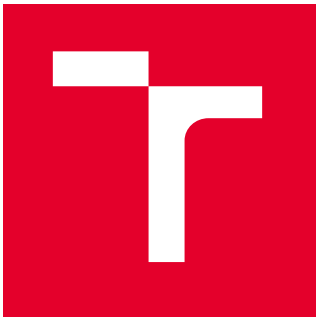


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

VYUŽITÍ NÁSTROJE SIMULINK PLC CODER PŘI ŘÍZENÍ REÁLNÉHO SYSTÉMU

A SYSTEM CONTROL USING SIMULINK PLC CODER TOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Viktor Musil**
AUTHOR

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

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Viktor Musil ***ID:*** 230139

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Využití nástroje Simulink PLC Coder při řízení reálného systému

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh a implementace demonstrační řídicí úlohy s reálnou soustavou a využitím automaticky generovaného kódu z prostředí MATLAB/Simulink.

1. Seznamte se s nástrojem pro automatické generování PLC kódu v prostředí MATLAB.
2. Seznamte se s platformou – simulátorem dynamických systémů, osaďte a oživte dodanou DPS.
3. Připojte simulátor na vstupy/výstupy PLC a změřte vybrané charakteristiky zadané soustavy.
4. Identifikujte model soustavy a navrhněte pro něj regulátor. Otestujte pomocí simulace.
5. Vygenerujte PLC kód navrženého regulátoru a implementujte do fyzického PLC.
6. Porovnejte dosažené výsledky se simulací a s výsledky získanými řízením soustavy využitím knihovních funkcí PLC.

DOPORUČENÁ LITERATURA:

[1] Get Started with Simulink PLC Coder [online]. MATHWORKS, 2022 [cit. 2022-09-06]. Dostupné z: https://www.mathworks.com/help/plccoder/getting-started-with-simulink-plc-coder.html

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 22.5.2023

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

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

ABSTRAKT

Práce se zabývá seznámením s nástrojem pro automatické generování PLC kódu v prostředí MATLAB a platformou pro simulování dynamických systémů. Práce obsahuje osazení a oživení platformy a následné simulovaní na platformě. Dále se zabývá propojením simulátoru s PLC a vytvoření regulační smyčky. Součástí práce je i návrh regulátoru s následným vygenerování PLC kódu. Výsledkem práce je řízení systému druhého řádu na PLC pomocí vygenerovaného kódu a knihovních funkcí PLC.

KLÍČOVÁ SLOVA

Simulink PLC Coder, Tia Portal, systém, simulátor, PLC

ABSTRACT

The thesis deals with the introduction to the tool for automatic generation of PLC code in MATLAB environment and the platform for simulation of dynamic systems. The work includes the installation and revival of the platform and then simulation on the platform. It also deals with connecting the simulator to the PLC and creating a regulation loop. The work also includes the design of the controller with subsequent generation of PLC code. The result of the work is the control of the second order system on the PLC using the generated code and PLC library functions.

KEYWORDS

Simulink PLC Coder, Tia Portal, system, simulator, PLC

MUSIL, Viktor. *Využití nástroje Simulink PLC Coder při řízení reálného systému*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 64 s. Bakalářská práce. Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Viktor Musil
VUT ID autora:	230139
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Využití nástroje Simulink PLC Coder při řízení reálného systému

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 bakalářské práce panu Ing. Miroslavu Jirglovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.
Dále bych rád poděkoval rodině za veškerou podporu a motivaci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cílem práce je se seznámit s nástrojem *Simulink PLC coder* a simulátorem dynamických systému, aby bylo možné využít a implementovat vygenerovaný kód do programu *Siemens TIA Portal*. Dalším cílem je osadit a oživit desku plošných spojů simulátoru. Následně propojit simulátor s PLC a změřit vybrané charakteristiky zvolené soustavy.

V první kapitole této bakalářské práce je teoretické seznámení s nástrojem *Simulink PLC coder* a postup pro vytvoření strukturovaného textu do fyzického PLC. Kapitola je ukončena teorií simulátoru dynamických systému. Zde je popsána analogová část simulátoru a členy, které jsou následně použity pro simulaci.

Druhá kapitola obsahuje postupné oživení simulátoru až po jeho propojením s PLC. Na simulátoru se navolí přenos a ten je simulován na fyzickém PLC. Přenos se změří a pomocí nástroje *System Identification Toolbox* je zpětně identifikován. V závěru kapitoly je kontrola funkčnosti simulátoru a oprava tlumení u systému druhého řádu.

Poslední kapitola se zabývá návrhem regulátoru metodou optimálního modulu s následnou implementací v Simulinku blokem *Discrete PID Controller*. Pro tenhle regulátor je generován PLC kód pomocí nástroje *Simulink PLC coder*. Dále je výsledný regulátor porovnán s knihovní funkcí PLC *PID compact* a dosažené výsledky jsou porovnány v závěru kapitoly.

1 PLC coder a simulátor dynamických systémů

Tahle kapitola reprezentuje teoretickou část bakalářské práce. Je zde seznámení s toolboxem PLC coder a platformou pro simulovaní dynamických systémů.

1.1 Simulink PLC coder

Simulink PLC Coder generuje hardwarově nezávislý strukturovaný text (ST - structured text) nebo žebříkový diagram (LD - ladder diagram) ze simulinkových modelů, funkcí MATLAB nebo diagramů stateflow. Dělá to podle normy IEC 61131-3. Žebříkové diagramy jsou generovány ve formátech souborů podporovaných *Rockwell Automation Studio 5000*. Strukturovaný text je generován v *PLCopen XML* a v dalších formátech souborů, které podporují široce používanými integrovanými vývojovými prostředími (IDE), včetně *3S-Smart Software Solutions CODESYS, Siemens TIA Portal* a další. Pomocí tohoto lze aplikaci zkompilovat a vložit do mnoha zařízení s programovatelnými logickými automaty (PLC) a programovatelnými automaty (PAC). [1]

PLC Coder umí generovat zkušební testy, které pomohou ověřit strukturovaný text a žebříkové diagramy pomocí IDE a simulačních nástrojů PLC a PAC. Poskytne poté zprávy o generování kódu se statickými metrikami kódu a obousměrnou sledovatelností mezi modelem a kódem. [1]

1.1.1 Příprava modelu pro generování ST

První pro přípravu na generování strukturovaného textu se musí nastavit určité parametry a zkontrolovat dané podmínky. Mezi to patří kontrola kompatibility modelu, nastavení solveru (řešitele) a určení, zda je model jedno nebo více rychlostní. [1]

Tasking mode

PLC coder pracuje pouze v *single-tasking* módu. To znamená, že subsystémy modelu pracují se stejnou periodou. Pokud jsou ale různé, jedná se o *multirate* mód. Jestli je tedy *multirate* model, musí se první explicitně nastavit *tasking mode* na *single-tasking*. [1]

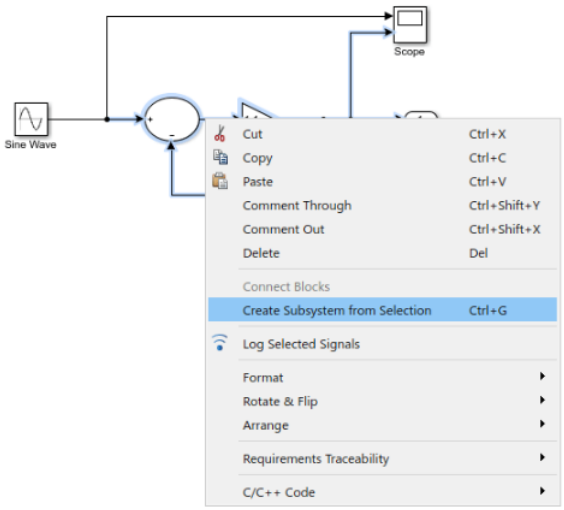
Výběr kroku výpočtu

Je na výběr ze dvou modelů: *Variable-step* (proměnný krok) ten optimalizuje velikost kroků na základě změn proměnných a *Fixed-step* (fixní krok), který řeší model ve fixních intervalech. Udělá to tak, že pokud se proměnné mění rychle, velikost

kroků se zjemní. Když se proměnné mění pomalu, tak zase naopak. Z důvodu lepší optimalizace je lepší použít mód *Variable-step*. [1] [2]

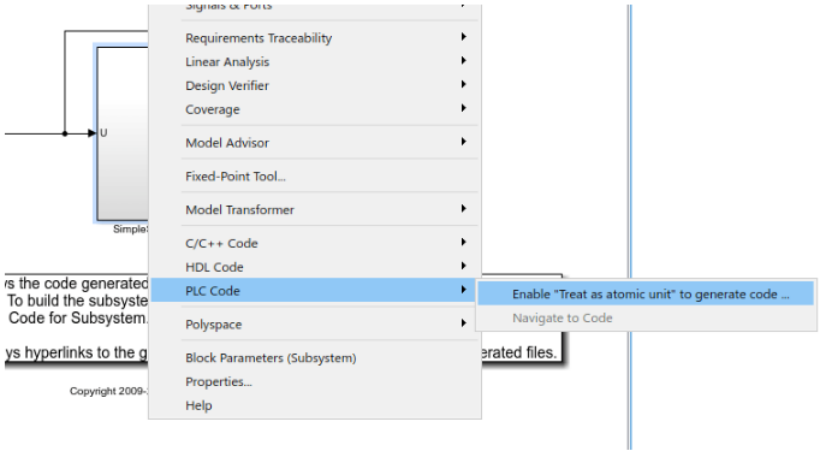
Vytvoření subsystému

Simulink PLC Coder je schopný pouze generovat kód pro *atomic subsystem* (nevirtuální subsystém). Nevirtuální subsystémy se oproti normálním subsystémům liší v tom, že jsou jako jeden celek. Při simulaci se první provedou všechny bloky v nevirtuálních subsystémech a poté simulace přejde k vykonání dalších bloků. Subsystém se vytvoří tak, že se označí zvolená oblast a pravým tlačítkem zvolíme *Create Subsystem from Selection*. Ukázka je na obrázku 3.11. [1]



Obr. 1.1: Vytvoření subsystému [1]

Následně se pravým tlačítkem zvolí daný subsystém a vybere se možnost PLC Code > Enable 'Treat as atomic unit' to generate code... Podle obrázku 1.2 [1]



Obr. 1.2: Nastavení *atomic subsystem* [1]

1.1.2 Generování a zkoumání ST

Po přípravě modelu v předchozí kapitole, lze nyní přejít na generování kódu. Jako první se musí otevřít *Simulink PLC Coder app* a sekce *PLC Code Generation*. Zde v seznamu *Target IDE* se zvolí *Siemens TIA Portal: Double precision*. Nakonec se povolí *Generate testbench for subsystem*, ten vytvoří testovací kód pro ověření jeho funkčnosti. Po veškerém nastavením je možné generovat kód. Vygeneruje se kliknutím pravým tlačítkem na subsystém a v záložce *PLC Code* zvolením *Generate Code for Subsystem*.[1]

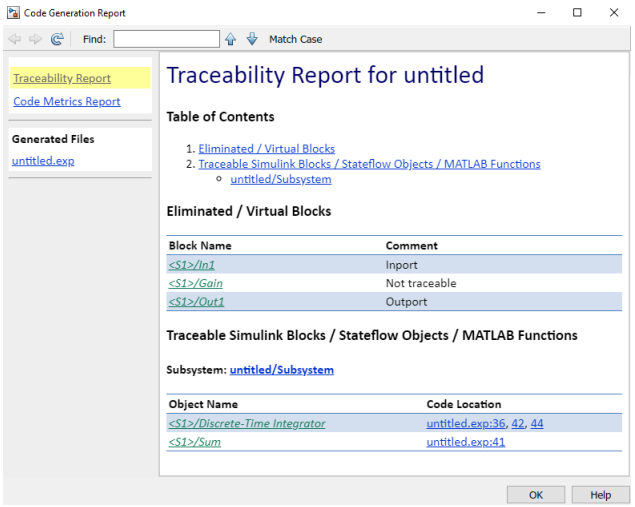
Důvod výběru *Siemens TIA Portal: Double precision* je ten, že podporuje *PLC SIMATIC S7-1500*. Tohle PLC a IDE se nachází v laboratoři a pracuje se s nimi v téhle práci.

Na obrázcích 1.3 a 1.4 je příklad pro jednoduchý systém s diskrétním integrátorem v záporné zpětné vazbě.

```
FUNCTION_BLOCK Subsystem
VAR_INPUT
    ssMethodType: SINT;
    In1: LREAL;
END_VAR
VAR_OUTPUT
    Out1: LREAL;
END_VAR
VAR
    c_DiscreteTimeIntegrator_DS: LREAL;
END_VAR
CASE ssMethodType OF
    SS_INITIALIZE:
        (* SystemInitialize for Atomic SubSystem: '<Root>/Subsystem' *)
        (* InitializeConditions for DiscreteIntegrator: '<S1>/Discrete-Time Integrator' *)
        c_DiscreteTimeIntegrator_DS := 0.0;
        (* End of SystemInitialize for SubSystem: '<Root>/Subsystem' *)
    SS_STEP:
        (* Outputs for Atomic SubSystem: '<Root>/Subsystem' *)
        (* Sum: '<S1>/Sum' incorporates:
         *  DiscreteIntegrator: '<S1>/Discrete-Time Integrator' *)
        Out1 := In1 - c_DiscreteTimeIntegrator_DS;
        (* Update for DiscreteIntegrator: '<S1>/Discrete-Time Integrator' *)
        c_DiscreteTimeIntegrator_DS := (0.2 * Out1) + c_DiscreteTimeIntegrator_DS;
        (* End of Outputs for SubSystem: '<Root>/Subsystem' *)
END_CASE;
END_FUNCTION_BLOCK
VAR_GLOBAL CONSTANT
    SS_INITIALIZE: SINT := 0;
    SS_STEP: SINT := 1;
END_VAR
```

Obr. 1.3: Příklad vygenerovaného kódu

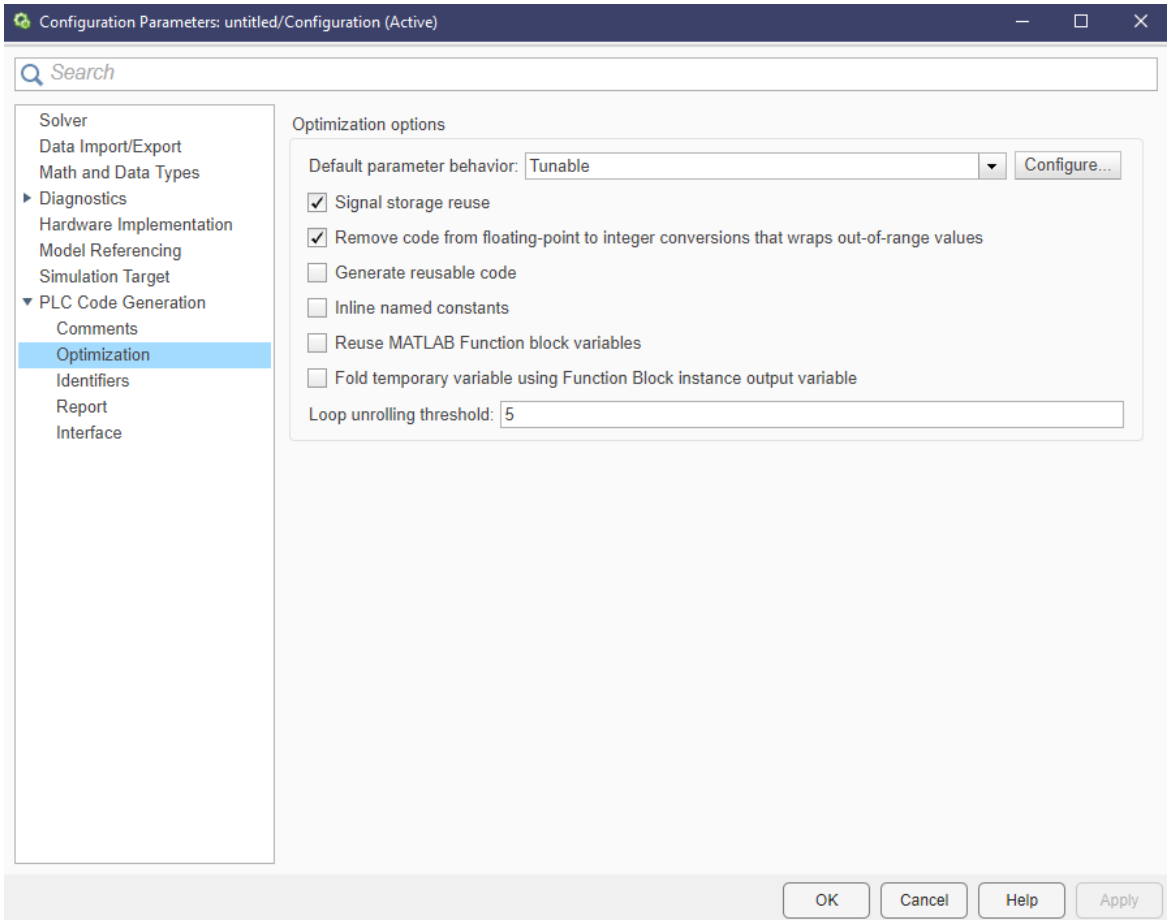
Po vygenerovaném kódu se otevře okno *Code Generation Report*. V něm lze nalézt seznam eliminovaných a virtuálních bloků, vygenerovaný kód a veškeré informace o kódu. [1]



Obr. 1.4: Okno *Code Generation Report*

1.1.3 Optimalizace

Simulink PLC Coder obsahuje i optimalizaci, kterou lze nastavit a tím zrychlit generování kódu. Je možné nastavit různé metody optimalizace, které jsou zobrazeny v obrázku 1.5. [2]



Obr. 1.5: Okno nastavení *Optimization*. [2]

V *default* nastavení je zapnuté:

- Default parameter behavior – 'Tunable', umožňuje laditelnost parametrů numerických bloků.
- Signal storage reuse – Opakovaně používá alokované paměti přidělené k I/O signálů bloku. Zmenšuje požadavky programu v reálném čase.
- Remove code from floating-point to integer conversions that wraps out-of-range values – Při převádění čísla s plovoucí desetinnou čárkou na celé číslo, se odebere část kódu, která zpracovává hodnoty mimo rozsah.

Ostatní nastavení se bude povolovat v průběhu projektu, podle složitosti kódu a potřebám systému. [2]

1.1.4 Importování ST do IDE

Jako poslední krok je přidání kódu do vývojového prostředí. Pro nás *TIA Portal*, které je nainstalované v laboratoři a je propojené s daným PLC.

Simulink PLC coder nabízí automatické vkládaní kódu do IDE, ale to nyní není potřeba používat. Jedná se o jeden kód a vložení do prostředí je velice jednoduché. Jako první se vytvoří v *TIA Portalu* projekt, pokud ještě není vytvořen. Vygenerovaný kód s příponou SCL se vloží do záložky *External source files*. Následně se musí vygenerovat blok obsahující kód. Kliknutím pravým tlačítkem na přidaný soubor a zvolení možnosti *Generate source from blocks*, se vytvoří v záložce *Program blocks* funkční blok s kódem. [3]

1.2 Simulátor dynamických systémů

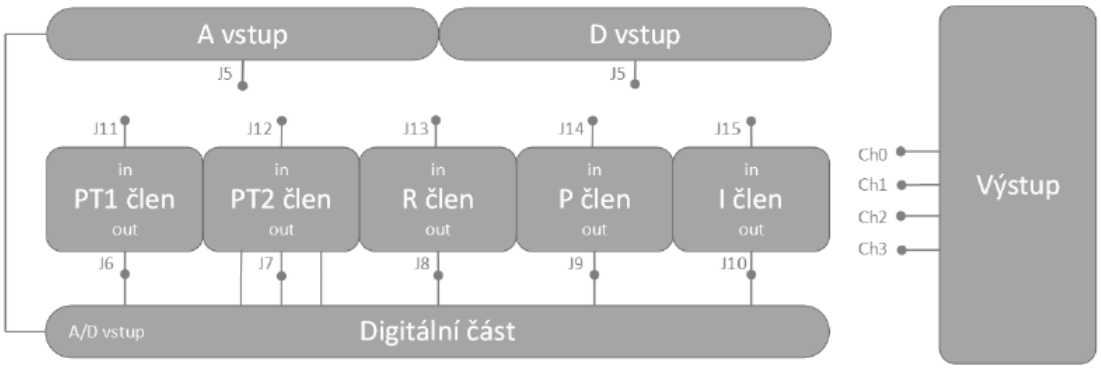
Pro simulování systému na PLC využijeme simulátor dynamických systému, jenž vypracoval Ing. Stanislav Pacal a byla to jeho diplomová práce. [3]

Simulátor slouží jako laboratorní přípravek na simulování jednoduchých technologických procesů. Je složen z analogové a digitální části. Analogová část má 5 modulů a každý z nich simuluje odlišnou dynamiku.

- PT1 člen – systém prvního řádů s nastavitelnou konstantou
- PT2 člen – systém druhého řádů s nastavitelným zesílením a tlumením
- P člen – proporcionální člen s nastavitelným zesílením
- I člen – systém s astatismem a nastavitelnou rychlostí integrace
- Rozdílový člen - simulátor poruchy působící na výstupu systému

Digitální část slouží pro vytvoření libovolného vstupního signálu, měření výstupních hodnot, nastavování podílu rozdílového členu a ovlivňování dynamiky setrvačného členu prvního řádu. [3]

V práci budu potřebovat pouze analogovou část a z ní dva členy: simulátor dynamiky systému druhého řádu (PT2) a proporcionální člen (P).



Obr. 1.6: Blokové schéma platformy pro simulování dynamických systémů [3]

1.2.1 Setrvačný nebo kmitavý člen druhého řádu

Patří do analogové části a je složen ze tří operačních zesilovačů. Ty realizují přenosové funkce druhého řádu s nastavitelnými parametry zesílení ω a tlumení d. Přenosová rovnice je 1.1 a její parametry tlumení 1.2 a zesílení 1.3. [3]

$$F(p) = -\frac{\omega^2}{p^2 + p \cdot 3 \cdot \omega \cdot d + \omega^2} \tag{1.1}$$

$$d = \frac{R_2}{R_2 + \Delta R_5} \tag{1.2}$$

Kde ΔR_5 je nastavovací odpor na přepínači SW4 (1M2, 620k, 330k, 160k, 47k, 2k7 Ω). Odpor R_2 má hodnotu 5,6 kΩ. [3]

Z důvodu malého tlumení jsem rezistor na přepínači SW4 přepájel z 620 kΩ na 1 kΩ. Problematika je rozebrána v kapitole 2.2.4. Nové hodnoty na přepínači SW4 jsou: 1M2, 1k, 330k, 160k, 47k, 2k7 Ω.

$$\omega^2 = \frac{1}{C \cdot \Delta R_4} \tag{1.3}$$

Kde je kapacita C = 4,7 μF a odpor ΔR_4 přepínač SW3 a SW6 (stejné hodnoty jak u SW4). [3]

1.2.2 Proporcionální člen

Člen simuluje proporcionální systém. Je tvořený invertujícím zapojením operačního zesilovače s nastavitelnými parametry. Pokud se nastaví zesílení K = 1, stane se z členu invertor signálu. Přenosová rovnice tohoto členu je 1.4. [3]

$$F(p) = -\frac{R_2}{\Delta R_1} \tag{1.4}$$

Kde ΔR_1 je nastavovací odpor na přepínači SW2 (22k, 8k2, 16k, 36k, 220k, 470k Ω). Odpor R_2 = 22 kΩ. [3]

1.2.3 Napájení

Pomocí stepdown stejnosměrného měniče *DCW08B-15* je zajištěno symetrické napájení operačních zesilovačů. Ten převádí vstupní napětí v rozsahu 18 až 35 V na výstupních ± 15 V [3].

Při používání simulátoru byla deska napájena pomocí laboratorního zdroje při nastaveném napětí 24 V.

2 Oživení simulátoru a měření systému

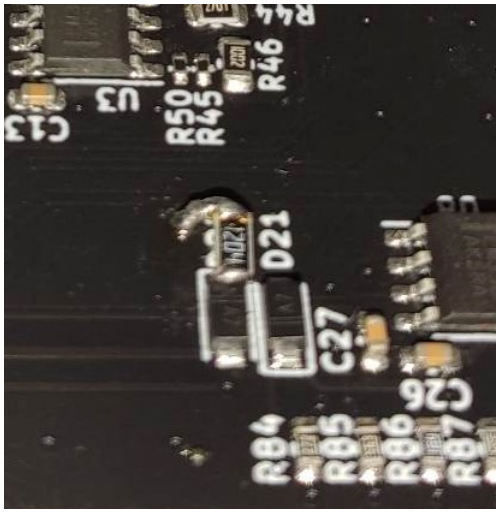
Tato a následující kapitoly představují praktickou část bakalářské práce. Tahle kapitola obsahuje oživení simulátoru, až po simulaci jednoduchého systému s následnou identifikací.

2.1 Práce se simulátorem dynamických systémů

První je nutné dodaný simulátor oživit a ověřit jeho funkčnost. Od vedoucího práce byla poskytnuta deska simulátoru. Měření proběhlo v laboratoři T12/SE 2.132 v Brně na VUT, kde se nachází PLC automaty na kterých, byl simulátor připojen. Zde se tedy ověřila jeho funkčnost a proběhly zde veškeré měření a regulace.

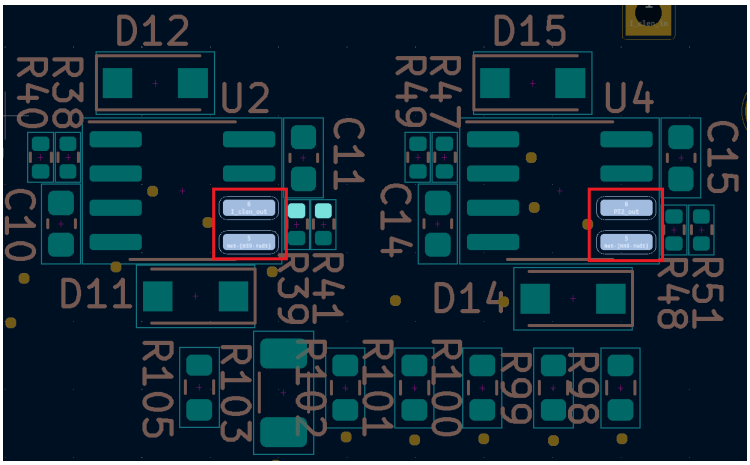
2.1.1 Osazení DPS

Deska byla z počátku polovičně již osazena a bylo zapotřebí napájet pouze vrchní část desky. Byly zde i chyby od výroby co se musely opravit. Na desce chyběl jeden rezistor hodnoty 1,2 MΩ. Ten musel být připájen zvlášť 2.1. Bez daného rezistoru na analogovém vstupu není 0 V v klidovém stavu. Na zmiňovaném vstupu bude přiváděn jednotkový skok ve formě napětí od 0 do 10 V.



Obr. 2.1: Připájený chybějící rezistor

Druhá chyba byla prohození dvou kontaktů u dvou operačních zesilovačů. Protože se jednalo o velice malé součástky, musel jsem improvizovat a pro prohození použít tenké vodiče. Prohozené piny v schématu na obrázku 2.2.



Obr. 2.2: Prohozené kontakty v operačních zesilovačích [3]

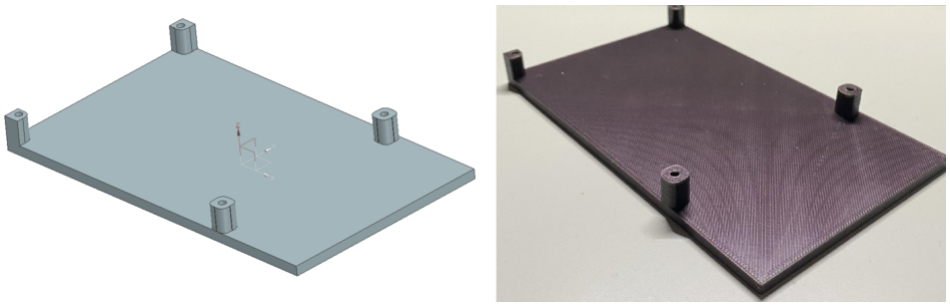
Schéma z obrázku 2.2 je z příloh diplomové práce [3]. Výsledná osazená deska plošných spojů je v příloze A.1 a A.2.

2.1.2 Oživení DPS

Pro ověření, zda je deska funkční, jsem ji připojil na napájení 24 V. Při analogovém režimu zde byla minimální hodnota v jednotkách mV (zaokrouhleno na 0). Z čehož plyne že deska je správně zapájena.

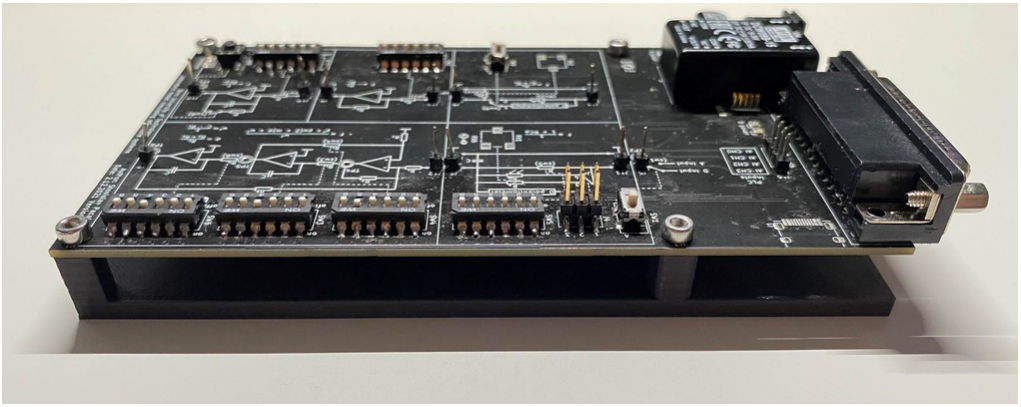
2.1.3 Podstavec k simulátoru

Pro lepší používání při měření a budoucímu využití desky v laboratorní úloze, jsem simulátoru navrhl podstavec na 3D tisk. 3D model jsem navrhl v programu *Siemens NX* a vytiskl na 3D tiskárně Bambu Lab X1C.



Obr. 2.3: 3D model podstavce v programu *Siemens NX* (vlevo) a vytisknutý podstavec (vpravo)

Deska má v sobě 4 montážní otvory o průměru 3 mm. Ty jsem využil pro přichycení podstavce.



Obr. 2.4: Podstavec připevněn k simulátoru

2.1.4 Zvolení soustavy na simulátoru

Pro ověření funkčnosti desky, budu simulovat na desce systém druhého řádu a na PLC změřím vybrané charakteristiky systému. Na setrvačném členu druhého řádu nastavím přepínač SW4 do polohy 1 (2,7 kΩ) a přepínače SW3 a SW6 do polohy 3 (330 kΩ). Nyní mohu dosadit do vzorečků 1.1 - 1.3 a dopočítat členy přenosů.

$$d = \frac{R_2}{R_2 + \Delta R_5} = \frac{5,6 \cdot 10^3}{5,6 \cdot 10^3 + 2,7 \cdot 10^3} = 0,67857 \tag{2.1}$$

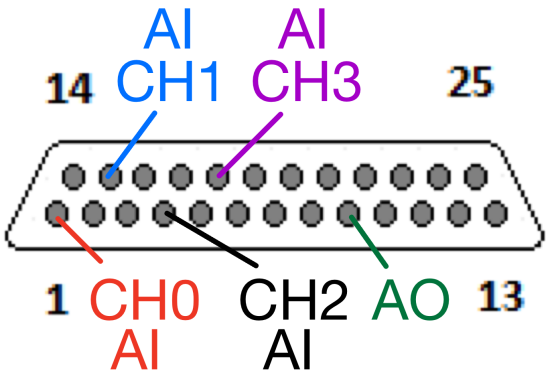
$$\omega^2 = \frac{1}{C \cdot \Delta R_4} = \frac{1}{4,7 \cdot 10^-6 \cdot 330 \cdot 10^3} = 0,64475 \tag{2.2}$$

Nakonec dosadím vypočítané členy do přenosové rovnice 1.1 a mám výsledný přenos.

$$F(p) = -\frac{0,65}{p^2 + 1,64p + 0,65} \tag{2.3}$$

2.1.5 Propojení simulátoru s PLC

Abych mohl simulátor propojit s PLC je první potřeba kontrola I/O na PLC a simulátoru. První zjistím, rozmístění pinů na konektoru simulátoru 2.5.



Obr. 2.5: Vstupní a výstupní piny na konektoru simulátoru

Jednotlivé zkratky značí: AI – Analog input, CH – Channel, AO – Analog output.

Následně je zkontroluji s konektorem CANON25, který vyvádí vstupy/výstupy PLC v laboratoři. Porovnání pinu, zda odpovídají konektorům je uvedeno v následující tabulce 2.1.

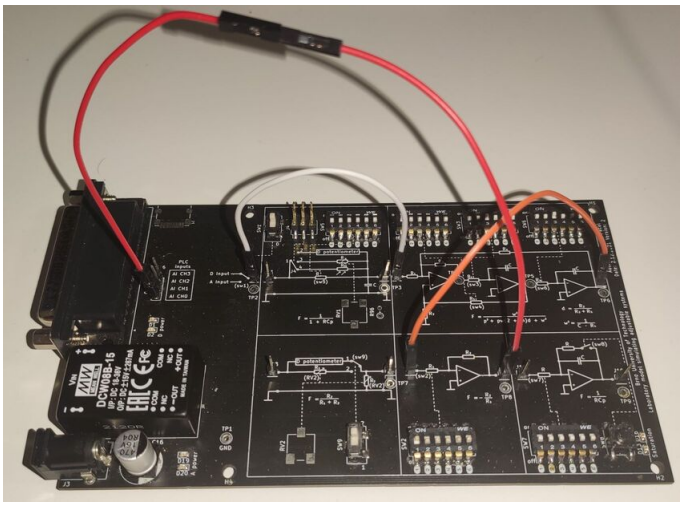
Tab. 2.1: Porovnání vstupů a výstupů konektoru PLC a simulátoru

I/O kanály	**PLC**	**Simulátor**
I CH0	1,2	1
I CH1	15,16	15
I CH2	4,5	4
I CH3	18,19	18
O CH0	9,22,10,23	9

Ostatní piny konektoru simulátoru, které nejsou zmíněné jsou připojené k zemi (proto jsou vypsány samostatné piny). PLC konektor má čtyřžilové propojení, a proto jsou v tabulce 2.1 dva piny. Lze vidět že piny odpovídají a mohu tedy analogovým kabelem propojit simulátor a PLC.

Aby ze simulátoru nevycházel záporný přenos (způsobený lichým počtem OZ zapojených v invertujícím zapojení) musím ještě za setrvačný člen druhého řádu připojit proporcionální člen. U něho nastavím ΔR_1 na přepínači SW2 do polohy 1 (22 kΩ). To způsobí podle vzorečku 1.4, že přenos bude invertovaný. To obrátí znaménko přenosu.

Poté na simulátoru nastavím přepínač SW1 do polohy A (použití analogové části desky) a vodičem propojím jednotlivé členy. Nyní je deska připravena pro simulování přenosu a vypadá následovně 2.6.



Obr. 2.6: Připravený simulátor pro připojení do PLC

2.2 Simulování systému

V téhle podkapitole naměřím data pro nastavený přenos a následně je porovnám s teoretickým přenosem. Jedná se ověření zda simulované přenosy odpovídají teoretickým.

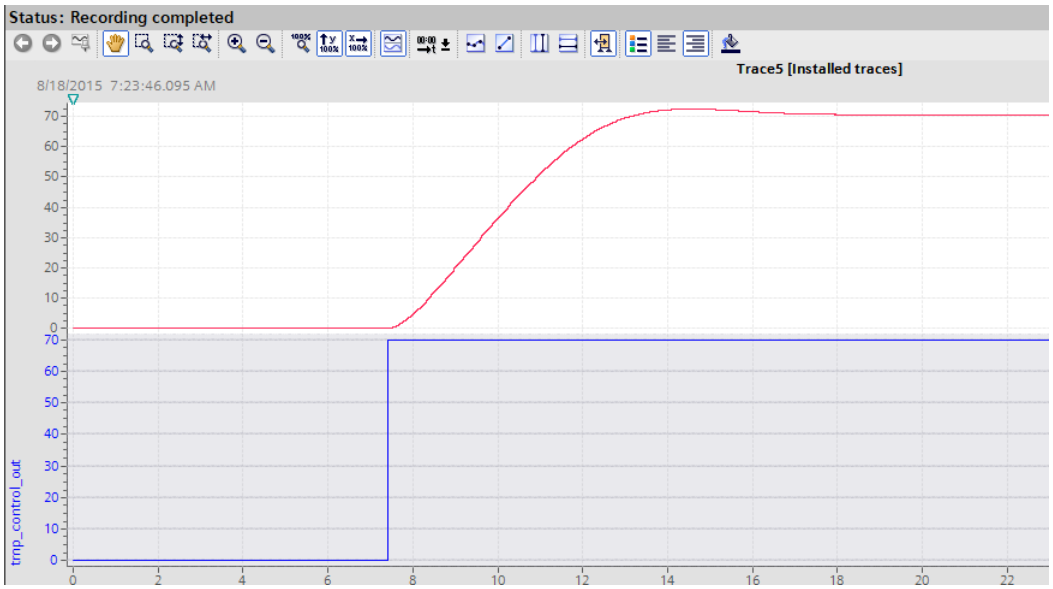
2.2.1 Měření na PLC

Ve vývojovém prostředí *TIA Portal* jsem vytvořil projekt pro změření simulovaného systému. Protože signál přichází do PLC v bitech musí se převést na sjednocenou jednotku (V téhle práci jsou to procenta). Pomocí funkce *Serialize* se převede výstupní signál ze simulátoru na 0 až 100 %, pro lepší zobrazení. Opačně je to provedeno pro výstup PLC neboli vstup do simulátoru. Tento signál představuje velikost skoku. Pro opačné převedení slouží funkce *Deserialize*. To znamená z hodnot 0 až 100 % na hodnoty v bitech.

Následně pomocí funkce *trace* jsem změřil přechodové charakteristiky různých amplitud nastaveného systému. Pro tuhle funkci je třeba nastavit I/O signály a následující specifické nastavení:

- Sample with - 'Cyclic interrupt', to znamená že se hodnota vezme při jednom cyklu hlavního bloku *Cyclic interrupt*. V mém případě je cyklování nastavené na 1 ms.
- Record every - '1', zaznamenání hodnoty při každém jednom cyklu.
- Trigger - 'Record immediately', měření začne při spuštění programu.
- Max. recording duration - '58250 samples', maximální doba měření. S aktuálním nastavení to vychází na 58 s.

Jednotlivé data z průběhu jsem vytáhl (v excel formátu) a nahrál do Matlabu. Na následujícím obrázku 2.7 je naměřený průběh pro 70 %.



Obr. 2.7: Naměřený průběh pro amplitudu 70 % v TIA portal

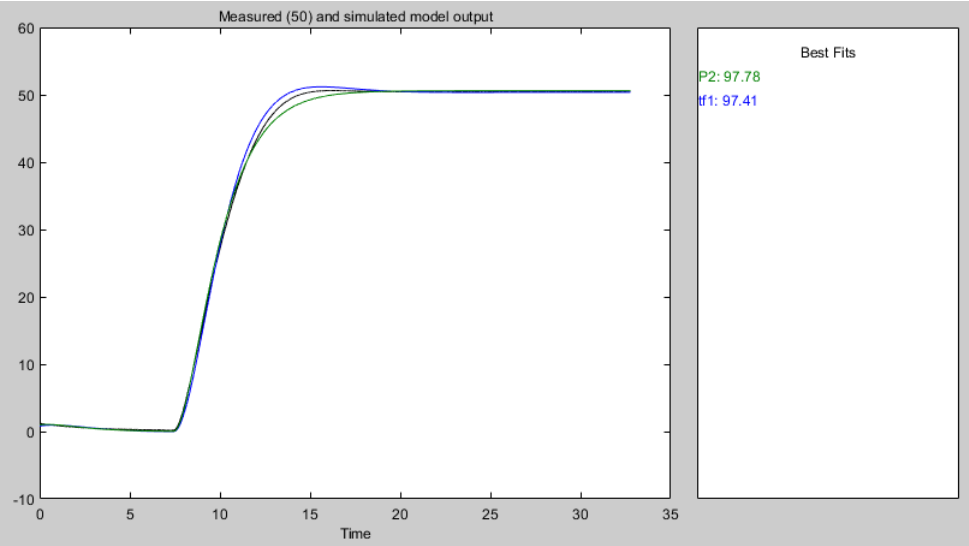
2.2.2 Identifikace systému

Naměřená data jsem importoval do *System identification toolbox*, který dokáže na základě vstupu a výstupu systému vypočítat přenosovou rovnici. Při vkládání dat jsem na vstup nastavil hodnoty z výstupu PLC (jednotkový skok) a na výstup jsem přiřadil vstup do PLC (výstup simulátoru). Následně jsem nastavil vzorkovací čas na 1 ms a startovací čas na 0 s. Nastavení musí být stejné jak při měření na PLC.

Data jsem následně pomocí funkce *Merge experiments* spojil dohromady pro přesnější odhad. Poté pomocí metod *Transfer function a Process model* jsem odhadl přenosy. Pro správnou identifikaci jsem metodám nastavil:

- Initial condition – 'Zero', nastaví iteračnímu algoritmu počáteční podmínku pro odhad modelu na 0. [4]
- Number of poles – '2', počet pólů přenosu.
- Number of zeros – '0', počet nul přenosu.

Zbytek mohu nechat v originálním nastavení (U metody *Process model* jsem musel navíc vypnout dopravní zpoždění). Podle funkce *Best Fits* byla nejlepší metoda *Process model* s přesností 97,78 %. Výsledné přenosy lze vidět v následujícím obrázku 2.8.



Obr. 2.8: Výsledná identifikace pomocí *System identification toolbox*

Identifikovaný přenos podle *System identification toolbox* metody *Process model* vyšel:

$$F(p) = \frac{0,53}{p^2 + 1,46p + 0,53} \tag{2.4}$$

2.2.3 Porovnání výsledného systému

Mohu nyní porovnat výsledné přenosy, kdy teoretické $F_1(p)$ představuje přenos 2.3 a měřené $F_2(p)$ je 2.4. Pomocí funkce *Stepinfo* v Matlabu jsem odečetl důležité parametry, které jsou v následujících obrázcích 2.9 a 2.10.

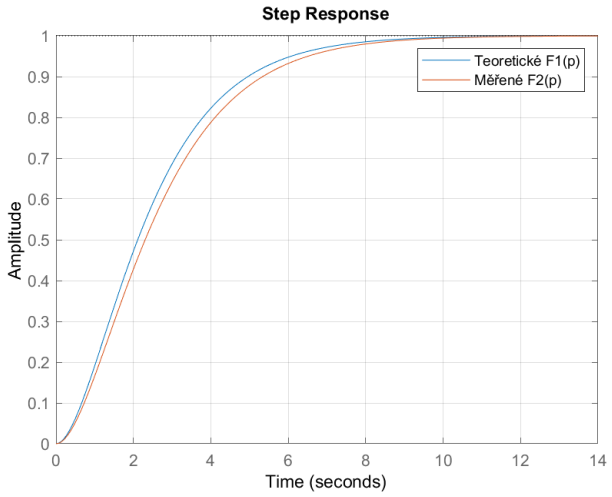
```
      RiseTime: 4.2932
 TransientTime: 7.5165
  SettlingTime: 7.5165
   SettlingMin: 0.9018
   SettlingMax: 0.9992
     Overshoot: 0
    Undershoot: 0
          Peak: 0.9992
      PeakTime: 12.3980
```

Obr. 2.9: Teoretický přenos $F_1(p)$ z rovnice 2.3.

```
      RiseTime: 4.6089
 TransientTime: 8.0065
  SettlingTime: 8.0065
   SettlingMin: 0.9020
   SettlingMax: 0.9999
     Overshoot: 0
    Undershoot: 0
          Peak: 0.9999
      PeakTime: 16.3692
```

Obr. 2.10: Měřený přenos $F_2(p)$ z rovnice 2.4.

Z hodnot lze vyčíst že $F_1(p)$ má rychlejší dobu náběhu a ustálení. Hodnoty pak nejsou moc odlišné a lze usoudit že simulátor simuluje správně systém, který jsem si předvolil. Výsledné přenosové charakteristiky jsou zobrazeny v následujícím obrázku 2.11.



Obr. 2.11: Porovnání teoretického a měřeného průběhu

2.2.4 Oprava tlumení v přenosu druhého řádu

Protože budu simulovat tepelný systém, očekávám že nebude mít překmit a bude takzvaně přetlumený. To nastane pouze pokud tlumení $\xi > 1$. Z rovnice 1.1 tlumení odpovídá:

$$\xi = \frac{3d}{2} \tag{2.5}$$

Pokud za d dosadím vztah 1.2 a položím $\xi > 1$ dostanu vztah:

$$\frac{R_2}{R_2 + \Delta R_5} > \frac{2}{3} \tag{2.6}$$

Kde $R_2 = 5,6k\Omega$ a ΔR_5 je nastavovací odpor. Více informací v kapitole 1.2.1. Tahle podmínka pro tlumení je splněna pouze pro přepínač 1. To odpovídá odporu $\Delta R_5 = 2,7k\Omega$. Tím pádem tlumení vychází $\xi = 1,01$.

To je velice na hraně a po měření systému na PLC, soustava vychází s překmitem. Chyba může být způsobena nepřesnosti odporů. Proto jsem jeden odpor na pozici 5, přepájel na odpor s hodnotou $1k\Omega$. Poté je $\xi = 1,27$. Po novém měření a identifikaci přenosu, je přenos bez překmitu.

2.2.5 Kontrola nového tlumení

Protože nyní mám nové tlumení a také vím že simulátor je plně funkční, volím nový přenos a ověřím, zda je tlumení stejné. Nová soustava bude představovat tepelný systém zahřívaní vody. Pro možnost realizace v laboratorních podmínkách jsou konstanty modelů mnohonásobně menší, než v reálných podmínkách a celý model je zjednodušený. Přepínače jsou nově nastaveny následovně: SW4 na pozici 5 ($1k\Omega$), SW3 a SW6 na pozici 2 ($620k\Omega$). Stejně jak v kapitole 2.2 postupuji a identifikuji přenos. Výsledná identifikace je:

$$F_s(p) = \frac{0,12}{p^2 + 0.86p + 0,12} = \frac{1,02}{(1 + 1,47p) \cdot (1 + 5,58p)} \tag{2.7}$$

Pro ověření, zda je systém přetlumený, vypočtu z přenosu ξ. První musím soustavu převést do správného tvaru:

$$F_s(p) = \frac{K_0}{T^2p^2 + 2\xi Tp + 1} = \frac{0,12}{p^2 + 0.86p + 0.12} = \frac{1}{8.33p^2 + 7,17p + 1} \tag{2.8}$$

Nyní lze ze soustavy vidět $T^2 = 8.33$. Poté je možné dopočítat ξ:

$$2\xi Tp = 7,17 \tag{2.9}$$

$$\xi = 1,24$$

Lze vidět podle předchozí podkapitoly 2.2.4, že tlumení vychází s malou odchylkou tři setiny. Soustava je tedy přetlumená a nekmitá. Budu ji tedy používat pro následnou regulaci. Veškerá měření proběhla pro skok 5V.

3 Regulace v PLC

Cílem téhle kapitoly je návrh regulátoru pro identifikovaný systém na simulátoru a následná implementace generování PLC kódu do fyzického PLC.

Dalším cílem je porovnání výsledků se simulací a knihovní funkcí PLC.

3.1 Návrh regulátoru

Pro návrh regulátoru jsem použil metodu optimálního modulu. Ta pracuje s požadovaným tvarem frekvenční charakteristiky uzavřené smyčky, která je daná přenosem řízení $F_w(p)$. Přechodný děj bude optimální když, $|F_w(j\omega)| = 1$ pro co nejvyšší frekvence a pro monotónní průběh. [5]

Metoda zahrnuje delší výpočty s rovnicemi o x neznámých. To ale není třeba počítat, protože existují tabulky pro optimální seřízení regulátoru podle kritéria optimálního modulu. Stačí, aby daný přenos soustavy existoval pro daný typ regulátoru. První přenos 2.7 přepočítám do následujícího tvaru:

$$F_S(p) = \frac{K_1}{(1+T_1p)\cdot(1+T_2p)} = \frac{1,02}{(1+1,47p)\cdot(1+5,58p)} \tag{3.1}$$

Následně z parametrů T_1 a T_2 dopočtu parametr r.

$$r = \frac{T_2}{T_1} = \frac{5,58}{1,47} = 3,79 \tag{3.2}$$

Tab. 3.1: Optimální seřízení regulátoru podle OM [6]

Typ regulátoru	**Vzorec pro daný regulátor**
P	$k_p = \frac{r^2+1}{2\cdot K_1\cdot r}$
I	$T_I = \frac{2\cdot K_1\cdot T_1}{1} \cdot \frac{1+r}{1}$
PI	$k_p = \frac{r^2+1}{2\cdot K_1\cdot r}$ $T_I = k_p \cdot \frac{2\cdot K_1\cdot T_1}{1} \cdot \frac{r\cdot(1+r)}{r^2+r+1}$
PID	$k_p \to \infty$ $T_I \to 0$ $T_D \to \infty$

Nyní můžu použít tabulku 3.1, která platí pro zvolený přenos a lze z ní dopočítat jednotlivé hodnoty regulátoru. Protože pro zvolený přenos nelze navrhnout regulátor PID touto metodou, použiji regulátor PI. Pokud bych chtěl PID regulátor, musel bych použít jinou metodu.

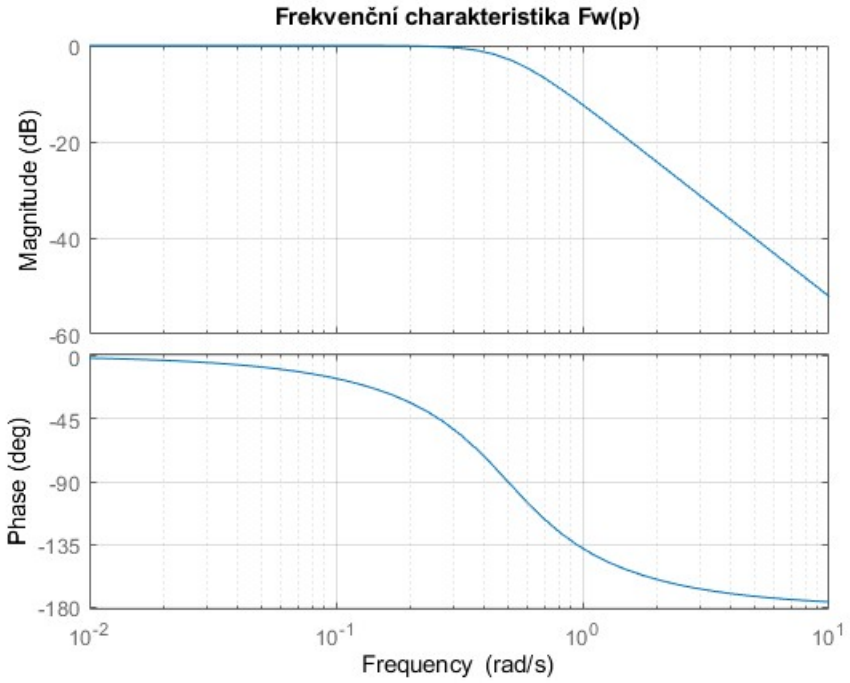
Pomocí tabulky jsem vypočítal hodnoty regulátoru a dosadil je do rovnice na PI regulátor:

$$F_R(p) = k_p \cdot \left(1 + \frac{1}{T_I \cdot p}\right) = 1,99 \cdot \left(1 + \frac{1}{5,66 \cdot p}\right) = \frac{11,27 \cdot p + 1,99}{5,66 \cdot p} \tag{3.3}$$

Nakonec dopočítám přenos řízení:

$$F_W(p) = \frac{F_R(p) \cdot F_S(p)}{1 + F_R(p) \cdot F_S(p)} = \frac{7,89p^4 + 8,17p^3 + 2,16p^2 + 0,17p}{32p^6 + 54,93p^5 + 39,26p^4 + 14,85p^3 + 2,63p^2 + 0,17p} \tag{3.4}$$

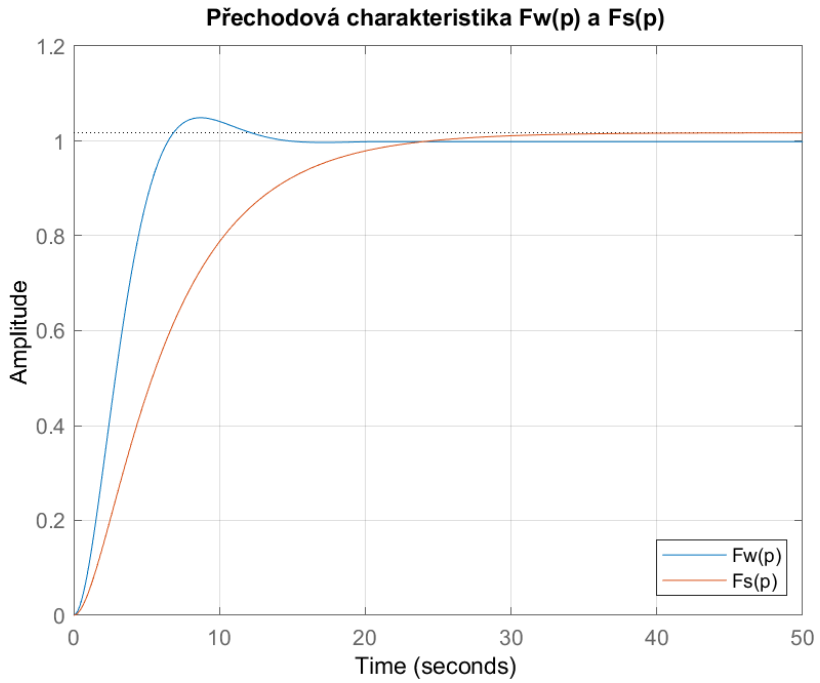
Pomocí funkce *bode* v Matlabu jsem vykreslil frekvenční charakteristika přenosu řízení:



Obr. 3.1: Frekvenční charakteristika $F_W(p)$

Z charakteristiky 3.1 lze vidět, že je průběh monotónní a neobsahuje rezonanční překmit. To splňuje požadavky pro optimální modul.

Pro porovnání jsem vykreslil přechodovou charakteristiku soustavy $F_S(p)$ a přenosu řízení $F_W(p)$ na obrázku 3.2. Implementováno v Matlabu funkcí *step*.



Obr. 3.2: Přechodová charakteristika $F_W(p)$ a $F_S(p)$

Lze vidět že odezva regulačního obvodu je rychlejší a je ustálená. Nyní regulátor budu simulovat v Simulinku a pro něj budu generovat PLC kód.

3.2 Simulování regulátoru v Simulink

V téhle kapitole navržený regulátor je simulován v Simulink a následně je pro něj generován PLC kód.

3.2.1 Discrete PID Controller

Pro simulaci regulátoru v prostředí Simulink použiji blok *Discrete PID Controller*. Ten je podporovaný pro generovaní PLC kódu. Regulátor lze nastavit podle potřeby na PID, PI, PD, P a I. Je možné také nastavit, aby pracoval buď ve spojité nebo diskrétní oblasti. Protože generátor PLC kódu umožňuje pouze práci v diskrétní oblasti, je nastavena tahle možnost.

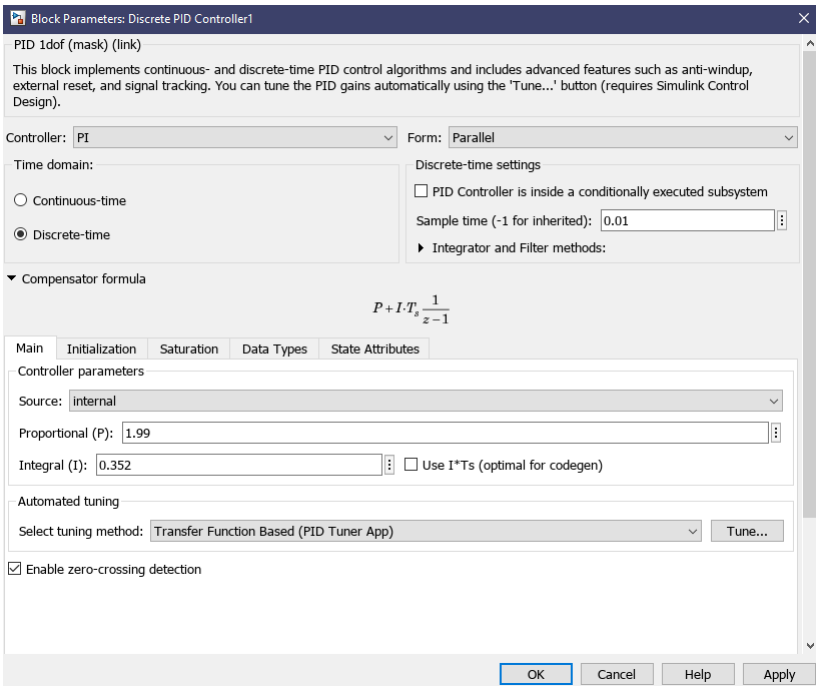
Regulátor je nastaven pro paralelní formu. To znamená že přenos regulátoru 3.3 se musí přepočítat do stejné formy, aby bylo možné vytáhnout parametry P a I, které se mohou vložit do nastavení. Přepočet do paralelní formy:

$$F_R(p) = 1,99 \cdot \left(1 + \frac{1}{5,66 \cdot p}\right) = 1,99 + 0,352 \cdot \frac{1}{p} \tag{3.5}$$

Z přepočtu lze odečíst parametry: P = 1,99 a I = 0,352. Rovnice nastaveného bloku *Discrete PID Controller* s dosazenými parametry je následující:

$$F_R(z) = P + I \cdot T_S \cdot \frac{1}{z-1} = 1,99 + 0,352 \cdot 0,01 \cdot \frac{1}{z-1} \tag{3.6}$$

Vzorkování T_S je nastaveno na 0,01 z důvodu stejného vzorkovaní v PLC. To je použité v bloku *cyclic interrupt* v TIA Portal.

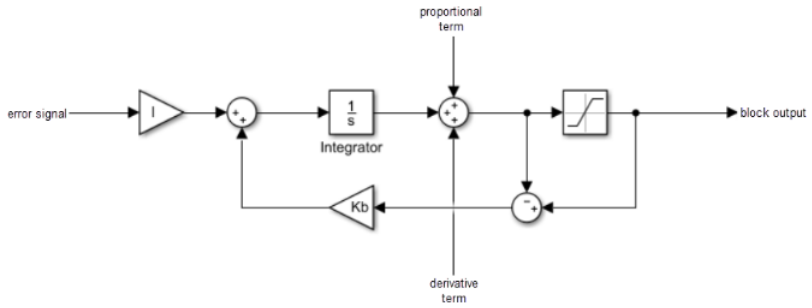


Obr. 3.3: Nastavení bloku *Discrete PID Controller*

Protože používám integrační složku může v obvodu dojít k nasycení akční veličiny. Proto je zde dobré zavést *Anti-windup* metodu. Podle teorie lze *Anti-windup* metodu zavést pomocí měření skutečné hodnoty akční veličiny nebo omezením akční veličiny. [7]

Blok *Discrete PID Controller* umožňuje zavést saturaci a *Anti-windup* v záložce *Saturation*. Zde jsem nastavil metodu pro *Anti-windup back-calculation* a nastavil saturaci na limity 0 až 100. Hodnoty limit saturace představují minimální a maximální hodnoty, které lze na simulátor přivést (0 až 10 V).

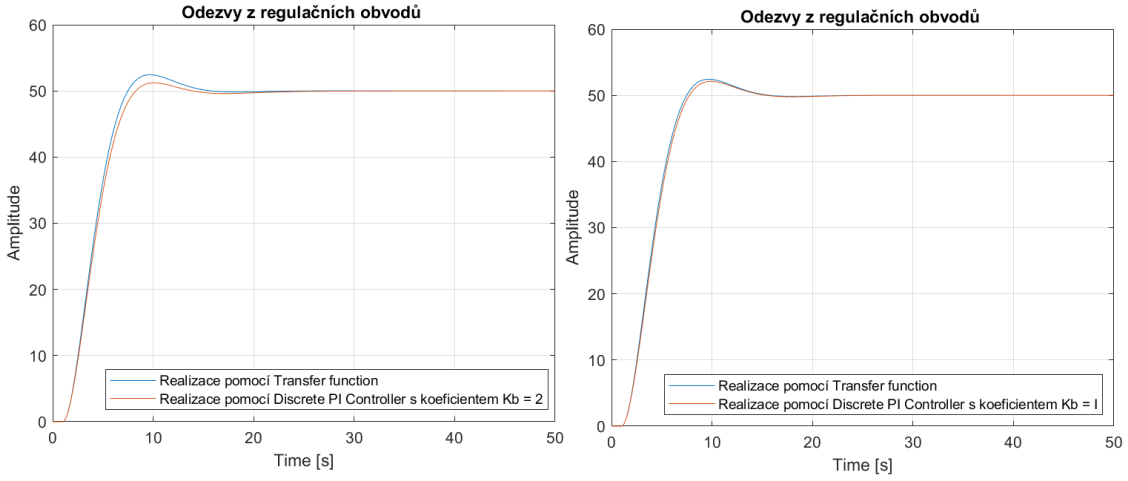
Metoda *Anti-windup back-calculation* funguje tak, že výstup bloku integrátoru saturuje a rozdíl mezi saturovanou a nesaturovanou hodnotou posílá zpět do integrátoru. Hodnota je ještě násobena nastavitelným koeficientem *Kb*. [8]. Schéma *back-calculation* v následujícím obrázku:



Obr. 3.4: Nastavení bloku *back-calculation* v *Discrete PID Controller* [8]

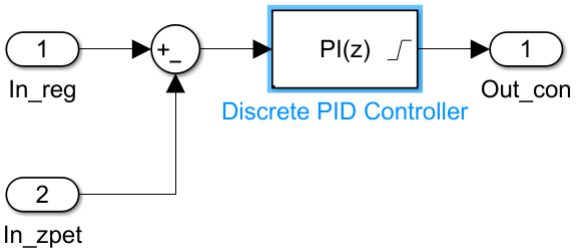
Podle dokumentace [8] je dobré nastavit koeficient Kb stejný jako hodnota I. Při tomhle nastavení došlo k nejpřesnějšímu přenosu pro teoretický přenos řízení. Pokud byl ale koeficient rovný 2, došlo k menšímu překmitu. Proto zůstalo nastavení $Kb = 2$.

Pro porovnání nastavení Kb jsem použil jeden regulační obvod tvořen pouze bloky *Transfer function* a druhý obvod, který má regulátor zaměněn za *Discrete PID Controller* s nastavitelným koeficientem Kb. Výsledné porovnání nastavení Kb je na obrázku 3.5.



Obr. 3.5: Odezvy z regulačních obvodů pro různé Kb

Regulátor je nyní nastavený a mohu se přesunout na generování PLC kódu. Jak je popsáno v kapitole 1.1 musím vytvořit subsystém. PLC slouží jako regulátor a do regulátoru vstupuje odchylka. To musím také nasimulovat a zahrnout v subsystému. Proto schéma regulátoru vypadá následovně:



Obr. 3.6: Schéma subsystém regulátoru

Kde:

- In_reg – Žádaná veličina (Bude amplituda 50% nebo 5V).
- In_zpet – Zpětná vazba (Výstup ze simulátoru).
- Out_con – Akční veličina (Výstup z PLC, Vstup do simulátoru).

3.2.2 Generování PLC kódu

Podle kapitoly 1.1 jsem nastavil subsystém a vygeneroval kód. Byl vygenerován pro IDE *Siemens TIA Portal: Double Precision* a bez *testbench* nastavení. Pro lepší popis jsem vygenerovaný kód rozdělil do tří části.

```
1   (*
2    *
3    * File: pid_reg.scl
4    *
5    * IEC 61131-3 Structured Text (ST) code generated for subsystem "pid_reg/Subsystem"
6    *
7    * Model name                    : pid_reg
8    * Model version                 : 1.3
9    * Model creator                 : xmusil72
10   * Model last modified by        : xmusil72
11   * Model last modified on        : Wed Apr 05 02:02:35 2023
12   * Model sample time             : 0s
13   * Subsystem name                : pid_reg/Subsystem
14   * Subsystem sample time         : 0.01s
15   * Simulink PLC Coder version    : 3.7 (R2022b) 13-May-2022
16   * ST code generated on          : Thu Apr 20 00:20:17 2023
17   *
18   * Target IDE selection          : Siemens TIA Portal: Double Precision
19   * Test Bench included           : No
20   *
21   *)
```

Obr. 3.7: První část kódu: Informace ze Simulink

První část kódu 3.7 obsahuje informace o modelu (jméno, verzi, kdo ho vytvořil ...) a provedené nastavená při generování kódu.

```
22  FUNCTION_BLOCK Subsystem
23  VAR_INPUT
24      ssMethodType: SINT;
25      In_reg: LREAL;
26      In_zpet: LREAL;
27  END_VAR
28  VAR_OUTPUT
29      Out_con: LREAL;
30  END_VAR
31  VAR
32      Integrator_DSTATE: LREAL;
33      rtb_Sum: LREAL;
34      rtb_Sum_e: LREAL;
35  END_VAR
```

Obr. 3.8: Druhá část kódu: Definice proměnných

V další části 3.8 je na začátku definovaní funkčního bloku (*Function block*) a následná definice veškerých proměnných. První jsou definované proměnné pro vstup a výstup. Následně jsou pomocné pro výpočty.

```
36  CASE ssMethodType OF
37      0:
38          (* SystemInitialize for Atomic SubSystem: '<Root>/Subsystem' *)
39          (* InitializeConditions for DiscreteIntegrator: '<S34>/Integrator' *)
40          Integrator_DSTATE := 0.0;
41          (* End of SystemInitialize for SubSystem: '<Root>/Subsystem' *)
42      1:
43          (* Outputs for Atomic SubSystem: '<Root>/Subsystem' *)
44          (* Sum: '<S1>/Sum' *)
45          rtb_Sum := In_reg - In_zpet;
46          (* Sum: '<S43>/Sum' incorporates:
47           *  DiscreteIntegrator: '<S34>/Integrator'
48           *  Gain: '<S39>/Proportional Gain' *)
49          rtb_Sum_e := (1.99 * rtb_Sum) + Integrator_DSTATE;
50          (* Saturate: '<S41>/Saturation' *)
51          IF rtb_Sum_e > 100.0 THEN
52              Out_con := 100.0;
53          ELSIF rtb_Sum_e >= 0.0 THEN
54              Out_con := rtb_Sum_e;
55          ELSE
56              Out_con := 0.0;
57          END_IF;
58          (* End of Saturate: '<S41>/Saturation' *)
59
60          (* Update for DiscreteIntegrator: '<S34>/Integrator' incorporates:
61           *  Gain: '<S27>/Kb'
62           *  Gain: '<S31>/Integral Gain'
63           *  Sum: '<S27>/SumI2'
64           *  Sum: '<S27>/SumI4' *)
65          Integrator_DSTATE := ((((Out_con - rtb_Sum_e) * 2.0) + (0.352 * rtb_Sum)) * 0.01) + Integrator_DSTATE;
66          (* End of Outputs for SubSystem: '<Root>/Subsystem' *)
67  END_CASE;
68  END_FUNCTION_BLOCK
69
```

Obr. 3.9: Třetí část kódu: Implementace regulátoru

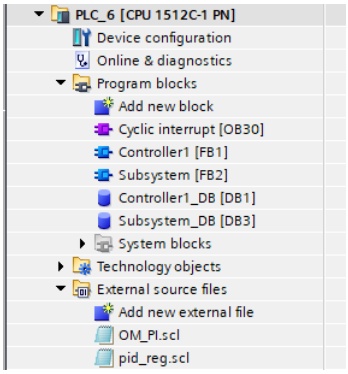
V poslední části 3.9 je samotný algoritmus vytvořený pro funkci PI regulátoru. Je tvořený *case* strukturou s parametrem *ssMethodType*. Ten podle hodnoty, zda je 0 nebo 1 spouští určitou část kódu. Na konci je ukončení *case* struktury a *Function block*.

3.3 Implementace regulátoru do PLC

Tahle kapitola popisuje implementaci regulátoru v PLC pomocí vygenerovaného PLC kódu a knihovních funkcí.

3.3.1 Regulátor generovaného PLC kódu

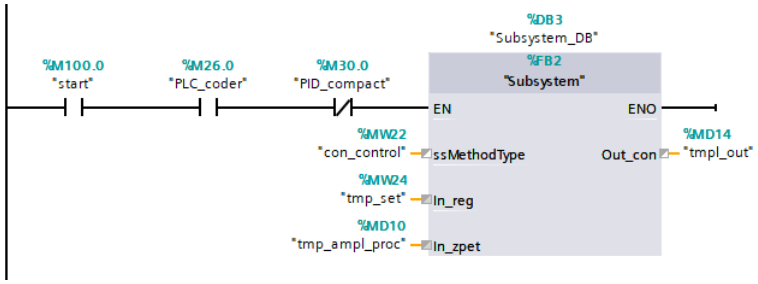
Výsledný kód jsem vložil do programu podle předchozí kapitoly 1.1.4 a vygeneroval *Function block*. Všechny výsledné bloky lze vidět:



Obr. 3.10: Bloky v programu

Předchozí projekt pro měření soustavy v kapitole 2.2.1 jsem změnil hlavně hlavním blokem *main*. Vyměnil jsem ho na blok *Cyclic interrupt*. Ten umožňuje vlastní nastavení cyklu bloku.

Nyní blok *Subsystem* reprezentuje vygenerovaný PLC kód PI regulátor ze Simulinku. Ten jsem vložil do nového hlavního bloku *Cyclic interrupt* s $T_s = 10$ ms.



Obr. 3.11: Funkční blok obsahující vygenerovaný PLC kód

Vstupní hodnoty do funkčního bloku jsou opět převedeny z bitů na amplitudu od 0 do 100 % (0 až 10 V) a výstupní hodnoty jsou opačným způsobem převedeny zpět (Funkční bloky *Serialize* a *Deserialize*).

3.3.2 Regulátor *PID compact*

Pro porovnání regulátoru jsem použil regulátor *PID compact*, z knihovních funkcí *TIA Portal*. Jedná se o technologický objekt, který poskytuje PID regulátor s integrovanou optimalizací. *PID compact* lze nastavit do automatického nebo manuálního režimu (Manuální režim umožňuje ručně zapsat hodnotu do procesoru a ihned vstoupit do platnosti). Algoritmus PID pracuje podle rovnice 3.7. [9]

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right] \tag{3.7}$$

Kde:

- a – Koeficient derivačního zpoždění
- b – Vyvažování proporcionálního zásahu
- c – Vyvažování derivačního zásahu
- y – Výstupní hodnota algoritmu PID
- x – Procesní hodnota
- w – Požadovaná hodnota
- K_p – Proporcionální zesílení
- T_I – Integrační časová konstanta
- T_I – Derivační časová konstanta
- s – Laplaceův operátor

Pro porovnání použiji tenhle regulátor ve třech režimech. Aby bylo možné regulátory srovnat, používám PI regulátor. Ten byl vypočten pro používanou soustavu. Poprvé nepoužiji jeho algoritmy pro výpočet parametrů a dosadím vypočtené hodnoty z kapitoly 3.1 optimálního modulu (Parametry a, b, a c jsou nastaveny na 1. To způsobí že regulátor by měl fungovat stejně jako regulátor vygenerovaný ze Simulinku). Následně využiji algoritmy (*Pretuning a Fine tuning*), pro výpočet nových parametrů PI. Počáteční hodnoty jsou opět vypočtené hodnoty optimálního modulu.

Pretuning

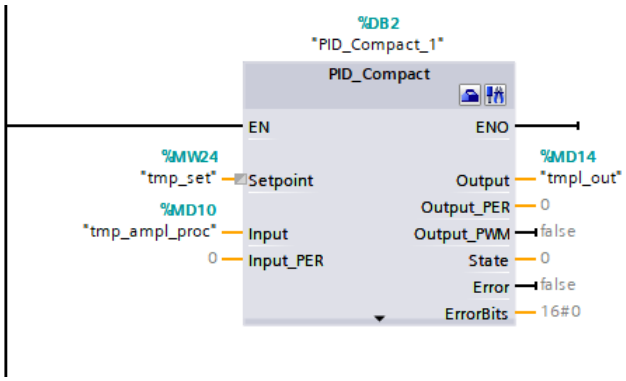
Stanoví odezvu procesu na skokovou změnu výstupní hodnoty a hledá inflexní bod. Parametry PID se poté vypočítají z maximální rychlosti nárůstu a doby průtahu řízeného systému. Čím stabilnější je procesní hodnota, tím snazší je výpočet parametrů PID. [9]

Fine tuning

Generuje konstantní omezené kmitání procesní hodnoty. Parametry PID jsou pro daný pracovní bod vyladěny na základě amplitudy a frekvence tohoto kmitání. Všechny PID parametry jsou přepočítány z výsledků. Parametry PID z *Fine tuning*

mají obvykle lepší vlastnosti nadřazeného řízení a rušení než parametry PID z *Pretuning*. [9]

PID compact implementován v programu vypadá následovně:



Obr. 3.12: Funkční blok *PID Compact*

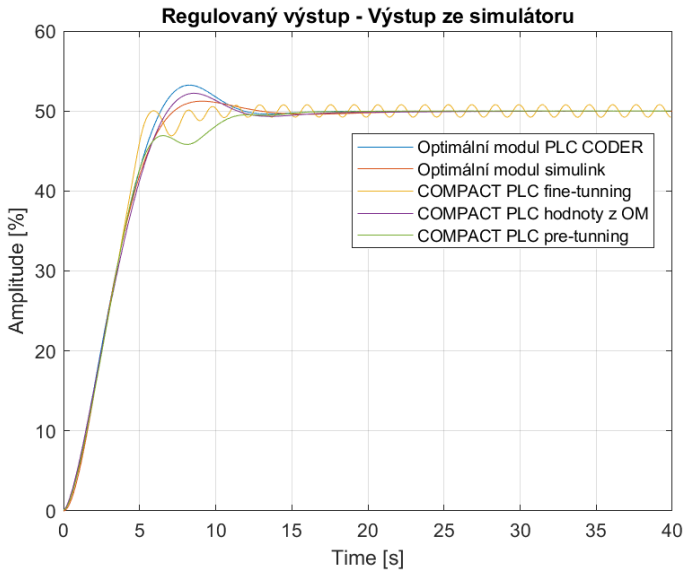
3.4 Výsledky regulace

Pro jednotlivé regulátory jsem změřil odezvu na změnu vstupní veličiny (Nastavená amplituda v PLC 50 %). Měření proběhlo pomocí funkce *trace*, tak jak je v kapitole 2.2.1 popsáno. Navíc jsem zde počítal odchylku e:

Obr. 3.13: Výpočet odchylky e

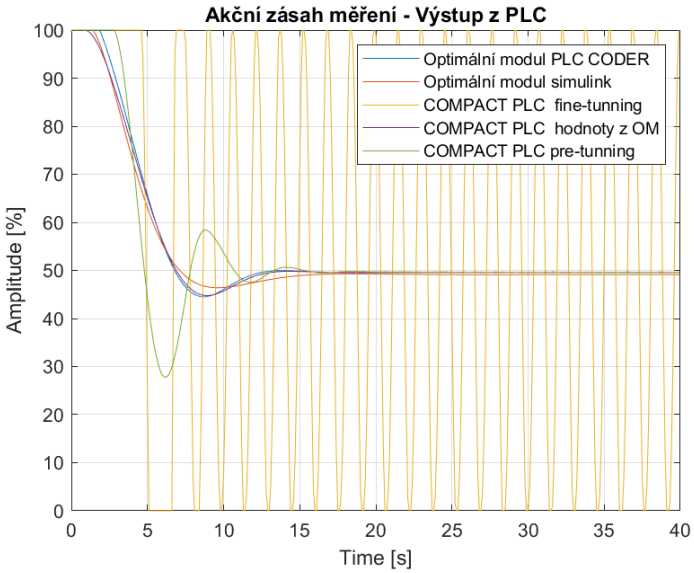
Další změna nastala pro vzorkovaní u regulátoru *PID compact* při používaní algoritmu *Fine tuning a Pretuning*. Regulátor nestihl počítat parametry a proto jsem musel periodu vzorkování snížit z 0,01 na 0,1 s.

V prvním grafu 3.14 lze vidět, že regulátor vytvořený pomocí PLC kódu má skoro stejný průběh jako regulátor simulován v Simulinku a regulátor *PI compact*. Regulátor *PI compact Pretuning* dokázal u regulovat soustavu, ale *PI compact Fine tuning* bohužel už ne. Hodnoty P a I z *PI compact Pretuning* a *Fine tuning* nelze porovnat, protože tyhle funkce už nastavují parametry a, b, a c.



Obr. 3.14: Výsledky regulace všemi regulátory

V dalším grafu 3.15, jsou data z výstupu PLC. To znamená akční zásah do simulátoru. Opět lze vidět že akční zásah *PI compact Pretuning* a *Fine tuning* má jiný průběh než ostatní regulátory. *Fine tuning* do nekonečna kmitá. To je způsobeno velice malou hodnotou integrační složky, kterou algoritmus nastavil.



Obr. 3.15: Výsledky akčního zásahu všech regulátoru

3.5 Kvalita regulace

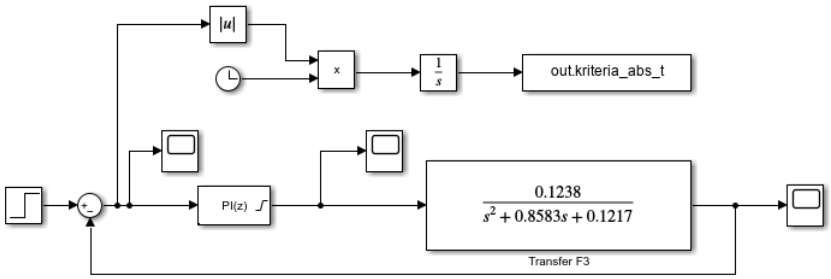
Pro porovnání regulátoru jsem použil hlavně ITAE (*Integral of Time multiplied by Absolute value of Error*) kritérium. ITAE je váhové kritérium. To znamená že váha odchylky narůstá lineárně s časem. Je definované vztahem: [5]

$$J_{ITAE} = \int_0^{\infty} |e(t) \; - \; e(\infty)| \, t \; dt \tag{3.8}$$

Výpočet ITAE jsem implementoval v Matlabu. Integrátor jsem nahradil sumou a tu jsem vynásobil časem vzorkovaní. Výsledná implementovaná rovnice:

$$J_{ITAE} = \; T_S \cdot sum(abs(e) \cdot t) \tag{3.9}$$

Data jsem vytáhl z měřených průběhů 3.14 od začátku měření až do 40 sekund, pro všechny regulátory. Pro ověření správnosti výpočtu jsem vytvořil ITAE kritérium i v Simulinku 3.16.



Obr. 3.16: Simulink schéma regulačního obvodu s výpočtem ITAE

Tab. 3.2: Kvalita regulace jednotlivých regulátorů

Druh regulátoru	**ITAE 40s** [-]	**Překmit** [%]	**Doba odezvy** [s]	**Doba regulace 1%** [s]
Simulink OM	418,45	2,45	7,27	11,51
PLC Coder OM	410,32	6,47	6,39	11,45
Compact PI OM	452,59	4,43	6,9	15,3
Compact PI pretunning	461,85	0,02	23,6	13,9
Compact PI fine tunning	681,84	1,6	5,9	-

Výsledné hodnoty ITAE a další důležité parametry jsou v tabulce 3.2. Mezi další zhodnocení regulace je zahrnuto: překmit, doba odezvy (čas kdy se dosáhlo amplitudy 50) a doba regulace pro 1 % (čas kdy nastalo ustálení pro ±1 % od amplitudy 50).

Z výsledků je patrné že PLC Coder má nejlepší ITAE, odezvu a dobu regulace. Má ale největší překmit. Podle grafu akčního zásahu 3.15, má PLC Coder skoro identický průběh jako Compact PI OM. Regulátor Compact PI má ale o něco rychlejší akční zásah. To způsobuje 2 % odlišný překmit.

Zajímavým výsledkem je Compact PI *pretunning*. Ten má minimální překmit a podle grafu regulace 3.14 dosáhl v podstatě ve stejné době jako ostatní regulátory požadované veličiny. Z grafu akčního zásahu 3.15 má ale dost kmitavý průběh.

Nejhůř je na tom regulátor Compact PI *fine tunning*. Jediný nedokázal u regulovat soustavu.

Závěr

Cílem téhle bakalářské práce bylo se seznámit s nástrojem *Simulink PLC coder* a simulátorem dynamických systémů. Dalším cílem bylo osadit a oživit desku plošných spojů simulátoru. Následně desku propojit s PLC, vytvořit regulační obvod, navrhnout regulátor na soustavu druhého řádu a vygenerovat pro regulátor PLC kód. Posledním cílem bylo řízení systému na PLC pomocí vygenerovaného kódu a knihovních funkcí PLC.

V práci je nejdříve seznámení s nástrojem *Simulink PLC coder* a simulátorem. Popsal jsem zde základní informace o nástroji a pak následný postup práce pro generování strukturovaného kódu. Postup obsahuje vytvoření subsystémů, nastavení důležitých parametrů, generování kódu a implementaci do *TIA Portal*. Dále je seznámení se simulátorem a jeho důležitými částmi, které jsou použité v prací.

V druhé kapitole je postupné oživení simulátoru. Při osazovaní desky jsem vyřešil chyby z výroby, viz obrázky 2.1 a 2.2. Vytvořil jsem pro simulátor podstavec pomocí programu *Siemens NX* a ten jsem 3D tiskem vytiskl. Následně jsem zkontroloval propojení PLC a simulátoru, kde jsem výsledné očíslování pinů uvedl v tabulce 2.1. Po oživení simulátoru a kontrole pinů, byla platforma připravená pro simulování navoleného přenosu 2.3. Vytvořil jsem projekt v *TIA Portal V15* pro měření systému a ověření funkčnosti. Měření jsem uskutečnil třikrát, pro tři různé odezvy výstupu na skok o různých velikostech. Výsledná naměřená data jsem předal do *System identification toolbox*, kde jsem systém zpětně identifikoval. Použil jsem metodu *Process model*, ze které výsledný přenos vyšel 2.4. Nakonec jsem přenosy porovnal a lze z grafu 2.11 vyčíst že vznikla malá odchylka. Ta může být způsobena nepřesností použitých součástek nebo nepřesným odhadem použitého toolboxu. V závěru kapitoly jsem opravil koeficient tlumení, tak aby byl větší než jedna a soustava byla přetlumena. Toho jsem docílil přepájením rezistoru R79 o hodnotě 620 $k\Omega$ na menší hodnotu 1 $k\Omega$.

V poslední kapitole je návrh regulátoru metodou optimálního modulu. Pro návrh jsem použil tabulku 3.1 a identifikovaný přenos 2.7. Vypočetl jsem přenos řízení s navrženým regulátorem 3.2 a vynesl jeho frekvenční charakteristiku 3.1. Navržený regulátor jsem nastavil v Simulinku pomocí bloku *Discrete PID Controller* a pomocí teorie z kapitoly 1.1 jsem vygeneroval PLC kód. Vygenerovaný kód jsem exportoval do PLC a vytvořil funkční blok, který reprezentoval regulátor 3.11. Jako další regulátor jsem implementoval knihovní funkci z PLC *PID compact*, kdy jsem do ní natavil hodnoty z navrženého regulátoru 3.6. Poslední dva regulátory jsem vytvořil pomocí algoritmu *PID compact Pretuning a Fine tuning*. Z veškerými regulátory jsem měřil odezvu výstupu na skok o hodnotě 5 V (amplituda 50 %). Všechny regulované výstupy a akční zásahy jsem zobrazil obrázcích 3.14 a 3.15. Pro zhodnocení

jednotlivých regulátoru jsem použil ITAE kritériu a další parametry pro vyhodnocení kvality regulace. Veškeré výsledky jsem uvedl v tabulce 3.2. Z výsledků je patrné, že regulátor vygenerovaný pomocí PLC kódu má nejlepší výsledky. Má také největší překmit, a to může být způsobeno implementovanou metodou *anti-windup* a nastaveným koeficientem Kb. Ten má vliv na překmit viz obrázek 3.5. Zjistil jsem také že algoritmy *Pretuning a Fine tuning* pro tenhle konkrétní přenos 2.7, slouží pouze pro přednastavení parametrů regulátoru a neslouží jako finální nastavení. Po použití algoritmů by se měli parametry doladit.

Z téhle bakalářské práce jsem navrhl laboratorní úlohu. Její hlavní záměr je použití PLC jako regulátor a simulátor jako soustavu. Student má v úloze za úkol navrhnout regulátor, vytvořit regulační smyčku pomocí PLC a simulátoru, implementovat regulátor a následně regulovat soustavu na zadané požadavky. Celá úloha obsahuje dokument *Regulace v PLC* v příloze C, který slouží jako návod do cvičení. Dále jsem do přílohy připravil projekt v *TIA Portal V15*, který studenti potřebují k úloze a naměřená data ze kterých se identifikovali přenosy pro úlohu.

Literatura

[1] *Get Started with Simulink PLC Coder* [online]. MATHWORKS, 2022. [cit. 09. 06. 2022]. Dostupné z URL: <https://www.mathworks.com/help/plccoder/getting-started-with-simulink-plc-coder.html>.

[2] *Simulink User's Guide* [online]. MATHWORKS, 2022. [cit. 09. 06. 2022]. Dostupné z URL: <https://www.mathworks.com/help/pdf_doc/simulink/>.

[3] PACAL, Stanislav. *Řízení reálného systému pomocí PLC s Využitím automaticky generovaného kódu* [online]. Brno, 2022 [cit. 09. 06. 2022]. Dostupné z URL: <https://www.vut.cz/studenti/zav-prace/detail/142565>. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Miroslav Jirgl.

[4] *System Identification Toolbox: User's Guide* [online]. MATHWORKS, 2022. [cit. 09. 06. 2022]. Dostupné z URL: <https://www.mathworks.com/help/pdf_doc/ident/ident_ug.pdf.

[5] BLAHA, Petr a Petr VAVŘÍN. *Řízení a regulace I: Základy regulace lineárních systémů - spojité a diskrétní.* Verze 1.3.9 [cit. 22. 04. 2023] .

[6] SLOVÁK, Tomáš a Zdeněk RIEDL. *Syntéza: Metody syntézy* [online] VŠB-Technická univerzita Ostrava FAKULTA STROJNÍ Katedra automatizační techniky a řízení, 2003 [cit. 25. 04. 2023] . Dostupné z URL: <http://books.fs.vsb.cz/SyntezaReg/text0305.htm.

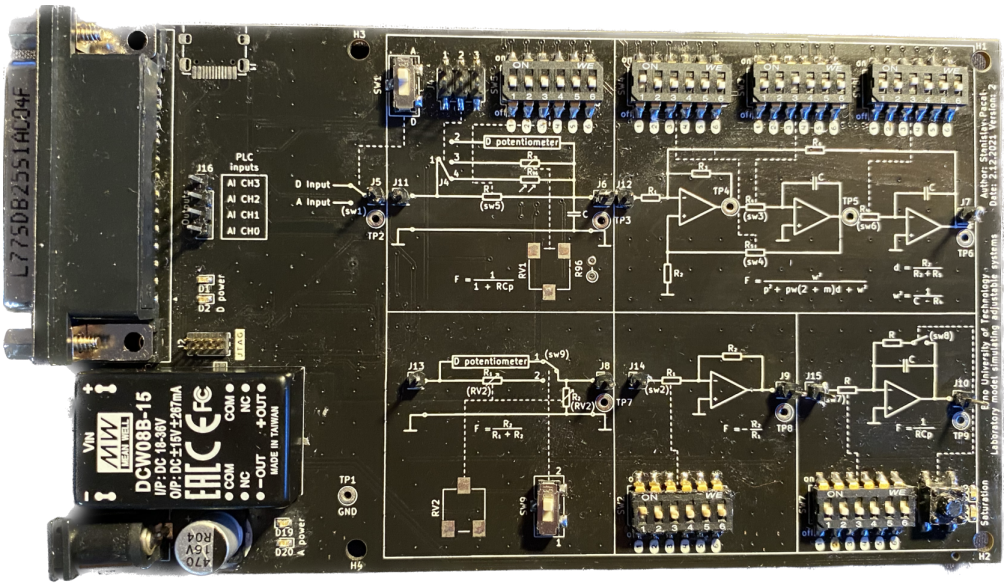
[7] ŠOLC, František, Pavel VÁCLAVEK a Petr VAVŘÍN. *Řízení a Regulace II: Analýza a řízení nelineárních systémů.* Verze 1.27. 4.ledna 2011 [cit. 23. 04. 2023].

[8] *Simulink* [online]. MATHWORKS, 2023. [cit. 23. 04. 2023]. Dostupné z URL: <https://www.mathworks.com/help/simulink/slref/discretepidcontroller.html?s_tid=doc_ta>.

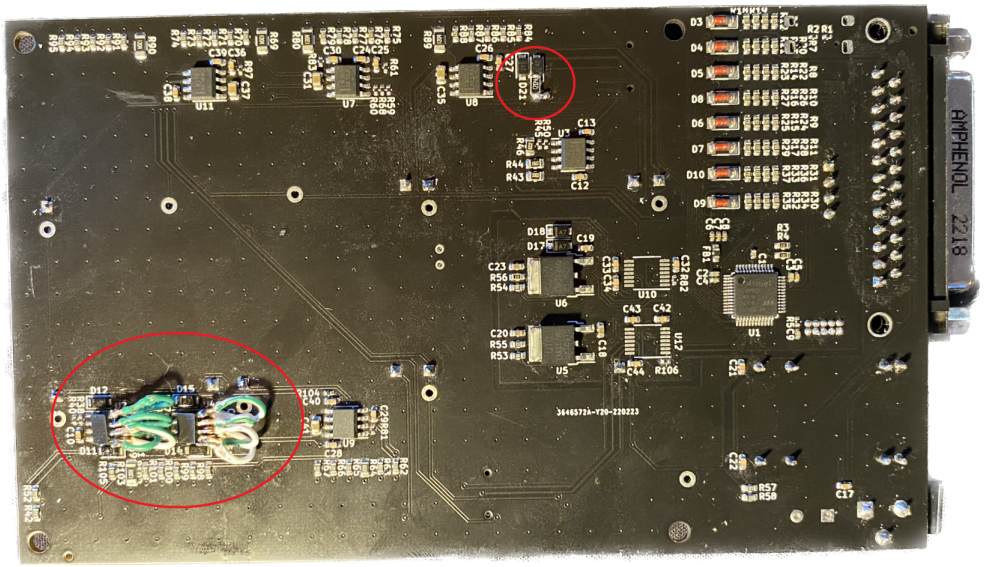
[9] SIEMENS FUNCTION MANUAL SIMATIC S7-1200, S7-1500 *PID control* [online] 11/2022. [cit. 29. 04. 2023]. <https://support.industry.siemens.com/cs/attachments/108210036/s71500_pid_control_function_manual_en-US_en-US.pdf?download=true.

Seznam příloh

A Plně osazena DPS



Obr. A.1: Horní strana desky



Obr. A.2: Spodní strana desky

B Vygenerovaný PLC kód

```
(*
 *
 * File: pid_reg.scl
 *
 * IEC 61131-3 Structured Text (ST) code generated for subsystem "pid_reg/Subsystem"
 *
 * Model name                     : pid_reg
 * Model version                  : 1.3
 * Model creator                  : xmusil72
 * Model last modified by         : xmusil72
 * Model last modified on         : Wed Apr 05 02:02:35 2023
 * Model sample time              : 0s
 * Subsystem name                 : pid_reg/Subsystem
 * Subsystem sample time          : 0.01s
 * Simulink PLC Coder version     : 3.7 (R2022b) 13-May-2022
 * ST code generated on           : Thu Apr 20 00:20:17 2023
 *
 * Target IDE selection           : Siemens TIA Portal: Double Precision
 * Test Bench included            : No
 *
 *)
FUNCTION_BLOCK Subsystem
VAR_INPUT
    ssMethodType: SINT;
    In_reg: LREAL;
    In_zpet: LREAL;
END_VAR
VAR_OUTPUT
    Out_con: LREAL;
END_VAR
VAR
    Integrator_DSTATE: LREAL;
    rtb_Sum: LREAL;
    rtb_Sum_e: LREAL;
END_VAR
CASE ssMethodType OF
    0:
        (* SystemInitialize for Atomic SubSystem: '<Root>/Subsystem' *)
        (* InitializeConditions for DiscreteIntegrator: '<S34>/Integrator' *)
        Integrator_DSTATE := 0.0;
        (* End of SystemInitialize for SubSystem: '<Root>/Subsystem' *)
    1:
        (* Outputs for Atomic SubSystem: '<Root>/Subsystem' *)
        (* Sum: '<S1>/Sum' *)
        rtb_Sum := In_reg - In_zpet;
        (* Sum: '<S43>/Sum' incorporates:
         *  DiscreteIntegrator: '<S34>/Integrator'
         *  Gain: '<S39>/Proportional Gain' *)
        rtb_Sum_e := (1.99 * rtb_Sum) + Integrator_DSTATE;
        (* Saturate: '<S41>/Saturation' *)
        IF rtb_Sum_e > 100.0 THEN
            Out_con := 100.0;
        ELSIF rtb_Sum_e >= 0.0 THEN
            Out_con := rtb_Sum_e;
        ELSE
            Out_con := 0.0;
        END_IF;
        (* End of Saturate: '<S41>/Saturation' *)

        (* Update for DiscreteIntegrator: '<S34>/Integrator' incorporates:
         *  Gain: '<S27>/Kb'
         *  Gain: '<S31>/Integral Gain'
         *  Sum: '<S27>/SumI2'
         *  Sum: '<S27>/SumI4' *)
        Integrator_DSTATE := ((((Out_con - rtb_Sum_e) * 2.0) + (0.352 * rtb_Sum)) * 0.01) + Integrator_DSTATE;
        (* End of Outputs for SubSystem: '<Root>/Subsystem' *)
END_CASE;
END_FUNCTION_BLOCK
```

Obr. B.1: Vygenerovaný PLC kód pro *Discrete PID Controller* ze schématu 3.6

C Laboratorní úloha *Regulace v PLC*

PDF verze navržené laboratorní úlohy.

Laboratorní cvičení

Regulace pomocí PLC

Autor: Viktor Musil

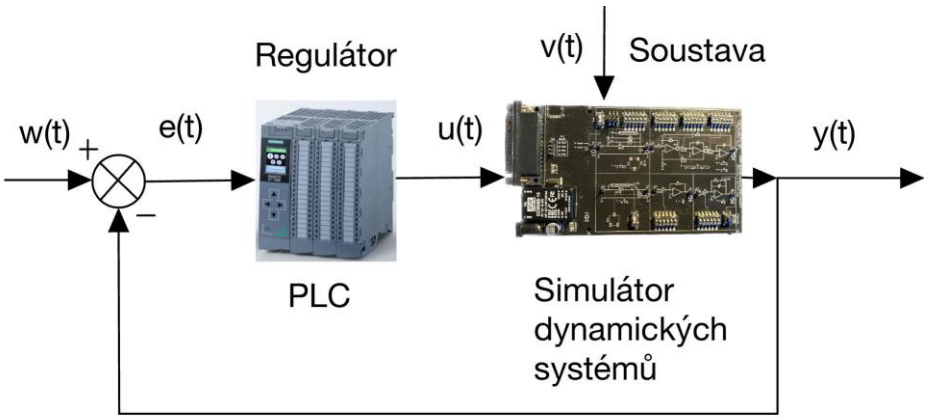
Verze: 1.1-2023

1. Teoretický rozbor

Hlavní záměr úlohy je sestrojit regulační smyčku pomocí PLC a simulátoru dynamických systémů. Následně PLC využít jako regulátor a regulovat soustavu druhého řádu.

1.1. Regulační smyčka

Pro následující úlohu bude platit následující regulační smyčka:



Obrázek 1 Principiální regulační smyčka

Kde je: *w* – požadovaná hodnota

e – regulační odchylka (Pozn. Regulační odchylka se počítá v PLC)

u – akční zásah

v – porucha

y – regulovaná veličina

Poruchu v téhle úloze nebudeme simulovat. Budeme využívat regulovanou veličinu *y(t)* (výstup ze simulátoru), požadovanou hodnotu *w(t),* akční zásah *u(t)* (výstup z regulátoru) a dopočteme regulační odchylku *e(t) (e = w – y).*

1.2. Regulovaná soustava

V téhle úloze bude regulovaná soustava systém druhého řádu. Pro takový systém platí:

$$F_s(p) = \frac{K_0}{T^2p^2+2\xi Tp+1} \qquad (1)$$

Kde je: T … časová konstanta

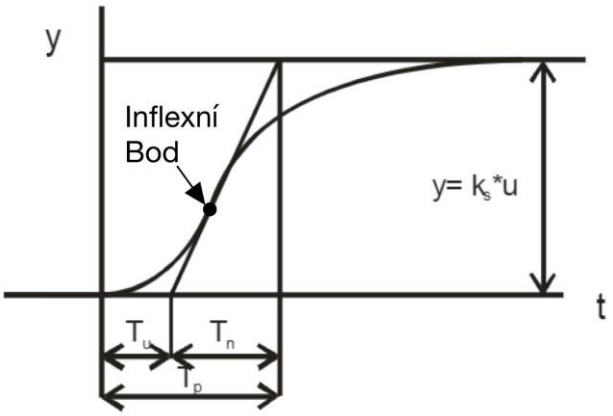
K_0 … zesílení soustavy

ξ … koeficient tlumení

Pokud je koeficient tlumení $\xi \in \langle 0, 1)$ tak platí, že soustava je kmitavá. Když je $\xi \geq 1$ tak je soustava přetlumená a nekmitá. Poslední možnost, pokud je $\xi = 1$ tak je soustava ne mezi aperiodicity.

Z přechodové charakteristiky, která se získá z odezvy na skokovou změnu vstupní veličiny lze identifikovat parametry: zesílení ***K***, doba průtahu T_u, doba náběhu T_n (Doba náběhu je definována jako časový interval mezi okamžikem, kdy se odezva začne pohybovat od 10 % na výsledné amplitudě a

okamžikem, kdy dosáhne 90 % výsledné amplitudy.) a doba přechodu T_p. Odečtení parametrů je možné pomocí tečny v inflexním bodě.



Obrázek 2 Určení parametrů přechodové charakteristiky pomocí tečny v inflexním bodě

1.3. PID regulátor (PLC)

Pro řízení LTI systémů se využívá standartní lineární PID regulátor (resp. PSD). Jedná se o proporcionálně (P) integračně (I) derivační (D) regulátor. Pro časové průběhy regulátoru platí:

$$x(t) = r_0 e(t) + r_d \frac{de(t)}{dt} + r_i \int_0^t e(\tau) d\tau + x(0) \qquad (2)$$

Pokud jsou nulové počáteční podmínky *x(0) = 0*, tak platí:

$$F_R(p) = \frac{X(p)}{E(p)} = r_0 + \frac{r_i}{p} + r_d p = \boldsymbol{K_P}\left(1 + \boldsymbol{T_D} p + \frac{1}{\boldsymbol{T_I} p}\right) \qquad (3)$$

Mezi konstanty v rovnici platí:

$$\boldsymbol{K_P} = r_0 \,, \boldsymbol{T_D} = \frac{r_d}{r_0} \,, \boldsymbol{T_I} = \frac{r_0}{r_i} \qquad (4)$$

Pro přehlednější a jednodušší řízení regulátoru se používá přenos z rovnice *(3)*, s parametry K_P, T_D a T_I. [1]

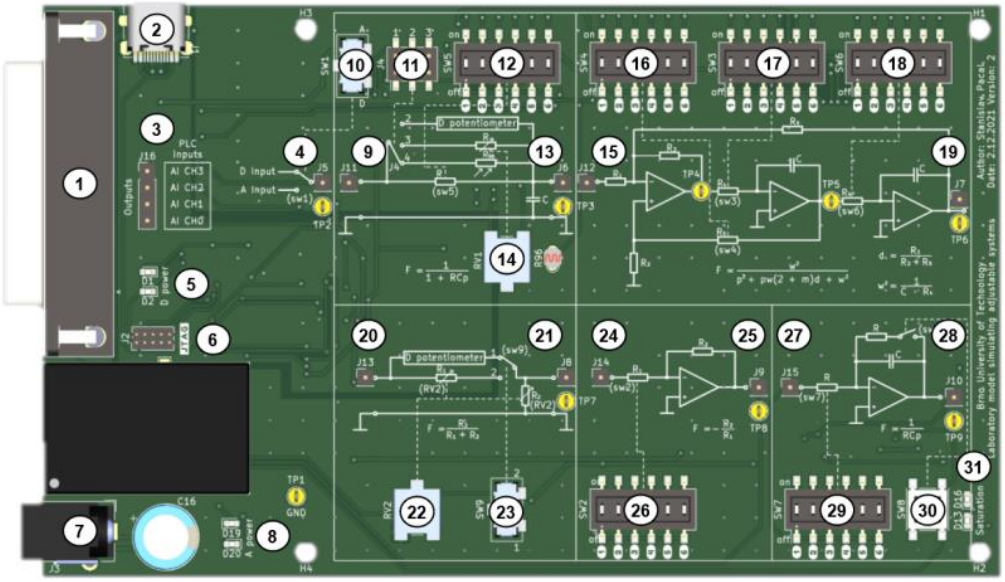
2. Popis SDS (Simulátor dynamických systémů)

Simulátor slouží jako laboratorní přípravek na simulování jednoduchých technologických procesů. Je složen z analogové a digitální části. Analogová část má 5 modulů a každý z nich simuluje odlišnou dynamiku. [2]

- PT1 člen – systém prvního řádů s nastavitelnou konstantou
- PT2 člen – systém druhého řádů s nastavitelným zesílením a tlumením
- P člen – proporcionální člen s nastavitelným zesílením
- I člen – systém s astatismem a nastavitelnou rychlostí integrace
- Rozdílový člen – simulátor poruchy působící na výstupu systému

Digitální část slouží pro vytvoření libovolného vstupního signálu, měření výstupních hodnot, nastavování podílu rozdílového členu a ovlivňování dynamiky setrvačného členu prvního řádu. Digitální část v téhle úloze nebudete potřebovat. [2]

Pro pochopení desky je následující popis ovládacích prvků:



Obrázek 3 Vrchní část SDS [2]

I/O část platformy	
1	D-SUB konektor sloužící 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*

Tabulka 1 Popis ovládacích prvků [2]

2.1. Setrvačný nebo kmitavý člen druhého řádu (PT2)

Je složen ze tří operačních zesilovačů. Ty realizují přenosové funkce druhého řádu s nastavitelnými parametry zesílení ω a tlumení d. Přenosová rovnice je:

$$F(p) = -\frac{\omega^2}{p^2+p\cdot 3\cdot\omega\cdot d+\omega^2} \qquad (5)$$

Kde ω je parametr frekvence a *d* je tlumení. [2]

Pomocí přepínačů SW4, SW3 a SW6 se jednotlivé parametry nastavují a lze vytvářet různé variace přenosu. SW4 nastavuje parametr tlumení a pro tuhle úlohu bude vždy v poloze 5. Tímto nastavení

bude tlumení vždy větší než 1. Vy budete nastavovat SW3 a SW6 dle libosti (**musí být vždy stejně nastaveny!** Jinak se tvoří paralelní kombinace odporů) do polohy 2, 3 nebo 4. Tyhle switche představují nastavení ω.

2.2. Proporcionální člen

Protože výstup z PT2 je záporný, využijeme i proporcionální člen. Je tvořený invertujícím zapojením operačního zesilovače s nastavitelnými parametry. Pokud se nastaví zesílení *K* = 1, stane se z členu invertor signálu. Přenosová rovnice tohoto členu je:

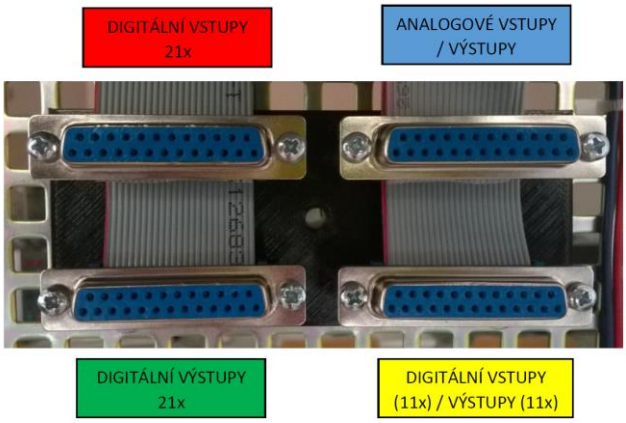
$$F(p) = -\frac{R_2}{\Delta R_1} \qquad (6)$$

Kde R_2 je odpor o hodnotě 22 kΩ a ΔR_1 je nastavovací odpor na přepínači SW2 (22k, 8k2, 16k, 36k, 220k, 470k Ω). [2]

SW2 nastavte na hodnotu 1 a tím hodnota přenosu bude 1 a změní pouze znaménko Vaší soustavě.

3. Popis stanoviště

Pracoviště je vybaveno PLC S7-1500 (typ 1512C -1 PN), které obsahuje 1x AI/AO kartu – 5 x AI, 2 x AO, 2x DI/DO kartu – 2 x (16 x DI, 16 x DO) a dále dotykovým panelem TP 700 Comfort. Všechny I/O jsou vyvedeny na konektory CANON25.



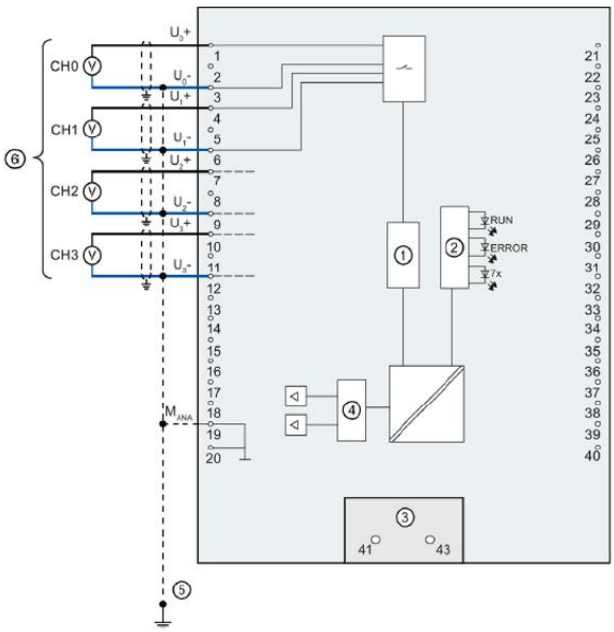
Obrázek 4 Konektory z PLC

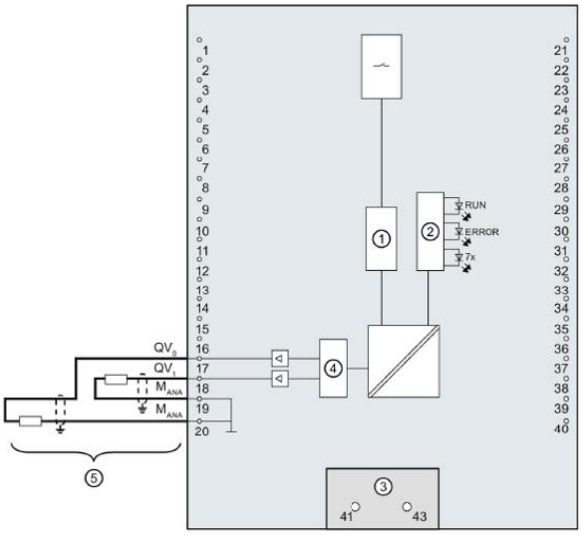
Zapojení analogového konektoru má propojení s PLC podle následující tabulky:

Žíla kabelu	Karta	Číslo svorky (karta)	Číslo pinu (konektor)
1	AI/AO	1	1
2	AI/AO	2	14
3	AI/AO	3	2
4	AI/AO	4	15
5	AI/AO	5	3
6	AI/AO	6	16
7	AI/AO	7	4
8	AI/AO	8	17
9	AI/AO	9	5
10	AI/AO	10	18
11	AI/AO	11	6
12	AI/AO	12	19
13	AI/AO	13	7
14	AI/AO	14	20
15	AI/AO	15	8
16	AI/AO	16	21
17	AI/AO	17	9
18	AI/AO	18	22
19	AI/AO	19	10
20	AI/AO	20	23
			11
			24
			12
			15
			13

Tabulka 2 Analogové vstupy / výstupy

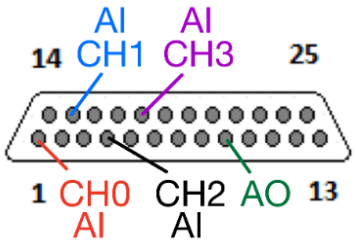
Analogový vstup / výstup na kartě:





Obrázek 5 Analogový vstup (vlevo) a výstup (vpravo)

Rozmístění pinu na simulátoru:



Obrázek 6 I/O pinů na konektoru simulátoru

Pokud jednotlivé propojení srovnáme dostaneme tabulku:

I/O kanály	PLC	SDS
I CH0	1,2	1
I CH1	15,16	15
I CH2	4,5	4
I CH3	18,19	18
O CH0	9,22,10,23	9

Tabulka 3 Porovnání I/O konektoru PLC a SDS

Nyní můžeme nadefinovat v TIA PORTAL jednotlivé vstupy a výstupy simulátoru.

4. Projekt v TIA PORTAL

Pro cvičení máte nachystaný projekt v TIA PORTAL, který už má nadefinované *tags* (přiřazení k adrese zařízení/PLC neboli proměnné):

Default tag table

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...
1	Analog_Input_CH0_0-10V	Int	%IW0	☐	☑	☑	☑
2	Analog_Output_0-10V	Int	%QW0	☐	☑	☑	☑
3	tmp_input	Real	%MD2	☐	☑	☑	☑
4	input_v_procentech	Real	%MD6	☐	☑	☑	☑
5	output_v_procentech	Real	%MD10	☐	☑	☑	☑
6	tmp_output	Real	%MD14	☐	☑	☑	☑
7	start	Bool	%M18.0	☐	☑	☑	☑

Obrázek 7 Tags

Kde: *Analog_Input_CH0_0-10V* – Vstup do PLC, Výstup z SDS

Analog_Output_0-10V – Výstup z PLC, Vstup do SDS

tmp_input – Pomocná proměnná pro funkci *Serialize*

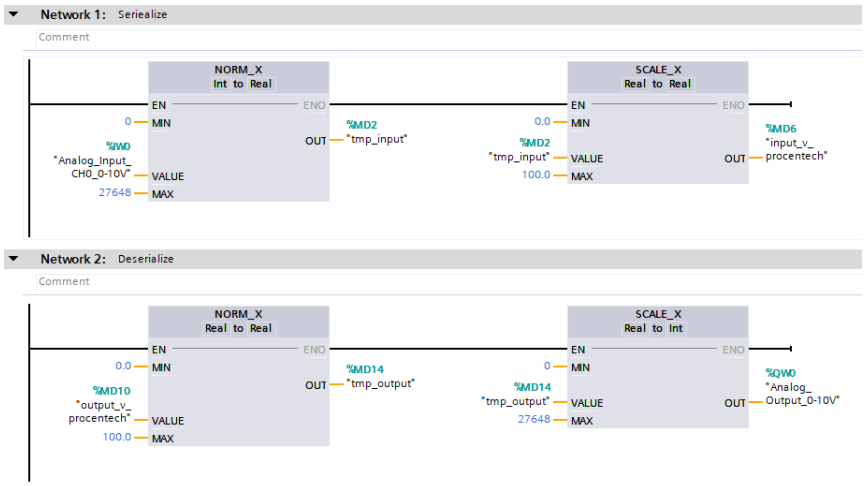
input_v_procentech – Vstup do PLC v procentech (***u* – akční zásah**).

output_v_procentech – Výstup z PLC v procentech (***y* – regulovaná veličina**).

tmp_output – Pomocná proměnná pro funkci *Deserialize*.

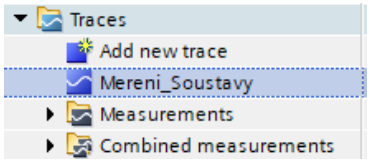
start – *Bool* proměnná pro odstartovaní měření.

Poté máte funkci *Serialize* v bloku *main*, která převádí výstupní signál ze simulátoru na 0 až 100% (0 až 10 V). Opačně je zde funkce *Deserialize.*



Obrázek 8 Funkce Serialize a Deserialize

Jako poslední máte nachystanou funkci *trace*. Pomocí téhle funkce budete schopni změřit odezvy na definovaný vstupní signál pro Vámi zvolený přenos. Najdete ji v levé části v záložce *Traces.*



Obrázek 9 Traces

Je nastavená tak aby měřila akční zásah *a* a regulovanou veličinu *y*. Dále má nastavené, aby vzorkovala podle bloku *main* (To znamená že zaznamená hodnotu při každém cyklu *main* → 1ms) a měření se spustilo při aktivovaní proměnné *start*.

***Tagy, Serialize a Deserialize* nemodifikujte**. Pro práci využijte nadefinované tagy.

Po úpravě kódu musíte program pokaždé nahrát do fyzického PLC a následně přejít do režimu online. Je důležité, aby se nahrání provedlo do správného PLC, to zkontrolujete adresou IP. Na každém stanovišti je štítek s označením pracoviště a příslušnými IP adresami. Pro nahrání je v horní liště tlačítko *Download to device* a tlačítko pro přechodu režimu online *Go online*:

Obrázek 10 Download to device, Go online

5. Úkoly

V rámci tohoto cvičení budete řídit Vámi zvolenou soustavu na SDS. Vše budete ovládat v přichystaném projektu v TIA PORTAL. Přenosy soustav představují zjednodušené modely ohřevu vody v nádrži. Pro možnost realizace v laboratorních podmínkách jsou konstanty modelů mnohonásobně menší než v reálných podmínkách. Požadavky jsou:

- Regulace byla na konstantní hodnotě 50 °C (jednotky jsou přepočítané z výstupního napětí SDS na amplitudu 0 až 100 °C)
- Nulová ustálená odchylka e (max ± 1 %)

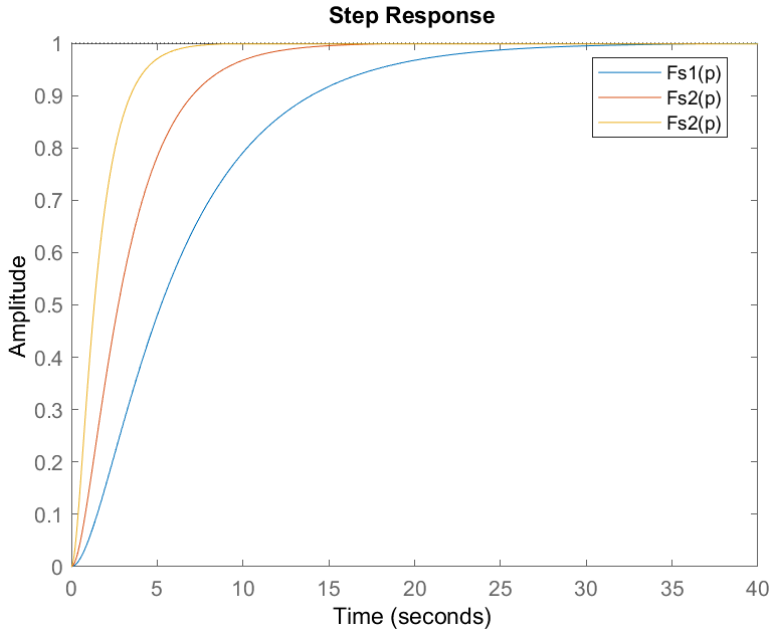
- Maximální rychlost přechodného děje.

Pro jednoduchost si můžete vybrat jednu ze tří soustav viz následující Tabulka 4. Pro tyto soustavy již byla provedena identifikace parametrů přenosu a zároveň vykreslena přechodová charakteristika (Obrázek 11).

Nastavení switchu na SDS	Identifikovaný přenos
SW4 = 5, SW3 a SW6 = 2	$F_{S1}(p) = \frac{1}{7{,}34p^2 + 6{,}75p + 1}$
SW4 = 5, SW3 a SW6 = 3	$F_{S2}(p) = \frac{1}{2{,}19p^2 + 3{,}49p + 1}$
SW4 = 5, SW3 a SW6 = 4	$F_{S3}(p) = \frac{1}{0{,}52p^2 + 1{,}69p + 1}$

Tabulka 4 Identifikované přenosy pro dané nastavení přepínačů

Pro lepší představu jsou zde přechodové charakteristiky jednotlivých systémů:



Obrázek 11 Přechodové charakteristiky jednotlivých přenosů z Tabulka 4

1) Pomocí vodičů propojte SDS a nastavte přepínače na Vámi zvolený přenos. V TIA PORTAL blokem *move* nastavte na vstup simulátoru amplitudu 50 a změřte funkcí *trace* přechodovou charakteristiku.
2) Výsledný průběh si uložte do .csv souboru. Z naměřených dat vykreslete změřenou přechodovou charakteristiku a porovnejte s teoretickou – viz Obrázek 11, a zjistěte jeho základní parametry.
3) Vámi vybranou metodou navrhněte parametry PI/PID regulátoru a výsledné parametry nastavte v bloku PID Compact. Průběh změřte a popřípadě dolaďte parametry, dokud průběh bude odpovídat výše uvedeným požadavkům.

6. Domácí příprava

1) Prostudujte u simulátoru členy: Setrvačný nebo kmitavý člen druhého řádu, Proporcionální člen.

2) Prostudujte/zopakujte si způsob identifikace systému z přechodové charakteristiky. Nakreslete si přechodovou charakteristiku setrvačného systému 2. řádu a vyznačte hodnoty základních parametrů (viz Teoretický rozbor).
3) Prostudujte si základní princip PID regulace a nakreslete odezvy jednotlivých složek – P, I a D na jednotkový skok.
4) Zvolte si přenos z tabulky výše a navrhněte pro něj PI/PID regulátor, Vámi vybranou metodou (Doporučené metody viz Přílohy). Navrhnutý regulátor vyzkoušejte v Simulink (vytvořte regulační smyčku) a ověřte, zda obvod splňuje požadavky.
   Pozn. Pokud budete navrhovat regulátor pomocí metody Ziegler-Nichols, přechodovou charakteristiku můžete simulovat v Simulink nebo pomocí funkce *step* v Matlab. Odtud odečtěte parametry T_n a T_u.

7. Doporučený postup

7.1. Úkol č. 1:

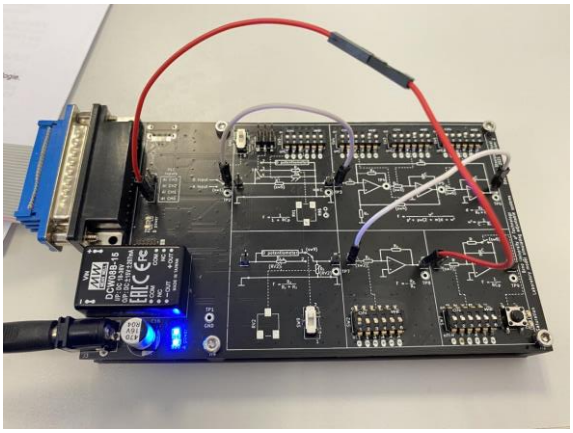
1) Propojte simulátor s PLC do konektoru na analogové I/O. Na simulátoru pomocí vodičů propojte jednotlivé složky následovně:
   - Vstupní signál z PIN J5 (4, očíslovaní v Obrázek 3) → Vstupní PIN J12 (15)
   - Výstupní PIN J17 (19) → Vstupní PIN J14 (24)
   - Výstupní PIN J9 (25) → PIN header J16 na AI CH0 (3)

   Nastavte přepínače:

   - SW4 (16), SW3 (17) a SW6 (18) do poloh podle Vámi zvoleného přenosu.
   - SW2 (26) do polohy 1 – invertor signálů.
   - SW1 (10) do polohy nahoru – vstupní signál generovaný pomocí PLC (analogový režim)

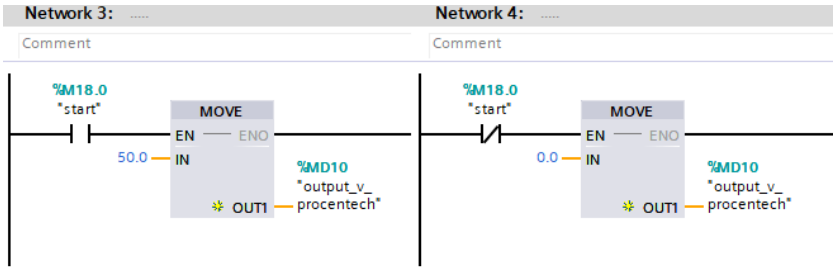
   Připojte napájení SDS.

   Výsledné zapojení SDS je následovné:



Obrázek 12 Propojení vodičů na SDS

2) V TIA PORTAL přejděte do hlavní bloku *main* a do nového *networku* přidejte funkci *move*. Funkce naleznete v pravé části v záložce Instructions . Před blok vložte otevřený kontakt, který bude ovládán *tagem start*. Na vstup bloku *move* nastavte požadovanou hodnotu 50 a na výstup *tag output_v_procentech*. To stejné udělejte v dalším *networku* ale požadovaná hodnota bude 0 a před blokem bude uzavřený kontakt. Výsledná implementace bude vypadat následovně:



Obrázek 13 Implementace bloků move

Tahle realizace představuje přivedení skoku o hodnotě 5V na simulátor.

3) Nahrajte program do PLC a přejděte do režimu *online.* Otevřete připravenou funkci *trace mereni_soustavy.* Funkce nemusí být nahrána v PLC (poznáte podle zeleného kruhu vedle bloku) a proto budete muset levým tlačítkem ji nahrát. Ve stejné liště klikněte na režim monitorování a následně na zapnutí měření . Nyní funkce měří a čeká na *trigger* proměnné *start.*

4) V bloku *main* přejděte do režimu monitorování tlačítkem , který naleznete ve vrchní liště. Nyní můžete vidět aktuální hodnoty proměnných a stavy jednotlivých *networků.* Pravým tlačítkem klikněte na otevřený kontakt *start* a pomocí *modife value* nastavte hodnotu na 1. Tím nastavíte kontakt na sepnutý a přivedete na simulátor nastavenou hodnotu 50 °C. Spustí se tím i měření a uvidíte aktuální průběh.

7.2. Úkol č. 2:

1) Výsledný průběh uložte pomocí tlačítka ve formátu csv. V Matlabu data přečtěte a zobrazte průběh. Funkce *trace* měří maximálně 32766 hodnot.

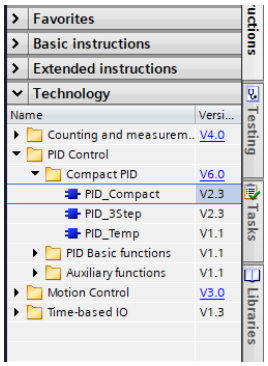
```
Fs = csvread('FILENAME.csv',1,1); % precteni souboru
VYSTUPZESIMULATORU = Fs(:,2); % Nacteni hodnot
time = 0:0.001:32.765; % prirazeni casu ke kazde hodnote podle vzorkovani
plot(time,VYSTUPZESIMULATORU); % vykresleni prubehu
```

Obrázek 14 Zobrazení průběhů v Matlab

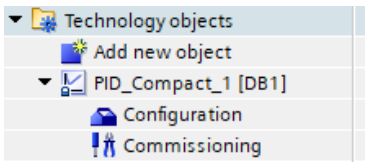
2) Výsledný průběh z Matlabu uložte a odečtěte z něho základní parametry.

7.3. Úkol č. 3:

1) Odstraňte předchozí vytvořené dva *networky* a vložte do nového *PID Compact*. Naleznete ho opět v pravé části v záložce *Instructions*. *PID Compact* je technologický objekt a přidá se Vám také v levé části v sekci *Technology objects.*

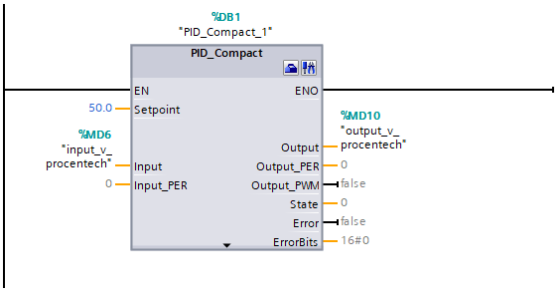


Obrázek 15 Umístění PID Compact



Obrázek 16 Technology objects

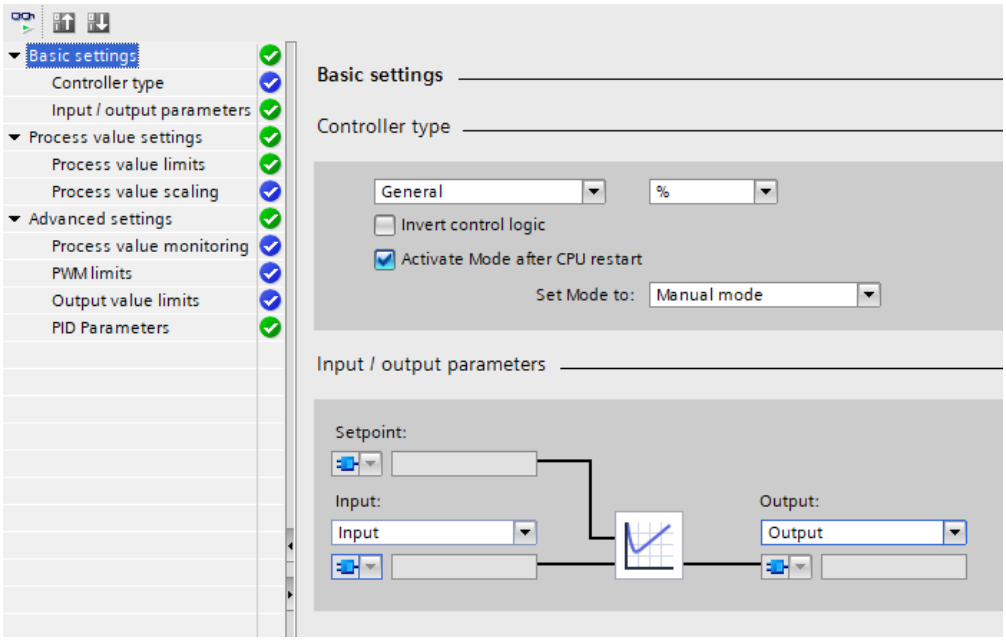
2) Nastavte PID Compact následovně:



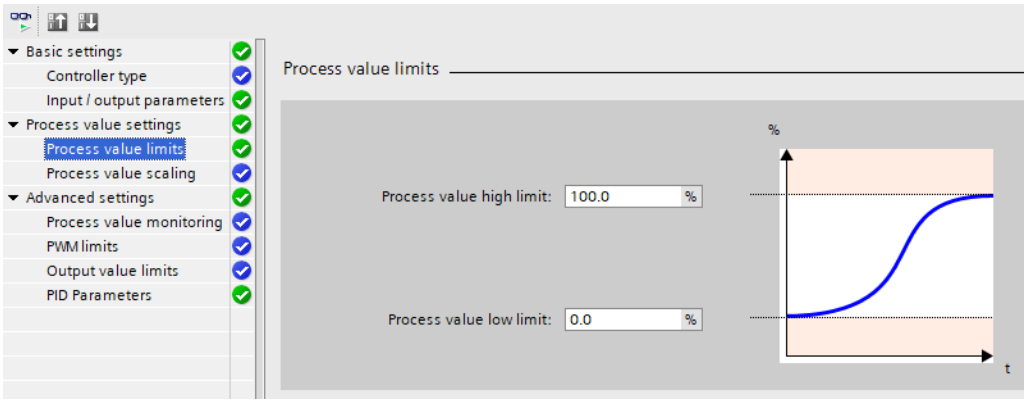
Obrázek 17 PID Compact

Jako vstup do regulátoru jde regulovaná veličina a výstup je akční zásah. *Setpoint* je požadovaná veličina. Odchylka *e* se počítá přímo v regulátoru.

3) V levé části otevřete Configuration a nastavte jednotlivé sekce následovně:

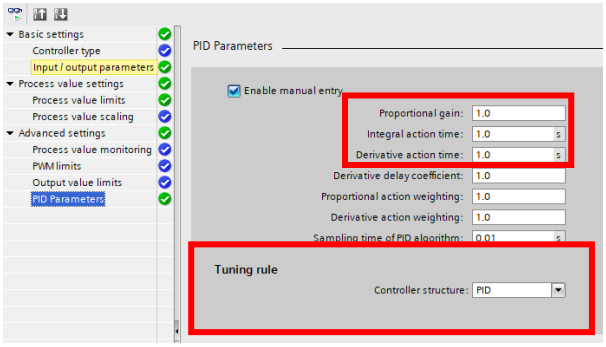


Obrázek 18 Nastavení Basic settings v PID Compact



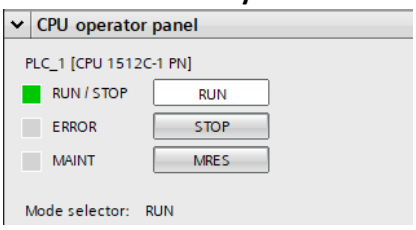
Obrázek 19 Nastavení Process value limits v PID Compact

4) Vypočtené hodnoty regulátoru nastavte v *PID Compact*. Regulátor umožňuje nastavení vah, které ovlivňují parametry. Ty nechte vždy rovny 1. Nastavujte pouze zvýrazněné hodnoty. Pokud máte PI regulátor přepněte *Controller structure* na PI.

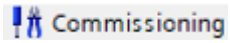


Obrázek 20 Parametry PID Compact

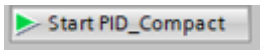
5) Pokud máte vše nastavené program opět nahrajte do PLC a přejděte do režimu *online*. Při prvním nahraní *PID Compact* je zapotřebí přepnout PLC do režimu STOP a následně do RUN. To provedete vpravo nahoře. **Musíte být v režimu *online*.**



Obrázek 21 RUN/STOP PLC

Otevřete 

a klikněte na *monitoring* režim  , který je vlevo nahoře. Nyní můžete zapnout měření které je také vlevo nahoře.

Nakonec zapněte regulátor 

. Změřte odezvu regulačního obvodu společně s akčním zásahem a udělejte screenshot obou průběhů.

6) Zamyslete se nad funkcí jednotlivých složek PID regulátoru při regulaci soustavy a pokuste se doladit regulátor tak, abyste získali co nejlepší odezvu z pohledu požadavků v zadaní.

8. Reference

[1] P. Blaha a P. Vavřín, „Základy regulace lineárních systémů - spojité a diskrétní,“ *Řízení a regulace I* , Verze 1.3.9.

[2] P. Stanislav, „Řízení reálného systému pomocí PLC s Využitím automaticky generovaného kódu,“ 2022.

[3] T. Slovák a Z. Riedl, „Syntéza: Metody syntézy,“ 2003. [Online]. Available: http://books.fs.vsb.cz/SyntezaReg/text0305.htm.

9. Přílohy

9.1. Návrh regulátoru pomocí metody Optimální modul

Metoda pracuje s požadovaným tvarem frekvenční charakteristiky uzavřené smyčky, která je daná přenosem řízení $Fw(p)$. Přechodný děj bude optimální když, $|Fw(j\omega)|$ = 1 pro co nejvyšší frekvence a pro monotónní průběh. [1]

Metoda zahrnuje delší výpočty s rovnicemi o x neznámých (výpočty jsou složitější a jsou více rozebrané v BPC-RR1). To ale není třeba počítat, protože existují tabulky pro optimální seřízení regulátoru podle kritéria optimálního modulu. Stačí nám, aby daný přenos soustavy existoval pro daný typ soustavy.

Metoda předpokládá přenos PID regulátoru:

$$F_R(p) = K_P\left(1 + T_D p + \frac{1}{T_I p}\right) \quad (7)$$

Pro náš přenos soustavy platí následující tabulka:

Typ regulátoru	Vzorec pro daný regulátor
P	$K_P = \frac{r^2+1}{2 \cdot K_1 \cdot r}$
I	$T_I = \frac{2 \cdot K_1 \cdot T_1}{1} \cdot \frac{1+r}{1}$
PI	$K_P = \frac{r^2+1}{2 \cdot K_1 \cdot r}$ $T_I = K_P \cdot \frac{2 \cdot K_1 \cdot T_1}{1} \cdot \frac{r \cdot (1+r)}{r^2+r+1}$
PID	$K_P \rightarrow \infty$ $T_I \rightarrow 0$ $T_D \rightarrow \infty$

Tabulka 5 Optimální seřízení regulátoru podle OM [3]

Postup:

1) Vámi zvolený přenos přepočítejte do tvaru:

$$F_S(p) = \frac{\boldsymbol{K_1}}{(1+\boldsymbol{T_1}p)\cdot(1+\boldsymbol{T_2}p)} \quad (8)$$

   Kde parametry jsou: $\boldsymbol{K_1}$ (zesílení), $\boldsymbol{T_1}$ a $\boldsymbol{T_2}$ (časové konstanty).
2) Dopočítejte parametr *r* pomocí vztahu:

$$r = \frac{\boldsymbol{T_2}}{\boldsymbol{T_1}} \quad (9)$$

3) Vyberte vhodný regulátor a pomocí Tabulka 5 dopočítejte jednotlivé hodnoty regulátoru.
4) Výsledné hodnoty můžete dosadit do rovnice na PID regulátor (Regulátor už můžete vyzkoušet v Simulink pomocí bloku PID Controller nebo Transfer Fcn)
5) Vypočítané hodnoty dosaďte do PID Compact v TIA PORTAL. (viz. Doporučený postup)

9.2. Návrh regulátoru pomocí metody Ziegler-Nichols

Princip metody spočívá v uvedení regulačního obvodu na mez stability. Soustava následně kmitá netlumenými kmity. Následně se odečítají kritické parametry. Tenhle způsob není bezpečný nebo z technologických důvodu pro všechny soustavy přípustný. Pro takové soustavy existují různé možnosti výpočtu a my vyzkoušíme jeden z nich. Jedná se o určení kritických parametrů z přechodové charakteristiky. [1].

Metoda předpokládá přenos PID regulátoru:

$$F_R(p) = K_P\left(1 + T_D p + \frac{1}{T_I p}\right) \qquad (10)$$

Postup:

1) Z přechodové charakteristiky zjistěte dobu průtahu T_u a dobu náběhu T_n.
2) Dopočítejte kritické parametry, pro které platí přibližné vztahy:

$$K_{krit} \approx \frac{\pi}{2} \cdot \frac{T_n}{T_u} + 1 \qquad T_{krit} \approx 4 \cdot T_u \qquad (11)$$

3) Na základě kritický parametrů dopočítejte parametry Vámi zvoleného regulátoru podle následují tabulky:

Typ regulátoru	K_P	T_I	T_D
P	$K_P = 0{,}5K_{krit}$	-	-
PI	$K_P = 0{,}45K_{krit}$	$T_I = 0{,}85T_{krit}$	-
PID	$K_P = 0{,}6K_{krit}$	$T_I = 0{,}5T_{krit}$	$T_D = 0{,}125T_{krit}$

Tabulka 6 Vzorce pro návrh parametrů regulátoru metodou Ziegler-Nicholse [1]

4) Výsledné hodnoty můžete dosadit do rovnice na PID regulátor (Regulátor už můžete vyzkoušet v Simulink pomocí bloku PID Controller nebo Transfer Fcn)
5) Vypočítané hodnoty dosaďte do PID Compact v TIA PORTAL. (viz. Doporučený postup)

D Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `3D Model podstavec`
    - `Podstavec3D.stl` ..... 3D objekt
    - `PodstavecNX.prt` ..... Software Siemens NX V1915
  - `Laboratorní úloha`
    - `Naměřené průběhy`
    - `TIA Portal Regulace na PLC` ..... Software TIA Portal V15
    - `Regulace pomocí PLC Laboratorni uloha xmusil72.docx`
    - `Regulace pomocí PLC Laboratorni uloha xmusil72.pdf`
  - `Matlab a Simulink projekt` ..... Verze R2022b
  - `Naměřená data` ..... Obsahuje data v excelu naměřená v TIA Portalu
    - `Identifikace`
    - `Nové tlumení 1kohm`
    - `Výsledné regulace`
  - `PLC kód` ..... Vygenerovaný kód pomocí Simulink PLC Coder
  - `TIA Portal projekt xmusil72` ..... Software TIA Portal V15
  - `BP Musil Viktor.pdf` ..... Zdrojový tvar písemné zprávy bakalářské práce