první je vhodné vyjádřit operátorový přenos motoru ve tvaru

$$F_{mot}(p) = \frac{\omega(p)}{U_a(p)}. \tag{3.1}$$

Dynamické chování motoru lze popsat diferenciálními rovnicemi rovnováhy napětí v obvodu kotvy [16]

$$u_a(t) = R_a \cdot i_a(t) + L_a \cdot \frac{di_a(t)}{dt} + u_i(t) \tag{3.2}$$

a rovnováhy momentů na hřídeli rotoru. [16]

$$M_m = M_z + J \cdot \frac{d\omega(t)}{dt} + B \cdot \omega(t), \tag{3.3}$$

kde B je viskózní tření motoru. Dále pro moment motoru a napětí indukované v kotvě motoru platí [16]

$$M_m = C\phi \cdot i_a(t), \tag{3.4}$$

$$u_i = C\phi \cdot \omega(t), \tag{3.5}$$

kde C je konstanta motoru a ϕ je magnetický tok buzení motoru. Dosazením rovnice 3.5 do rovnice 3.2 a následnou aplikací Laplaceovy transformace je získána rovnice v operátorovém tvaru

$$U_a(p) = R_a \cdot I_a(p) + p \cdot L_a \cdot I_a(p) + C\phi \cdot \omega(p). \tag{3.6}$$

Při zanedbání viskózního tření B a zatěžovacího momentu M_z je po aplikaci Laplaceovy transformace na diferenciální rovnici rovnováhy momentů na hřídeli rotoru získána rovnice v operátorovém tvaru

$$M_m = p \cdot J \cdot \omega(p). \tag{3.7}$$

Moment a indukované napětí v kotvě motoru v operátorovém tvaru

$$M_m = C\phi \cdot I_a(p), \tag{3.8}$$

$$U_i = C\phi \cdot \omega(p). \tag{3.9}$$

Vyjádřením $\frac{U_a(p)}{\omega(p)}$ z rovnice 3.6 je získána rovnice

$$\frac{U_a(p)}{\omega(p)} = \frac{R_a \cdot I_a(p)}{\omega(p)} + \frac{p \cdot L_a \cdot I_a(p)}{\omega(p)} + C\phi. \tag{3.10}$$

Dosazením rovnice 3.8 do rovnice 3.7, vyjádřením $I_a(p)$ a následným dosazením do rovnice 3.10 je získána rovnice

$$\frac{U_a(p)}{\omega(p)} = \frac{R_a \cdot p \cdot J \cdot \omega(p)}{C\phi \cdot \omega(p)} + \frac{p^2 \cdot L_a \cdot J \cdot \omega(p)}{C\phi \cdot \omega(p)} + C\phi. \tag{3.11}$$

Postupnými úpravami je obdržen výsledný přenos

$$F_{mot}(p) = \frac{\omega(p)}{U_a(p)} = \frac{1}{C\phi \cdot \left(\frac{R_a \cdot J \cdot p}{C\phi^2} + \frac{p^2 \cdot L_a \cdot J}{C\phi^2} + 1\right)}. \tag{3.12}$$

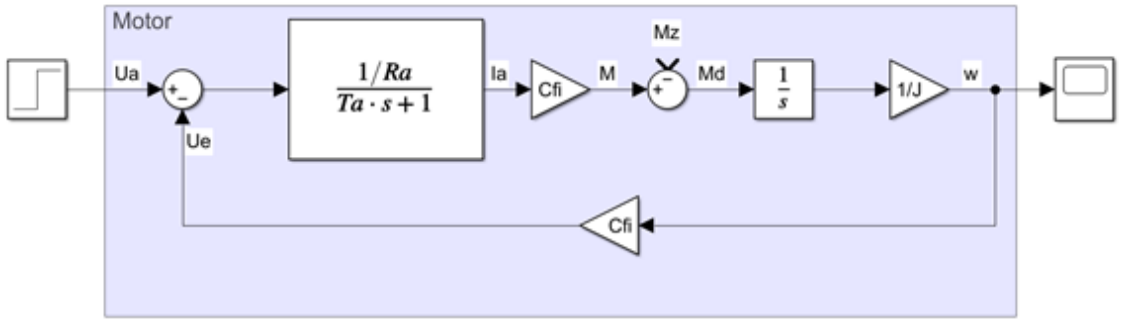
Tento přenos je možné přepsat do jiného tvaru vyjádřením elektromagnetické časové konstanty τ_a a elektromechanické časové konstanty τ_m. [17]

$$\tau_a = \frac{L_a}{R_a}, \tag{3.13}$$

$$\tau_m = \frac{R_a \cdot J}{C\phi^2}, \tag{3.14}$$

$$F_{mot}(p) = \frac{\omega(p)}{U_a(p)} = \frac{1}{C\phi \cdot (p^2 \cdot \tau_a \cdot \tau_m + p \cdot \tau_m + 1)}. \tag{3.15}$$

Odpovídající blokové schéma modelu stejnosměrného motoru vytvořené v nástroji *Simulink* programu *MATLAB* vypadá následovně.



Obr. 3.2: Blokové schéma stejnosměrného motoru

3.1.1 Postup při identifikaci

Jelikož regulovaný systém popsaný v kapitole 3.1 je pouze simulovaný na realizované platformě, není možné změřit parametry motoru jako takové, dosadit je do přenosové rovnice 3.12 a začít s návrhem regulátoru.

Nejdříve je nutné na platformě zvolit vhodný člen a nastavit jeho parametry tak, aby výsledný přenos zhruba odpovídal systému, který má být simulován. V rámci této demonstrační úlohy, kdy je simulován stejnosměrný motor s permanentními magnety, je na realizované platformě připojen setrvačný člen druhého řádu s proměnnými parametry odporů nastavenými tak, aby se jednalo o systém zhruba odpovídající systému modelovanému.

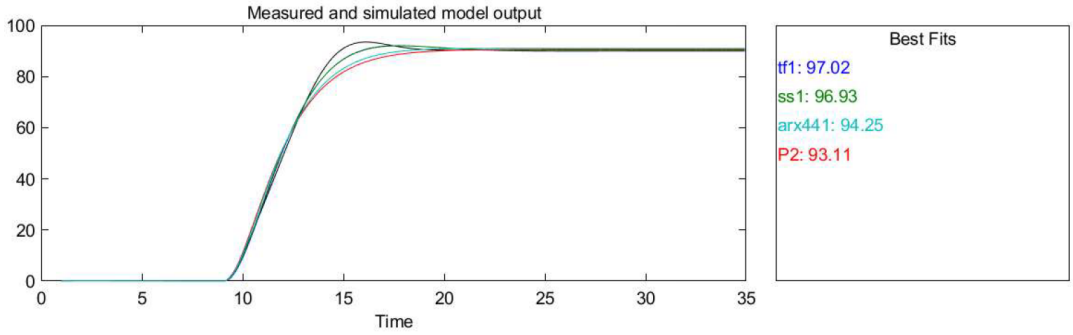
Tedy vstupní signál z pinu $J5$ je přiveden na vstup Setrvačného členu druhého řádu, pin $J12$, a nastavitelné odpory R_5 a R_4 jsou navoleny pomocí DIP switchů na polohy $SW4 \rightarrow poloha\ 1$ a $SW3/6 \rightarrow poloha\ 3$. Výstup tohoto členu, pin $J17$, bude oproti vstupnímu signálu invertovaný, je tudíž vhodné tento výstup připojit dále k proporcionálnímu členu, který je tvořen proporcionálním zesilovačem v invertujícím zapojení a nastavit jeho zesílení na hodnotu jedna.

Po nastavení zvoleného členu je nutné identifikovat jeho přenos. Platformu je možné připojit například k PLC a neměřit přechodovou charakteristiku zvoleného

systému na skokovou změnu vstupního signálu. K identifikaci lze využít *MATLAB System Identification Toolbox*, kapitola 1.5.

Pro identifikaci v toolboxu bylo naměřeno několik průběhů odezvy systému na různé hodnoty skoku vstupní veličiny. Tyto naměřené hodnoty byly v toolboxu pomocí funkce předzpracování *Merge experiments* spojeny dohromady a použity jako trénovací data. Pro odhad modelu byly poté použity metody *State space, Transfer function, Process model* a *ARX model*. Nejlepších výsledků podle kritéria *Best Fits* dosahovala metoda *Transfer function*, jak je možné vidět v grafu 3.3.

Výsledný identifikovaný přenos odpovídá přenosu v rovnici 3.17.



Obr. 3.3: Výsledek identifikace systému pomocí *System Identification Toolboxu*

Výsledný identifikovaný přenos je

$$F_s(p) = \frac{0,389}{p^2 + 1,088p + 0,389}. \tag{3.16}$$

3.2 Návrh regulátoru

Aby definovaný přenos modelu stejnosměrného motoru 3.15 odpovídal přenosu identifikované soustavy 3.16, je předpokládáno, že simulovaný motor má následující parametry.

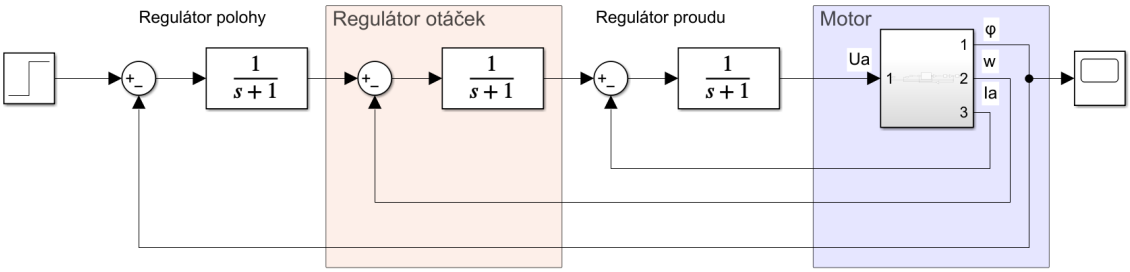
Tab. 3.1: Parametry motoru

Symbol	**Hodnota**	**Veličina**	**Popis**
R_a	0,5	Ω	odpor vinutí kotvy
L_a	0,46	mH	indukčnost vinutí kotvy
J	5,14	kg/m^2	setrvačnost motoru
$C\phi$	1	Nm/A	konstanta motoru

Vyčíslením přenosové rovnice stejnosměrného motoru je získán přenos

$$F_{mot}(p) = \frac{0,389}{p^2 + 1,088p + 0,389}. \tag{3.17}$$

Pokud by regulace probíhala na opravdovém stejnosměrném motoru, bylo by možné navrhnout kaskádní regulaci polohy, otáček a proudu tak, jak je naznačeno ve schématu 3.4.Jelikož na realizované platformě je simulován pouze jeden jediný průběh, v tomto případě odpovídající otáčkové odezvě na změnu vstupního napětí, bude navržen pouze otáčkový regulátor.



Obr. 3.4: Blokové schéma stejnosměrného motoru

Návrh PID regulátoru vychází z metody standardních tvarů charakteristického polynomu z hlediska integrálního kritéria ITAE. Standardní formy charakteristického polynomu závisí na čitateli přenosové funkce řízení. Jako první je tedy nutné stanovit si přenosovou funkci řízení.

Přenos PID regulátoru je ve tvaru

$$F_R(p) = k_p + \frac{k_i}{p} + K_d p = \frac{k_p p + k_i + k_d p^2}{p}. \tag{3.18}$$

Z rovnic 3.17 a 3.18 je určen přenos řízení jako

$$F_W(p) = \frac{F_R(p) \cdot F_{mot}(p)}{1 + F_R(p) \cdot F_{mot}(p)}. \tag{3.19}$$

Po úpravě je získán přenos

$$F_W(p) = \frac{p^3 + 1,088p^2 + 0,389p}{p^3 + p^2(1,088 + 0,389K_d) + p(0,389 + 0,389K_p) + 0,389K_i}. \tag{3.20}$$

Pro přenosovou funkci řízení se třemi koeficienty v čitateli jsou standardní charakteristiky polynomů ve tvaru zobrazeném v následující tabulce.

Tab. 3.2: Standardní tvary charakteristického polynomu jmenovatele přenosu uzavřené regulační smyčky [18]

Řád polynomu	**Standardní tvar charakteristického polynomu**
3	$p^3 + 2,97ap^2 + 4,94a^2p + a^3$
4	$p^4 + 3,71ap^3 + 7,88a^2p^2 + 5,93a^3p + a^4$
5	$p^5 + 3,81ap^4 + 9,94a^2p^3 + 13,44a^3p^2 + 7,36a^4p + a^5$

Porovnáním jmenovatele přenosové funkce řízení z rovnice 3.20 a standardního tvaru charakteristického polynomu z tabulky 3.2 jsou získány rovnice

$$1,088 + 0,389K_d = 2,97a, \tag{3.21}$$

$$0,389 + 0,389K_p = 4,94a^2, \tag{3.22}$$

$$0,389K_i = a^3. \tag{3.23}$$

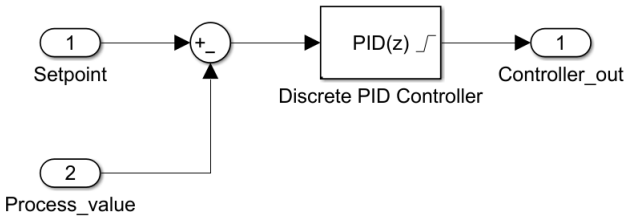
Pro vyhodnocení nejlepších parametrů PID regulátoru bylo využito ITAE kritérium zobrazené ve schématu v příloze B.1. Postupnou iterací parametru K_p a poté vyhodnocením nejmenší hodnoty integrálního kritéria byly získány ideální parametry regulátoru. Pomocí stejného postupu byly získány i parametry filtrace derivační složky a koeficient K_b zesilující zásah použité anti windup metody, viz schéma 3.7.

Výsledný přenos regulátoru je

$$F_R(p) = \frac{3,01p^2 + 8,2p + 5,51}{p}. \tag{3.24}$$

3.2.1 Implementace regulátoru v prostředí MATLAB/Simulink

Po získání ideálních parametrů PID regulátoru je v simulinku řidící člen implementován pomocí bloku *Discrete PID Controller*, který je jedním z knihovny bloků podporovaných PLC coderem.



Obr. 3.5: Schéma regulátoru vytvořeného pomocí bloku *Discrete PID Controller*

V bloku *Discrete PID Controller* je potřeba zvolit typ regulátoru jako PID a jeho časovou doménu jako *Discrete-time*. Pro *Continuous-time* by se nepodařilo vygenerovat kód. Dále je nutné zvolit formu regulátoru libovolně jako paralelní/ideální a doplnit parametry P, I a D.

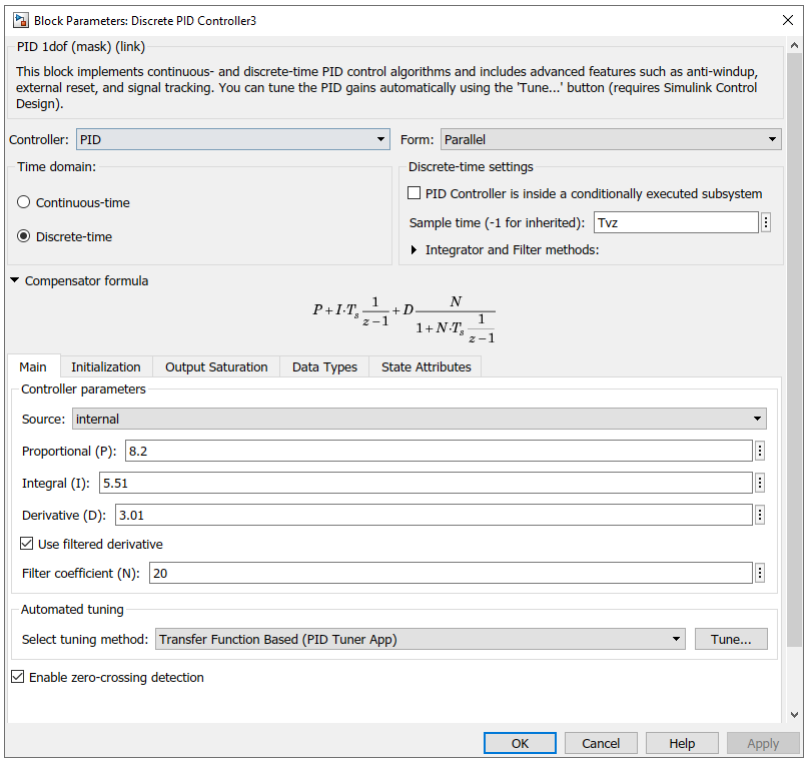
V této práci byla použita paralelní forma regulátoru. Diskrétní přenos regulátoru 3.24 v paralelní formě vypadá následovně

$$Kp + Ki \cdot \frac{T_s}{z-1} + K_d \cdot \frac{N}{1 + N \cdot \frac{T_s}{z-1}}, \tag{3.25}$$

kde $K_p = 8,2$, $K_i = 5,51$, $K_d = 3,01$, perioda vzorkování $T_s = 0,1$ a filtrační koeficient derivační složky $N = 20$.

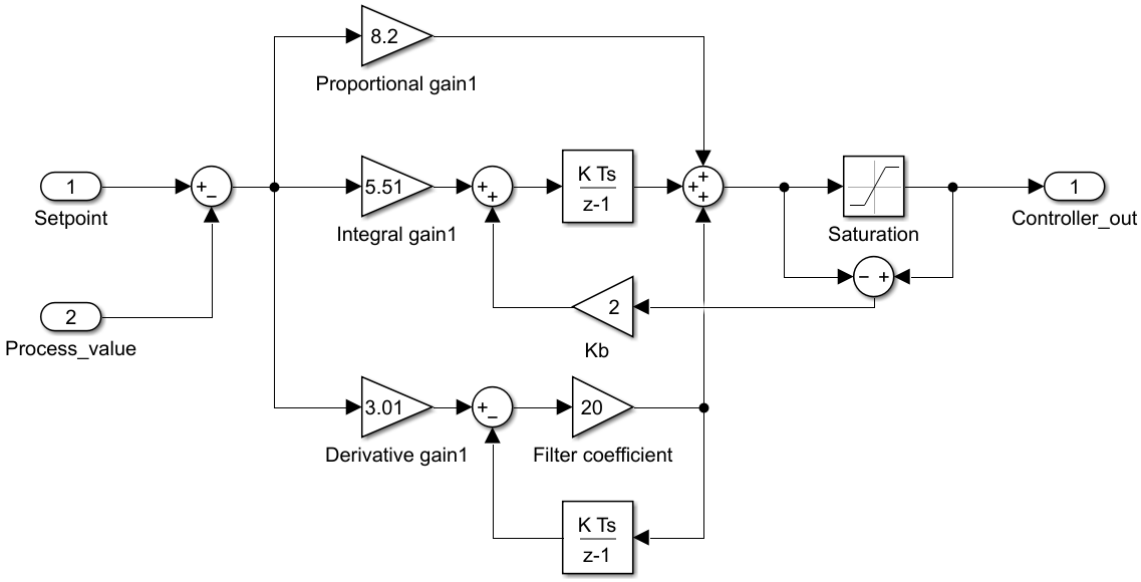
Při použití regulátoru s integrační složkou je vhodné použít i anti windup metodu pro umělé snížení regulační odchylky při saturaci akčního členu. Anti windup metodu je možné zvolit v záložce *Output Saturation* v bloku *Anti-windup* volbou metody výpočtu jako *back-calculation* nebo případně *clamping*.

Následně je možné vygenerovat kód pro zvolené IDE tak, jak je vysvětleno v kapitole 1.6. Vygenerovaný kód je zobrazen v kapitole 3.2.2.



Obr. 3.6: Nastavení parametrů bloku *Discrete PID Controller*

Regulátor spolu s použitou metodou anti windupu lze implementovat i pomocí základních bloků z knihovny podporované PLC coderem. Toto zapojení je zobrazeno ve schématu 3.7. Výsledný vygenerovaný kód pro takto zadrátovaný subsystém má stejnou strukturu jako kód generovaný s využitím bloku *Discrete PID Controller*, viz příloha B.1.



Obr. 3.7: Diskrétní PI regulátor sestavený ze základních bloků

3.2.2 Ukázka generovaného kódu

V této kapitole je zobrazen vygenerovaný kód pro subsystém s regulátorem vytvořeným pomocí bloku *Discrete PID Controller* a s rozborem jednotlivých části vygenerovaného kódu. Subsystém lze vidět ve schématu 3.5. Vygenerovaný kód odpovídá i zapojení ze schématu 3.7.

První řádky obsahují informace o vygenerovaném kódu, jako jsou například název simulinkového modelu, ze kterého byl kód vygenerován, verze modelu, poslední modifikace a další informace vygenerované mezi řádky 1 až 21.

```
1  (*
2  *
3  * File: DP_uloha_motor_ReSys.scl
4  *
5  * IEC 61131-3 Structured Text (ST) code generated for subsystem "
       DP_uloha_motor_ReSys/Controller"
6  *
7  * Model name                      : DP_uloha_motor_ReSys
8  * Model version                   : 1.5
9  * Model creator                   : SPacal
10 * Model last modified by          : SPacal
11 * Model last modified on          : Thu Mar 31 12:43:08 2022
12 * Model sample time               : 0.01s
13 * Subsystem name                  : DP_uloha_motor_ReSys/Controller
14 * Subsystem sample time           : 0.01s
15 * Simulink PLC Coder version      : 3.4 (R2021a) 14-Nov-2020
16 * ST code generated on            : Thu Mar 31 17:17:39 2022
17 *
18 * Target IDE selection            : Siemens TIA Portal: Double Precision
19 * Test Bench included             : No
```

```
*
*)
```

Následuje začátek funkčního bloku se stejným jménem jakým byl pojmenován subsystém v simulinkovém modelu.

```
FUNCTION_BLOCK Controller
```

Další část obsahuje deklarování vstupních a výstupních parametrů funkčního bloku.

```
VAR_INPUT
    ssMethodType: SINT;
    Setpoint: LREAL;
    Process_value: LREAL;
END_VAR
VAR_OUTPUT
    Controller_out: LREAL;
END_VAR
```

Dále deklarování vnitřních proměnných funkčního bloku.

```
VAR
    Integrator_DSTATE: LREAL;
    Filter_DSTATE: LREAL;
    rtb_Sum: LREAL;
    c_rtb_FilterCoeffici: LREAL;
    rtb_Sum_b: LREAL;
END_VAR
```

Nakonec následuje *CASE* struktura ovládaná vstupním parametrem *ssMethodType*, ve které dochází k inicializaci a implementaci regulátoru s anti windup metodou back calculation.

V této části kódu lze vidět, že před jednotlivými příkazy se nachází komentáře s výčtem bloků ze simulinku, ze kterých je daný příkaz složen. Díky tomuto odkazování na bloky použité v simulinku je kód velmi dobře čitelný.

```
CASE ssMethodType OF
    0:
        (* SystemInitialize for Atomic SubSystem: '<Root>/Controller1'
         *
         * Block description for '<Root>/Controller1':
         *  PID controller for DC motor *)
        (* InitializeConditions for DiscreteIntegrator: '<S34>/Integrator' *)
        Integrator_DSTATE := 0.0;
        (* InitializeConditions for DiscreteIntegrator: '<S29>/Filter' *)
        Filter_DSTATE := 0.0;
        (* End of SystemInitialize for SubSystem: '<Root>/Controller1' *)
    1:
        (* Outputs for Atomic SubSystem: '<Root>/Controller1'
         *
         * Block description for '<Root>/Controller1':
         *  PID controller for DC motor *)
        (* Sum: '<S1>/Sum' *)
        rtb_Sum := Setpoint - Process_value;
        (* Gain: '<S37>/Filter Coefficient' incorporates:
         *  DiscreteIntegrator: '<S29>/Filter'
```

```
56          *  Gain: '<S28>/Derivative Gain'
57          *  Sum: '<S29>/SumD' *)
58         c_rtb_FilterCoeffici := ((3.01 * rtb_Sum) - Filter_DSTATE) * 20.0;
59         (* Sum: '<S43>/Sum' incorporates:
60          *  DiscreteIntegrator: '<S34>/Integrator'
61          *  Gain: '<S39>/Proportional Gain' *)
62         rtb_Sum_b := ((8.2 * rtb_Sum) + Integrator_DSTATE) + c_rtb_FilterCoeffici;
63         (* Saturate: '<S41>/Saturation' *)
64         IF rtb_Sum_b >= 100.0 THEN
65             Controller_out := 100.0;
66         ELSIF rtb_Sum_b > 0.0 THEN
67             Controller_out := rtb_Sum_b;
68         ELSE
69             Controller_out := 0.0;
70         END_IF;
71         (* End of Saturate: '<S41>/Saturation' *)
72
73         (* Update for DiscreteIntegrator: '<S34>/Integrator' incorporates:
74          *  Gain: '<S27>/Kb'
75          *  Gain: '<S31>/Integral Gain'
76          *  Sum: '<S27>/SumI2'
77          *  Sum: '<S27>/SumI4' *)
78         Integrator_DSTATE := ((((Controller_out - rtb_Sum_b) * 2.0) + (5.51 *
    rtb_Sum)) * 0.01) + Integrator_DSTATE;
79         (* Update for DiscreteIntegrator: '<S29>/Filter' *)
80         Filter_DSTATE := (0.01 * c_rtb_FilterCoeffici) + Filter_DSTATE;
81         (* End of Outputs for SubSystem: '<Root>/Controller1' *)
82 END_CASE;
```

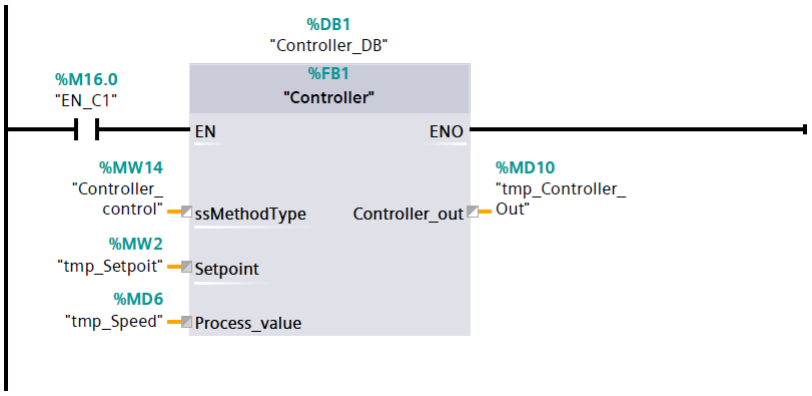
Posledním řádkem generovaného kódu je ukončení funkčního bloku.

```
68 END_FUNCTION_BLOCK
```

3.3 Výsledky regulace na realizované platformě

V této kapitole jsou zobrazeny výsledky regulace na realizované platformě pomocí kódu automaticky generovaného ze simulinku v porovnání s výsledky regulace za použití funkčních bloků *PID Compact* použitého *IDE TIA Portal*.

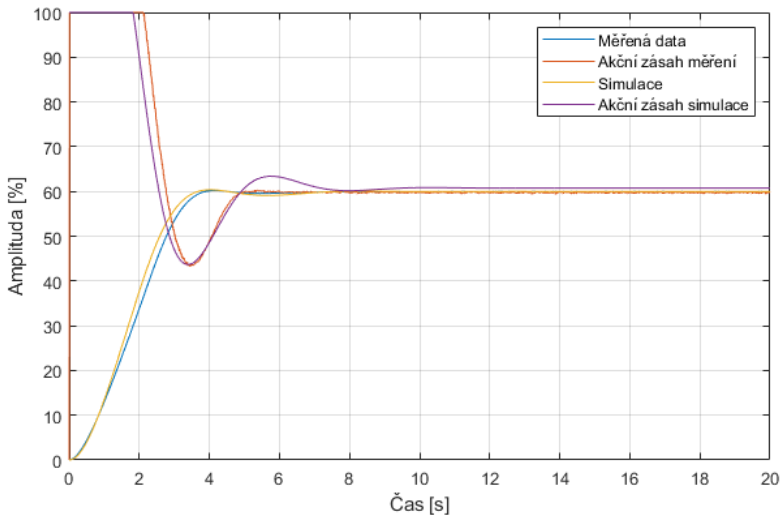
Po provedení implementace vygenerovaného kódu tak, jak je popsáno v kapitole 1.6.2, je funkční blok s vygenerovaným kódem umístěn do cyklicky volaného bloku OB30 s periodou volání stejnou jako je použita v simulinkové simulaci.



Obr. 3.8: Funkční blok vygenerovaného kódu pomocí *Simulink PLC Coderu*

K bloku *Controller* jsou jako vstupní proměnné přivedeny standardizovaná hodnota procesní veličiny, žádaná hodnota a proměnná pro ovládání vnitřní case struktury. Výstup tohoto funkčního bloku dále prochází destandardizací a na výstupu analogové karty je poté generováno napětí v rozsahu $0V \div 10V$, které je připojeno na vstup platformy.

V grafu 3.9 je zobrazena odezva regulované soustavy s PID regulátorem implementovaným pomocí automaticky generovaného kódu v porovnání s výsledkem regulace modelu v simulinku.



Obr. 3.9: Výsledek regulace pomocí generovaného kódu v porovnání s výsledkem regulace v simulinku

Jako další je vytvořen regulátor pomocí bloku *PID Compact*, jehož parametry jsou nastaveny stejným způsobem jako jsou nastaveny parametry PID regulátoru

v bloku *Discrete PID Controller* v simulinku.

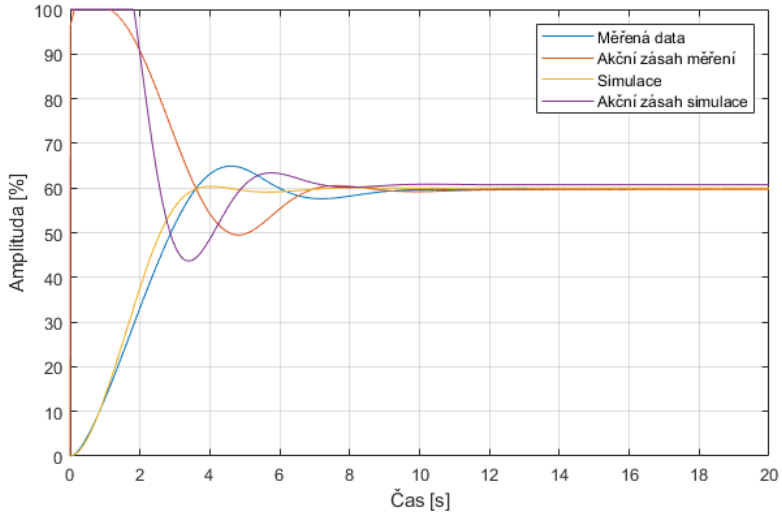
Algoritmus univerzálního PID regulátoru *PID Compact* lze popsat rovnící 3.26. Význam jednotlivých členů této rovnice je popsán v tabulce 3.3. Pro nastavení parametrů stejně jako v simulaci tedy stačí zvolit typ regulátoru jako PID a za proměnné a, b a c dosadit hodnotu jedna. Výsledný přenos poté bude odpovídat přenosu nakonfigurovanému v simulaci. [19]

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_i \cdot p} (w - x) + \frac{T_D \cdot p}{a \cdot T_D \cdot p + 1} (c \cdot w - x) \right]. \tag{3.26}$$

Tab. 3.3: Význam jednotlivých členů rovnice 3.26 [19]

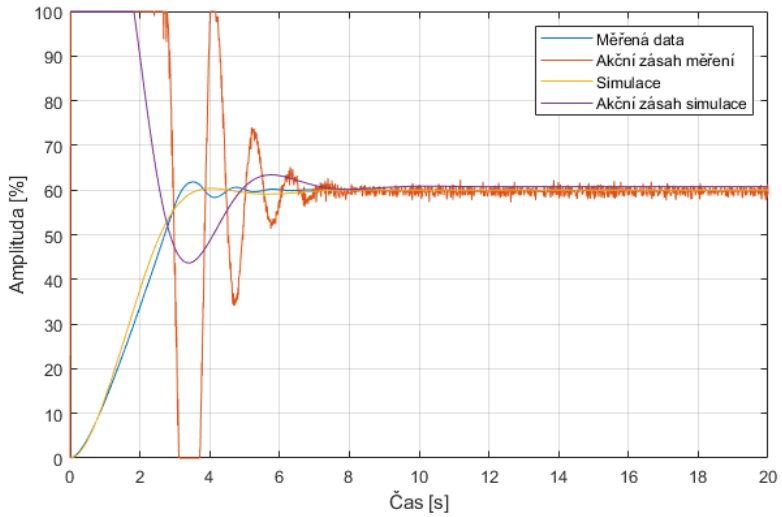
Symbol	**Popis**	**Symbol**	**Popis**
y	výstupní hodnota algoritmu	x	procesní hodnota
K_p	proporcionální zesílení	T_i	integrační časová konstanta
p	Laplaceův operátor	a	derivační koeficient zpoždění
b	váhování proporcionálního zásahu	T_D	derivační časová konstanta
w	žádaná hodnota	c	váhování derivačního zásahu

V grafu 3.10 je zobrazen výsledek regulace pomocí takto implementovaného regulátoru.



Obr. 3.10: Výsledek regulace s regulátorem *PID Compact* v porovnání s výsledkem regulace v simulinku

Poslední regulátor je implementován opět pomocí bloku *PID Compact*, ale tentokrát s využitím jeho funkcí pro automatické ladění a to *Pretuning* a *Fine tuning*. Blok *PID Compact* si pomocí těchto funkcí sám vyladí parametry PID regulátoru. Odezva regulovaného systému s tímto regulátorem v porovnání s výsledkem regulace v simulinku je zobrazena v grafu 3.11.



Obr. 3.11: Výsledek regulace se samonastaveným regulátorem *PID Compact* v porovnání s výsledkem regulace v simulinku

3.3.1 Zhodnocení výsledků regulace

V následující tabulce 3.4 je zhodnocena kvalita regulace pomocí integrálního kritéria ITAE, které je počítáno do 15s po spuštění regulace, velikosti překmitu, doby dosažení 95% žádané hodnoty a doby ustálení. Integrální kritérium ITAE je počítáno v PLC pomocí algoritmu generovaného *Simulink PLC Coderem* popsaného v kapitole 3.4. ITAE sestavené pomocí základních bloků podporovaných PLC coderem je zobrazeno v příloze ve schématu B.1 spolu s dalšími integrálními kriterii ISE, IAE a IE vytvořenými čistě pro ukázku generovaného kódu. Generovaný kód pro integrální kriteria lze vidět v příloze B.2.

Tab. 3.4: Zhodnocení kvality regulace jednotlivých regulátorů

	ITAE 15s	**Překmit [%]**	**t_r 95% [s]**	**Doba ustálení [s]**
PID Com	22164	8,16	3,35	9,47
PID Tun	12448	3,05	3,01	4,43
PID Gen	12984	0,37	3,3	3,7
PID Sim	11970	0.67	3,11	6,6

Z grafu 3.9 a z tabulky 3.4 je patrné, že průběhy regulace na reálné platformě s regulátorem implementovaným pomocí kódu generovaného PLC coderem a jeho ekvivalentem vytvořeným s využitím bloku *Discrete PID Controller* pro regulaci modelu v simulinku mají podle předpokladů téměř totožnou odezvu.

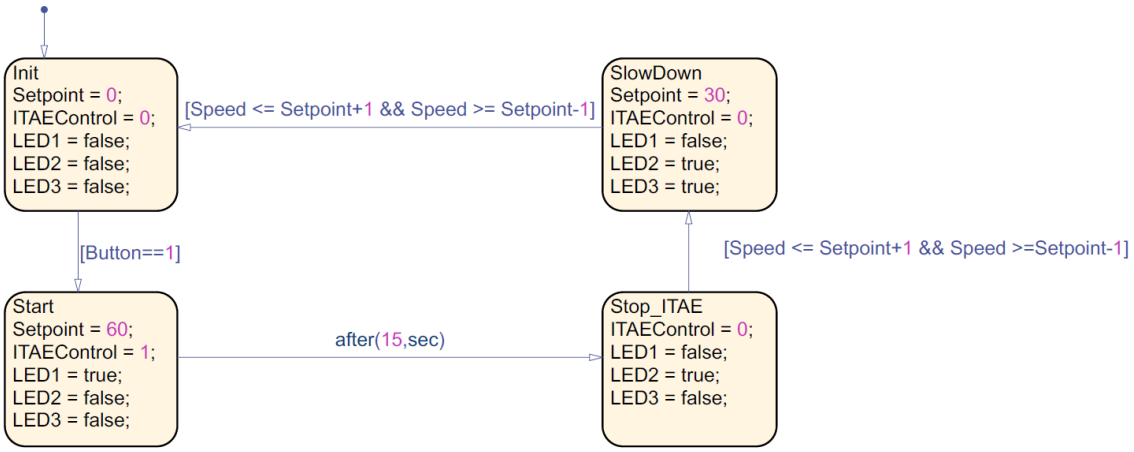
Regulátor implementovaný pomocí bloku *PID Compact* se stejnými parametry jako regulátory *PID Gen* a *PID Sim* má, jak je patrné z tabulky 3.4, daleko větší překmit než ostatní implementované regulátory. To je způsobeno, jak lze vidět v grafu 3.10, odlišným průběhem akčního zásahu.

Regulátor *PID Compact* naladěný pomocí systémových funkcí *Pretuning* a *Fine tuning* má nejrychlejší náběh, ale oproti ostatním implementovaným regulátorům o poznání kmitavější akční zásah, který není vhodný pro většinu akčních členů.

3.4 Stavový automat pro výpočet ITAE kritéria

Součástí implementované demonstrační úlohy je i stavový automat řídicí výpočet integrálního kritéria ITAE použitého pro zhodnocení kvality jednotlivých regulátorů.

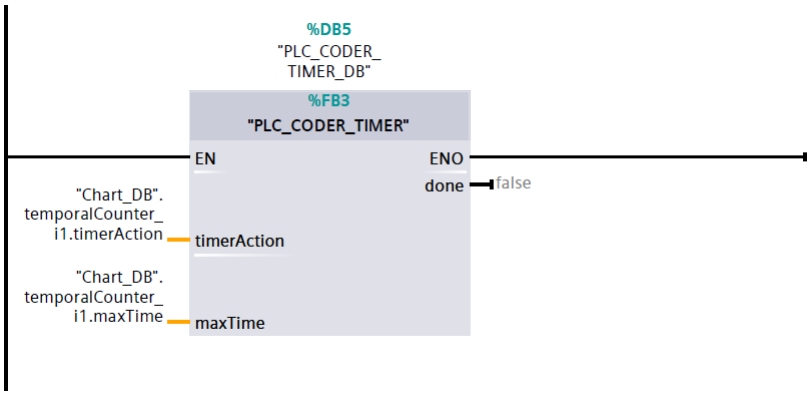
Stavový automat je navržen pomocí stateflow diagramu v prostředí *MATLAB/-Simulink* a skládá se ze čtyř stavů.



Obr. 3.12: Jednoduchý stavový automat pro výpočet ITAE kritéria

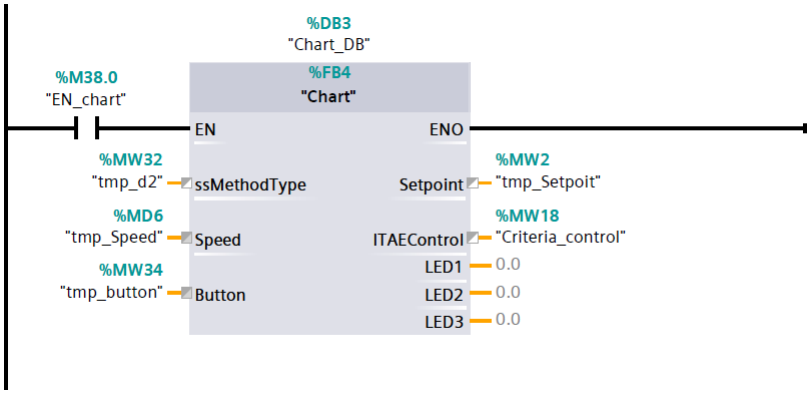
Po spuštění stavového automatu dojde v prvním stavu *Init* k inicializaci všech proměnných. Ve stavu *Init* se čeká na stisknutí tlačítka pro přechod do dalšího stavu. Ve druhém stavu *Start* dojde k nastavení žádané hodnoty a spuštění integrálního kritéria. Po patnácti vteřinách dojde k přechodu do třetího stavu *Stop ITAE*, ve kterém je zastaven výpočet integrálního kritéria. Pokud je procesní veličina stejná jako žádaná hodnota, dojde k přechodu do posledního stavu *SlowDown*, ve kterém je snížena žádaná hodnota a po jejím dosažení dojde k přechodu do stavu *Init*.

Implementovaný kód vygenerovaný pomocí tohoto stateflow diagramu se skládá ze dvou funkčních bloků zobrazených v 3.13 a 3.14. První obsahuje obsluhu časovače a druhý obsluhu jednotlivých stavů. Vygenerovaný kód je dostupný v přílohách.



Obr. 3.13: Funkční blok obsluhy časovače

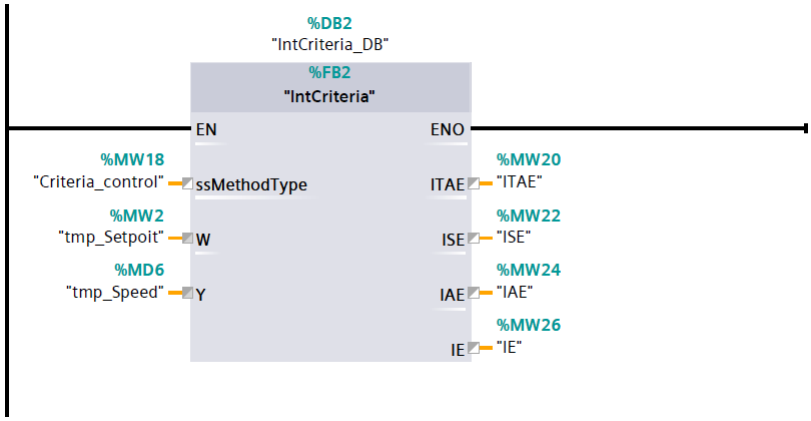
Vstupy bloku obsluhujícího časovač jsou vnitřní proměnné bloku *Chart*, nastavující časovač.



Obr. 3.14: Funkční blok obsluhy stavů

Vstupy bloku obsahujícího stavový automat jsou *ssMethodType* pro spuštění stavového automatu, vstup aktuální hodnoty procesní veličiny *Speed* a vstup *Button* pro přechod mezi stavem *Init* a *Start*. Výstupy tohoto bloku jsou *LED 1 ÷ 3* pro indikaci průběhu měření, výstup *ITAE Control* pro spouštění bloku *IntCriteria* a *Setpoint* pro nastavení žádané hodnoty.

Generovaný kód pro samotný výpočet integrálních kritérií je implementován v bloku *IntCriteria*



Obr. 3.15: Funkční blok výpočtu integrálních kritérií

Vstupem tohoto bloku je parametr *Criteria control* generovaný stavovým automatem, dále vstup žádané hodnoty *W* a aktuální hodnota procesní veličiny *Y*. Výstupy tohoto bloku jsou hodnoty integrálních kritérií *ITAE, ISE, IAE* a *IE*.

Celý program lze ovládat z HMI panelu, jehož vizualizace je zobrazena na následujícím obrázku.

Obr. 3.16: Vizualizace na HMI panelu pro výpočet integrálních kritérií

V levé části vizualizace uživatel nejdříve pomocí tlačítek *Reg 1 ÷ 3* vybere jeden ze tří implementovaných regulátorů. Po výběru regulátoru stisknuté tlačítko změní barvu. Zrušit výběr lze stisknutím tlačítka *Vypnout reg* a změnit výběr jednoduše stisknutím jiného tlačítka *Reg 1 ÷ 3*.

Po výběru regulátoru stisknutím tlačítka *Start* dojde ke spuštění měření a postupně probliknou všechny signalizační diody *LED 1 ÷ 3*. Stisknutím tlačítka *Průběh regulace* je zobrazena další obrazovka, na které je vykreslován graf aktuálního průběhu akčního zásahu a procesní veličiny. Po ukončení měření je v pravé části vizualizace zobrazena výsledná hodnota jednotlivých integrálních kritérií a lze zvolit další regulátor a provést další měření.

3.5 Zhodnocení Simulink PLC Coderu

Tato kapitola se zabývá zhodnocením možností a limitů automatického generování kódu pomocí *Simulink PLC Coderu*.

3.5.1 Generování kódu

Jak již bylo řečeno v kapitole 1.6, PLC Coder generuje nezávisle na hardwaru kód strukturovaného textu a ladder diagramu ze simulinkových modelů, stateflow diagramů a matlab funkcí podle standardu IEC 61131-3.V tabulce 3.5 jsou zobrazeny

všechny v současné době podporované IDE s datovými formáty, ve kterých je uložen výsledný vygenerovaný kód.

Tab. 3.5: Všechny v současné době PLC Coderem podporované IDE

Výrobce	**IDE**	**Datový formát**
3S-Smart Software Solutions	CoDeSys 2.3	.exp
	CoDeSys 3.3	.xml
	CoDeSys 3.5	.xml
B&R	Automation Studio IDE	.pkg, .xml, .st
Beckhoff	TwinCAT 2.11	.exp
	TwinCAT 3	.xml
KW-Software	MULTIPROG 5.0	.xml
Phoenix Contact	PC WORX 6.0	.xml
Rockwell Automation	RSLogix 5000	.L5X, .xml
Siemens	SIMATIC STEP 7 IDE	.scl, .asc
	TIA Portal IDE	.scl
Generic	(Čistě strukturovaný text, jestliže cílové IDE není podporováno)	.st
PLCopen XML	(Strukturovaný text používající PLCopen XML standard)	.xml
Rexroth	IndraWorks	.xml
OMRON	Sysmac Studio	.xml

Přestože všechny IDE podporují stejný standard, ne každé používá stejný formát pro generovaný kód a je tedy nutné při generování zvolit správné IDE.

Pro většinu dostupných IDE je v současné době nutné implementovat generovaný kód ručně. Například implementování generovaného kódu do IDE *TIA Portal* použitého v rámci této diplomové práce je popsáno v kapitole 1.6.2. Automatické generování a importování kódu do IDE je dostupné pouze pro *CoDeSys 2.3* společnosti *3S-Smart Software Solutions*.

Simulink PLC Coder má velmi zjednodušené uživatelské rozhraní, které umožňuje pouze minimální zásahy do procesu generování kódu, což je předpokládatelné

z důvodu určité univerzálnosti generovaného kódu. To jak bude generovaný kód efektivní záleží tedy na tom, jak dobře je daný model v simulinku zpracován.

Díky automatickému generování kódu v kombinaci s modelováním systémů pomocí simulinku lze proces vývoje zkrátit a zajistit rychlejší a efektivnější pracovní postup. Simulace umožňuje rychlejší odhalení chyb a tím dovoluje snížení nákladů na vývoj

3.5.2 Ověření generovaného kódu

Při automatickém generování kódu pro složitější modely zůstává otázka, je-li možné se na generovaný kód spolehnout. Automaticky generovaný kód je standard, tím pádem jsou vyloučeny chyby při psaní kódu programátorem, ale na druhé straně generovaný kód je generován strojem a může dojít k chybě.

Pro ověření funkčnosti generovaného kódu existují v podstatě tři možnosti. První z nich je kontrola generovaného kódu řádek po řádku. Druhou možností je testovaní funkcionality každého bloku zvlášť a třetí možností je využití statistického ověření založeného na simulaci.

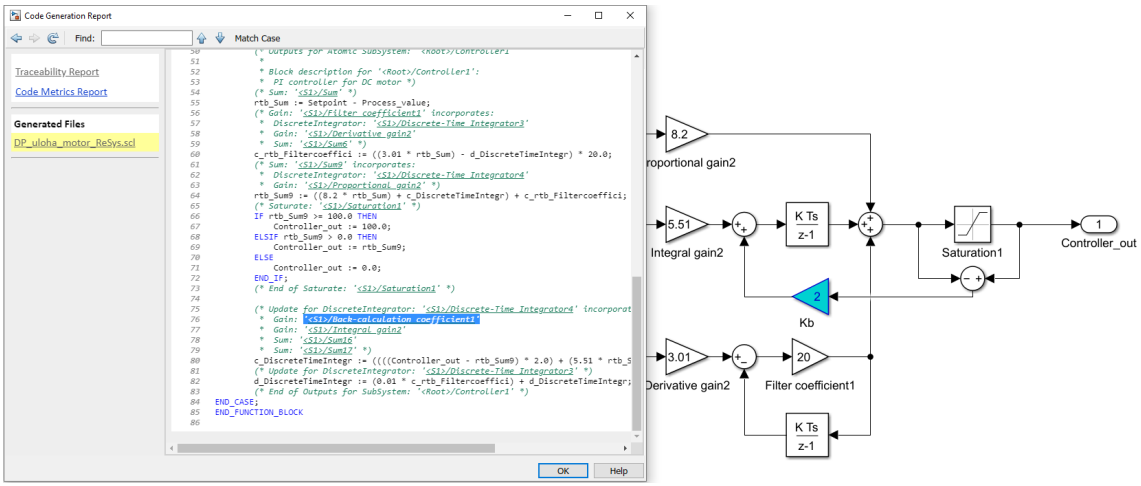
Simulink PLC Coder nabízí možnost generování testovacího kódu, tzv. *testbench*, který na základě numerických metod vyhodnotí, je-li generovaný kód statisticky srovnatelný se simulinkovým modelem. Způsob jakým je toto testování prováděno je popsán v kapitole 1.6.4.

Fáze testování je nedílnou součástí každého projektu. Je to finální fáze potvrzující dosažení požadovaných funkcionalit produktu. V některých případech může být tato fáze velmi nákladná a velmi časově náročná. Navíc důkladné testování nemusí být zcela proveditelné, pokud není dostupný všechen potřebný hardware. PLC coder tedy nabízí možnost, jak tuto fázi urychlit.

3.5.3 Automatické generování komentářů

Je velmi náročné porozumět jakémukoliv delšímu kódu, který neobsahuje komentáře autora. Pro zkvalitnění čitelnosti generovaného kódu nabízí *Simulink PLC Coder* několik možností pro automatické generování komentářů. PLC coder dokáže vytvořit krátké komentáře stručně popisující k jakému kroku dochází a jaké bloky modelu jsou v daném příkazu použity.

Komentáře obsahující bloky modelu, které jsou v daném příkazu použity, jsou reference, které po kliknutí odkazují na daný blok v simulinkovém schématu, jak lze vidět na obrázku 3.17. Tato funkce umožňuje opravdu velmi dobrou čitelnost kódu, ovšem bude-li k dispozici pouze generovaný kód bez příslušného simulinkového modelu, jsou generované komentáře fakticky nic neříkající.

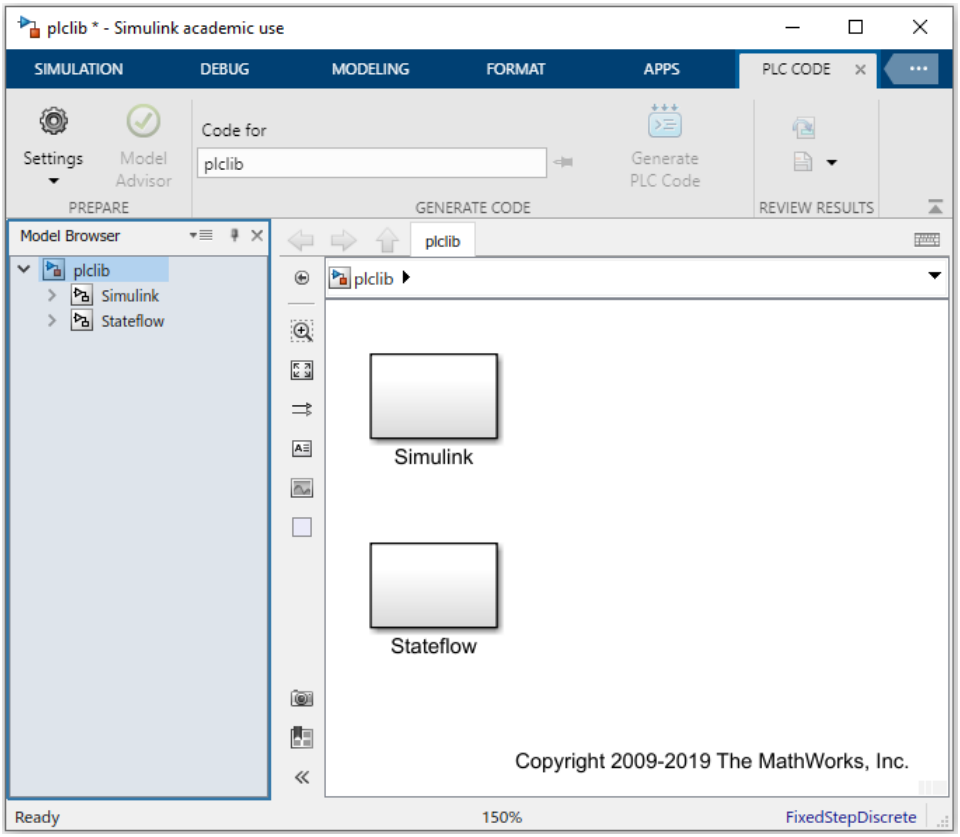


Obr. 3.17: Simulinková reference v generovaném kódu

3.5.4 Bloky použitelné pro generování kódu

Takzvaná trvalá omezení PLC Coderu většinou souvisí se strukturou jazyka strukturovaného textu. Tento jazyk běží na digitálním prostředí, tím pádem lze v simulinku používat pouze diskrétní bloky a nevirtuální subsystémy. Pro vytvoření modelu vhodného pro generování kódu stačí využívat knihovnu bloků *plclib*.

Knihovna *plclib* je soubor bloků, které jsou kompatibilní se *Simulink PLC Coderem*. Tato knihovna v dnešní době obsahuje více než 160 bloků pro tvorbu simulinkových modelů a všechny stateflow struktury pro tvorbu stavových automatů. V dalších verzích *Simulink PLC Coderu* lze očekávat nárůst počtu podporovaných bloků, tzv. *ready-made* bloků, což jsou komplexní bloky plnící komplexní funkce sestavené ze základních bloků.



Obr. 3.18: Knihovna bloků podporovaných *Simulink PLC Coderem*

Závěr

Cílem této diplomové práce bylo realizovat novou verzi laboratorní platformy vybavené analogovými obvody a doplnit nedostatky předchozí verze. Dalším cílem práce bylo prozkoumat možnostmi automatického generování kódu pro PLC s využitím *Simulink PLC Coderu*, seznámit se s nástrojem *System Identification Toolbox* a následně aplikovat tyto nástroje na úlohu řešící regulaci jednoduchého reálného systému.

V rámci této diplomové práce byla nejdříve realizována nová verze laboratorní platformy, viz obrázek 2.19. Naměřené průběhy na této platformě, které lze vidět v grafu 2.20, odpovídají teoretickým předpokladům. Nepřesnosti teoretických a měřených průběhů mohou být způsobeny až 10% tolerancí některých použitých součástek.

Dále byl v této diplomové práci navržen jednoduchý model reálného systému stejnosměrného motoru s permanentními magnety, pro který byl vytvořen simulinkový model s PID regulátorem. Celá úloha se skládá ze simulinkového modelu a *Stateflow* diagramu, pro které byl generován kód a implementován do použitého IDE *TIA Portal V15*. Výsledky regulace s využitím realizované platformy jsou uvedeny v grafech 3.9 až 3.11 a v tabulce 3.4. Z uvedeného grafu 3.9 lze vidět, že výsledek regulace simulinkového modelu odpovídá výsledku regulace na realizované platformě s regulátorem implementovaným pomocí generovaného kódu.

Z tabulky 3.4 vyplývá, že hodnota integrálního kritéria pro regulátor s generovaným kódem *PID Gen* a regulátor *PID Tun*, který je implementovaný pomocí bloku *PID Compact* a vyladěný s využitím systémových funkcí *Pretuning* a *Fine tuning* má téměř stejnou hodnotu. Ovšem v ostatních kritériích, jako je překmit, doba dosažení 95% žádané hodnoty a doba ustálení má lepší parametry regulátor implementovaný pomocí generovaného kódu.

V závěru poslední kapitoly je uvedeno zhodnocení použití nástroje *Simulink PLC Coder*.

V použité verzi R2021a *Simulink PLC Coder* nabízí přes 160 bloků pro tvorbu simulinkových modelů a podporuje všechny struktury pro tvorbu *Stateflow* diagramů. Dále podporuje velké množství IDE a v případě, že používané IDE coder nepodporuje, je možné generovat kód čistě do formátu .st nebo .xml.

S využitím možností optimalizací a generování automatických komentářů generuje *Simulink PLC Coder* velmi dobře čitelný a optimalizovaný kód, který je možné ihned implementovat do podporovaného IDE a po ověření funkčnosti, například pomocí automaticky generovaného ověřovacího kódu *testbench*, použít.

Literatura

[1] FIKAR, Miroslav a Ján MIKLEŠ. *Identifikácia systemov*. Skriptum. STU Bratislava, 2006. ISBN 80-227-1177-2.

[2] BALÁTĚ, Jaroslav. *Automatické řízení*. Praha: BEN-technická literatura, 2003. ISBN 80-730-0020-2.

[3] KAMLER, Radomír. *Identifikace spojitých soustav pomocí zvolených obrazových přenosů*. Liberec, 1996. Diplomová práce. Technická univerzita v Liberci.

[4] NOSKIEVIČ, Petr, Zora JANČÍKOVÁ a Jiří DAVID. *Modelování a identifikace systémů*: učební text. Ostrava: Montanex, 1999. ISBN 80-7225-030-2.

[5] VROŽINA, Milan, Zora JANČÍKOVÁ a Jiří DAVID. *Identifikace systémů*: učební text. Ostrava: Vysoká škola báňská - Technická univerzita, 2012. ISBN 978-80-248-2594-6.

[6] LJUNG, Lennart. *System identification: Theory for the user*. 2nd. Edition. Upper Saddle River, New Jersey 07458: Prentice Hall PTR, 1999. ISBN 0-13-656695-2.

[7] LJUNG, Lennart. *System Identification Toolbox: User's Guide* [online]. R2022. © 1994-2022 The MathWorks [cit: 2022-04-26]. Dostupné z: <`https://www.mathworks.com/help/ident/`>

[8] FRIEDMAN, Jerome, Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning*. 2001. Springer series in statistics New York.

[9] LATTIN, James M, J Douglas Carroll, and Paul E Green. *Analyzing multivariate data*. Thomson Brooks/Cole Pacific Grove, CA, 2003.

[10] *Simulink PLC Coder: User's Guide* [online]. R2022a. © 1994-2022 The MathWorks [cit. 2022-04-26]. Dostupné z: <`https://www.mathworks.com/help/plccoder/`>

[11] *Simulink: User's Guide* [online]. R2022a. © 1994-2022 The MathWorks [cit. 2022-04-26]. Dostupné z: <`https://www.mathworks.com/help/pdf_doc/simulink`>

[12] *SIEMENS. SIMATIC STEP 7 (TIA Portal) options Target 1500S™ for Simulink® V4.0 Update 2: Programing Manual* [online]. © Siemens AG 2016 - 2020., 2020 [cit. 2022-05-16]. Dostupné z: <`https://cache.industry.siemens.com/dl/files/754/109741754/att_921301/v1/s71500_target1500s_manual_en-US_en-US.pdf`>

[13] PACAL, Stanislav. *Realizace laboratorního modelu pro demonstraci řízení jednoduchých technologických procesů* [online]. Brno, 2019 [cit. 2019-05-20]. Dostupné z: <https://www.vutbr.cz/studenti/zav-prace/detail/119065>. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Miroslav Jirgl.

[14] ROZKOPAL, T. *Vliv topologie operačních zesilovačů na kvalitu audio signálu.* Brno, 2017. 76 s. Vedoucí diplomové práce doc. Ing. Jiří Háze, Ph.D. FEKT VUT v Brně

[15] PUNČOCHÁŘ, Josef. *Operační zesilovače v elektronice.* 5. vyd. Praha: BEN - technická literatura, 2002. ISBN 80-730-0059-8.

[16] OGATA, Katsuhiko. *System Dynamics.* 4th. ed. Upper Saddle River, c2004. ISBN 0-13-142462-9.

[17] ČERVINKA, Dalibor a Ivo PAZDERA. *Učební text do předmětu BEPB.* Brno, 2013. Učební text. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[18] SLOVÁK, Tomáš a Zdeněk RIEDL. *Syntéza: Metody syntézy* [online]. VŠB - Technická univerzita Ostrava FAKULTA STROJNÍ Katedra automatizační techniky a řízení, 2003 [cit. 2022-05-16]. Dostupné z: <http://books.fs.vsb.cz/Identifikace/>

[19] *SIEMENS. SIMATIC S7-1200, S7-1500 PID control* [online]. [cit. 2022-05-16]. Dostupné z: <https://cache.industry.siemens.com/dl/files/036/108210036/att_896030/v1/s71500_pid_control_function_manual_enUS_en-US.pdf>

Seznam příloh

A Doplnění popisu hardwaru realizovné platformy

Tab. A.1: Tabulka nastavitelných odporů na DIP přepínačích

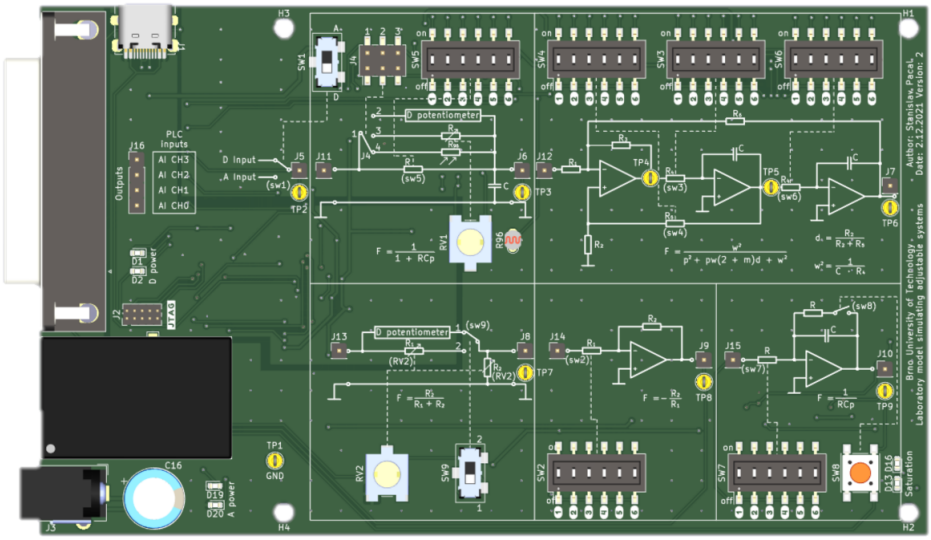
	SW2	**SW3**	**SW4**	**SW5**	**SW6**	**SW7**
1	$22k\Omega$	$1M2\Omega$	$2k7\Omega$	$2k7\Omega$	$1M2\Omega$	$4k7\Omega$
2	$8k2\Omega$	$620k\Omega$	$22k\Omega$	$56k\Omega$	$620k\Omega$	$10k\Omega$
3	$16k\Omega$	$330k\Omega$	$56k\Omega$	$160k\Omega$	$330k\Omega$	$56k\Omega$
4	$36k\Omega$	$160k\Omega$	$160k\Omega$	$330k\Omega$	$160k\Omega$	$160k\Omega$
5	$220k\Omega$	$47k\Omega$	$620k\Omega$	$620k\Omega$	$47k\Omega$	$620k\Omega$
6	$470k\Omega$	$2k7\Omega$	$1M2\Omega$	$1M2\Omega$	$2k7\Omega$	$1M2\Omega$

Maximální odpor použitých otáčkových potenciometrů *RV1* a *RV2* je $100k\Omega$. Maximální odpor použitých číslicových potenciometrů je $10k\Omega$.

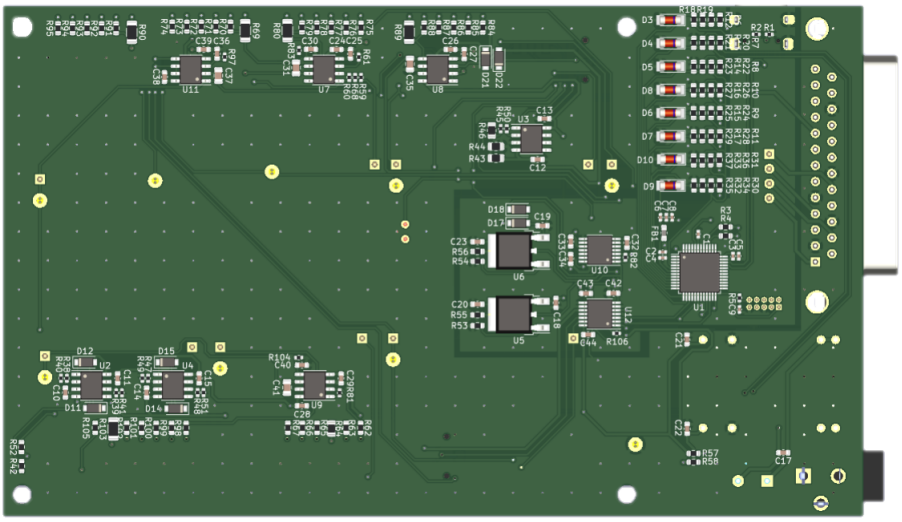
Tab. A.2: Zapojení plochého propojovacího kabelu k PLC

Svorky (karta)	**Piny (konektor)**	**Značení**	**Popis**
1	1	U_0+	Analogový vstup PLC
3	2	U_0-	Analogový vstup PLC
4	15	U_1+	Analogový vstup PLC
6	16	U_1-	Analogový vstup PLC
7	4	U_2+	Analogový vstup PLC
9	5	U_2-	Analogový vstup PLC
10	18	U_3+	Analogový vstup PLC
12	19	U_3-	Analogový vstup PLC
17	9	QV_0	Analogový výstup PLC
20	23	M_{ANA}	Analogový výstup PLC

A.1 Realizovaná platforma



Obr. A.1: Horní strana realizované platformy



Obr. A.2: Spodní strana realizované platformy

B Generovaný kód pomocí PLC Coderu

B.1 Generovaný kód pro subsystém ze schématu 3.7

```
(*
*
* File: DP_uloha_motor_ReSys.scl
*
* IEC 61131-3 Structured Text (ST) code generated for subsystem "
    DP_uloha_motor_ReSys/Controller"
*
* Model name                      : DP_uloha_motor_ReSys
* Model version                   : 1.15
* Model creator                   : SPacal
* Model last modified by          : SPacal
* Model last modified on          : Thu Apr 21 22:27:05 2022
* Model sample time               : 0.01s
* Subsystem name                  : DP_uloha_motor_ReSys/Controller
* Subsystem sample time           : 0.01s
* Simulink PLC Coder version      : 3.4 (R2021a) 14-Nov-2020
* ST code generated on            : Sat Apr 23 18:33:17 2022
*
* Target IDE selection            : Siemens TIA Portal: Double Precision
* Test Bench included             : No
*
*)
FUNCTION_BLOCK Controller
VAR_INPUT
    ssMethodType: SINT;
    Setpoint: LREAL;
    Process_value: LREAL;
END_VAR
VAR_OUTPUT
    Controller_out: LREAL;
END_VAR
VAR
    c_DiscreteTimeIntegr: LREAL;
    d_DiscreteTimeIntegr: LREAL;
    rtb_Sum: LREAL;
    c_rtb_Filtercoeffici: LREAL;
    rtb_Sum14: LREAL;
END_VAR
CASE ssMethodType OF
    0:
        (* SystemInitialize for Atomic SubSystem: '<Root>/Controller1'
         *
         * Block description for '<Root>/Controller1':
         *  PI controller for DC motor *)
        (* InitializeConditions for DiscreteIntegrator: '<S1>/Discrete-Time
    Integrator5' *)
        c_DiscreteTimeIntegr := 0.0;
        (* InitializeConditions for DiscreteIntegrator: '<S1>/Discrete-Time
    Integrator2' *)
        d_DiscreteTimeIntegr := 0.0;
        (* End of SystemInitialize for SubSystem: '<Root>/Controller1' *)
    1:
        (* Outputs for Atomic SubSystem: '<Root>/Controller1'
```

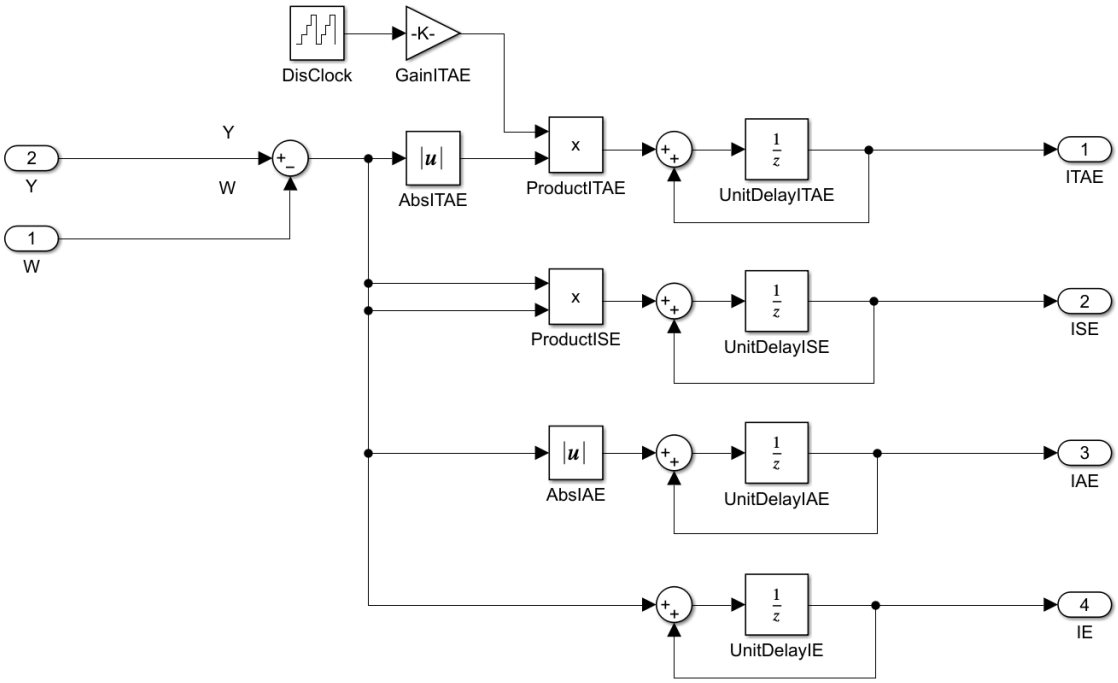
```
         *
         * Block description for '<Root>/Controller1':
         *  PI controller for DC motor *)
        (* Sum: '<S1>/Sum' *)
        rtb_Sum := Setpoint - Process_value;
        (* Gain: '<S1>/Filter coefficient' incorporates:
         *  DiscreteIntegrator: '<S1>/Discrete-Time Integrator2'
         *  Gain: '<S1>/Derivative gain1'
         *  Sum: '<S1>/Sum13' *)
        c_rtb_Filtercoeffici := ((3.01 * rtb_Sum) - d_DiscreteTimeIntegr) * 20.0;
        (* Sum: '<S1>/Sum14' incorporates:
         *  DiscreteIntegrator: '<S1>/Discrete-Time Integrator5'
         *  Gain: '<S1>/Proportional gain1' *)
        rtb_Sum14 := ((8.2 * rtb_Sum) + c_DiscreteTimeIntegr) +
    c_rtb_Filtercoeffici;
        (* Saturate: '<S1>/Saturation' *)
        IF rtb_Sum14 >= 100.0 THEN
            Controller_out := 100.0;
        ELSIF rtb_Sum14 > 0.0 THEN
            Controller_out := rtb_Sum14;
        ELSE
            Controller_out := 0.0;
        END_IF;
        (* End of Saturate: '<S1>/Saturation' *)

        (* Update for DiscreteIntegrator: '<S1>/Discrete-Time Integrator5'
    incorporates:
         *  Gain: '<S1>/Back-calculation coefficient'
         *  Gain: '<S1>/Integral gain1'
         *  Sum: '<S1>/Sum15'
         *  Sum: '<S1>/Sum5' *)
        c_DiscreteTimeIntegr := ((((Controller_out - rtb_Sum14) * 2.0) + (5.51 *
    rtb_Sum)) * 0.01) + c_DiscreteTimeIntegr;
        (* Update for DiscreteIntegrator: '<S1>/Discrete-Time Integrator2' *)
        d_DiscreteTimeIntegr := (0.01 * c_rtb_Filtercoeffici) +
    d_DiscreteTimeIntegr;
        (* End of Outputs for SubSystem: '<Root>/Controller1' *)
END_CASE;
END_FUNCTION_BLOCK
```

B.2 Generovaný kód pro subsystém ze schématu B.1

```
   (*
    *
    * File: DP_uloha_motor_ReSys.scl
    *
    * IEC 61131-3 Structured Text (ST) code generated for subsystem "
    DP_uloha_motor_ReSys/IntCriteria"
    *
    * Model name                      : DP_uloha_motor_ReSys
    * Model version                   : 1.15
    * Model creator                   : stani
    * Model last modified by          : stani
    * Model last modified on          : Thu Apr 21 22:27:05 2022
    * Model sample time               : 0.01s
    * Subsystem name                  : DP_uloha_motor_ReSys/IntCriteria
```

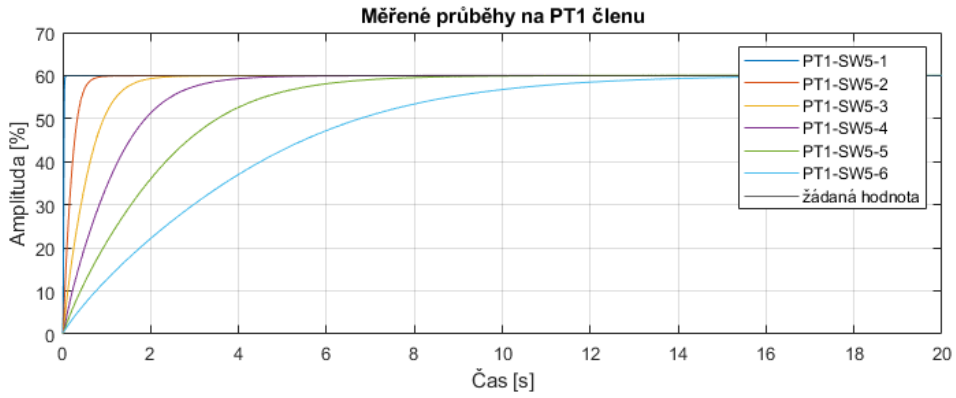
```
14      * Subsystem sample time           : 0.01s
15      * Simulink PLC Coder version      : 3.4 (R2021a) 14-Nov-2020
16      * ST code generated on            : Sat Apr 23 20:46:00 2022
17      *
18      * Target IDE selection            : Siemens TIA Portal: Double Precision
19      * Test Bench included             : No
20      *
21      *)
22  FUNCTION_BLOCK IntCriteria
23  VAR_INPUT
24      ssMethodType: SINT;
25      W: LREAL;
26      Y: LREAL;
27  END_VAR
28  VAR_OUTPUT
29      ITAE: LREAL;
30      ISE: LREAL;
31      IAE: LREAL;
32      IE: LREAL;
33  END_VAR
34  VAR
35      Output_DSTATE: UINT;
36      UnitDelayIE_DSTATE: LREAL;
37      UnitDelayIAE_DSTATE: LREAL;
38      UnitDelayISE_DSTATE: LREAL;
39      UnitDelayITAE_DSTATE: LREAL;
40      rtb_RegOdchylka: LREAL;
41      rtb_ProductITAE: LREAL;
42      rtb_ProductITAE_tmp: LREAL;
43  END_VAR
44  CASE ssMethodType OF
45      0:
46          (* SystemInitialize for Atomic SubSystem: '<Root>/IntCriteria' *)
47          (* InitializeConditions for UnitDelay: '<S2>/Output' *)
48          Output_DSTATE := 0;
49          (* InitializeConditions for UnitDelay: '<S1>/UnitDelayITAE' *)
50          UnitDelayITAE_DSTATE := 0.0;
51          (* InitializeConditions for UnitDelay: '<S1>/UnitDelayISE' *)
52          UnitDelayISE_DSTATE := 0.0;
53          (* InitializeConditions for UnitDelay: '<S1>/UnitDelayIAE' *)
54          UnitDelayIAE_DSTATE := 0.0;
55          (* InitializeConditions for UnitDelay: '<S1>/UnitDelayIE' *)
56          UnitDelayIE_DSTATE := 0.0;
57          (* End of SystemInitialize for SubSystem: '<Root>/IntCriteria' *)
58      1:
59          (* Outputs for Atomic SubSystem: '<Root>/IntCriteria' *)
60          (* Sum: '<S1>/RegOdchylka' *)
61          rtb_RegOdchylka := Y - W;
62          (* Abs: '<S1>/AbsITAE' incorporates:
63           *  Abs: '<S1>/AbsIAE' *)
64          rtb_ProductITAE_tmp := ABS(rtb_RegOdchylka);
65          (* Product: '<S1>/ProductITAE' incorporates:
66           *  Abs: '<S1>/AbsITAE'
67           *  Gain: '<S1>/GainITAE'
68           *  UnitDelay: '<S2>/Output' *)
69          rtb_ProductITAE := ((41943.0 * DINT_TO_LREAL(UINT_TO_DINT(Output_DSTATE)
    )) * 2.384185791015625E-7) * rtb_ProductITAE_tmp;
70          (* Outport: '<Root>/ITAE' incorporates:
71           *  UnitDelay: '<S1>/UnitDelayITAE' *)
```

```
72          ITAE := UnitDelayITAE_DSTATE;
73          (* Outport: '<Root>/ISE' incorporates:
74           *  UnitDelay: '<S1>/UnitDelayISE' *)
75          ISE := UnitDelayISE_DSTATE;
76          (* Outport: '<Root>/IAE' incorporates:
77           *  UnitDelay: '<S1>/UnitDelayIAE' *)
78          IAE := UnitDelayIAE_DSTATE;
79          (* Outport: '<Root>/IE' incorporates:
80           *  UnitDelay: '<S1>/UnitDelayIE' *)
81          IE := UnitDelayIE_DSTATE;
82          (* Update for UnitDelay: '<S2>/Output' incorporates:
83           *  Constant: '<S3>/FixPt Constant'
84           *  Sum: '<S3>/FixPt Sum1'
85           *  Switch: '<S4>/FixPt Switch' *)
86          Output_DSTATE := UDINT_TO_UINT(UINT_TO_UDINT(Output_DSTATE) + 1);
87          (* Update for UnitDelay: '<S1>/UnitDelayITAE' incorporates:
88           *  Sum: '<S1>/Sum' *)
89          UnitDelayITAE_DSTATE := rtb_ProductITAE + UnitDelayITAE_DSTATE;
90          (* Update for UnitDelay: '<S1>/UnitDelayISE' incorporates:
91           *  Product: '<S1>/ProductISE'
92           *  Sum: '<S1>/Sum2' *)
93          UnitDelayISE_DSTATE := (rtb_RegOdchylka * rtb_RegOdchylka) +
    UnitDelayISE_DSTATE;
94          (* Update for UnitDelay: '<S1>/UnitDelayIAE' incorporates:
95           *  Sum: '<S1>/Sum4' *)
96          UnitDelayIAE_DSTATE := rtb_ProductITAE_tmp + UnitDelayIAE_DSTATE;
97          (* Update for UnitDelay: '<S1>/UnitDelayIE' incorporates:
98           *  Sum: '<S1>/Sum6' *)
99          UnitDelayIE_DSTATE := rtb_RegOdchylka + UnitDelayIE_DSTATE;
100         (* End of Outputs for SubSystem: '<Root>/IntCriteria' *)
101     END_CASE;
102     END_FUNCTION_BLOCK
```

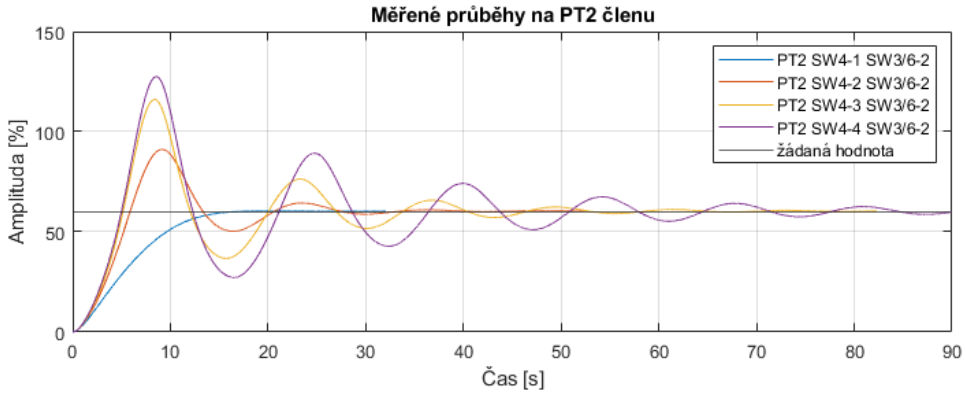


Obr. B.1: Integrální kriteria ITAE, ISE, IAE a IE

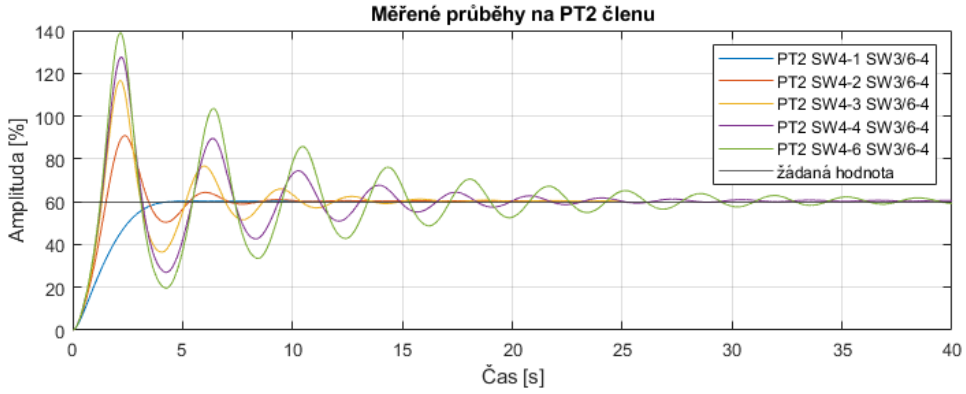
C Výsledky měření na realizované platformě



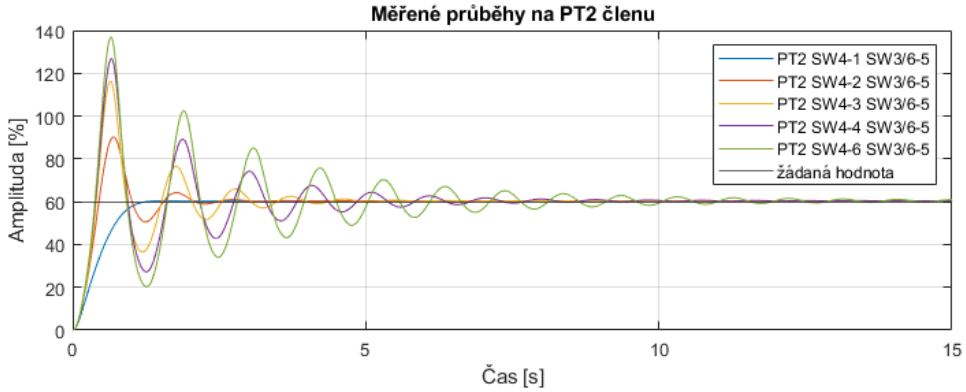
Obr. C.1: Měřené průběhy na setrvačném členu prvního řádu



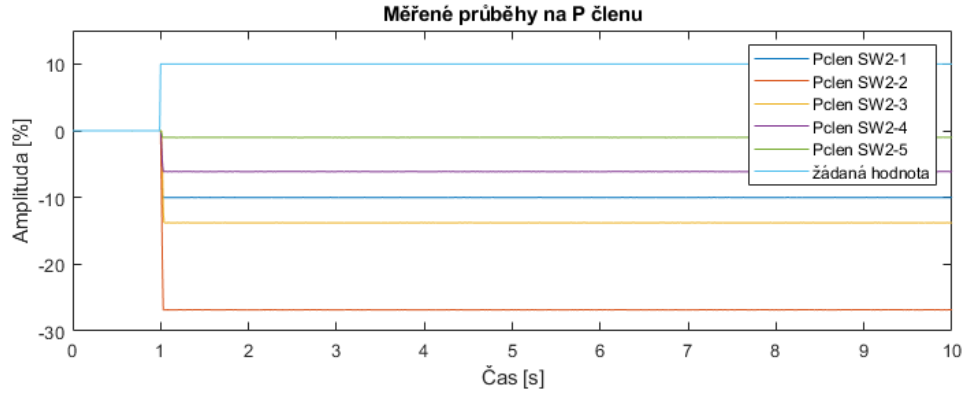
Obr. C.2: Měřené průběhy na setrvačném členu druhého řádu, první



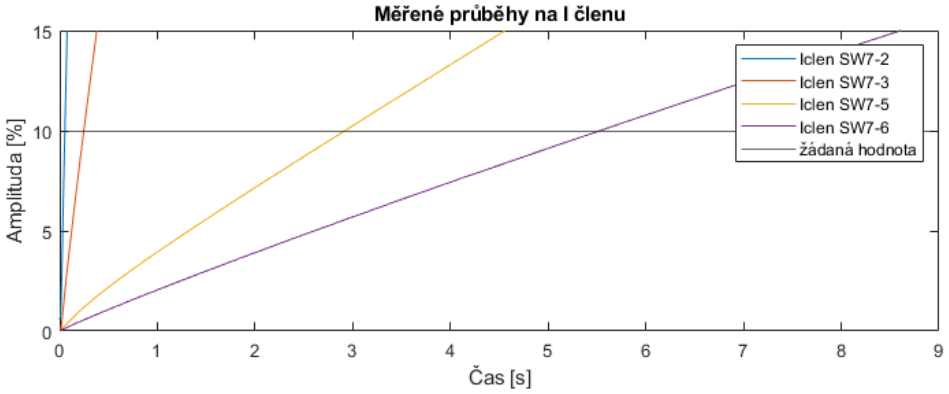
Obr. C.3: Měřené průběhy na setrvačném členu druhého řádu, druhé



Obr. C.4: Měřené průběhy na setrvačném členu druhého řádu, třetí



Obr. C.5: Měřené průběhy na proporcionálním členu



Obr. C.6: Měřené průběhy na integračním členu

D Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `Výsledky měření na platformě`
    - `Teoretické versus měřené průběhy`
    - `odezvy jednotlivých bloků na skok`
  - `Hardware` ..... deska je navržena v programu KiCad
  - `Matlab projekt` ..... použitá verze softwaru je MATLAB 2021a
    - `DP uloha motor ReSys chart.slx`
    - `DP uloha motor.m`
  - `TIA Portal projekt` ..... použitá verze softwaru je TIA Portal V15
  - `Generovaný kód` ..... kód generovaný pomocí Simulink PLC Coderu
    - `PID základní bloky`
    - `PID blok Discrete PID control`
    - `ITAE`
    - `Stateflow`
  - `Diplomová práce.pdf`