UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

STM32 JAKO HARDWAROVÝ SIMULÁTOR SPOJITÝCH DYNAMICKÝCH SOUSTAV

Tomáš Líbal

Bakalářská práce
2016

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Líbal**
Osobní číslo:	**I12076**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**STM32 jako hardwarový simulátor spojitých dynamických soustav**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Postup:
Cílem práce je návrh hardwarového simulátoru dynamických SISO soustav pomocí modulu STM32. Výsledkem by mělo být uzavřené zařízení s jedním napěťovým vstupem, jedním napěťovým výstupem a miniUSB portem, které bude možno použít pro demonstraci automatické regulace.
Teoretická část:
Popis zařízení STM32, popis možností numerického řešení diferenciálních rovnic
Praktická část:
Vytvoření programu implementujícího numerické řešení diferenciálních rovnic, vytvoření hardwarového řešení, testování spolehlivosti, porovnání se simulovanou soustavou o stejných parametrech

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

COLLATZ, Lothar. Funkcionální analýza a numerická matematika. Praha: Státní nakladatelství technické literatury, 1970, 418 s. STM32 datasheet. STMicroelectronics. [online]. 15.10.2015 [cit. 2015-10-15]. Dostupné z: http://www.st.com/stonline/stappl/resourceSelector/app?page=fullResourceSelector&doctype=datasheet&SeriesID=1574

Vedoucí bakalářské práce: **Ing. Petr Doležel, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **20. listopadu 2015**
Termín odevzdání bakalářské práce: **13. května 2016**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2016

Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 8.5.2016

Tomáš Líbal

Poděkování

Rád bych poděkoval Ing. Petru Doleželovi, Ph.D za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce. Dále bych chtěl poděkovat mé rodině za podporu po celou dobu studia.

V Pardubicích dne 8.5.2016

Tomáš Líbal

ANOTACE

Práce popisuje tvorbu hardwarového simulátoru SISO soustav pomocí modulu STM32. Je zde popsána tvorba programu pro modul STM32, matematický popis zvolené soustavy a popsána PID regulace vytvořeného modelu pomocí PLC Simatic 1200.

KLÍČOVÁ SLOVA

hardwarový simulátor, modul STM32, PID regulace, PLC Simatic 1200.

TITLE

STM32 MODULE AS HARDWARE SIMULATOR OF DYNAMIC SYSTEMS

ANNOTATION

The work describes the creation of a hardware simulator of SISO system using the STM32 module. There is a description of program creation for STM32 module, a mathematical description of chosen system and there is also described PID regulation of the model using PLC Simatic 1200.

KEYWORDS

Hardware simulator, Module STM32, PID regulation, PLC Simatic 1200.

Obsah

Seznam zkratek

PLC	programovatelný logický automat
PID	proporcionálně integračně derivační (regulátor)
AD	analogově digitální (převodník)
DA	digitálně analogový (převodník)
SISO	soustava s jedním vstupem a jedním výstupem

Seznam značek (symbolů proměnných a funkcí)

Q_{in}	přítok do nádrže, $m^3 \cdot s^{-1}$
Q_{out}	odtok z nádrže, $m^3 \cdot s^{-1}$
h	výška hladiny v nádrži, m
D_s	průměr nádrže, m
D_f	průměr výtokového otvoru, m
S	plocha nádrže, m^2
F	plocha výtokového otvoru, m^2
U	elektrické napětí, V
t	čas, s
e	regulační odchylka
w	požadovaná hodnota
y	akční veličina
x	regulovaná veličina

Seznam obrázků

ÚVOD

Cílem této práce bude vytvoření hardwarového simulátoru pomocí modulu STM32 a následná demonstrace automatického řízení. Simulovanou soustavou bude nádrž s otvorem ve dnu a s regulovatelným přítokem. Tento modul je pro vytvoření simulátoru vhodný díky tomu, že má již zabudovaný AD převodník a USB port. Pomocí USB portu bude do modulu nahrán program a bude pomocí něj i celý modul napájen. Dále bude požit jeden analogový výstup, který bude představovat aktuální výšku hladiny v nádrži a jeden analogový vstup, který bude představovat přítok do nádrže. Simulátor bude následně řízen pomocí regulátoru vytvořeného v PLC Simatic 1200.

Nejprve budou v práci popsány základní pojmy, které je nutné znát. Poté bude popsán hardware vybraného modulu STM32. Dále bude popsána simulovaná soustava, kde bude znázorněno značení jednotlivých proměnných, napsána bilanční rovnice soustavy a odvozena diferenční rovnice pro numerický výpočet hladiny. Poté bude znázorněna tvorba program pro simulátor a demonstrace jeho funkčnosti. Na závěr bude popsán program pro regulátor vytvořený pomocí PLC a nastavení PID regulátoru. Kvalita regulace bude zhodnocena vizuálně pomocí grafu obsahujícího průběh regulačního pochodu při žádané hodnotě.

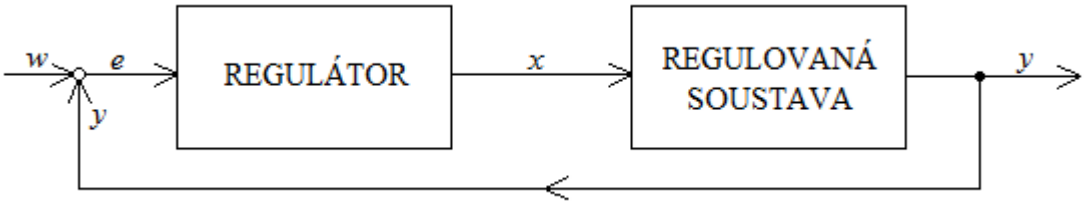
1 TEORETICKÁ ČÁST

Automatizací se označuje proces, který využívá řídicích systémů k řízení průmyslových zařízení a procesů. Jde o krok následující po mechanizaci, která lidem poskytuje zařízení k usnadnění jejich práce. Automatizace snižuje potřebu přítomnosti člověka v procesu. Komplexní automatizace by mohla člověka z výrobního procesu dokonce vyřadit úplně. V praxi se toto zatím nepoužívá a člověk má v procesu vždy svoje místo.

Řízený proces je proces, u kterého za použití regulátorů, počítačů a různých snímačů dosáhneme předem daného požadovaného výsledku.

PLC (programovatelný logický automat) je průmyslový počítač používaný pro automatizaci procesů v reálném čase. Využívá se k řízení strojů, nebo výrobních linek. Pro PLC je charakteristická vysoká spolehlivost, zvýšená odolnost vůči prostředí a cyklické vykonávání programu.

V regulační smyčce je regulovaná veličina snímána a srovnávána v komparátoru s požadovanou hodnotou. Rozdíl požadované hodnoty w a regulované veličiny y je regulační odchylka e, která se tedy dá vyjádřit vztahem $e = w - y$, která je vstupem do regulátoru. Jehož výstupem je akční veličina x, která ovlivňuje regulovanou veličinu y. Podrobněji tuto problematiku řeší (Balátě, 2003).



Obr. 1.1 – Regulační smyčka

1.1 POPIS STM32

STM32 je deska sloužící pro vývoj aplikací a je založená na mikroprocesoru STM32F100RBT6B

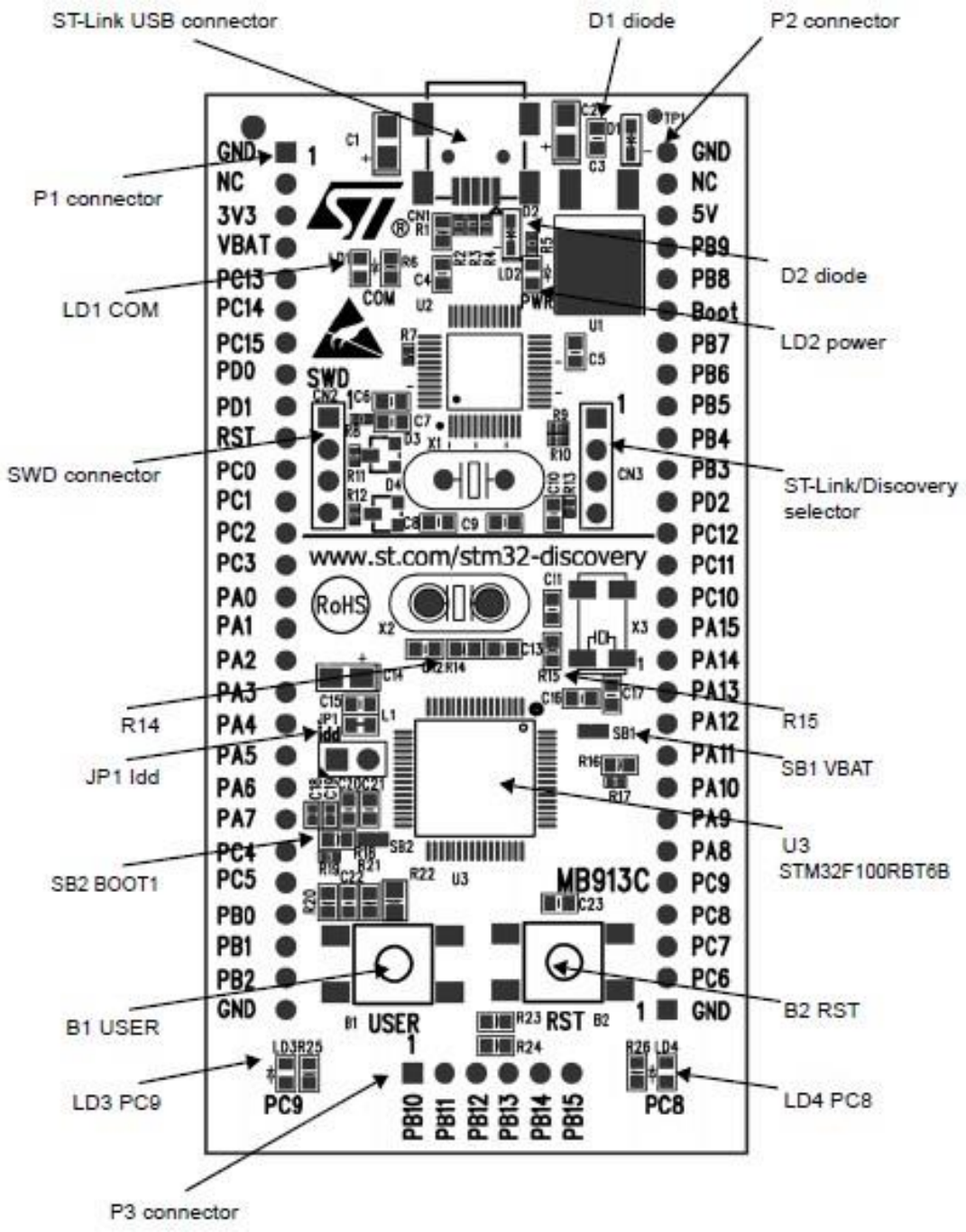
Deska je vybavena:

- mikroprocesorem STM32F100RBT6B, 128 KB Flash paměti, 8 KB RAM
- přepínačem režim použití kitu jako samostatného ST-Linku, nebo s použitím SWD konektoru pro programování a ladění
- dvěma červenými LED diodami - LD1 pro signalizaci komunikace USB, LD2 zapnuté napájení

- může být napájen buď přes USB nebo z externího zdroje 5 V, nebo 3,3 V
- Vytvořené aplikace mohou pracovat s 5 V, nebo s 3 V.
- Dvě uživatelské LED diody, LD3 a LD4 (zelená a modrá)
- dvěma tlačítky (modré uživatelské a černé resetovací)

  Systémové požadavky:
- Windows PC (2000, XP, Vista)
- Pro připojení k počítači slouží USB kabel s konektory A/Mini -B
- Možné vývojové prostředí
    - Atollic, TrueSTUDIO®
    - IAR, Embedded Workbench® for ARM
    - Keil, MDK-ARMTM

Toto jsou systémové požadavky uvedené v datasheetu tohoto modulu. V této práci budu popisovat vytvoření programu v prostředí Atollic, TrueSTUDIO. Vše mi fungovalo i na operačním systému Windows 7.

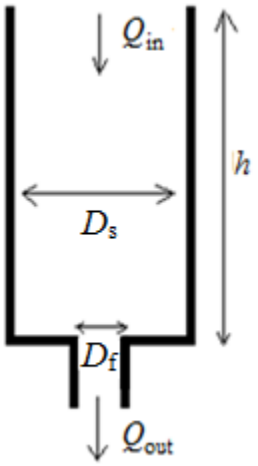


Obr. 1.2 – Rozložení součástek na desce

Jumper CN3 slouží pro výběr módu. Zapnutý pro programování a ladění desky přes USB a vypnutý pro programování přes SWD konektor. Ve výchozím nastavení je sepnutý, což nám vyhovuje. Podrobnější informace o modulu jsou uvedeny v (*STM32 datasheet*, 2013)

1.2 POPIS SIMULOVANÉ SOUSTAVY

Simulovanou soustavou bude nádrž s otvorem ve dnu, u které bude regulován přítok tak, aby byla nastavena požadovaná hladina.



Obr. 1.3 – Simulovaná soustava

Q_{in} – přítok do nádrže, l/s,

Q_{out} – odtok z nádrže, l/s,

h – výška hladiny v nádrži, m,

D_s – průměr nádrže, m,

D_f – průměr výtokového otvoru, m.

Bilanční rovnice pro danou soustavu je vyjádřena vztahem

$$Q_{in} = Q_{out} + S\frac{dh}{dt}, \tag{1.1}$$

kde S – plocha nádrže, m^2,

$\frac{dh}{dt}$ – přírůstek hladiny, l/s.

Odtok z nádrže je závislí na výšce hladiny a je dán vztahem

$$Q_{out} = F\sqrt{2 \cdot g \cdot h} \tag{1.2}$$

kde F – plocha výtokového otvoru, m^2,

g – gravitační konstanta, m/s^2.

Vztah pro výpočet plochy nádrže

$$S = \frac{\pi \cdot (D_s)^2}{4} \tag{1.3}$$

Vztah pro výpočet plochy výtokového otvoru

$$F = \frac{\pi \cdot (D_f)^2}{4} \tag{1.4}$$

Vyjádření přírůstku z bilanční rovnice

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{Q_{in} - Q_{out}}{S} \tag{1.5}$$

Nyní se daná rovnice převede na rovnici diferenční. Přesnější jsou metody Runge-Kutta, čím vyšší řád této metody zvolíme, tím je výpočet přesnější, ale také složitější. Pro převedení rovnice použijeme Eulerovu metodu, která sice není tak přesná, ale je nejednoduší. Tuto problematiku řeší např. (Ollatz, 1970).

$$\frac{h - h_0}{t - t_0} = \frac{Q_{in} - Q_{out}}{S} \tag{1.6}$$

kde h_0 – výška hladiny v předešlém stavu, m,

$t - t_0$ – vzorkovací čas, s.

Z předešlého vztahu se vyjádří aktuální výška hladiny, protože simulátor bude mít jen jeden napěťový výstup, který bude reprezentovat aktuální výšku hladiny.

$$h = \frac{Q_{in} - Q_{out}}{S}(t - t_0) + h_0 \tag{1.7}$$

2 TVORBA HARDWAROVÉHO SIMULÁTORU

Simulátor je vytvořen z modulu STM32. Pro psaní programu používám vývojové prostředí Atollic TrueSTUDIO. Tento program je zdarma ke stažení na internetu a je plně funkční na operačním systému Windows 7. Dalším programem, který je vhodné si nainstalovat, je STM32 ST-LINK Utility, v kterém snadno poznáte, jestli jste správně připojeni k zařízení a můžete zde vyčítat hexadecimálně program, který je nahraný v modulu STM32.

Modul by měl mít jeden napěťový vstup, který bude reprezentovat přítok a jeden napěťový výstup, který bude reprezentovat aktuální výšku hladiny. Z toho vyplývá, že bude za potřebí AD převodník pro vstup a DA převodník pro výstup. Modul bude na základě analogové hodnoty na vstupu počítat hladinu. Pro výpočty použijeme vztahy, které jsme si uvedli v předchozí kapitole.

2.1 POPIS TVORBY PROGRAMU PRO SIMULÁTOR

Před psaním programu je nejdříve potřeba vytvořit si v Atolic TrueSTUDIO nový projekt. Takže se vytvoří „c projekt“ a zvolí se správný typ desky tedy STM32 Discovery. Po vytvoření projektu se otevře hlavní program, který je ve složce „src“. Program není prázdný, ale má na prvních řádcích komentář, kde je uveden název projektu, název programu, pro jaký modul je program napsán a distribuce. Dále program již obsahuje, jaké knihovny má pro daný hardware používat. V novém projektu jsou vypsány všechny, ale ostatní jsou napsány jen jako komentář, tudíž se mohou odstranit pro větší přehlednost.

Na začátku programu budou tedy vybrány knihovny takto:

```
/* Includes */
#include <stddef.h>
#include "stm32f10x.h"
#include "STM32vldiscovery.h"

  #define MESSAGE5   " program built with "
       #define MESSAGE6   " Atollic TrueSTUDIO "
```

2.1.1 Inicializace konstant, proměnných a funkcí

Poté je potřeba vytvořit a nadefinovat konstanty a proměnné příslušného datového typu a přidělit konstantám danou hodnotu a proměnným jejich počáteční hodnotu. Značení odpovídá značení u popisu simulované soustavy v oddíle 1.2. Proměnou d*t* určujeme, jak

často se bude načítat hodnota vstupního napětí a v závislosti na této hodnotě a stavu předchozím se nastaví výstupní napětí.

```
uint16_t Vstup_ADC=0;
float S=0.002;                          /* obsah nadrze, m2 */
float F=0.0000197;                      /*obsah vytokoveho otvoru, m2*/
float g=9.81;                           /* gravitacni konstanta, m/s2*/
float h=0;                              /* vyska hladiny, m */
float Qin=0;                            /* přítok, l/s */
float Qout=0;                           /* odtok, l/s */
float dt=100;                           /* vzorkovaci cas, ms */
__IO uint16_t Vystup_DAC1 = 0;
```

V další části programu se nadefinují funkce, které se budou v programu volat. Definuje se zde, jakého datového typu bude vstupní parametr funkce a jakého datového typu bude návratová hodnota. Pokud funkce nemá žádný vstupní parametr, napíšeme před danou funkci parametr „void“, nebo žádnou návratovou hodnotu, napíše se „void“ do závorky za danou funkci, jak je vidět například u první funkce „DCA_Inicializace“. U funkce „ADC1_Read“, je vidět, jak vypadá zápis, pokud má funkce nějakou návratovou hodnotu.

```
void DAC_Inicializace(void);
void GPIO_Inicializace(void);
void ADC_Inicializace(void);
void SysTick_Handler(void);
void UbyvaniCasu_Delay(void);
uint16_t ADC1_Read(void);
```

2.1.2 Vytvoření potřebných funkcí

Jako první si vytvořím funkci „UbyvaniCasu_Delay“, která slouží k nastavení vzorkovacího času. Tato funkce spolu s použitím přerušení zabezpečí, že program se bude vykonávat předem definovaný čas, což je důležité pro přesnost výpočtu. Nejprve si nastavíme, jak často nám má nastat přerušení od systémových hodin. Nastavení hodin se nachází v stm32f10x.c, kde pokud nic nezměníme, budou hodiny pro SystemClock dány interním oscilátorem 8 MHz. Nastavíme přerušení na každých 8000 hodinových taktů a díky tomu se vyvolá přerušení přesně každou ms. Návratová hodnota funkce „SysTick_Config“ je dle úspěchu nastavení.

```
if(SysTick_Config(SystemCoreClock / 1000))
        {
          /* Capture error */
          while (1);
             }
```

Tímto je nastaveno jak často má přerušení nastat, poté je potřeba napsat co má přerušení vykonat. Obsluha všech přerušení je v souboru „src/stm32f1xx_it.c“, kde upravíme příslušnou funkci nasledovně.

```
void SysTick_Handler(void)

{ UbyvaniCasu_Delay();
 }
```

Tímto se nám při každém přerušení zavolá funkce „UbyvaniCasu_Delay“, ve které se při každém jejím zavolání sníží hodnota proměnné „CasProDelay“ o jedna.

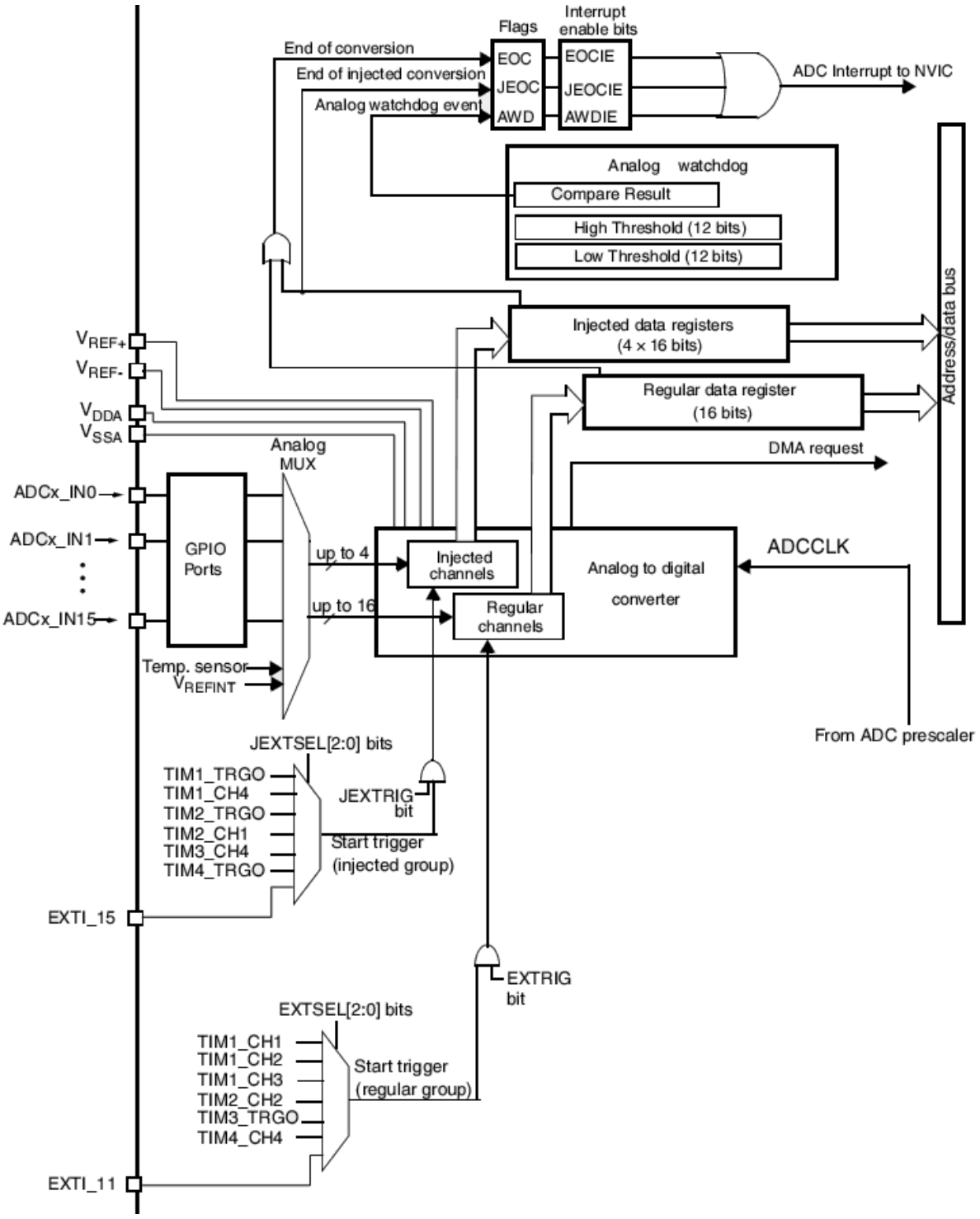
```
void UbyvaniCasu_Delay(void)
 {
   if (CasProDelay != 0x00)
   {
     CasProDelay--;
   }
      }
```

Další potřebnou funkcí, která se musí vytvořit, je „ADC_Inicializace“. Integrovaný AD převodník, který je použit v modulu STM32, umožňuje velmi různá nastavení. V této funkci lze nastavit parametry AD převodníku, od nepřetržitého běhu či jednoho převodu, přes dobu převodu a přesnost, po možnost vyvolání přerušení po dokončení převodu a také jaký vstup AD převodníku se bude používat.

Vlastnosti A/D převodníku v STM32 Value Line:

- 12bit přesnost
- celkem 16 vstupních kanálů
- generování přerušení na konci konverze
- analogový watchdog
- single a kontinuální módy převodu
- scan mód pro automatickou konverzi vybraného počtu kanálů
- samo-kalibrace převodníku
- zarovnání dat dle požadovaného způsobu
- programovatelný čas vzorkování pro každý kanál zvlášť
- možnost externího trigeru
- doba převodu: 1.17 µs pro 24 MHz hodinový takt
- požadavek na napájení ADC: 2.4V to 3.6V
- generace DMA požadavku v průběhu převodu napětí kanálu

Analogový watchdog je modul, který vyvolá přerušení mikroprocesoru, pokud konvertované napětí klesne, nebo stoupne (dle nastavení) nad zadanou hodnotu.



Obr. 2.1 – Blokové schéma AD převodníku

Na tomto schématu je vidět, že na analogový multiplexor, co je na obrázku, si jako vstupní hodnotu můžete vybrat kromě vstupních analogových kanálů také interní teplotní čidlo a interní napěťovou referenci.

Nezbytné pro nastavení převodníku je povelní hodin na daný port, nastavení rychlosti převodníku, výběr pinu daného portu, povolení hodin na vstup AD převodníku, výběr kanálu převodníku, konfigurace a kalibrace AD převodníku.

Nastavení AD převodníku pro daný simulátor může vypadat následovně. Přímo v programu jsou v komentářích vysvětleny a popsány nejdůležitější parametry AD převodníku.

```
void ADC_Inicializace(void)
{
  ADC_InitTypeDef  ADC_InitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;

  /* Nastavení frekvence hodin AD převodníku 24/2 = 12MHz*/
  RCC_ADCCLKConfig(RCC_PCLK2_Div2);

  /* Povolení hodin na portu */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);

  /* Nastavení pinu, na který bude připojeno napěti. */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;    //
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  /* Povolení hodin do AD převodníku */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

  /* Konfigurace AD převodníku-------------------------------------*/
  ADC_InitStructure.ADC_Mode                = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode        = DISABLE;
  ADC_InitStructure.ADC_ContinuousConvMode  = DISABLE;
  ADC_InitStructure.ADC_ExternalTrigConv    =
ADC_ExternalTrigConv_None;
  ADC_InitStructure.ADC_DataAlign           = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel        = 1;
  ADC_Init( ADC1, &ADC_InitStructure );

  /* Nastavení příslušného kanálu AD převodníku. */
  ADC_RegularChannelConfig( ADC1, ADC_Channel_14,  1,
ADC_SampleTime_239Cycles5);

  /* Povolení externího spouštění AD převodníku. */
  ADC_ExternalTrigConvCmd( ADC1, ENABLE );
```

```
  ADC_Cmd(ADC1, ENABLE);

  /* Dále provedeme kalibraci převodníku. */
  /* Provedeme reset registru kalibrace převodníku. */
  ADC_ResetCalibration(ADC1);
  /* Zkontrolujeme, jestli se reset registru kalibrace dokončil. */
  while(ADC_GetResetCalibrationStatus(ADC1));
  /* Spustíme kalibraci AD převodníku. */
  ADC_StartCalibration(ADC1);
  /* Zkontrolujeme, jestli se kalibrace převodníku dokončila. */
  while(ADC_GetCalibrationStatus(ADC1));
      }
```

Následné převedení hodnoty na analogovém vstupu na hodnotu digitální poté již probíhá jen pomocí funkce „ADC1_Read“, která nejprve spustí převod, poté počká, až se převod dokončí a nastaví výsledek jako svou návratovou hodnotu.

```
uint16_t ADC1_Read(void)
{
  /* Začátek převodu. */
  ADC_SoftwareStartConvCmd(ADC1, ENABLE);
  /* Čekání na dokončení převodu */
  while(ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC) == RESET);
  /* Nastavení výsledku převodu na návratovou hodnotu funkce */
  return ADC_GetConversionValue(ADC1);
      }
```

Další vytvořenou funkcí bude „DAC_Inicializace“, která je potřebná ke správné funkčnosti DA převodníku. Z datasheetu, kde jsou popsány piny modulu a jejich alternativní funkce, je vidět, že DA převodník má dva kanály DAC1 a DAC2. Při použití prvního kanálu DAC1 bude výstup na pinu PA4. DA převodník na daném modulu je 12bit, maximální hodnota, kterou lze poslat na výstup je v hexadecimálním tvaru 0xFFF, což odpovídá napájecímu napětí, tedy 3 V. DA převodník umí sám generovat některé signály, například bílý šum a pilu.

Pro použití DA převodníku je nezbytné, stejně jako u AD převodníku, povolit hodiny na daný port, povolit hodiny do DA převodníku, nastavit pin daného portu, nastavit příslušný kanál převodníku a nakonfigurovat jej. DA převodník umožňuje povolení výstupního bufferu. Pokud buffer není povolen, tak po připojení zátěže na výstupní pin DA převodníku bude klesat napětí. Inicializace pro DA převodníku muže vypadat následovně.

```
void DAC_Inicializace(void)
{
  /* Povolení hodin na portu */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```

```
  /* Povolení hodin do AD převodníku */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);
  /* Zde si zvolíme příslušný pin daného portu */
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  DAC_InitTypeDef
  DAC_InitStructure;

  /* Konfigurace DA převodníku. */
  /* Není potřeba žádné vnitřní spouštění převodu. */
  DAC_InitStructure.DAC_Trigger = DAC_Trigger_None;
  /* Zde je možnost nastavit automatické generování výstupního
signálu. Tato možnost bude vypnuta. */
  DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_None;
  /* Zde se povolí použití výstupního bufferu, pokud byjsme ho
nepovolili a výstup zatížili, výstupní napětí by nám kleslo. */
  DAC_InitStructure.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
  DAC_Init(DAC_Channel_1, &DAC_InitStructure);
  DAC_Init(DAC_Channel_2, &DAC_InitStructure);

  /* Povolení příslušného kanálu pro převod. */
  DAC_Cmd(DAC_Channel_1, ENABLE);
  }
```

Převedení vypočtené hodnoty na analogovou a nastavení jako výstup na pin PA4 se provede následujícím příkazem.

```
    /*nastaveni vystupu*/
    DAC_SetChannel1Data(DAC_Align_12b_R, (uint32_t)Vystup_DAC1);
```

To jsou všechny potřebné funkce, které jsou nezbytné pro správnou funkci programu.

2.1.3 Vytvoření hlavního programu

Nyní popis hlavního programu, ve kterém se volají některé z funkcí popsaných a kde se provádí samotný výpočet a simulace dané soustavy. Jedná se o nekonečnou smyčku. Pokud chceme spustit daný program od začátku a nastavit tak parametry soustavy na inicializační hodnoty, musíme odpojit modul od napájení a znovu jej připojit, nebo resetovat modul pomocí příslušného tlačítka.

Na začátku smyčky se inkrementuje hodnota proměnné „i“. Nastaví se vzorkovací frekvence. Spustí se odpočet 100 ms. Na konci smyčky se čeká, a smyčka se znovu spustí po dokončení odpočtu 100 ms. Tento odpočet běží po celou dobu běhu programu. Tím se zajistí

dodržení stejného času v každém cyklu. Navíc si na začátku cyklu rozsvítím zelenou LED diodu pro signalizaci běhu programu.

```
while (1)
  {
      /* Hodnota i se při každém cyklu zvýší o jedna, což muže pomoci
při ladění programu. */
      i++;
      /* Nastavení vzorkovací frekvence. Od tohoto místa se spouští
odpočet 100 ms. */
    { CasProDelay = dt;
      /* Rozsvícení zelené led diody. */
      STM32vldiscovery_LEDOn(LED3);
```

V další části si nejprve načteme a převedeme vstupní napětí na digitální hodnotu a uložíme ji do proměnné „Vstup_ADC“. Poté hodnotu tohoto napětí převedeme na přítok, kde bude odpovídat 1 V na vstupu přítoku 1 l/min. Dále vypočítáme, jaký bude odtok z nádrže podle rovnice (1.2). Funkce „sqrt“ je funkce odmocniny, která je součástí knihoven v Atollic TrueSTUDIO. Při použití, jaké dále ukáži, vypíše program varování, které si mi nepovedlo odstranit. Toto varování však nijak neovlivňuje funkčnost programu a podle mého názoru bude souviset s tím, jaké hodnoty mohou být jako vstupní do funkce odmocniny.

```
      /* Načtení vstupu pomocí funkce „ADC1_Read“ a uložení výsledku
do proměnné „Vstup_ADC“. */
      Vstup_ADC = ADC1_Read();
      /* Přepočtení vstupu na přítok. /
      Qin=((Vstup_ADC*1.221)/100000000);
      /* Vypočtení odtoku podle odvozeného vztahu */
      Qout=F*(sqrt(2*g*h));
```

V další časti provedeme výpočet aktuální výšky hladiny podle vzorce (1.7). Hodnotu d*t* zde dělíme tisícem, abychom ji převedli na základní jednotku v sekundách, protože je zadávána v milisekundách. Výška hladiny použitá ve vzorci na pravé straně od rovnítka je výška hladiny v předchozím cyklu. Na levé straně je aktuální výška hladiny, která se přepočte tak, aby výšce hladiny 10 cm odpovídalo napětí 1 V. Tato hodnota se na konci cyklu nastaví na DA převodník, který na výstupní pin nastaví příslušné napětí.

```
      /* Vypočtení aktuální výšky hladiny. */
      h=((((Qin-Qout)/S)*(dt/1000))+h);
      /* Přepočtení výšky hladiny. */
      Vystup_DAC1=h*13650;
```

Výstupní napětí, nemůže být, vyšší než 3 V. Této hodnotě napětí odpovídá hodnota 4095 na vstupu DA převodníku. Proto nastavíme omezení, které bude simulovat přetečení nádrže. Pokud tedy bude vypočtená hodnota výstupu vyšší, nastavíme na výstup

právě tuto hodnotu. Také nastavíme, že výška hladiny nesmí být záporná, pokud tato hodnota vyjde, nastavíme na výstup 0.

```
      /* Simulace přetečení nádrže. */
       if (Vystup_DAC1>=4095)
         Vystup_DAC1=4095;
      /* Ošetření záporné výšky hladiny. */
       if (Vystup_DAC1<=0)
{Vystup_DAC1=0;
        h=0;}
```

Nakonec se vypočtená hodnota odešle do DA převodníku, který nastaví příslušné napětí na svůj výstup a počká se, až odpočet, který zajišťuje stejnou dobu každého cyklu a tudíž je i hodnota d*t* v každém cyklu stejná.

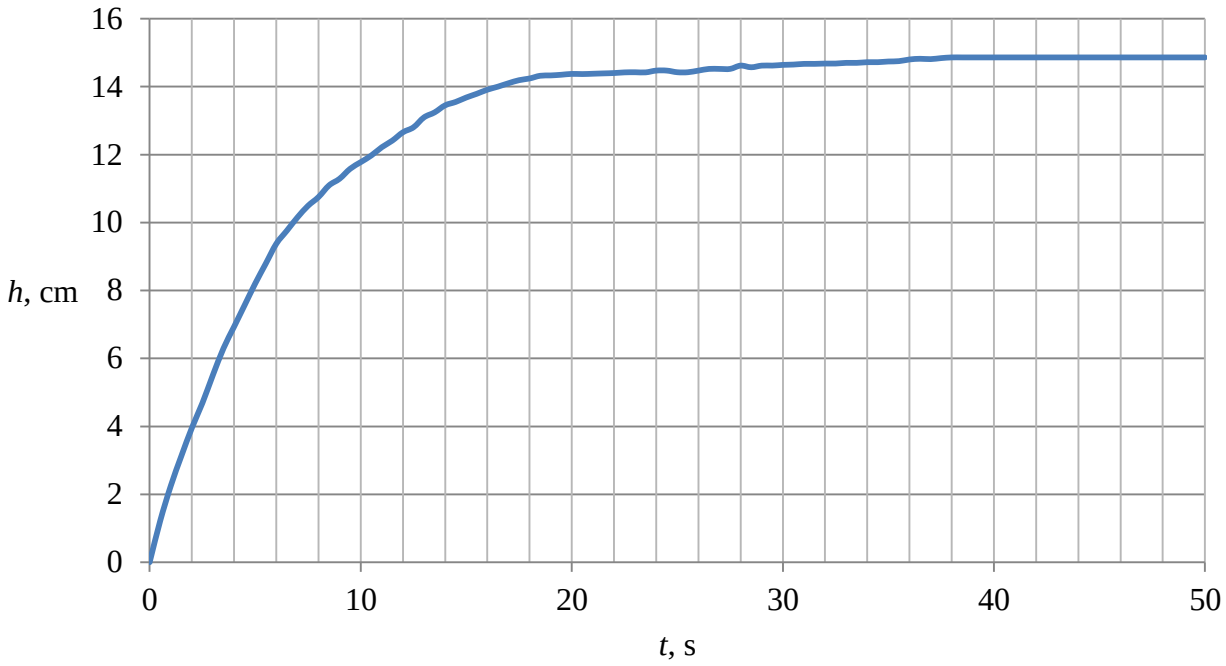
```
      /* Odeslání vypočtené hodnoty do DA převodníku. */
       DAC_SetChannel1Data(DAC_Align_12b_R, (uint32_t)Vystup_DAC1);
      /* Čeká dokud „CasProDelay“ nebude roven 0. */
       while (CasProDelay !=0);
    }
```

Podrobněji s jazykem C se čtenář může seznámit v (Kernighan, 2013)

2.2 ODEZVA SIMULÁTORU

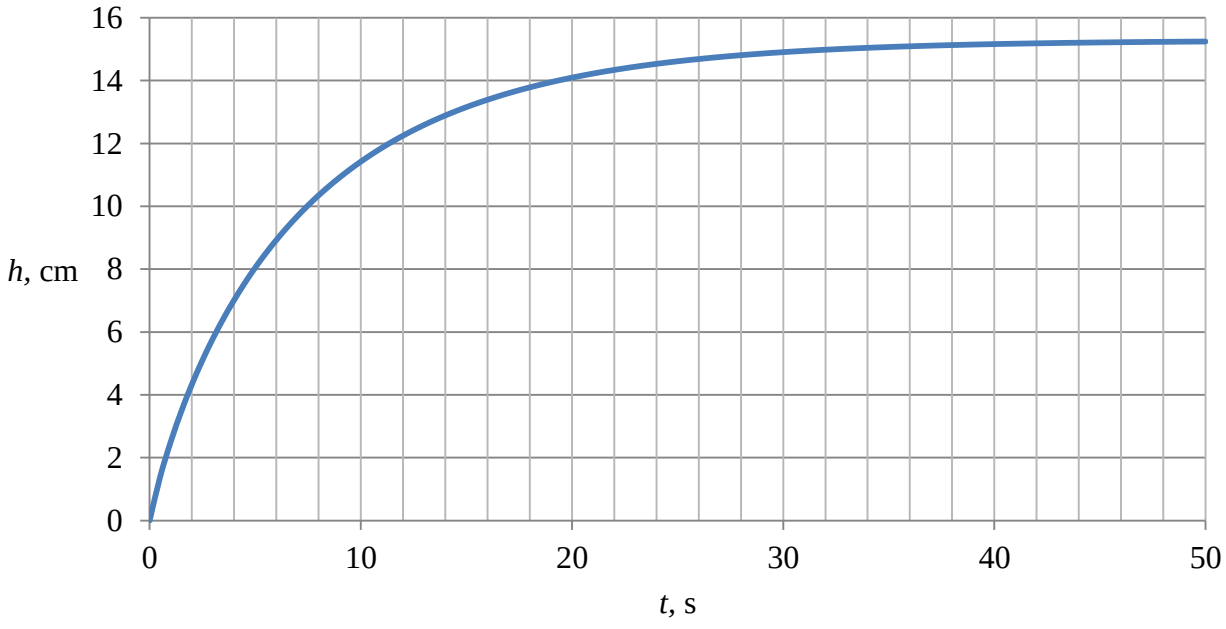
Po vytvoření programu v prostředí Atollic TrueSTUDIO se program nahraje pomocí USB portu do modulu STM32. Program se začne vykonávat okamžitě po nahrání do modulu, což nám signalizuje rozsvícení zelené LED diody. Nyní je tedy simulátor připravený a simuluje danou soustavu nádrže. Simulátor má jeden vstup na pinu PC4. Pokud na tento pin přivedeme napětí v rozsahu od 0 do 3 V, bude nám toto napětí simulovat přítok do nádrže v rozsahu od 0 do 3 l/min. Takto vytvořený simulátor má také jeden výstup na pinu PA4, na kterém je možné měřit napětí od 0 do 3 V. Toto napětí nám reprezentuje aktuální výšku hladiny, kde 1 V odpovídá výšce hladiny 10 cm.

Pokud tedy na vstup simulátoru, pin PC4, přivedeme napětí, začne nám v nádrži růst hladina a tím i napětí na výstupním pinu PA4. Hladina bude vstoupat až do chvíle, kdy se přítok bude rovnat odtoku z nádrže. Tento stav nastává kvůli tomu, že odtok se z rostoucí hladinou zvyšuje, což vyplývá ze vzorce (1.2). Tím, že jsme omezili výšku nádrže na 30 cm, tak pokud se hladina neustálí do této výšky, nádrž přeteče a hladina přestane růst. Omezení bylo nutné z důvodu, že na výstupu DA převodníku nelze nastavit větší napětí než 3 V. Samozřejmě by bylo možné přepsat v programu přepočet hladiny na výstupní napětí tak, že by například 1 V na výstupu odpovídal 1 m.



Obr. 2.2 – Odezva simulátoru na skokový signál

Na zobrazeném grafu je vidět průběh výšky hladiny na přivedený skokový signál. Přivedený signál na vstup byl 2 V, což odpovídá přítoku 2 l/min. Z grafu vyplývá, že výška hladiny se nám ustálí přibližně po 30 s a ustálí se přibližně na 15 cm. Na dalším grafu je znázorněn průběh výšky hladiny v čase u soustavy stejných parametrů vytvořené v Simulinku. Grafy jsou téměř shodné a z toho vyplívá, že vytvořený simulátor funguje správně.



Obr. 2.3 – Odezva soustavy na skokový signál

3 ŘÍZENÍ SOUSTAVY

Pokud chceme řídit nějakou soustavu, musíme si nejprve vybrat, jaký typ regulace zvolíme. Podle účasti člověka je možné regulace dělit na ruční regulaci, při které přebírá některé z funkcí v regulační smyčce člověk a na automatickou regulaci, která pobíhá bez vlivu člověka, s výjimkou zadávání hodnoty řídicí veličiny, tj. požadované hodnoty. Při regulaci na konstantní hodnotu se regulátor stále pokouší uvést skutečnou hodnotu regulované veličiny do souladu s požadovanou hodnotou.

U automatické regulace je na výběr z několika možností regulátorů. Spínací regulátory například dvoustavový regulátor, který není příliš vhodný, pokud chceme mít řízení přesné, protože tento regulátor má vždy nějakou odchylku od skutečné hodnoty, a kolem požadované hodnoty kmitá. Tyto vlastnosti jsou dané hysterezí regulátoru. Tento typ regulátoru je vhodný spíše pro pomalejší soustavy, kde není vyžadována taková přesnost. Další možností je použít PID regulátor. PID regulátor patří mezi spojité regulátory, je složený z proporcionální, integrační a derivační části. U PID regulátoru nemusíme vždy použít všechny jeho složky. Pokud použijeme pouze proporcionální složku, tak bude mít regulátor trvalou regulační odchylku. Proporcionální člen v PID regulátoru zesiluje vstupní signál a zrychluje regulaci, ale při velkém zesílení je regulátor náchylný ke kmitání. Integrační složka úplně odstraňuje regulační odchylku. Derivační složka dokáže rychle reagovat na změny regulační odchylky. Poslední možností je použít číslicový regulátor.

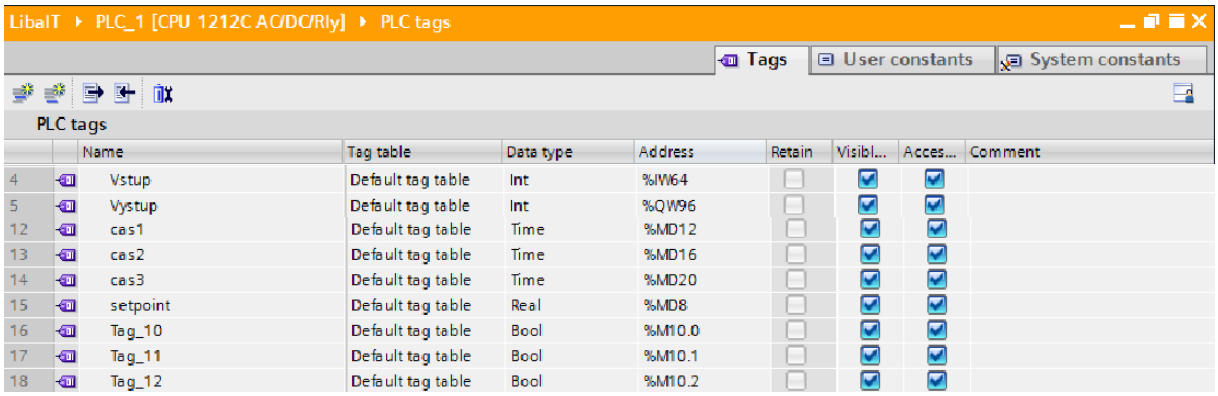
3.1 POPIS REGULÁTORU

Pro řízení vytvořeného simulátoru budeme používat PLC Siemens 1200. Program pro PLC bude tvořen v prostředí TIA PORTAL. Bude se používat softwarový PID regulátor, který je pro toto PLC v prostředí již naprogramován. Bude se tedy jednat o číslicovou regulaci, protože PLC pracuje v cyklech a i když používá PID regulátor, nejedná se o spojitou regulaci. Použité PLC disponuje jedním napěťovým vstupem, na který bude připojen výstup z modelu, tedy aktuální výška hladiny. Napěťový výstup PLC nemá a je tudíž nutné použít rozšiřující kartu, která tento výstup má a na který bude připojen vstup do soustavy, tedy přítok.

3.2 TVORBA PROGRAMU PRO PLC

Při zapnutí programu TIA PORTAL se nejprve založí nový projekt, pojmenuje se a nastaví se adresář, kam se bude ukládat. Poté se musí vytvořit hardwarová konfigurace, kde jde o to si vybrat, pro jaké zařízení chcete daný program psát. V našem případě budeme mít v hardwarové konfiguraci zařízení dvě, PLC Siemens 1200 a přídavnou kartu, která má analogový výstup.

Poté je potřeba vytvořit tabulku tagů. V této tabulce si můžeme vytvářet proměnné přiřazovat jim vstupy, výstupy, nastavovat jejich datový typ, a pokud je třeba, psát si k nim komentáře.

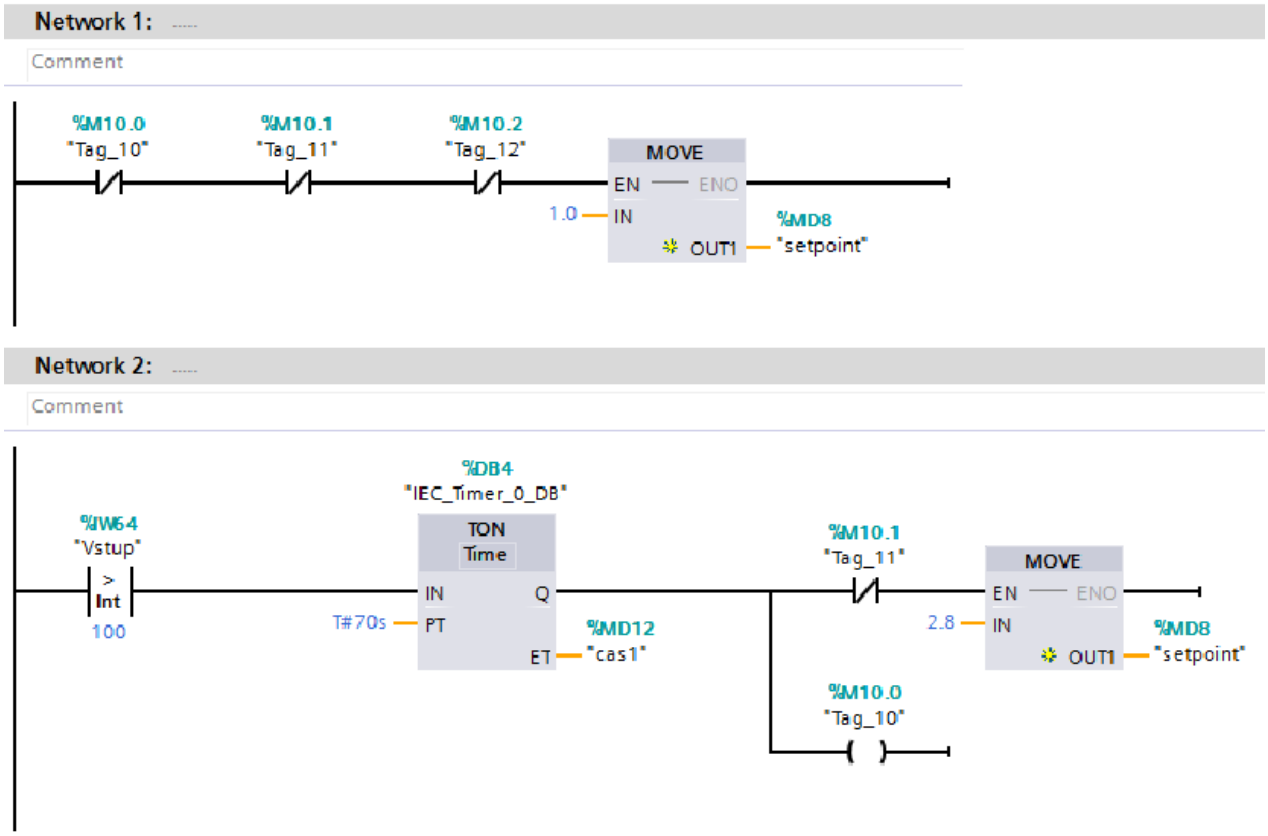


	Name	Tag table	Data type	Address	Retain	Visibl...	Acces...	Comment
4	Vstup	Default tag table	Int	%IW64	☐	☑	☑	
5	Vystup	Default tag table	Int	%QW96	☐	☑	☑	
12	cas1	Default tag table	Time	%MD12	☐	☑	☑	
13	cas2	Default tag table	Time	%MD16	☐	☑	☑	
14	cas3	Default tag table	Time	%MD20	☐	☑	☑	
15	setpoint	Default tag table	Real	%MD8	☐	☑	☑	
16	Tag_10	Default tag table	Bool	%M10.0	☐	☑	☑	
17	Tag_11	Default tag table	Bool	%M10.1	☐	☑	☑	
18	Tag_12	Default tag table	Bool	%M10.2	☐	☑	☑	

Obr. 3.1 – Tabulka tagů

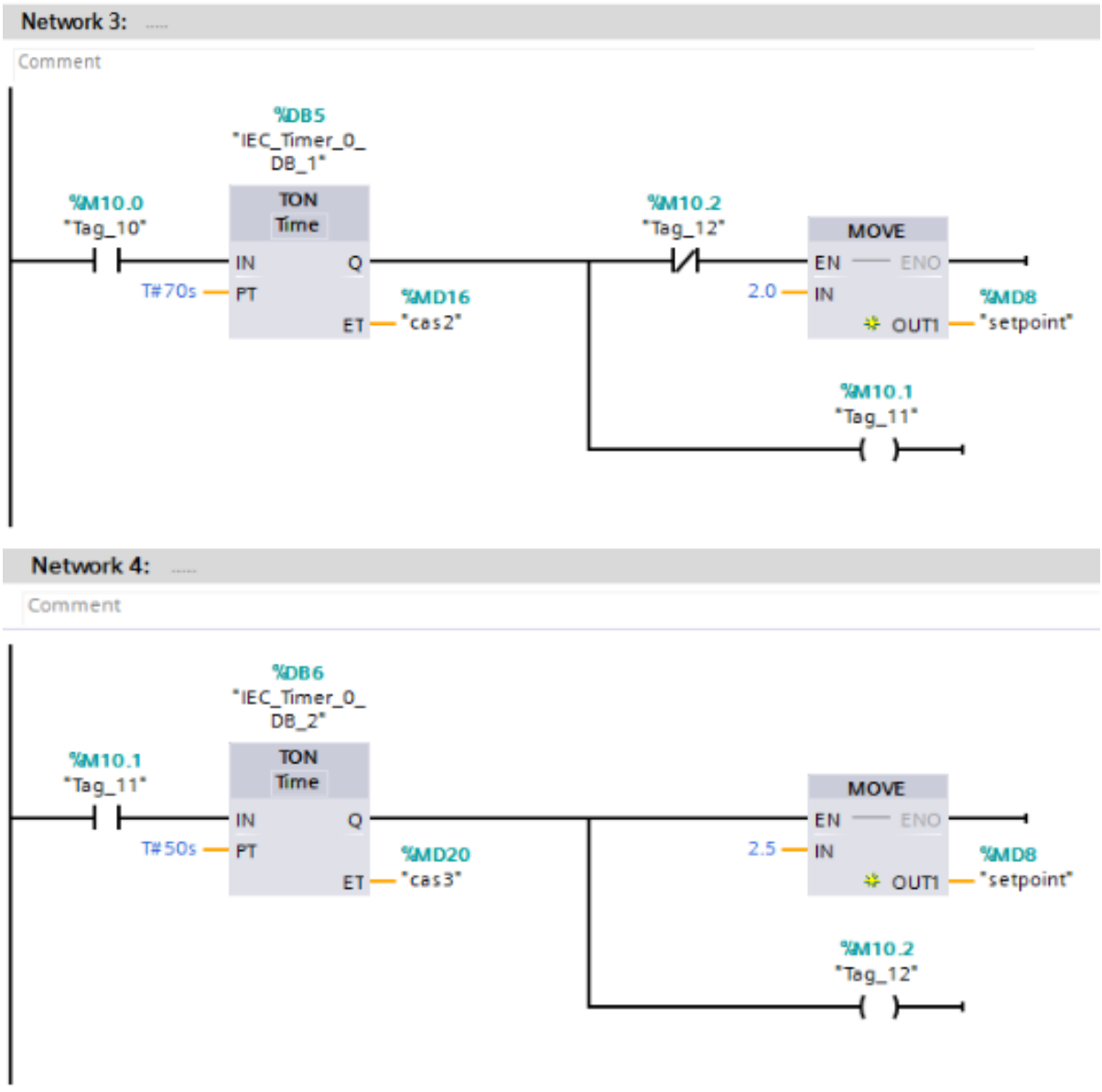
V programu PLC je použit, jak již bylo napsáno v předešlé kapitole, softwarový PID regulátor. U PLC Siemens 1200 je na výběr z několika PID regulátorů. Použitý regulátor je PID compact. Tento regulátor je již naprogramovaný a musí se jen nastavit. Dále je v programu PLC, vytvořena změna žádané hodnoty v určitých časech. Prostředí TIA PORTAL nabízí několik programovacích jazyků. Já sem pro vytvoření programu použil Ladder Diagram. Ladder Diagram je grafický jazyk, určený pro programování automatických systémů. Program je velice jednoduchý a přehledný, obsahuje 5 linií, neboli Networků.

V prvním Networku jsou tři rozpínací kontakty a blok „MOVE“, který přesouvá hodnotu na levé straně do proměnné na pravé straně. Proměnná na pravé straně je „setpoint“, je to požadovaná hodnota výšky hladiny, která je dále použita jako vstupní parametr do PID regulátoru. Rozpínací kontakty jsou zde proto, aby se po nastaveném čase 70 s, jak je vidět v Networku 2, nastavila jiná požadovaná hodnota do proměnné „setpoint“. V Networku 2 je použit časovač, který začíná časovat, pokud je splněna podmínka, že na analogovém vstupu je hodnota větší než 100. Na tento vstup je připojen výstup z modulu. Tato podmínka je splněna téměř hned po zapnutí. Prostředí TIA PORTALU vám nedovolí, nemít před časovačem žádnou podmínku. Po uplynutí času 70 s, se nastaví pomocí „MOVE“ jiná požadovaná hodnota a nastaví se pomocný bit markru „M10.0“ do jedničky. Tím se zajistí, že není splněna podmínka v Networku 1 a do proměnné „setpoint“ se nastavuje pouze jedna hodnota.



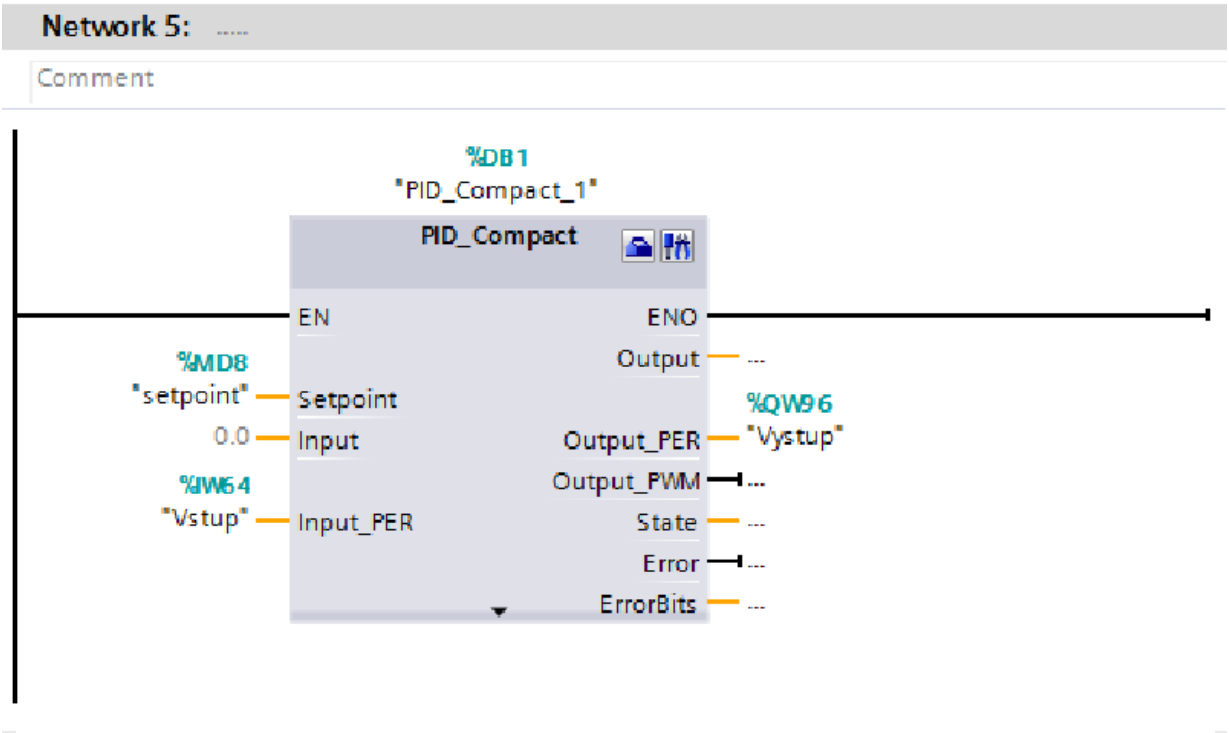
Obr. 3.2 – Program pro PLC 1. část

V další časti programu jsou vidět další 2 Networky, které slouží ke změně nastavení požadované hodnoty po určitém čase. V Networku 3 použit další časovač, který se spouští pokud je marker „M10.0“ v jedničce. Tento marker se nastavuje v Networku 2. Pokud je tedy marker „M10.0“ v jedničce spustí se časovač, po uplynutí nastaveného času, se nastaví nová hodnota do proměnné „setpoint“ a nastaví se do jedničky pomocný bit markeru „M10.1“ Který blokuje nastavování hodnoty v Networku 1 a 2 a zároveň spouští poslední použitý časovač v Networku 4. Po nastaveném čase se nám opět změní hodnota nastavovaná do proměné „setpoint“ a nastaví se do jedničky pomocný marker bit „10.2“, který blokuje nastavení jiný hodnoty do proměnné „setpoint“ v Networku 1,2 a 3.



Obr. 3.3 – Program pro PLC 2. část

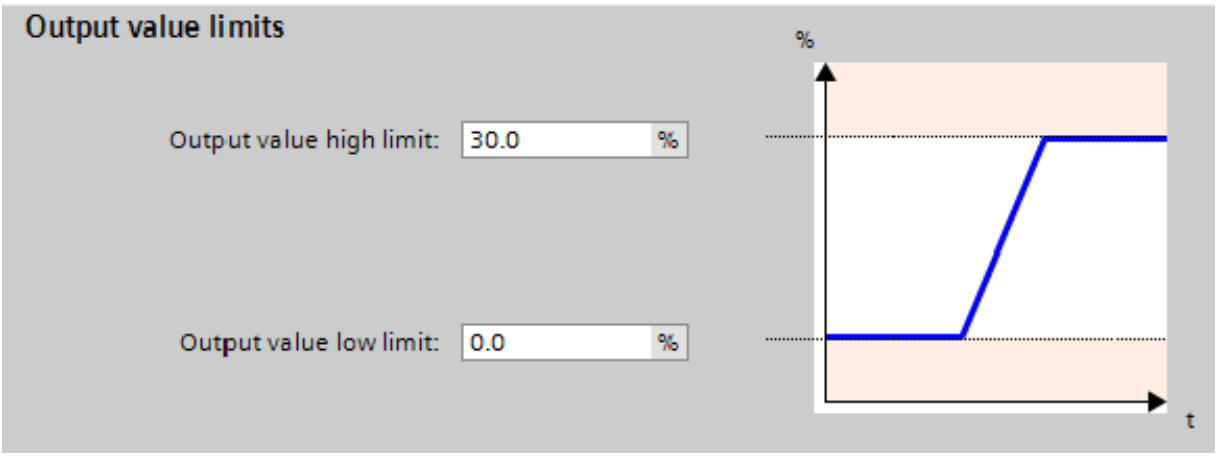
V poslední části programu je vidět pouze blok „PID_compact“ který má použity dva vstupy a jeden výstup. Jeden vstup je „setpoint“ kde nastavujeme požadovanou hodnotu. Tuta hodnota se nastavuje ve voltech. Přepočet na výšku hladiny v centimetrech je však jednoduchý a byl uveden v kapitole při tvorbě simulátoru, 1 V odpovídá 10 cm výšky hladiny. Další vstupem do regulátoru je vstup „IW64“ na který je připojen výstup modulu a tedy aktuální výška hladiny. Výstup z bloku „PID_compact“ je použít analogový „Output_PER“ a je posílán na přidanou kartu na analogový výstup „QW96“. „PID_compact“ umožňuje použití i jiných vstupů nebo výstupů. Na obr. 3.4 je například vidět výstup „Output_PWM“ což je pulzně šířkově modulovaný výstup.



Obr. 3.4 – Program pro PLC 3. část

3.3 NASTAVENÍ PARAMETRŮ PID REGULÁTORU

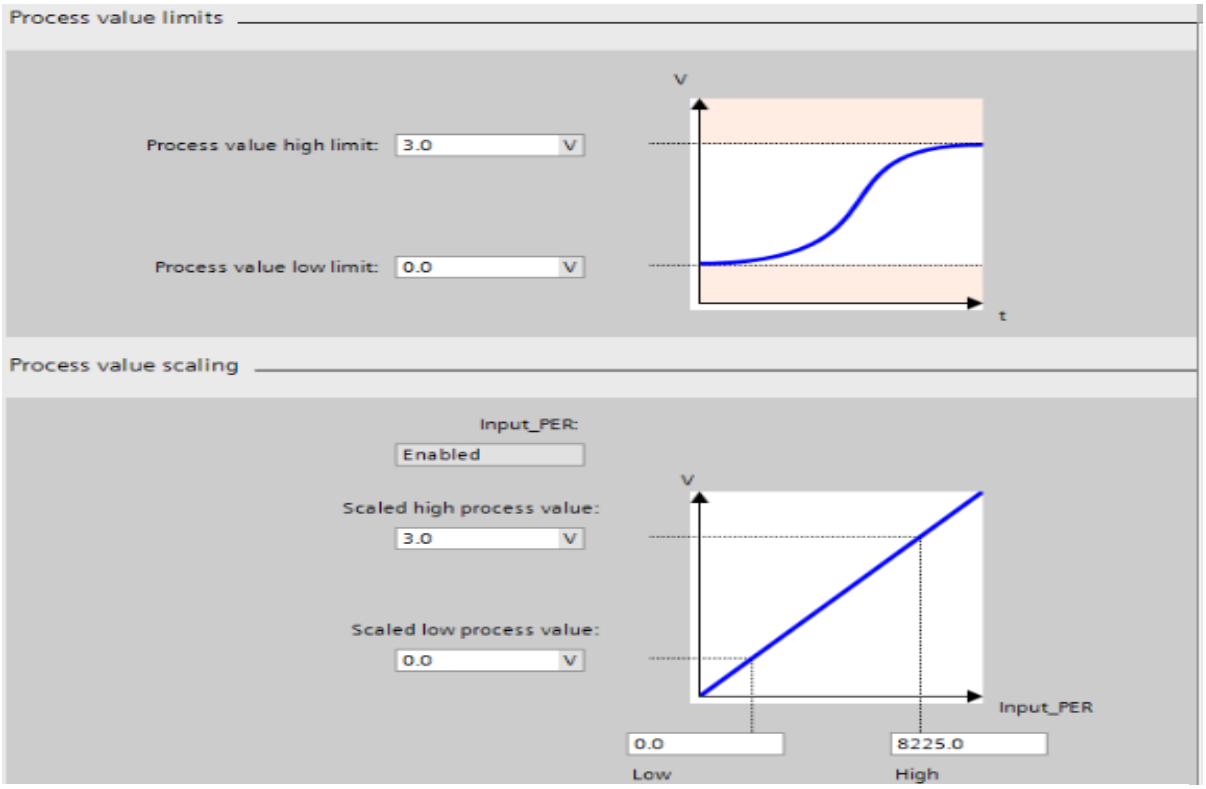
Nastavení správných parametrů v bloku PID_Compact je velice důležité. Jako první nastavíme parametry omezující hodnotu výstupu. Při použití analogového výstupu „Output_PER“ nastavujeme hodnotu v procentech.



Obr. 3.5 – Omezení výstupu regulátoru

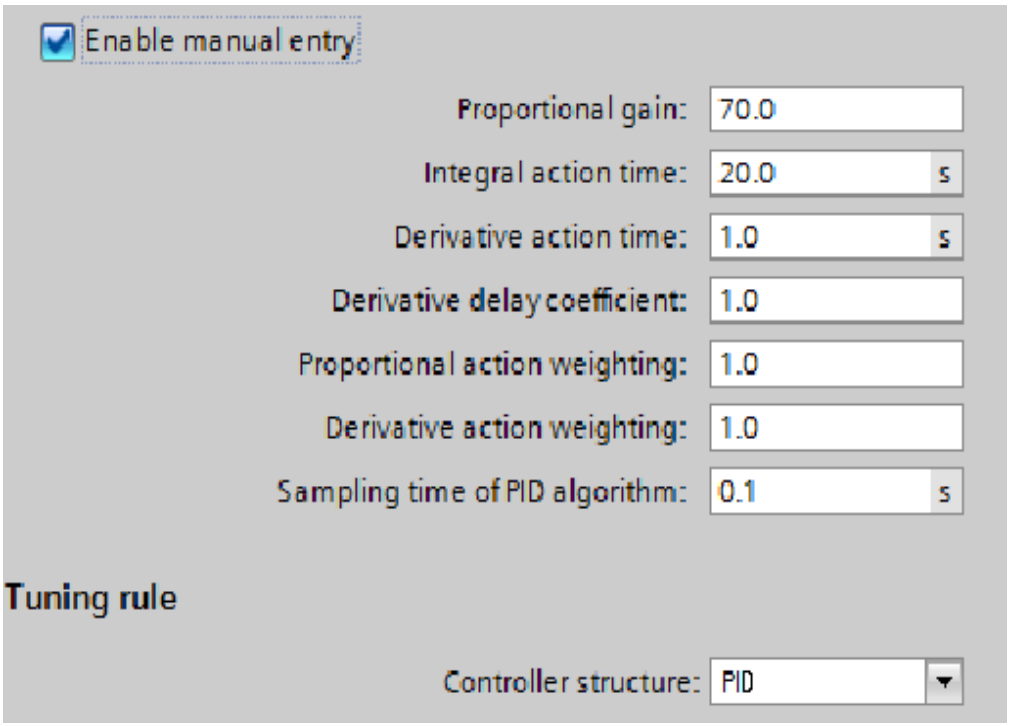
Tento nastavení je velice důležité provést. Doporučuji jej nastavit hned při prvním nahrání programu. Horní limit je nastaven na třicet procent, protože analogový výstup má rozsah 0 až 10 V, ale vstup na AD převodník použitý v našem simulátoru je pouze na 3,6 V. Samozřejmě by bylo možné nastavit 36 procent, ale ponecháme nějakou rezervu. Toto omezení nám tedy udává jaký maximální a minimální bude přítok do nádrže.

Další nastavení jsou limity vstupu PID regulátoru a scale. Limity se nastavují ve voltech. Minimum nastavíme na 0 V a maximum na 3 V. Maximum je dáno simulátorem, který na výstup dokáže nastavit maximálně právě 3 V a tato hodnota odpovídá výšce 30 cm. Což je pro náš simulátor výška nádrže, tudíž výška hladiny nemůže být nikdy větší.



Obr. 3.6 – Omezení a scaling vstupu

Posledními nastavovanými parametry jsou jednotlivé složky regulátoru. Tyto parametry můžeme zadávat ručně, nebo využít možnosti autotuningu, kde si tyto hodnoty nastaví regulátor sám. Nejsou zde pouze tři parametry, ale je jich zde sedm. A navíc je zde ještě možnost výběru struktury regulátoru - nemusíme využívat všechny tři složky, ale například jen dvě u PI regulátoru. Jednotlivé složky jsem nastavil metodou pokus omyl. Nejdříve nastavíme zesílení, aby regulátor dostatečně reagoval na změnu, poté se postupně zvyšuje integrační složka, aby se odstranila regulační odchylka. Na závěr jsem nastavil derivační složku. Parametrem „Sampling time of PID algorithm“ se nastavuje čas, který určuje, jak často se bude funkce PID regulátoru volat.



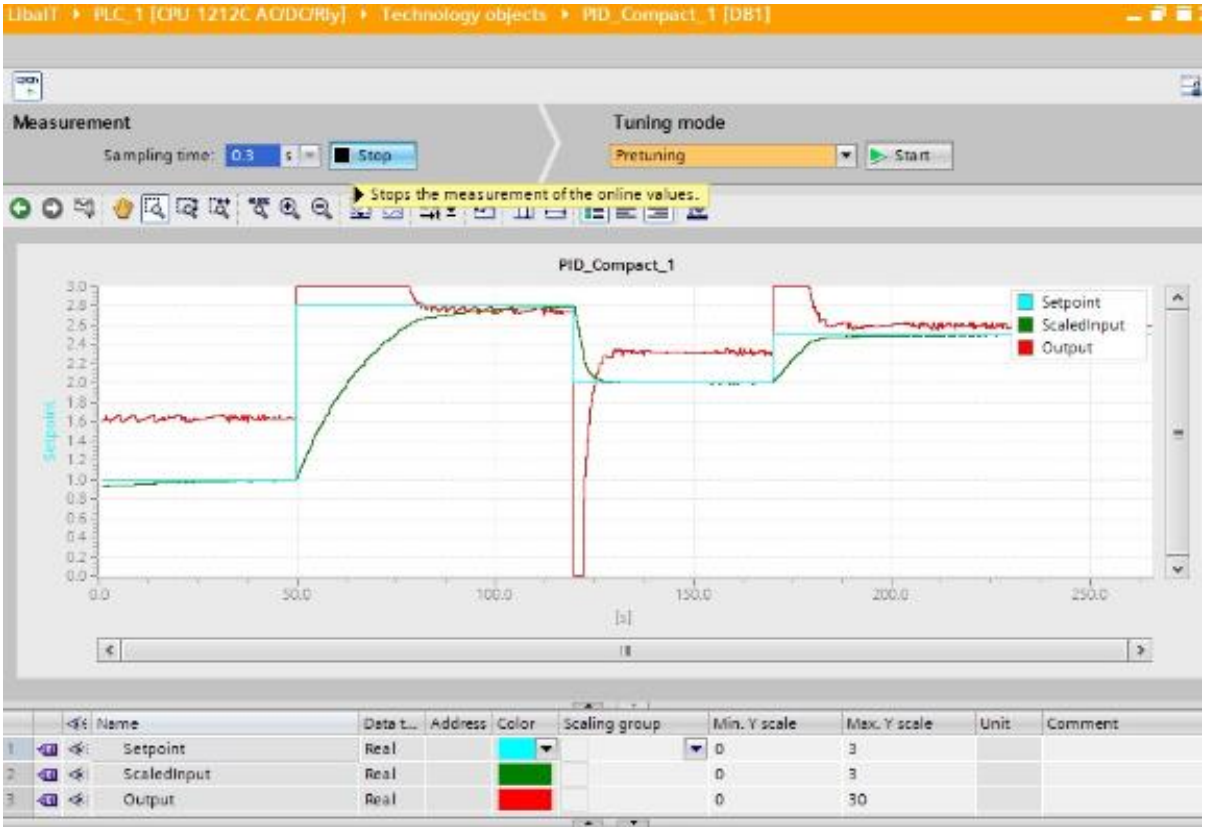
Obr. 3.7 – Nastavení parametrů regulátoru

3.4 ZÁZNAM HODNOT V ONLINE REŽIMU

Po vytvoření simulátoru a po vytvoření programu pro regulátor, kterým je v tomto případě PLC Siemens 1200, nehrajeme program z počítače do PLC. Poté propojíme PLC s modulem, až poté připojíme napájení PLC, které je 230 V. Na závěr připojíme k napájení modul STM 32 pomocí USB kabelu. Simulátor začne ihned vykonávat program. Pro zobrazení aktuálních hodnot vstupujících do regulátoru i akčních zásahů regulátoru se nechá připojené PLC k počítači a v prostředí Tia Portalu se přepne na režim online. V tomto režimu vidíme aktuálně nahraný program v našem PLC, je zde také vidět, jaké podmínky jsou

splněny, jaké části programu se vykonávají, u čítačů, které mají splněny podmínku, která je před nimi, je vidět také aktuální načítaný čas. Na posledním Networku s blokem PID regulátoru je vidět aktuální požadovaná hodnota, vstupní hodnota do regulátoru a aktuální akční veličina regulátoru, která je posílána přes analogový výstup PLC na vstup simulátoru.

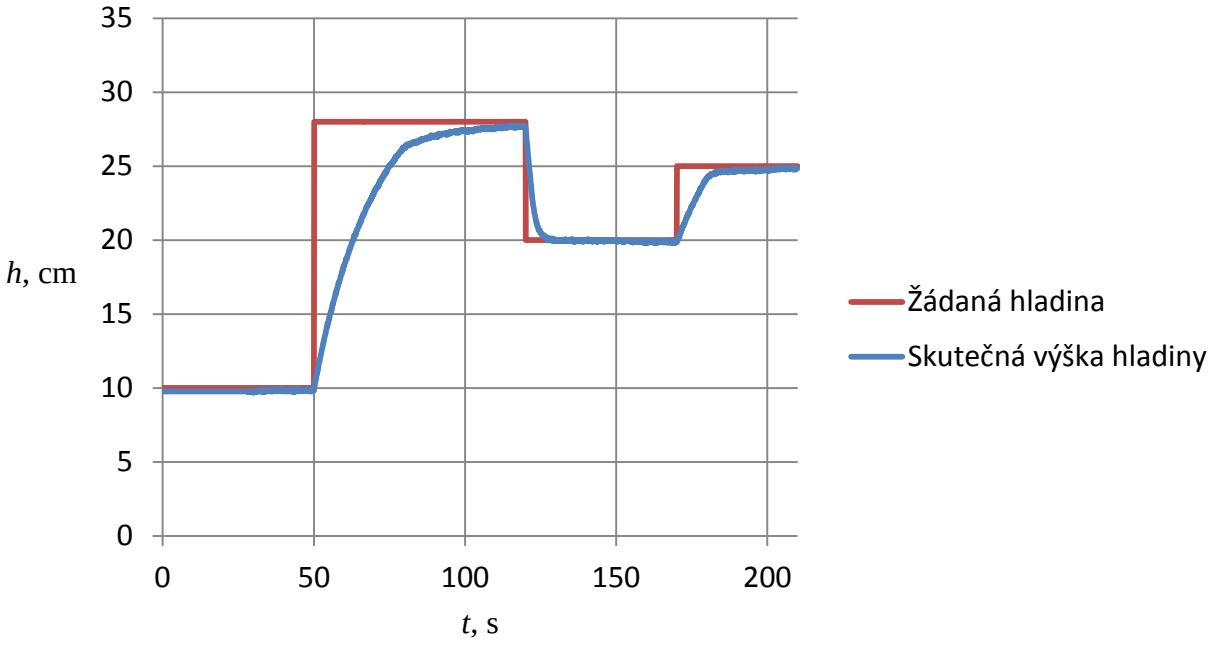
Všechny tyto hodnoty umožňuje Tia Portal také zobrazit v grafu. Graf se zobrazí po kliknutí na ikonu nastavení bloku „PID_Compact“, která se nachází v pravém horním rohu tohoto bloku. V této chvíli je graf prázdný a musí se nejprve spustit zaznamenávaní hodnot. Vybereme si, jak často chceme hodnoty do grafu zaznamenávat (Sampling time) a spustíme záznam. Na ose x se plynule mění měřítko, aby byl vidět celý zaznamenávaný průběh. Podrobněji s programem Tia Portal se čtenář může seznámit v (Berger, 2013).



Obr. 3.8 – Záznam pomocí TIAPORTALU

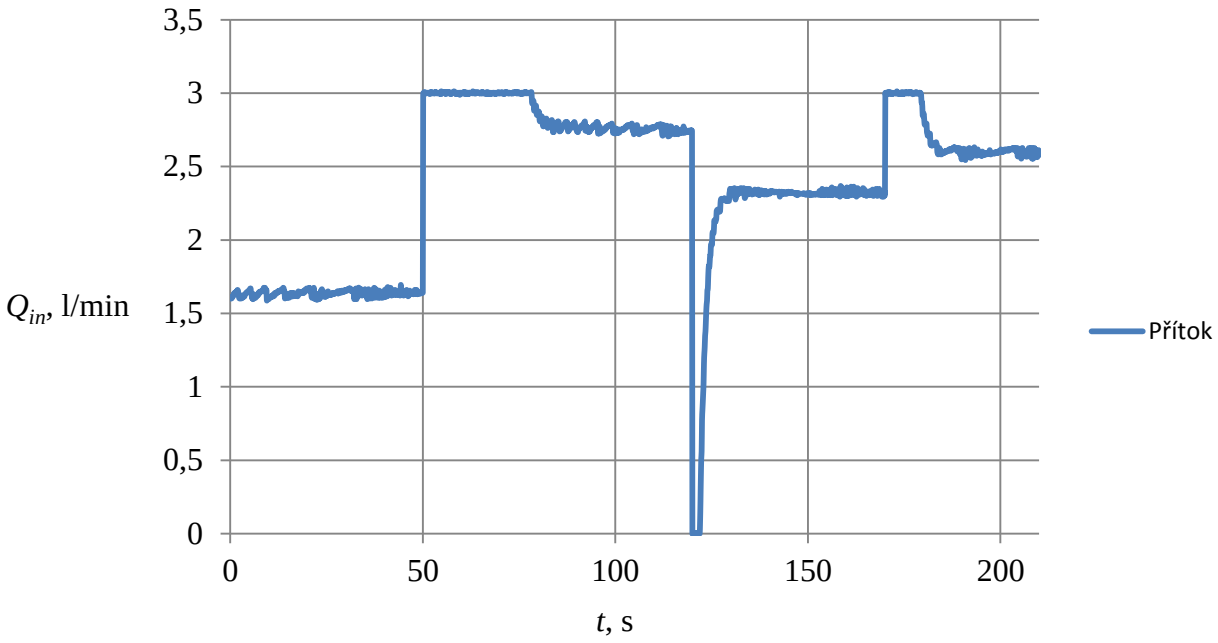
4 ZHODNOCENÍ

V předchozí kapitole již byly vidět nějaké výsledky, ale graf nezobrazoval výšku hladiny v centimetrech, ale zobrazoval jen, jak se v čase měnilo napětí na vstupu regulátoru. Přepočtením hodnoty napětí na výšku hladiny zobrazíme, jaká byla výška hladiny v čase. Dále si v grafu zobrazíme požadovanou hodnotu výšky hladiny, aby byla vidět odchylka mezi těmito hodnotami.



Obr. 4.1 – Průběh výšky hladiny v čase

Z grafu je vidět, že regulátor kolem žádané hodnoty nekmitá a na změnu žádané hodnoty reaguje okamžitě. Nejpomalejšího nastavení požadované hodnoty dosahuje při změně na vysokou žádanou výšku hladiny. To je ovšem dáno omezením přítoku do nádrže na 3 l/s. při změnách žádané hodnoty směrem dolu, jak je tomu například vidět na grafu v čase 120 s, je rychlost dosažení požadované hodnoty dána velikostí výtokového otvoru, protože regulátor nemůže nastavit přítok do nádrže na zápornou hodnotu. V následujícím grafu je znázorněna změna akční veličiny v čase při změně žádané hodnoty. Takto regulátor měnil akční veličinu na svém výstupu, aby dosáhl výšky hladiny z obr. 4.1.



Obr. 4.2 – Průběh akční veličiny v čase

Na tomto grafu je vidět, že akční veličina mírně kmitá, ovšem v tak rychlých časech a zanedbatelně, že se to na výšce hladiny v nádrži neprojevilo. Také je na grafu vidět, že regulátor v čase 50 s, reagoval na změnu žádané hodnoty okamžitě a nastavil přítok na maximum. V čase 120 s je zase vidět, že přítok do nádrže na chvíli úplně zavřel. Z těchto grafů vyplývá, že regulátor je správně nastavený, a byl by proto vhodný k použití regulace soustavy, jejíž model jsme vytvořili pomocí simulátoru.

5 ZÁVĚR

Cíle této práce byly splněny. Byl vytvořen hardwarový simulátor SISO soustavy a byla ověřena jeho funkčnost. Tento simulátor by mohl po změně hlavní části programu, kde je rovnice popisující danou soustavu a přepočet vstupního napětí na přítok a přepočet výstupního napětí na hladinu, simulovat téměř jakoukoli spojitou SISO soustavu. Zbylé funkce použité v programu simulátoru by mohly být ponechány a použity. Nebylo by příliš vhodné simulovat soustavu, kde je velký rozsah jejího vstupu, nebo výstupu, a zároveň požadována vysoká přesnost. Zde by funkčnost simulátoru omezoval zabudovaný AD, DA převodník.

Dále byla provedena regulace vytvořeného simulátoru za použití PLC Siemens 1200. Do PLC byl vytvořen program, který v časových intervalech mění žádanou výšku hladiny a pomocí softwarového regulátoru obsaženého v PLC byla provedena regulace simulátoru. Vytvořený regulátor nekmital, a jeho reakce na změnu žádané hodnoty byly rychlé.

Za použití vytvořeného simulátoru je možné testovat odezvy soustavy a regulátoru na změnu žádané hodnoty a to i pro stavy, které by byly v praxi těžké, nebo nevhodné testovat.

LITERATURA

BALÁTĚ, J. 2003. *Automatické řízení*. Praha: BEN, 654 s. ISBN 80-7300-020-2.

BERGER, B. 2013. *Automating with SIMATIC S7-1200: configuring, programming and testing with STEP 7 Basic, visualization with HMI Basic*. 2nd enlarged and revised edition. Erlangen: Publicis Publishing. ISBN 9783895783852.

KERNIGHAN, B.; RITCHIE, D. 2013. *Programovací jazyk C*. Brno: Computer Press, 2006. ISBN 80-251-0897-X.

OLLATZ, L. 1970. *Funkcionální analýza a numerická matematika.* Praha: Státní nakladatelství technické literatury. 418 s.

STM32 datasheet. STMicroelectronics. 2013 [online]. [cit. 1. 5. 2016]. Dostupné z: http://www.st.com/stonline/stappl/resourceSelector/app?page=fullResourceSelector&doctype=datasheet&SeriesID=1574.

PŘÍLOHY

A - CD

Příloha A

Příloha k bakalářské práci

Stm32 jako hardwarový simulátor spojitých dynamických soustav

Tomáš Líbal

CD

Obsah

1 Text bakalářské práce ve formátu PDF

2 Projekt v Atollic, TrueSTUDIO pro modul STM32

3 Projekt v Tia Portalu pro PLC Siemens 1200