UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

SOFTWARE PRO OBSLUHU A NASTAVOVÁNÍ PRŮMYSLOVÉHO REGULÁTORU

Miroslav Musílek

Diplomová práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Miroslav Musílek**
Osobní číslo:	**I15185**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Název tématu:	**Software pro obsluhu a nastavování průmyslového regulátoru**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cílem práce je navrhnout a realizovat software pro komunikaci průmyslového regulátoru s počítačem, který bude umožňovat nastavování parametrů regulátoru, ukládání dat do počítače, příp. jejich dalšího zpracování a vizualizace.
Teoretická část: Rešerše různých softwarových řešení používaných pro komunikaci průmyslových regulátorů s PC. Popis komunikačních protokolů s důrazem na standard EIA-485 (RS485) a protokol Modbus RTU. Popis konkrétního průmyslového regulátoru, který bude v rámci řešení práce vybrán a pořízen.
Implementační část: Realizace software pro obsluhu a nastavování vybraného průmyslového regulátoru. Ověření činnosti software při řízení zvolené laboratorní soustavy.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

BOBÁL, V.; BOHM, J.; FESSL, J.; MACHÁČEK, J. 2005. Digital Self-tuning Controllers: Algorithms, Implementation and Application. Londýn: Springer. 317 s. ISBN 978-1-85233-980-7.
ASTROM, K.; HAGGLUND, T. 1995. PID Controllers: Theory, Design, and Tuning. 2 vyd. Research Triangle Park (USA): Instrument Society of America. 343 s. ISBN 1-55617-516-7.
HLAVA, J. 2000. Prostředky automatického řízení II: analogové a číslicové regulátory, elektrické pohony, průmyslové komunikační systémy. Praha: ČVUT. 160 s. ISBN 80-01-02221-8.

Vedoucí diplomové práce: **Ing. Libor Kupka, Ph.D.**
Katedra řízení procesů

Datum zadání diplomové práce: **24. října 2016**
Termín odevzdání diplomové práce: **17. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 01. 05. 2017

Miroslav Musílek

Poděkování

Na tomto místě bych rád poděkoval Ing. Liborovi Kupkovi Ph.D. za vedení mé práce. Dále bych rád poděkoval své rodině za podporu během studia a v neposlední řadě svým přátelům.

V Pardubicích dne 01. 05. 2017

Miroslav Musílek

ANOTACE

Práce je věnována problematice průmyslových regulátorů s důrazem na důležitost komunikačního rozhraní. V rámci práce je vypracována aplikace pro ovládání zvoleného průmyslového PID regulátoru. Tato aplikace slouží pro ovládání parametrů zařízení a sběr naměřených dat včetně vizualizace. Ověření činnosti zařízení a programu proběhlo na příkladu zvolené regulační soustavy.

KLÍČOVÁ SLOVA

průmyslový regulátor, modbus, EIA/TIA 485, C#, XML

TITLE

INDUSTRIAL CONTROLLER SETTING AND OPERATING SOFTWARE

ANNOTATION

The thesis deals with the issue of industrial controllers and emphasizes the importance of communication interface. In the framework of the thesis an application is developed for controlling a specified industrial PID controller. This application is used to control device parameters and to collect measured data including visualization. The device and the application are verified by the example of selected control system.

KEYWORDS

Industrial control, Modbus, EIA/TIA 485, C#, XML

OBSAH

SEZNAM ZKRATEK A ZNAČEK

ADU	aplikační datový rámec
ASCII	americký standardní kód pro výměnu informací
CRC	cyklický redundantní součet
DOM	objektový model dokumentu
EEPROM	paměť pouze pro čtení vymazatelná elektricky
EPROM	paměť pouze pro čtení vymazatelná UV zářením
EXE	spustitelný soubor
LRC	podélný redundantní součet
LT	zakončovací odpor
MBAP	Hlavička modbus aplikačního protokolu
MIT	Massachusettský technologický institut
MV	měřená hodnota
PC	osobní počítač
PDU	protokolový datový rámec
PID	proporcionálně integračně derivační (regulátor)
PLC	programovatelný logický automat
PV	procesní hodnota
PWM	pulzně šířková modulace
RAM	paměť s přímým přístupem
ROM	paměť pouze pro čtení
RTU	vzdálená terminálová jednotka
SAX	základní aplikační programovací rozhraní pro XML
SSR	relé v pevné fázi
SV	nastavená hodnota
TCP	transportní kontrolní protokol
UART	univerzální synchronní / asynchronní sériové rozhraní
USB	univerzální sériová sběrnice
USD	americký dolar
WPF	Windows presentation foundation
XML	rozšiřitelný značkovací jazyk

SEZNAM SYMBOLŮ PROMĚNNÝCH VELIČIN A FUNKCÍ

α	koeficient filtrace
C	elektrická kapacita, F
D	derivační časová konstanta paralelní reprezentace, s
$d(k)$	diskrétní derivační složka
$D(s)$	Laplaceův obraz derivační složky
$E(s)$	Laplaceův obraz regulační odchylky
e(t)	regulační odchylka
$F(s)$	obrazový přenos regulované soustavy
f_0	maximální frekvence obsažena v signálu, Hz
f_m	vzorkovací frekvence, Hz
I	integrační časová konstanta paralelní reprezentace, s
K_P	proporcionální zesílení
K_{PS}	proporcionální zesílení sériové reprezentace
P_b	pásmo proporcionality, %
R	elektrický odpor, Ω
s	Laplaceův operátor
T_0	vzorkovací perioda, s
T_a	dominantní časová konstanta, s
T_D	derivační časová konstanta, s
T_{DS}	derivační časová konstanta sériové reprezentace, s
T_I	integrační časová konstanta, s
T_{IS}	integrační časová konstanta sériové reprezentace, s
T_Σ	souhrnná časová konstanta.
$U(s)$	Laplaceův obraz akční veličiny
$u(t)$	akční veličina
$u_s(t)$	akční veličina v saturačním stavu akčního členu
$U_1(s)$	Laplaceův obraz vstupního signálu
$u_1(t)$	vstupní signál, V
$U_2(s)$	Laplaceův obraz výstupního signálu
$u_2(t)$	výstupní signál, V
$w(t)$	žádaná veličina
$y(t)$	regulovaná veličina

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

V současnosti se v průmyslových aplikacích algoritmy řízení nejčastěji implementují jako součást nadřazených systémů. Zpravidla jsou implementovány v řídicím automatu (PLC). V jednoduchých, samostatných aplikacích si ovšem uživatel často vystačí bez přítomnosti dalších vstupně výstupních či měřicích karet. Modulární řídicí automaty jsou pro tyto aplikace příliš drahé a ve většině případů jsou určeny pro rozsáhlejší aplikace. V těchto případech je dostačujícím řešením kompaktní průmyslový regulátor. Ten je vybaven pouze omezeným počtem vstupů a výstupů a může být vybaven i komunikačním rozhraním. Pro toto komunikační rozhraní často není výrobcem popsán použitý komunikační protokol. Tím je komunikace omezena pouze na možnost spojení s nadřazenými systémy téhož výrobce. Některé regulátory však mají komunikační protokol popsán podrobněji a umožňují připojení ke standardizované průmyslové sběrnici. Na základě této specifikace je možné napsat program pro komunikaci se zařízením.

V rámci práce je vybrán a pořízen kompaktní průmyslový regulátor. Hlavním parametrem při výběru zařízení je především cena a přítomnost komunikačního rozhraní s alespoň minimálním popisem komunikačního protokolu a rozhraní. Dále je navrhnuta a vyvinuta aplikace pro komunikaci regulátoru s počítačem, pro pohodlnější obsluhu zařízení a pro ukládání a vizualizaci měřených dat. Činnost programu a vlastností samotného zařízení jsou ověřeny při řízení zvoleného modelu soustavy.

1 PID REGULÁTORY

PID regulátory jsou v průmyslu nejpoužívanějšími regulátory. Zákonem řízení zůstává standardní PID algoritmus ovšem doplněný pokročilými funkcemi a modifikacemi pro jejich praktické využití.

1.1 IDEÁLNÍ PID REGULÁTOR

Ideální tvar rovnice spojitého PID regulátoru je

$$u(t) = K_{\mathrm{P}}\left[e(t) + \frac{1}{T_{\mathrm{I}}}\int_0^t e(\tau)\mathrm{d}\tau + T_{\mathrm{D}}\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right], \tag{1.1}$$

kde $e(t) = w(t) - y(t)$ – regulační odchylka,

$u(t)$ – akční veličina,

$w(t)$ – žádaná veličina,

$y(t)$ – regulovaná veličina.

Parametry PID regulátoru (1.1) jsou proporcionální zesílení K_{P}, integrační časová konstanta T_{I} a derivační časová konstanta T_{D}. Pomocí Laplaceovy transformace je možné převést rovnici do obrazového tvaru

$$U(s) = K_P\left[1 + \frac{1}{T_{\mathrm{I}}s} + T_D s\right]E(s), \tag{1.2}$$

kde $E(\mathrm{s})$ – Laplaceův obraz regulační odchylky,

$U(\mathrm{s})$ – Laplaceův obraz akční veličiny,

s – Laplaceův operátor.

Tato rovnice se nazývá ideálním spojitým PID regulátorem (Bobál, 2005).

1.2 IDEÁLNÍ PSD REGULÁTOR

Nahrazením spojité integrace a derivace v rovnici PID regulátoru je získána rovnice tzv. proporcionálně sumačně diferenčního regulátoru. Derivační složka je v nejjednodušším případě aproximována první zpětnou diferencí

$$\frac{\mathrm{d}e}{\mathrm{d}t} \approx \frac{e(k) - e(k-1)}{T_0} = \frac{\Delta \mathrm{e(k)}}{T_0}, \tag{1.3}$$

kde $e(k)$ – diskrétní vzorek regulační odchylky v okamžiku $t = kT_0$,

T_0 – perioda vzorkování, s.

Spojitý integrál může být aproximován v nejjednodušším případě obdélníkovou metodou, a to buď přímou

$$u(k) = K_P \left\{ e(k) + \frac{T_0}{T_I} \sum_{i=1}^{k} e(i-1) + \frac{T_D}{T_0} \left[e(k) - e(k-1) \right] \right\}, \tag{1.4}$$

kde $u(k)$ – diskrétní vzorek akční veličiny v okamžiku $t = kT_0$,

$e(k)$ – diskrétní vzorek regulační odchylky v okamžiku $t = kT_0$,

K_P – proporcionální zesílení,

T_I – integrační časová konstanta, s,

T_D – derivační časová konstanta, s,

T_0 – perioda vzorkování, s,

nebo zpětnou

$$u(k) = K_P \left\{ e(k) + \frac{T_0}{T_I} \sum_{i=1}^{k} e(i) + \frac{T_D}{T_0} \left[e(k) - e(k-1) \right] \right\}. \tag{1.5}$$

V případě požadavku na přesnější aproximaci je použita lichoběžníková metoda

$$u(k) = K_P \left\{ e(k) + \frac{T_0}{T_I} \left[\frac{e(0) + e(k)}{2} + \sum_{i=1}^{k-1} e(i) \right] + \frac{T_D}{T_0} \left[e(k) - e(k-1) \right] \right\}. \tag{1.6}$$

Tím je průběh regulační odchylky nahrazen nikoliv po částech konstantní funkcí, ale lineárně rostoucími nebo klesajícími úseky. Pokud je vzorkovací frekvence dostatečně vysoká, není významný rozdíl mezi chybou způsobenou použitím odlišné metody integrálního přiblížení. Nejčastěji se používá zpětná obdélníková metoda.

V uvedených vztazích je již aproximace integrační i derivační složky dosazena přímo do rovnice regulátoru. Tyto rovnice jsou označovány jako polohové nebo nerekurentní rovnice. Jedná se o přímou diskrétní analogii k rovnici (1.1) a jejich výstupem je okamžitá hodnota akční veličiny. Pro tento tvar platí, že pro výpočet integrace a akční veličiny je nutné znát všechny předešlé hodnoty regulační odchylky $e(k - i)$, kde $i = 1, 2 \ldots k$. To je velice nepraktická vlastnost. Bylo by nutné uchovávat všechny hodnoty v paměti. Dále změna parametrů T_I nebo K_P vede k okamžité změně hodnoty celé integrální části, což způsobuje přetížení celého výpočtu. Z tohoto důvodu není polohový tvar rovnic vhodný ani pro změnu parametrů v regulátoru.

Proto se často používají tzv. přírůstkové algoritmy, které jsou použitelné pro rekurzivní výpočet nebo ve spolupráci s akčními členy s integračním charakterem. Hodnota výstupu

regulátoru $u(k)$ je vypočítána součtem dříve zaznamenané hodnoty $u(k-1)$ a vypočítaného přírůstku $\Delta u(k)$. Algoritmy pro výpočet $\Delta u(k)$ se označují jako přírůstkové, rychlostní nebo inkrementální algoritmy.

Za předpokladu použití obdélníkové zpětné metody pro integrální přiblížení je odečtením rovnic (1.5) pro časové okamžiky k a $k-1$ získán rekurentní vztah

$$\Delta u(k) = u(k) - u(k-1), \tag{1.7}$$

$$\Delta u(k) = K_{\mathrm{P}}\left\{e(k) - e(k-1) + \frac{T_0}{T_{\mathrm{I}}}e(k) + \frac{T_{\mathrm{D}}}{T_0}\left[e(k) - 2e(k-1+e(k-2))\right]\right\}. \tag{1.8}$$

Tento obecný tvar lze převést na univerzálně platný rekurentní tvar PSD regulátoru

$$u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) + u(k-1), \tag{1.9}$$

kde pro rekurentní rovnici s aproximací integrační složky pomocí zpětné obdélníkové metody, jsou parametry

$$q_0 = K_{\mathrm{P}}\left(1 + \frac{T_0}{T_{\mathrm{I}}} + \frac{T_{\mathrm{D}}}{T_0}\right), \tag{1.10}$$

$$q_1 = -K_{\mathrm{P}}\left(1 + \frac{2T_{\mathrm{D}}}{T_0}\right), \tag{1.11}$$

$$q_2 = K_{\mathrm{P}}\frac{T_{\mathrm{D}}}{T_0}. \tag{1.12}$$

Parametry regulátoru q_0, q_1 a q_2 jsou funkce proporcionálního zesílení K_{P}, časové konstanty integrační T_{I} a derivační T_{D} a periody vzorkování T_0. Z toho vyplívá podobnost s PID regulátorem a možný přepočet hodnot parametrů q z parametrů spojitého PID (Bobál, 2005; Hlava, 2000).

1.3 PARALELNÍ A SÉRIOVÁ STRUKTURA

Tvar rovnice PID algoritmu (1.2) se nazývá standartním. Tyto regulátory se postupně stále výrazněji prosazují. Ovšem dosud nejrozšířenější při implementaci v průmyslových regulátorech je forma s tzv. sériovou strukturou

$$U(s) = K_{\mathrm{PS}}\left(1 + \frac{1}{T_{\mathrm{IS}}s}\right)\left(1 + T_{\mathrm{DS}}s\right)E(s), \tag{1.13}$$

kde K_{PS} – proporcionální zesílení sériové reprezentace,

T_{IS} – integrační časová konstanta sériové reprezentace, s,

T_{DS} – derivační časová konstanta sériové reprezentace, s.

Lze se s touto formou setkat u analogových, elektronických i číslicových regulátorů. K realizaci sériové struktury PID regulátoru stačí pouze jeden zesilovací prvek. Z historického hlediska jsou jednodušší na realizaci. V současnosti hlavním důvodem implementace v elektronických regulátorech není úspora nákladů, ale snaha respektovat tradiční a v praxi zažitou podobu PID regulátoru.

V některých případech se používá implementace regulátoru s paralelní strukturou

$$U(s)=\left(K_{\mathrm{P}}+\frac{1}{Is}+Ds\right)E(s), \tag{1.14}$$

kde I – integrační časová konstanta paralelní reprezentace, s,

D – derivační časová konstanta paralelní reprezentace, s.

Parametry mezi vztahy standartní (1.2) a paralelní struktury (1.14) lze převést jednoduše. Přepočet parametrů regulátoru v sériovém tvaru (1.13) na parametry regulátoru standartní struktury (1.2) je možný pomocí vztahů

$$K_{\mathrm{P}}=K_{\mathrm{PS}}\left(1+\frac{T_{\mathrm{DS}}}{T_{\mathrm{IS}}}\right),\quad T_{\mathrm{I}}=T_{\mathrm{IS}}+T_{\mathrm{DS}},\quad T_{\mathrm{D}}=\frac{T_{\mathrm{IS}}T_{\mathrm{DS}}}{T_{\mathrm{IS}}+T_{\mathrm{DS}}}. \tag{1.15}$$

Při použití sériové struktury se velikost jednotlivých parametrů během jejich nastavování vzájemně ovlivňuje. Dochází k tzv. interakci konstant regulátoru. Pokud je regulátor nastavován experimentálně, jsou komplikace při přepočtu parametrů poměrně nepodstatné. Udává se, že regulátory se sériovou strukturou se experimentálně nastavují snáze než regulátory s paralelní strukturou.

U řady komerčně dostupných regulátorů není uvedena konkrétní struktura regulátoru, která je implementována. Není zřejmé, zda se jedná o zadávané parametry T_I, T_{IS}, I, respektive o T_D, T_{DS}, D. Skutečnost jiné reprezentace parametrů regulátoru může vést k velmi odlišnému chování (Bobál, 2002; Hlava, 2000).

1.4 FILTRACE DERIVAČNÍ SLOŽKY

Pokud je derivační složka PID regulátoru ovlivněna šumem, a má být aproximována jednoduchou diferencí prvního řádu, může šum způsobovat nežádoucí změny na výstupu regulátoru. Složka derivace musí být omezena. K tomu se nejčastěji používá filtr prvního nebo

druhého řádu, který snižuje zisk na vyšších frekvencích. Většinou je použit jednoduchý filtr prvního řádu. Derivační složka je pak ve tvaru

$$D(s) = K_{\mathrm{P}} \frac{T_{\mathrm{D}} s}{\frac{T_{\mathrm{D}}}{\alpha} s + 1} E(s), \tag{1.16}$$

kde α – koeficient filtrace,

$D(s)$ – Laplaceův obraz derivační složky,

$E(s)$ – Laplaceův obraz regulační odchylky,

K_{P} – proporcionální zesílení,

T_{D} – derivační časová konstanta, s.

Hodnota koeficientu filtrace je obvykle nastavitelná a pohybuje v rozmezí od 3 do 20. Když je hodnota koeficientu 10, tak filtr derivační složky má časovou konstantu desetkrát menší, než je derivační časová konstanta. Spojitá derivační složka (1.16) je nejčastěji aproximována za použití zpětné obdélníkové integrace ve tvaru

$$d(k) = \frac{T_{\mathrm{D}} d(k-1) + K_{\mathrm{P}} T_{\mathrm{D}} \alpha [e(k) - e(k-1)]}{T_{\mathrm{D}} + \alpha T_0}, \tag{1.17}$$

kde $d(k)$ – diskrétní derivační složka (Bobál, 2005).

1.5 VÝBĚR VZORKOVACÍ PERIODY

Vzorkovací perioda je nastavitelným parametrem pro digitální regulátory a její velikost se obvykle pohybuje mezi jednou setinou sekundy a jednou sekundou. Jedná se o jeden z hlavních parametrů. Pokud platí vztah

$$T_0 << T_a, \tag{1.18}$$

kde T_0 – perioda vzorkování, s,

T_{a} – dominantní časová konstanta regulovaného systému, s,

je možné z hlediska návrhu pracovat s diskrétním PID regulátorem prakticky stejně jako se spojitým. Obecně platí, že zkrácení periody vzorkování T_0 zlepšuje kvalitu kontroly a schopnost reagovat na poruchy. Příliš krátká vzorkovací perioda ovšem zvyšuje nároky na akční člen. Dochází ke zbytečnému namáhání akčního členu a snižuje se tak jeho životnost. Přílišné zkrácení periody vzorkování může také způsobit i zhoršení kvality regulace a nestabilitu systému. Snižování vzorkovací periody obvykle vede k nárůstu spotřeby energie. Ta je nejčastěji posuzována dle integrálního kvadratického kritéria např.

$$J = \int_0^{\infty} u^2(t)\,\mathrm{d}t\,, \tag{1.19}$$

kde $u(t)$ – akční veličina.

Prodloužením periody vzorkování T_0 se ve většině případů zpomaluje proces kontroly, ale významně se snižují změny akční veličiny a hodnota kritéria (1.19). Volba periody vzorkování je především ovlivněna požadovanou úrovní kontroly (požadavky na rychlost odezvy na řízenou veličinu, na změnu regulační odchylky a akční veličiny), dynamikou řízeného systému a vlastnostmi pohonu (doba odezvy, povolený počet sepnutí akčního členu za hodinu atd.).

Minimální vzorkovací frekvence je dána Shannonovým teorémem. Jedná se o minimální možnou frekvenci potřebnou k tomu, aby ze vzorků signálu bylo vůbec možné rekonstruovat původní průběh. Tato problematika je podrobněji popsána v pododdílu 1.6.3. Pro dosažení rozumné přesnosti je nutné použít kratší periodu vzorkování, než by odpovídalo těmto minimálním požadavkům. Velice zjednodušeně je doporučováno, aby vzorkovací perioda byla přibližně desetinou T_a pro soustavy s jednou dominantní časovou konstantou

$$T_0 < 0{,}1T_\mathrm{a}\,. \tag{1.20}$$

kde T_0 – perioda vzorkování, s,

T_a – dominantní časová konstanta regulovaného systému, s.

U systémů s několika dominantními časovými konstantami lze požadovat

$$T_0 < 0{,}1T_\Sigma\,, \tag{1.21}$$

kde T_Σ – souhrnná časová konstanta, s.

To platí pro statický nekmitavý systém s přenosem

$$F(s) = K_\mathrm{S}\,\frac{(T_{1\mathrm{N}}s+1)(T_{2\mathrm{N}}s+1)\cdots(T_{m\mathrm{N}}s+1)}{(T_1s+1)(T_2s+1)\cdots(T_ns+1)}\,s^{-T_\mathrm{D}s}\,, \tag{1.22}$$

kde F(s) – obrazový přenos regulované soustavy,

K_s – zesílení systému,

je souhrnná časová konstanta definovaná vztahem

$$T_\Sigma = T_1 + T_2 + \cdots + T_n - T_{1\mathrm{N}} - T_{2\mathrm{N}} - \cdots - T_{m\mathrm{N}} + T_\mathrm{D}\,. \tag{1.23}$$

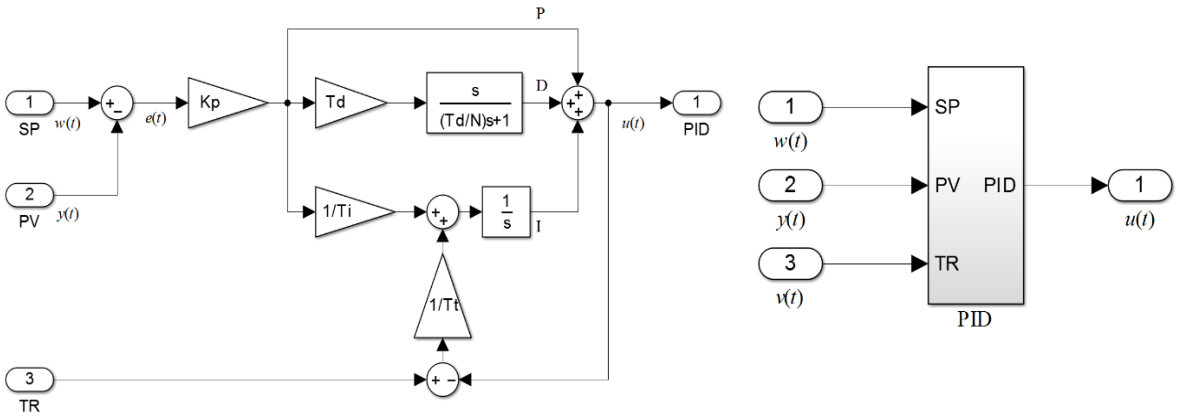
Protože lze většinu běžných regulovaných procesů považovat za relativně pomalé, umožňují rychlosti současných mikroprocesorů splnit podmínku s rezervou a volit periodu vzorkování spíše v setinách souhrnné časové konstanty (Bobál, 2005; Hlava, 2000).

1.6 PID REGULÁTORY PRO PRAKTICKÉ POUŽITÍ

Aby byly základní algoritmy pro regulaci použitelné pro praktické využití, je nutné je doplnit funkcemi jako je filtrace vstupních signálů, dopředná vazba, přepínání sad parametrů regulátoru, ošetření mezních stavů, beznárazové přepínání režimů atd.

1.6.1 Beznárazové přepínání regulačních algoritmů a parametrů

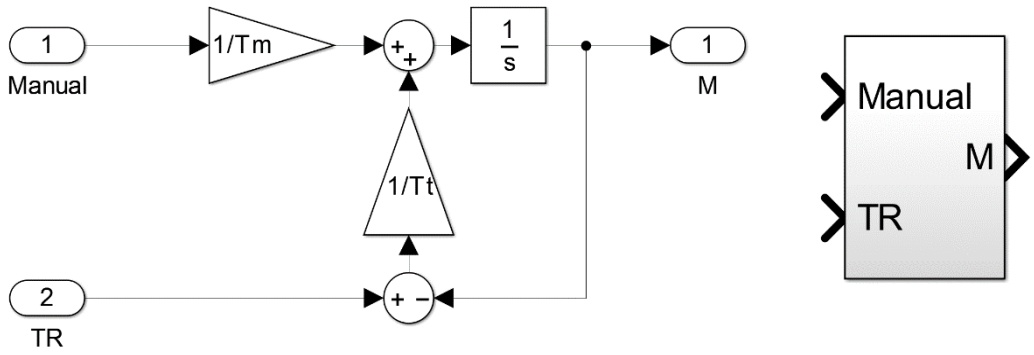
Prakticky všechny regulátory lze spustit minimálně ve dvou režimech. A to většinou v ručním nebo automatickém režimu. Některé regulátory implementují i více automatických regulačních algoritmů mezi kterými lze přepínat. V ručním režimu je obvykle výstup regulátoru nastavitelný obsluhou pomocí tlačítek na přístroji. Když je systém přepnut v ručním režimu, tak regulační algoritmus vytváří řídicí signál nezávisle na signálu z ručního řízení. Jejich výstupy se liší, a proto při následném přepnutí zpět na automatický režim může dojít na výstupu regulátoru k přechodovému jevu. Proto se u spojitých PID regulátorů zavádí na vstup integrátoru zpětná vazba odvozená od rozdílu mezi výstupem součtového členu a ručně nastaveným výstupem (obr. 1.1). Výstup součtového členu pak sleduje ručně nastavovanou hodnotu.



Obr. 1.1 – Spojitý PID regulátor se sledovacím vstupem

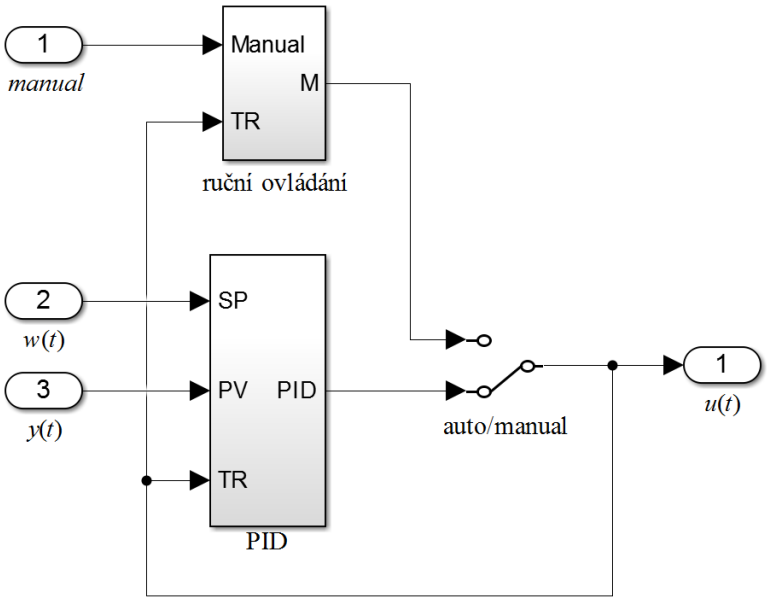
Blok ručního řízení (obr. 1.2) disponuje vstupem pro sledování TR a vstupem pro ruční řídicí příkazy Manual. Systém má dva parametry, časovou konstantu T_m pro ruční zadávání a

T_t pro dobu resetu. Blok ručního řízení obsahuje samostatný integrátor, který přidává přírůstkové změny z ručního řízení (Aström, 1995).



Obr. 1.2 – Blok ručního řízení se sledovacím vstupem

Kombinací bloku regulátoru PID se vstupem pro sledování na obr. 1.1 a bloku ručního ovládání na obr. 1.2 lze jednoduše sestavit kompletní regulátor s beznárazovým přepínáním. Sledovací vstupy zajišťují nastavení integrátoru v PID jednotce na správnou hodnotu, když regulátor je v manuálním režimu. Stejně tak integrátor spojený s ručním ovládáním je nastaven na správnou hodnotu, když je řídicí jednotka v automatickém režimu. To může být realizováno pomocí obvodu znázorněného na obr. 1.3. Přepnutí je hladké i v případě, že regulační odchylka nebo její derivace se liší od nuly v okamžiku spínání. Tento jev je založen na principu sledovacího mechanismu, který zajišťuje, že integrátor v manuálním ovládání sleduje výstup regulátoru a naopak (Aström, 1995).



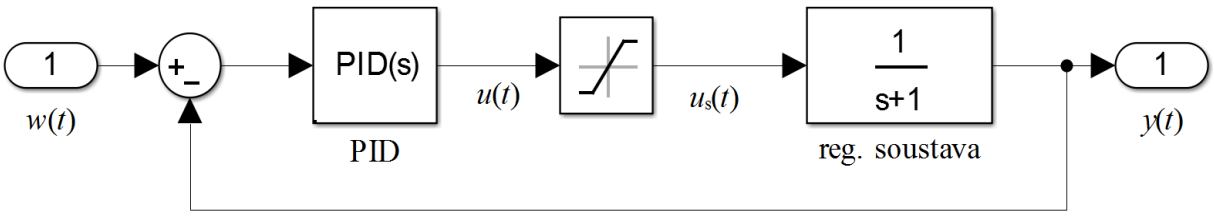
Obr. 1.3 – Beznárazové přepínání automatického a ručního řízení

U číslicových PSD regulátorů je nejjednodušším řešením použití řídicího algoritmu v přírůstkové podobě. Tento tvar rovnic regulátoru zajišťuje beznárazové přepnutí již ze své podstaty. Vzhledem k tomu, že přepínání ovlivňuje pouze inkrementaci, tak nedojde k žádnému velkému přechodovému jevu. Hodnota, na níž byl výstup ručně nastaven, se vezme jako počáteční podmínka $u(0)$ a k ní jsou pak přičítány přírůstky počítané například podle (1.5).

Podobný princip se při použití PSD regulátoru v přírůstkové podobě uplatní i při změně parametrů regulátoru. Přírůstkový tvar ovlivňuje pouze změnu akční veličiny a při změně parametrů regulátoru nedojde na výstupu regulátoru k žádnému velkému přechodovému jevu (Aström, 1995; Bobál, 2005; Hlava, 2000).

1.6.2 Unášení integrační složky

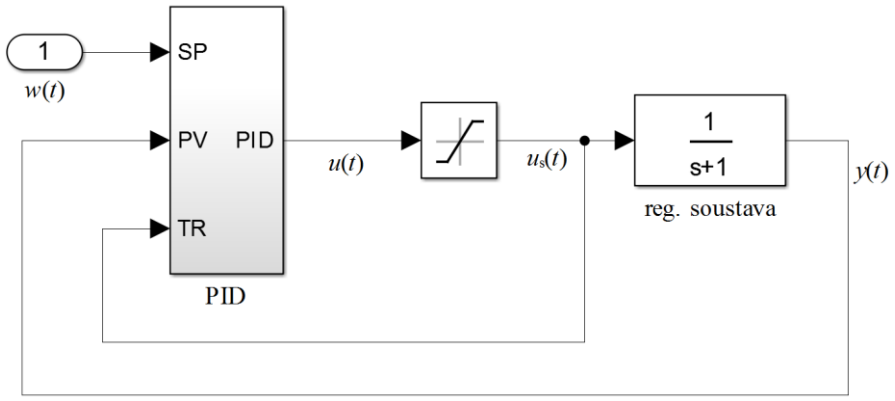
Každý akční člen je schopen realizovat řídicí signál jen v určitém rozsahu hodnot, protože například ventil nemůže být otevřen více než na maximum apod. To zavádí do regulačního obvodu nelinearitu. V okamžiku, kdy akční veličina $u(t)$ dosáhne limitu akčního členu $u_s(t)$, může být skutečný výstup procesu různý od výstupu regulátoru. V tomto případě dochází k přerušení zpětné vazby a systém běží jako v otevřené smyčce. U proporcionální a derivační složky regulátoru se tento jev většinou projeví pouze zpomalením regulačního pochodu oproti očekávání. Zpomalení je způsobeno menšími hodnotami akční veličiny. Dosáhne-li integrační složka saturační hodnoty realizovatelné akční veličiny a regulační odchylka ještě neklesla na nulu, tak integrační složka dále roste. To se ale neprojevuje na výstupu regulátoru. Integrační složka je tzv. unášena. U číslicového regulátoru může narůstat prakticky neomezeně. V momentě, kdy dojde ke změně znaménka regulační odchylky, začne velikost integrační složky opět klesat. Trvá dlouhou dobu, než se regulační obvod vrátí zpět do lineárního režimu, aby se pokles integrační složky projevil na hodnotě akční veličiny. Důsledkem je velký překmit opačným směrem a regulační pochod je kmitavý s dlouhou dobou ustálení (Hlava, 2000).



Obr. 1.4 – URO vznik unášení integrační složky

Řešení tohoto problému je celá řada. Implementovaná řešení v průmyslových regulátorech jsou většinou kombinací jednotlivých mechanismů. Nejjednodušším způsobem řešení je omezení rozsahu žádané veličiny tak, že se akční veličina nemůže dostat do limitního stavu akčního členu. Lepším řešením je dynamické omezení integrační složky, která používá modifikaci polohového tvaru rovnice s rekurzivním výpočtem integrační složky. Při překročení hodnoty integrační složky mimo realizovatelný rozsah je aktuální hodnotě přiřazena minulá hodnota.

V případě použití přírůstkového algoritmu, kde je sumace přírůstků akční veličiny prováděna akčním členem s integračním charakterem vně regulátoru, je ochrana proti unášení integrační složky zajištěna automaticky. Přírůstky vedoucí na hodnoty akční veličiny mimo rozsah prostě nejsou realizovány, protože to není fyzicky možné. Ovšem častěji se přírůstkový tvar používá pouze k výpočtu a následná sumace přírůstků je prováděna uvnitř regulátoru. Na výstup regulátoru je pak uplatňován některý z uvedených způsobů omezení. Ekvivalentního výsledku pak lze dosáhnout tím, že přírůstek, který by vedl k překročení mezí akční veličiny, bude pokládán za nulový. Této problematice se do větších podrobností věnuje (Aström, 1995).



Obr. 1.5 – Řešení unášení integrační složky regulátorem se sledovacím vstupem

Na obr. 1.5 je zobrazeno řešení unášení integrační složky pomocí regulátoru se sledovacím vstupem, uvedeným v předchozím pododdíle 1.6.1. Jakmile se dostane akční veličina do stavu saturace, potom je zpětnovazební signál $u(t) - u_s(t)$ kladný a působí na stav integrátoru tak, aby byla obnovena rovnost $u(t) = u_s(t)$. Rychlost konvergence $u(t) \rightarrow u_s(t)$ je dána časovou konstantou T_t. Tedy výstup regulátoru $u(t)$ je přesně roven horní saturační mezi $u_s(t)$ a tím je odstraněno unášení integrační složky (Aström, 1995; Hlava, 2000).

1.6.3 Filtrování měřené veličiny

U spojitého PID regulátoru jsou velké a rychlé změny regulační odchylky a akční veličiny přirozeně potlačeny. Diskrétní systém ovšem přijímá hodnoty žádané veličiny jako skokové změny proměnné v programu a nové hodnoty akční veličiny získává výpočtem dle (1.5). Měřená veličina je ovlivněna šumem a její vzorky jsou ovlivněny náhodnými chybami. Tyto šumy a chyby se projevují nežádoucími změnami regulační odchylky. Ty způsobují změny akční veličiny a tím i regulované veličiny. Dochází tak k nežádoucímu dynamickému namáhání akčního členu. Není narušena pouze činnost derivační složky regulátoru, ale méně výrazně i činnost ostatních složek. Negativní vliv nežádoucích změn regulační odchylky může být potlačen buď předzpracováním regulační odchylky nebo úpravou výchozího diskrétního algoritmu PID. Číslicové regulátory obvykle umožňují použít filtr regulované veličiny zabezpečující, že do všech složek regulátoru vstupuje signál, v němž byly rušivé složky alespoň částečně potlačeny. Obvykle se jedná o jednoduché filtry prvního řádu s uživatelsky nastavitelnou časovou konstantou a případnou deaktivací.

Rušivý signál vysoké frekvence by v analogovém regulačním obvodu nezpůsobil výraznější problémy a byl by utlumen buď vstupním filtrem regulátoru nebo samotnou regulovanou soustavou. Takový signál může být ovšem procesem vzorkování transformován na signál velmi nízké frekvence, tak může výrazně ovlivnit výpočet výstupu regulátoru a tím narušit průběh regulačního pochodu. Tomuto jevu se říká aliasing. Pro zajištění, aby frekvence diskrétního navzorkovaného signálu odpovídala frekvenci vstupního analogového signálu, je nutné splnit tzv. Shannonův teorém

$$f_{\mathrm{m}} > \frac{f_0}{2}, \tag{1.24}$$

kde f_{m} – maximální frekvence v signálu, Hz,

f_0 – vzorkovací frekvence, Hz.

Tato podmínka musí být bezpodmínečně splněna pro užitečný signál. Do obvodu vstupující rušivé signály vyšších frekvencí je nutné eliminovat. K tomuto účelu slouží tzv. antialiasingový filtr. Jedná se o dolnopropustní filtr, který odstraní ze signálu frekvence, které nesplňují podmínku (1.24). Filtr zároveň omezuje vliv šumu vzniklého při měření atd.

Perioda vzorkování běžných regulátorů určených k regulaci pomalých tepelných soustav se pohybuje řádově v desetinách sekundy. Analogový filtr s dostatečnou strmostí a mezní frekvencí splňující (1.24) pro tyto soustavy by měl složitější strukturu a požadavky na velké kondenzátory. Vzhledem k rozměrům potřebných kondenzátorů se často kombinují

analogová a číslicová filtrace. V signálové cestě za analogovým filtrem je realizován ještě číslicový filtr, který pracuje s menší periodou vzorkování než samotný regulační algoritmus. Tím se požadavek na mezní frekvenci pro analogový filtr posune podstatně výše. Poté bude struktura analogového filtru jednoduší a jeho kondenzátory pak budou výrazně menší. Často postačí jednoduchý RC článek (Bobál, 2005; Hlava, 2000).

1.6.4 Pásmo proporcionality

Proporcionální složka je často specifikována pásmem proporcionality (proportional band)

$$P_{\mathrm{b}} = \frac{1}{K_{\mathrm{P}}} 100, \tag{1.25}$$

kde P_{b} – pásmo proporcionality, %.

Veličina udává nepřímo zesílení regulátoru. Říká, jak velká změna regulační odchylky způsobí 100 % změnu akční veličiny. Pokud se zvolí malé proporcionální pásmo, tak už malá regulační odchylka stačí pro dosažení maximálního akčního zásahu. Zesílení se tedy zvyšuje s menším proporcionálním pásmem. Regulátor reaguje při nastavené nízké hodnotě proporcionálního pásma rychleji a častěji. Příliš malé proporcionální pásmo může vést k zákmitům na regulovaném obvodu. Změna proporcionálního pásma při použití (1.2) změní ve stejné míře také reakci integrační a derivační složky (Hlava, 2000).

2 PRŮMYSLOVÉ REGULÁTORY

Průmyslové regulátory jsou elektronická zařízení sloužící k řízení technologických procesů. Jedná se buď o relativně pomalé tepelné procesy nebo rychlé procesy např. ovládání servomotorů. V těchto zařízeních jsou implementovány diskrétní řídicí algoritmy, které jsou ve většině případů realizovány jednočipovými mikrokontrolery a doplňujícími periferními obvody. Mikrokontroler na jednom čipu integruje samotný mikroprocesor, potřebné paměti (ROM/EPROM, Flash EEPROM, RAM) a další funkční bloky jako A/D a D/A převodníky, časovací obvody, různé hlídací obvody atd. Číslicový regulátor řídí procesy spojité povahy, proto jsou zapotřebí A/D a D/A převodníky. Časovací obvody zajišťují stabilní časovou základnu pro zajištění konstantní vzorkovací frekvence. Dále jsou implementovány pomocné hlídací obvody zabezpečující definované chování regulátoru v případě nestabilního napájecího napětí či zacyklení programu v důsledku skryté programové chyby nebo rušení. V nejjednodušších regulátorech určených např. k jednosmyčkové regulaci teploty se běžně používají osmibitové jednočipové mikropočítače. Ve složitějších zařízeních se zpravidla používají více bitové mikroprocesory doplněné o další integrované obvody.

Algoritmy řízení implementované v komerčně dostupných kompaktních regulátorech se obecně velmi liší. Bohužel často nejsou detailně popsány ani v příslušné uživatelské příručce. A není tedy jasně dán přesný význam zadávaných parametrů a jak bude regulátor reagovat v nestandardních režimech (Bobál, 2005; Hlava, 2000).

2.1 ROZDĚLENÍ PRŮMYSLOVÝCH REGULÁTORŮ

Z hlediska konstrukčního provedení regulátorů je možné rozlišit na kompaktní a modulární regulátory, které lze dále rozdělit podle typu montáže. Kompaktní regulátory v jednom pouzdře obsahují vlastní mikrokontrolér, vstupy a výstupy včetně přizpůsobovacích obvodů, případně komunikační rozhraní i zobrazovací jednotku a tlačítka pro komunikaci s obsluhou. Zařízení jsou většinou určena pro panelovou montáž. Výrobce těchto přístrojů nabízí dodání v řadě variant provedení vstupů a výstupů, komunikačního rozhraní apod. Konfigurace těchto přístrojů probíhá formou specifikace požadovaných vlastností při objednávce. Následná uživatelská změna konfigurace přístroje je velice omezená. Většinou lze použít např. jiný typ teplotního snímače atd. Rozsáhlejší změna konfigurace zpravidla není možná. U modulárních regulátorů počet a provedení vstupů a výstupů záleží na tom, jaké moduly jsou použity a jejich výměnou či doplněním lze konfiguraci regulátoru v širokém rozsahu měnit.

Dále lze regulátory dělit dle míry možnosti uživatele ovlivnit chování samotného regulátoru. Většinou jsou možnosti uživatele ovlivnit chování regulátoru na programové úrovni omezeny pouze na nastavení konstant regulátoru, volbu mezi několika typy regulačního algoritmu (standardní PID, dvoupolohový regulátor apod.), zapnutí samočinného nastavování parametrů regulátoru apod. Existují však i programovatelné regulátory, kde uživatel může kombinací připravených funkčních bloků s časovými a logickými operacemi vytvářet poměrně snadno složité regulační struktury a v některých případech i programovat vlastní regulační algoritmy (Hlava, 2000).

2.2 KOMUNIKAČNÍ ROZHRANÍ

Komunikační rozhraní umožňuje provozovat zařízení v rámci rozsáhlejšího distribuovaného řídicího systému, nikoliv jen jako izolované zařízení. I v případě jednoduchého jednosmyčkového regulátoru může být užitečná např. změna žádané veličiny nebo případné monitorování jeho činnosti na dálku z nadřazeného počítače. U kompaktních regulátorů je většinou styk s okolím umožňující vytváření složitějších struktur řídicích systémů řešen pomocí sériových komunikačních rozhraní. Vybavení regulátoru komunikačním rozhraním často znamená, že je na něm k dispozici konektor pro připojení rozhraní TIA/EIA 232, 422 nebo 485, ale samotný komunikační protokol není popsán. To znamená, že komunikace je omezena pouze na možnost spojení s jinými zařízeními či nadřazenými systémy téhož výrobce. A proto je praktická použitelnost omezená. Některé regulátory však mají rozhraní popsáno podstatně lépe a umožňují tak připojení k nějaké standardizované průmyslové sběrnici (CAN, Profibus, Modbus apod.) (Hlava, 2000).

V nejjednodušším případě se jedná o rozhraní dovolující pouze dvoubodovou komunikaci mezi jedním přijímačem a jedním vysílačem TIA/EIA 232. Postupně vznikly požadavky provozovat komunikaci na větší vzdálenosti, mezi více zařízeními, a především s větší robustností vůči rušení. Proto vznikl standart TIA/EIA 485, který nevýhody TIA/EIA 232 eliminuje. Této problematice se podrobněji věnuje pododdíl 3.3.3.

2.3 VSTUPY A VÝSTUPY

A/D převodníky integrované v průmyslových regulátorech pracují s napěťovými signály o určité úrovni a rozsahu. Součástí číslicových regulátorů jsou proto analogové vstupní a přizpůsobovací obvody umožňující připojení různých rozsahů a typů signálů. Většina průmyslových regulátorů, především pro regulaci teploty, umožňuje přímé připojení běžných

čidel. Jedná se především o termočlánky různých typů (J, K…) a platinové měřicí odpory (PT100).

Zařízení jsou standardně vybavena vstupy a výstupy zpracovávající analogové signály unifikovaných rozsahů. Nejčastěji používaná je tzv. proudová smyčka o rozsahu od 4 mA do 20 mA. Tento rozsah umožňuje připojení tzv. dvouvodičových převodníků, které jsou po jednom vedení napájeny a zároveň zvyšováním svého odběru nad základní napájecí proud 4 mA vysílají informaci o hodnotě snímané veličiny. Skutečnost, že nulové hodnotě, resp. dolnímu konci rozsahu měřené veličiny, odpovídá nenulový proud 4 mA, umožňuje jednoduchou detekci přerušení vedení či poruchy snímače. Vedle tohoto základního proudového rozsahu bývají regulátory vybavovány i dalšími rozsahy: typicky od 0 mA do 20 mA. Mohou být implementovány i různé varianty napěťových rozsahů (typicky od 0 V do 10 V, od 2 V do 10 V). Napěťové signály jsou pro nevhodné pro přenos na delší vzdálenosti v průmyslovém prostředí zejména pro svoji velkou citlivost na rušení a možné znehodnocení v důsledku úbytku napětí na vedení. Často se používají také pomocné binární vstupy, jejichž pomocí lze na dálku přepínat mezi ručním a automatickým režimem provozu, regulací na konstantní hodnotu a sledováním hodnoty z pomocného analogového výstupu apod.

Důležitou součástí regulátoru jsou binární výstupy sloužící ke spínání menších zátěží. Jedná se zpravidla o kontakty elektromechanického relé realizované buď jako přepínací nebo pouze spínací či rozpínací. Proudová zatížitelnost bývá nanejvýše v jednotkách ampér. Při nutnosti spínat větší proudy jsou využity pro spínání externích výkonových relé. Alternativně jsou regulátory vybaveny logickými výstupy, obvykle s otevřeným kolektorem sloužící pro spínání například externích relé v pevné fázi. Relé v pevné fázi (SSR) je většinou určeno ke spínání střídavého proudu a je realizováno triakem. Slouží ve výkonových aplikacích jako náhrada elektromechanických relé, a to zejména z důvodu omezené životnosti mechanických kontaktů. Využívá se především v aplikacích PWM, kde dochází k častému spínání akčního členu.

Regulátory často disponují dalšími pomocnými výstupy. Například kontakty relé fungující jako alarmy spínající např. při vybočení hodnot regulované veličiny z přípustných mezí nebo v jiných uživatelsky konfigurovatelných situacích (Hlava, 2000).

2.4 PRŮMYSLOVÉ REGULÁTORY ŘADY ST828

Pro účely této práce byl vybrán kompaktní PID regulátor. Hlavními kritérii pro výběr byly cena koncového zařízení a přítomnost komunikačního rozhraní s alespoň minimálním

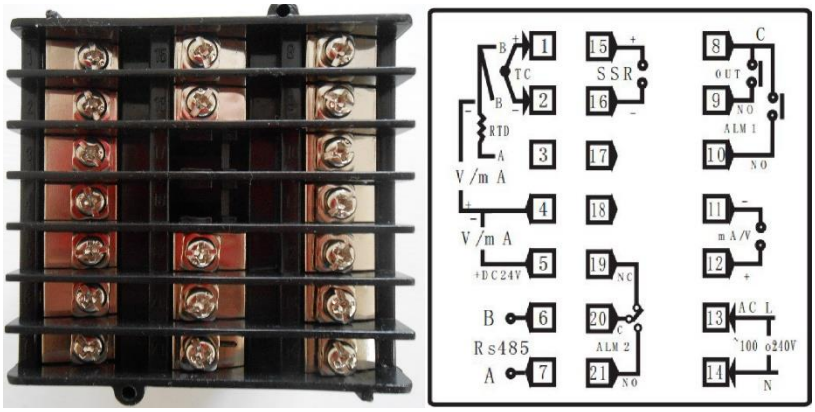
popisem komunikačního protokolu. Po analýze trhu padla volba na zařízení řady ST828 od výrobce Morning Eletronics Co. Zmiňovaná společnost má sídlo v Číně a zabývá se výrobou průmyslové elektroniky a obchodem s ní přes čínského internetového giganta alibaba.com. Jedná se o multifunkční inteligentní regulátory řízené mikroprocesorovou jednotkou určené pro řízení především pomalých tepelných soustav. Tato zařízení vynikají především poměrem cena výkon. Podobně vybavený regulátor renomovaných výrobců by cenu tohoto zařízení převyšoval několikanásobně. Pořizovací cena tohoto zařízení je v době zpracování práce a v plné konfiguraci 44,4 USD. Jedná se o kompaktní regulátory určené k panelové montáži. Na přední straně přístroje je realizován ovládací panel, sloužící jako rozhraní pro komunikaci s obsluhou ve formě tlačítek, dvouřádkového displeje a indikačních diod. Zadní strana zařízení obsahuje šroubovou svorkovnici. Rozsah napájecích napětí je velice široký. Zařízení jsou tak použitelná pro různé světové napájecí sítě.

Konfigurace zařízení je upřesněna číslem modelu objednávaného zařízení. Možnosti konfigurace jsou uvedeny v přiložené dokumentaci. Je možné vybrat velikost pouzdra regulátoru, druh řídicího algoritmu, druh a rozsah výstupu akční veličiny, počet a druh alarmových výstupů. Zařízení může být vybaveno pomocným napájecím zdrojem 24 VDC a komunikačním rozhraním.

Na ovládacím panelu jsou umístěny dva čtyřmístné displeje. První z displejů zobrazuje, v základním řídicím režimu, hodnotu regulované veličiny. Displej je označen zkratkou PV, neboli procesní hodnota, někdy bývá taky označováno zkratkou MV, jako měřená hodnota. Druhý displej zobrazuje velikost žádané veličiny. Označen je zkratkou SV, neboli nastavená hodnota. Přední strana obsahuje čtyři tlačítka a pět diod indikujících stav výstupu regulátoru, alarmů a komunikace. Přes toto ovládací rozhraní je možná konfigurace jednotlivých parametrů zařízení. Podrobný popis samotného ovládání je uveden v návodu k obsluze (Morning eletronics, 2016).

Obr. 2.1 – Ovládací panel ST828D

Na zadní straně regulátoru je umístěna šroubová svorkovnice. Obr. 2.2 zobrazuje fyzicky i schematicky zadní stranu regulátoru ST828D při plné konfiguraci. Svorky 1 a 2 slouží k připojení dvouvodičového termočlánku. Piny 1, 2 a 3 slouží k připojení odporového teplotního senzoru např. PT100. Piny 2 a 4 slouží dle konfigurace jako proudový nebo napěťový analogový vstup daného rozsahu. Pin 5 slouží jako přídavný zdroj 24 VDC, pro případné napájení senzorů. Piny 6 a 7 slouží k připojení komunikační sběrnice EIA/TIA 485. Piny 8 a 9 slouží jako jednoduchý reléový výstup regulátoru. Piny 8 a 10 slouží jako spínací kontakt prvního alarmového výstupu. Piny 11 a 12 slouží jako proudový či napěťový výstup regulátoru. Piny 13 a 14 slouží pro napájení zařízení ze sítě 230 VAC. Piny 15 a 16 slouží jako výstup ke spínání relé v pevné fázi. Piny 17 a 18 jsou nevyužity. Piny 19, 20 a 21 slouží jako přepínací kontakt druhého alarmového výstupu.



Obr. 2.2 – Svorkovnice ST828D

Dokumentace k zařízení není volně dostupná na internetu. Výrobce jí poskytl až na osobní vyžádání elektronickou poštou. Vyskytují se v ní drobné chyby a nejasnosti. Rozsahy vstupů a výstupů jsou zmatené. Není jasně uvedeno, jedná-li se na konkrétních svorkách o vstupy či výstupy atd. Součástí specifikace zařízení je obsažena konfigurace alarmových výstupů. Ty jsou ale zcela programově konfigurovatelné a nastavitelné v různých režimech dle potřeby (Morning eletronics, 2016).

2.4.1 Specifikace regulátoru řady ST828

Během zpracování této práce byla postupně vybrána a pořízena celkem dvě zařízení z této řady, která se od sebe liší především druhem použitých výstupů.

Modelové číslo prvního zařízení je ST828D-199-2-1100-21-1. Jedná se tedy o kompaktní průmyslový PID regulátor s pouzdrem pro panelovou montáž o rozměrech

72 × 72 mm (šířka × výška). Zařízení disponuje reléovým výstupem, výstupem pro relé v pevné fázi, 24 VDC zdrojem, nastavitelnými alarmovými výstupy, nastavitelným vstupem pro regulovanou (měřenou) veličinu a komunikačním rozhraním RS485. O samotném komunikačním rozhraní a implementovaném protokolu pojednává oddíl 3.5 této práce. Dále disponuje spojitým výstupem reprezentujícím regulovanou veličinu, pro externí zpracování měřeného signálu. Toto zařízení se více hodí pro praktickou aplikaci. V kombinaci s relé v pevné fázi lze regulovat skutečný tepelný děj, kde jako akční člen je použito elektrické odporové těleso.

Modelové číslo druhého zařízení je ST828D-199-4-1113-01-1. Regulátor se liší pouze použitým výstupem a konfigurací alarmů. Disponuje analogovým výstupem s rozsahem od 4 mA do 20 mA. Výstup pro SSR chybí. Toto zařízení se lépe hodí pro laboratorní měření. U tohoto zařízení byl během lazení programu poškozen analogový výstup a komunikační modul. Výsledné naměřené průběhy se spojitým výstupem regulátoru jsou proto pouze omezena na dostupná data.

Tab. 2.1 – Specifikace vstupu ST828D

Kód	1	2	3	4	5	6	7	8	9	10	11	12	13	14	99
Vstup	K	E	J	N	Wu3	S	T	R	B	AN4	AN3	AN2	AN1	Pt100	FULL
Rozsah	1 300 °C	600 °C	800 °C	1 300 °C	2 00 °C	1 600 °C	400 °C	1 700 °C	1 800 °C	2-10 V 1-5 V 4-20 mA	0-10 V 0-5 V 0-20 mA	0-50 mV	0-20 mV	800 °C	

Obě zařízení disponují univerzálním nastavitelným vstupem, který umožňuje připojení různých termočlánků, PT100, tak i unifikovaných proudových či napěťových signálů. Z tab. 3.1, převzaté z uživatelského manuálu plyne, že pro napěťové i proudové vstupy je nastavení stejná. Výrobce řeší tuto problematiku tak, že všechny vstupy AN1, AN2, AN3 a AN4 jsou napěťové a elektronicky se mění pouze jejich rozlišení. Při potřebě proudového vstupu, musí být obvod doplněn převodníkem proud/napětí. V nejjednodušším případě paralelně zapojený rezistor ke vstupu. Hodnota rezistoru je daná požadovaným rozsahem hodnot pomocí Ohmova zákona. Signál je tak převeden z proudového charakteru signálu na napěťový a signál může být tedy vyhodnocen napěťovým vstupem.

V zařízeních je implementován diskrétní řídící algoritmus, který není výrobcem podrobněji popsán. Jednotlivé parametry regulátoru jsou reprezentovány jako parametry PID regulátoru se sériovou strukturou. Proporcionální složka je nastavována pásmem proporcionality P_b, integrační složka integrační časovou konstantou sériové reprezentace T_{IS} a derivační složka derivační časovou konstantou sériové reprezentace T_{DS}. Regulátor disponuje

pevně danou vzorkovací periodou $T_0 = 0{,}5$ s, filtrací derivační složky, odstraněním unášení integrační složky, beznárazovým přepínáním mezi ručním a automatickým režimem řízení, beznárazovým přepínáním parametrů a funkcí automatického nastavení parametrů. Řídicímu algoritmu je předřazen deaktivovatelný digitální filtr s nastavitelnou časovou konstantou.

3 MODBUS

Pokud výrobce průmyslových kompaktních regulátorů uvádí použitý protokol pro komunikaci s nadřazenými systémy, jedná se ve většině případů o komunikační protokol Modbus provozovaný na sériové lince s použitím na fyzické vrstvě standartu EIA/TIA 485. Toto rozhraní slouží zpravidla k nastavení parametrů zařízení a k případnému odečítání hodnot regulačních veličin.

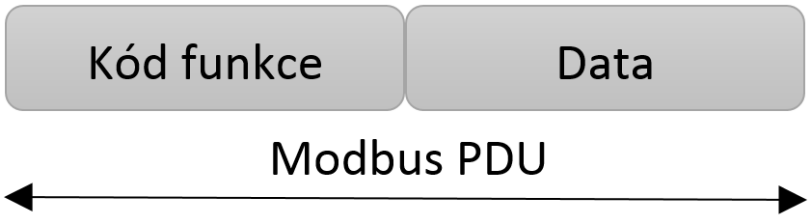
Protokol byl vytvořen již v roce 1979 firmou MODICON (dnešní Schneider Electric). I přes starší datum vydání je stále hojně využíván, především pro svoji jednoduchost, spolehlivost a robustnost. Protokol je určen ke komunikaci typu klient-server mezi zařízeními na různých typech sběrnic či sítí. Pro implementaci se používá především TCP/IP (ethernet) nebo asynchronní sériový přenos (EIA/TIA232, EIA/TIA485 atd.). Komunikace probíhá metodou požadavek-odpověď a druh požadavku odpovídá kódu funkce, který je jeho součástí. Tedy klient žádá o data a server následně data poskytuje formou odpovědi (Modbus.org, 2012).

Tab. 3.1 –Definice Modbus protokolu pomocí OSI/ISO modelu

ISO/OSI model		MODBUS	
Č. vrstvy	Název vrstvy	MODBUS RTU	MODBUS TCP
7	Aplikační	Aplikační vrstva Modbus	
6	Prezentační	-	-
5	Relační	-	Modbus na TCP
4	Transportní	-	TCP
3	Síťová	-	IP
2	Spojová	Master/Slave	Ethernet II/802.3
1	Fyzická	EIA/TIA-232, EIA/TIA-485	Ethernet

3.1 APLIKAČNÍ VRSTVA

Protokol Modbus je definován na sedmé aplikační vrstvě referenčního modelu ISO/OSI. Jeho definice obsahuje strukturu zprávy na úrovni protokolu (PDU) nezávisle na typu použitého komunikačního média. Tedy nezávisle na nižších vrstvách modelu.



Obr. 3.1 – Modbus PDU

Rámec PDU se skládá z kódu funkce a datové části. Kód funkce říká serveru, jakou operaci by měl provést. Velikost kódu je 1 bajt. Kódy jsou tedy v rozmezí od 1 do 255, přičemž kódy od 128 do 255 jsou vyhrazeny pro oznámení chybové odpovědi.

Zpráva poslaná klientem může obsahovat datovou část, například s adresou či počtem vstupů, které má server přečíst. Tato složka slouží k upřesnění operace a může zcela chybět.

Odpověď serveru, v případě úspěšného provedení požadované operace, obsahuje kód funkce provedené (požadované) funkce a v datové části předá server klientovi požadovaná data (pokud nějaká jsou).

Pokud při vykonávání požadované operace dojde k chybě, je vrácen kód požadované funkce s nastaveným nejvyšším bitem indikujícím neúspěch (kód funkce + 80_H). V datové části je vrácen chybový kód upřesňující důvod neúspěchu. Na straně klienta je implementován časový limit pro přijetí odpovědi, aby klient nečekal na odpověď, která nemusí přijít (Modbus.org, 2012). Protokol Modbus definuje 3 základní typy zpráv (viz tab. 3.2).

Tab. 3.2 – Modbus základní typy zpráv PDU

Typ zprávy	Kód funkce	Data
Požadavek	1 B Kód funkce	m B adresa, počet proměnných…
Odpověď	1 B Kód funkce	m B přečtené registry, stav zařízení…
Záporná odpověď	1 B kód funkce + 80_H	1 B chybový kód

3.2 KÓDY FUNKCÍ

Modbus protokol definuje tři základní skupiny kódů funkcí a to veřejné, uživatelsky definované a rezervované kódy. Veřejné kódy jsou jasně definované, zdokumentované, a proto se pro běžnou práci se sběrnicí využívají právě tyto kódy. Z množiny veřejných Modbus funkcí (tab. 3.3) nemusí být v konkrétním zařízení implementovány všechny tyto funkce. Výrobce zařízení musí uvést v přiloženém manuálu přesný výčet použitých funkcí (Modbus.org, 2012). Funkce slouží například ke čtení a zápisu jednotlivých bitů nebo celých registrů, k diagnostice atd.

Tab. 3.3 – Veřejné funkce Modbus

Funkce	**Dec**	**Hex**
Čti cívky	1	1
Čti diskrétní vstupy	2	2
Čti uchovávající registry	3	3
Čti vstupní registr	4	4
Zapiš jednu cívku	5	5
Zapiš jeden registr	6	6

Tab. 3.3 – Veřejné funkce Modbus – pokračování

Funkce	**Dec**	**Hex**
Čti stav	7	7
Diagnostika	8	8
Čti čítač kom. událostí	11	B
Čti záznam kom. událostí	12	C
Zapiš více cívek	15	F
Zapiš více registrů	16	10
Sděl identifikaci	17	11
Čti záznam ze souboru	20	14
Zapiš záznam do souboru	21	15
Zapiš registr s maskováním	22	16
Čti/zapiš více registrů	23	17
Čti FITO frontu	24	18
Čti identifikaci zařízení	43	2B

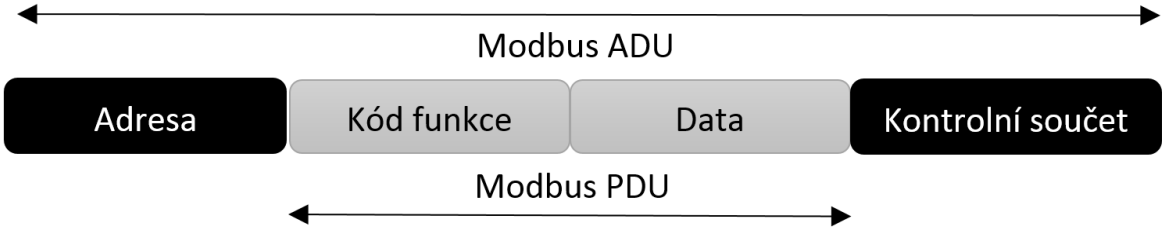
3.3 MODBUS NA SÉRIOVÉ LINCE

Modbus kromě aplikační vrstvy ISO/OSI modelu definuje i některé implementace protokolu na konkrétní typ komunikačního média. V případě implementace Modbus protokolu na sériové lince se jedná o master-slave komunikaci definovanou na druhé úrovni referenčního modelu ISO/OSI. Tedy na linkové vrstvě. Na fyzické vrstvě může protokol používat různá fyzická rozhraní (EIA/TIA 485, EIA/TIA 232). Aplikační vrstva v tomto případě slouží pouze k propojení logiky master-slave a klient-server. Role klienta je poskytnuta master uzlu a slave uzly fungují jako servery (Modbus.org).

3.3.1 Linková vrstva

Modbus protokol provozovaný na sériové lince řídí přístup k médiu na principu master-slave. Ke sběrnici je v jeden okamžik připojen pouze jeden master uzel, který vydává explicitní příkazy k jednomu ze slave uzlů. Slave uzly typicky nepřenáší data bez žádosti z master uzlu a nekomunikují s ostatními slave uzly. Ke sběrnici může být připojeno až 247 slave jednotek.

Master posílá požadavky slave jednotkám ve dvou režimech. Buďto v režimu unicast, kdy master adresuje požadavek jedné konkrétní slave jednotce, a ta pošle odpověď. Nebo v režimu broadcast, ve kterém master posílá požadavek všem jednotkám a žádná jednotka neodpoví.



Obr. 3.2 – Modbus ADU na sériové lince

Rámec PDU je rozšířen o další části v závislosti na typu sběrnice či sítě, na které je protokol použit. Tak vznikne zpráva na aplikační úrovni (ADU). Při implementaci na sériové lince se rozšiřuje konkrétně o adresu slave zařízení (8 b) a o kontrolní součet (16 b). Pole adresy obsahuje adresu slave jednotky, pro kterou je požadavek určen, v rozsahu od 0 do 247. Adresa 0 je určena pro broadcast vysílání. Master uzel nemá žádnou specifickou adresu a slave jednotky musí mít unikátní adresu v rámci sběrnice. Kontrolní součet slouží k detekci chyb a obsahuje CRC nebo LRC kód v závislosti na vysílacím režimu (Modbus.org, 2006).

3.3.2 Vysílací režimy

Modbus protokol definuje dva sériové vysílací režimy. Jedná se o pro implementaci povinný RTU režim a nepovinný ASCII. Všechny jednotky na jedné sběrnici musí pracovat ve stejném vysílacím režimu. Kontrolní součty jsou v obou případech povinné, ale použití parity je volitelné. Pokud není parita použita, musí být nahrazena dalším stop bitem.

V režimu Modbus RTU se přenáší 8 b informace v jednom 11 b rámci jako dva 4bitové hexadecimální znaky (start bit, 8 datových bitů, bit sudé parity a stop bit). Vysílání zprávy musí být souvislé, mezery mezi znaky nesmějí být delší než 1,5 znaku. Začátek a konec zprávy je dám odmlčením se na dobu delší než 3,5 znaku. Integrita celé zprávy je zaručena CRC součtem a pomocí sudého paritního bitu. Výhodou oproti ASCII režimu je vyšší propustnost dat.

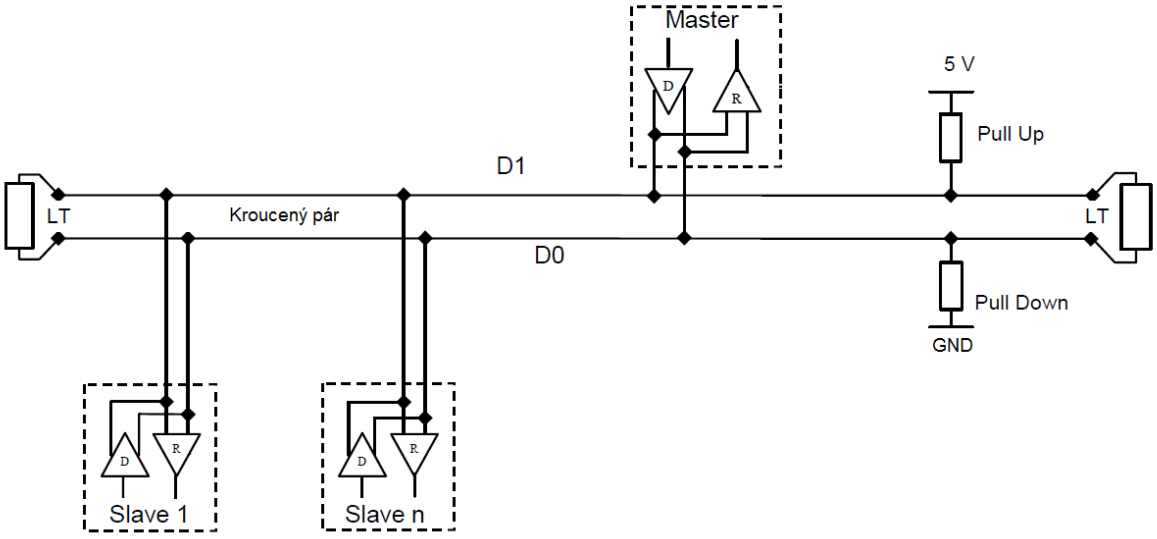
V režimu Modbus ASCII je každých 8 b informace posíláno jako dvojice ASCII znaků a ty jsou poslány ve dvou 10 bitových rámcích (start bit, ASCII znak 7 bitů, bit sudé parity a stop bit). Oproti režimu RTU je tedy nižší propustnost, ale umožňuje vysílat znaky s časovými mezerami až 1 s. Začátek zprávy je totiž indikován znakem „:“ a konec zprávy dvojicí řídicích znaků CR, LF. Integrita zprávy je zajištěna LRC součtem (Modbus.org, 2006).

3.3.3 Fyzická vrstva (EIA/TIA 485)

Pro implementaci protokolu Modbus na sériové lince se nejčastěji využívá standardu EIA/TIA 485 (známý jako RS 485). Jedná se o standard sériové komunikace z roku 1983 pro

průmyslové prostředí. V současné době jde o široce používané komunikační rozhraní pro sběr dat a řízení aplikací. Umožňuje vytvoření vícebodového poloduplexního sériového spoje.

Vychází částečně ze standardu RS 232. Liší se především jinou definicí napěťových úrovní a nepřítomností modemových signálů. Základní konfiguraci sběrnice tvoří tzv. dvoudrátová konfigurace. Signály jsou přenášeny pomocí symetrického krouceného vedení tvořeného dvěma vodiči. Přijímač RS 485 porovnává rozdíl napětí mezi oběma vodiči. Několik voltů rozdíl v absolutní hodnotě vysílače a přijímače nezpůsobuje žádné problémy. To brání existenci zemních smyček a odstraňuje problémy se společným zdrojem. Použitím diferenciálních signálů a krouceného symetrického vedení je zajištěna větší robustnost vůči rušení a šumu.



Obr. 3.3 – Topologie sběrnice EIA/TIA 485 (Modbus.org, 2006)

Ke sběrnici v jednom okamžiku může být připojeno pouze jedno master zařízení a jedno nebo více slave zařízení. Všechna zařízení jsou spojena paralelně. Bez použití opakovače je na sběrnici možné provozovat až 32 zařízení. S použitím opakovačů lze realizovat systém s až 247 slave zařízeními.

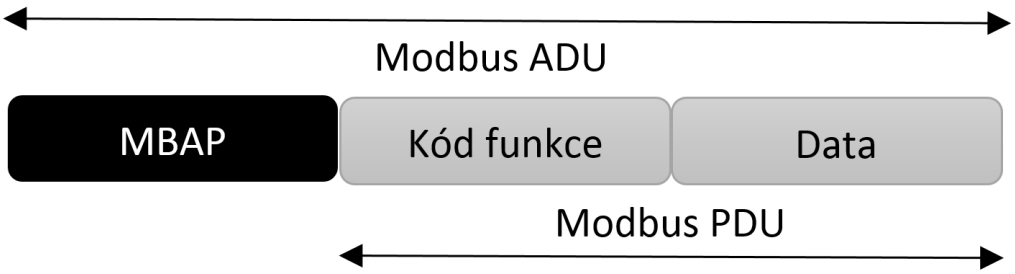
Maximální délka sběrnice závisí na přenosové rychlosti, na kapacitě a impedanci kabelu a počtu zátěží na sběrnici. Např. u spojů do 10 m může být přenosová rychlost až 10 $Mb \cdot s^{-1}$. Dále pro přenosovou rychlost 9 600 $b \cdot s^{-1}$ a průřez vedení 0,12 mm^2 je maximální délka sběrnice 1 000 m. Při implementaci do zařízení jsou přenosové rychlosti 9 600 $b \cdot s^{-1}$ a 19,2 $kb \cdot s^{-1}$ povinné. Dále mohou být realizovány rychlosti 1 200, 2 400, … 38 400 $b \cdot s^{-1}$, 56 $kb \cdot s^{-1}$, 115 $kb \cdot s^{-1}$…

Při komunikaci na vyšší vzdálenosti musí být vedení na obou stranách zakončeno zakončovacími odpory. Tyto odpory slouží k impedančnímu přizpůsobení konců vedení a tím zabraňují odrazům signálu na koncích vedení. Na obr. 3.3 jsou zobrazeny a označeny jako LT. Je důležité, aby sběrnice byla ukončena na obou stranách, protože šíření signálů a tím i odrazů, je obousměrné. Zakončovací odpory jsou připojeny mezi dvěma vodiči vyváženého páru. Hodnota odporů je kolem 150 Ω (0,5 W). V případě použití polarizačních rezistorů je vhodnější volbou impedanční přizpůsobení pomocí kombinace sériového kondenzátoru (1 nF, minimálně 10 V) a 120 Ω rezistoru.

V případě že nedochází k žádné aktivitě dat na sběrnici, stane se vedení náchylné vůči šumu a rušení. K zajištění konstantní úrovně v tomto případě lze použít tzv. pull-up a pull-down rezistory. Tím je zajištěna polarizace sběrnice. Každé zařízení připojené ke sběrnici by mělo mít v dokumentaci uvedeno, zda pro svojí správnou funkci vyžaduje polarizaci nebo jestli zařízení implementuje polarizaci samo. Hodnota těchto rezistorů se pohybuje v rozmezí od 450 Ω do 650 Ω. Vyšší hodnota rezistoru umožňuje vyšší počet zařízení na sběrnici. Polarizace vedení smí být provedena pouze na jednom místě pro celou sběrnici. To se většinou realizuje na master uzlu. Ostatní zařízení nesmí provést žádnou polarizaci. Jako mechanické rozhraní nejčastěji využívá šroubových svorek, konektor RJ45, nebo konektor typu D-sub 9 pin (Modbus.org, 2006).

3.4 MODBUS NA TCP/IP

Slouží k provozování komunikačního protokolu Modbus na fyzické ethernetové vrstvě s využitím TCP/IP protokolu.



Obr. 3.4 – Modbus ADU na ethernetu

Pro identifikaci Modbus ADU je použita MBAP hlavička (Modbus Application Protocol Header). Pro posílání Modbus/TCP ADU je na TCP vyhrazen registrovaný port 502 (Modbus.org, 2012).

3.5 KOMUNIKAČNÍ PROTOKOL ZAŘÍZENÍ ŘADY ST828

V zařízení je implementován protokol Modbus RTU a je vybaven dvouvodičovým rozhraním pro asynchronní sériový přenos EIA/TIA 485. Rychlost sběrnice je nastavitelná na hodnoty 2 400, 4 800, 9 600, 19 200 $b \cdot s^{-1}$. Tovární nastavení je 9 600 $b \cdot s^{-1}$. Zařízení má implementovány Modbus funkce č. 03_H, 06_H a 10_H. Funkce 03_H (čti uchovávající registry) slouží ke čtení jednoho nebo více 16 b registrů. V požadavku master uzlu je udána adresa prvního registru a jejich počet. Funkce 06_H (zapiš jeden registr) slouží k zápisu jednoho 16 b registru. V požadavku je specifikována adresa registru a hodnota, která se má do něj zapsat. Funkce 10_H (zapiš více registrů) slouží k zápisu jednoho nebo více 16 b registrů. V požadavku je uvedena adresa prvního registru, který se má zapsat, jejich počet a hodnoty, které se mají do nich zapsat. Datový formát je složen z jednoho start bitu, osmi data bitů, paritní bit je nahrazen jedním stop bitem a jeden klasický stop bit.

Přiložený manuál obsahuje základní popis vlastností komunikačního rozhraní uvedený výše a dále obsahuje tabulku adres jednotlivých parametrů regulátoru s vysvětlivkami a možnými rozsahy jednotlivých hodnot parametrů. Tato tabulka je velice důležitá pro tvorbu programu pro nastavování jednotlivých parametrů a načítání jejich hodnot. Zařízení má kvůli délce PDU omezen maximální počet čtených a zapisovaných registrů pro jeden příkaz, a to na hodnotu 36 registrů pro zápis a 37 pro čtení. S tím je nutné počítat při implementaci v programu. Zařízení má registrů určených pro komunikaci celkem 54, a proto nelze přečíst/zapsat všechny registry zařízení najednou (Morning eletronics, 2016).

4 APLIKACE

Výběr programovacího jazyka je závislý především na platformě, na které aplikace poběží a na šířce nabídky programovacích nástrojů. Cílem je aplikace s grafickým rozhraním pro komunikaci, nastavování parametrů a vizualizaci naměřených dat z průmyslového regulátoru. Z těchto rozličných požadavků vyplývá požadavek na více vláknové zpracování a vykreslování grafů měřených hodnot v reálném čase. Tyto požadavky splňuje například Java nebo C#. Hlavní výhoda Javy je její nezávislost na platformě a tím na operačním systému. C# je závislý na platformě Windows. Aplikace má sloužit ke komunikaci mezi PC a regulátorem, a proto nám tato skutečnost nevadí. Volba tedy padla na C# s využitím WPF. Vydavatelem je firma Microsoft a pro zpracování je použito vývojové prostředí Visual Studio. C# je moderní skriptovací jazyk. Samozřejmostí je veliká podpora uživatelské základny formou příkladů a návodů.

4.1 KOMUNIKACE

Regulátor je připojen k PC prostřednictvím sběrnice EIA/TIA 485. Jedná se o sériovou linku realizovanou dvouvodičovým párovým vedením. Počítač je dovybaven komunikačním převodníkem 485/USB, který slouží k převodu komunikační logiky. Implementuje v sobě integrované obvody CH340G a MAX485. Obvod CH340G slouží jako USB/UART rozhraní. Obvod MAX485 slouží k následnému přizpůsobení napěťových úrovní standartu sběrnice. Jedná se vlastně o EIA/TIA 485 vysílač/přijímač.

Obr. 4.1 – Komunikační převodník USB/RS 485

Pro správnou funkci obvodu je nutné do PC nainstalovat odpovídající ovladače (Arduined.eu, 2017). Pro jednotlivé verze operačního systému (Windows 7, Windows 10) je třeba jiná verze ovladače. Převodník se pak chová jako virtuální port sériové linky COM a lze k němu ze systému tak i přistupovat. Aplikační rámec .NET v sobě implementuje funkce a metody pro práci se sériovým portem. PC vystupuje jako master uzel a regulátor vystupuje

jako slave uzel. Při použití klient-server logiky je PC klientem, který žádá o data. Následně čeká na odpověď od serveru, tedy od připojeného zařízení.

4.2 POUŽITÉ PROGRAMOVÉ NÁSTROJE

Pro vývoj aplikace byly použity následující knihovny a nástroje. Některé jsou implementovány přímo v .NET, některé jsou externí.

4.2.1 EasyModbus

Cílová aplikace pro implementaci aplikační vrstvy komunikačního protokolu Modbus využívá externí knihovnu EasyModbus. Knihovna je vyvíjena Německou společností Stefan Rossmann Engineering Solutions s otevřenou veřejnou licencí. Je podporována v .NET a JAVA. Obsahuje metody a komponenty pro obsluhu komunikace přes protokol Modbus RTU i Modbus TCP. Dále obsahuje nástroje pro usnadnění vývoje softwaru např. Modbus Server Simulator. Knihovna zajišťuje rychlý a bezpečný přístup z PC, embedded systémů, případně z PLC nebo dalších komponentů pro průmyslovou automatizaci. Je zapotřebí pouze několik řádků kódu ke čtení nebo zápisu dat z nebo do systému. Knihovna implementuje Modbus funkce uvedené v tab. 4.1.

Tab. 4.1 – EasyModbus implementované funkce

Funkce	**Dec**	**Hex**
Čti cívky	1	1
Čti diskrétní vstupy	2	2
Čti uchovávající registry	3	3
Čti vstupní registr	4	4
Zapiš jednu cívku	5	5
Zapiš jeden registr	6	6
Zapiš více cívek	15	F
Zapiš více registrů	16	10
Čti/zapiš více registrů	23	17

Web vydavatele obsahuje příklady jednoduchých implementací kódu, dokumentaci k metodám a vlastnostem obsažených v knihovně. Metody jsou velice jednoduché a jsou dobře popsány. Z uživatelského hlediska je použití velice jednoduché a intuitivní (Rossmann Engineering, 2016).

4.2.2 OxyPlot

Pro vizualizaci naměřených dat je použita knihovna OxyPlot. Jedná se o knihovnu s otevřenou licencí MIT, která je velice tolerantní a umožňuje použití i pro komerční účely. Na webu vydavatele je obsažena dokumentace s nepřeberným množstvím příkladů přímo od vývojářů i uživatelů. Nástroj je jednoduchý na implementaci. Je šířen jako tzv. nuget balíček, který je stažitelný přes rozhraní přímo ve Visual studiu. (Oxyplot.org, 2016).

4.2.3 Serializace a deserializace

Pro ukládání konfiguračních a naměřených dat je použit formát XML. Jedná se o velmi univerzální značkovací jazyk, který je podporován řadou programovacích jazyků a aplikací. XML dokument (soubor) obsahuje tagy (značky), elementy a atributy. Možnosti implementace zpracování XML dokumentu v .NET jsou široké. Lze k souboru přistupovat jednoduchým kontinuálním čtením/zápisem pomocí SAX parseru, nebo pomocí objektové struktury DOM. Dalším přístupem je serializace a deserializace. Serializace slouží k uchování stavu objektu. Vytvořený objekt konvertuje na proud bytů a ten uloží do paměti. Poté pomocí deserializace je možné převést proud bytů na kopii objektu. Je tedy možné uložit stav objektu a potom pomocí deserializace objekt kdykoliv znovu vytvořit. Používá se například pro ukládání nastavení aplikace nebo k výměně dat mezi rozdílnými systémy. Všechny zmíněné postupy jsou implementované v .NET. V této aplikaci je využit přístup pomocí serializace a deserializace (Microsoft, 2017; Kosek, 2000).

4.2.4 Windows Presentation Foundation (WPF)

WPF je knihovna tříd, která umožňuje snadnou tvorbu grafického rozhraní aplikace. Je součástí aplikačního rámce .NET firmy Microsoft od verze 3.0. Jedná se o nástupce knihovny Windows Form. Při tvorbě aplikace je oddělena implementace vzhledu (XAML kód) a funkcionality (tzv. kód na pozadí). XAML je rozšiřitelný aplikační značkovací jazyk, je založený na XML. Používá se k implementaci vzhledu aplikace. To se obvykle používá k vytvoření okna, dialogového okna, stránky, uživatelských ovládacích prvků (kontrolek) nebo grafiky. Kód na pozadí slouží k implementaci funkcionality, kdy reaguje na události (například stisknutí tlačítka) a slouží k volání hlavní logiky programu. Kód je spojený s řídicími prvky obsaženými právě v XAML a jejich události obsluhuje.

WPF disponuje nepřeberným množstvím implementovaných kontrolek, které slouží k zobrazování dat a ovládání aplikace uživatelem. Jsou to například tlačítka, zobrazovače dat, kalendáře, dialogová okna, dokumenty, různá menu, zaškrtávací políčka atd.

Aplikace je většinou vytvořena s cílem poskytnout uživateli prostředky pro prohlížení a úpravu dat. Poté co jsou požadovaná data načtena do objektů aplikace je možné zkopírovat data z řízených objektů do kontrolek, kde lze data zobrazit a upravovat. Následně je třeba zajistit, aby změny provedené na datech pomocí ovládacích prvků byly zkopírovány zpět do objektů. Pro automatické provádění těchto kroků WPF obsahuje nástroje pro data binding.

Dojde-li ke změně vlastnosti, tak se o tom aplikační okno nedozví a zobrazuje v kontrolce starou hodnotu. Obnovení kontrolek v grafickém rozhraní aplikace je řešeno implementací rozhraní INotifyCollectionChanged. Pokud ho implementujeme v nějaké třídě, je možné reagovat na změnu vlastností této třídy. Rozhraní obsahuje pouze událost PropertyChanged. Ta je vyvolána v případě změny hodnoty vlastnosti a ta se projeví i v okně aplikace.

Většina dat v této konkrétní aplikaci je uložena do instance třídy ObservableCollection<T>. Třída je určena ke shromažďování dat a má implementované právě INotifyCollectionChanged rozhraní. Představuje seznam, který umí vyvolat událost změny při přidání položky, odebrání, nebo při její aktualizaci v celém seznamu. Díky tomuto mechanismu se automaticky obnoví všechny kontrolky na formuláři, které mají nastavený jako zdroj dat tuto kolekci. Pak není nutné ručně aktualizovat desítky kontrolek ve formuláři v případě změny, to by bylo velice nepřehledné (Microsoft, 2017).

4.2.5 Paralelní programování

Po spuštění výsledné aplikace dojde k vytvoření hlavního okna a tím ke spuštění hlavního vlákna. Při požadavku spuštění vizualizačního okna, dojde k jeho spuštění v novém vlákně. Každé z těchto oken tedy běží ve vlastním vlákně. Kdyby tomu tak nebylo, tak okno v pozadí se stane neaktivním. Nebylo by tedy možné při spuštěné vizualizaci měnit parametry zařízení. Paralelní programování s sebou přináší problém synchronizace. Je nutné v kritických částech programu zabránit chybám v přístupu ke společným proměnným z různých vláken. Kritickým místem této konkrétní aplikace je zejména přístup k portu sériové linky. Požadavek přístupu k sériové lince může vzniknout z obou vláken v jednom okamžiku. A to v případě, když probíhá periodické měření ve vizualizačním okně a zároveň dojde k požadavku na čtení či zápis do konfigurace zařízení v hlavním okně. Pomocí konstrukce příkazu lock lze zajistit

exkluzivní přístup k dané části kódu. Část programu je zamčena pro ostatní vlákna do doby, než dojde k jeho odemčení. Všechna vlákna, která se pokusí ke kódu přistoupit, jsou zablokována a řadí se do fronty v pořadí, ve kterém přišla. Když bude chtít systém vlákno uspat, musí počkat, až se dostane z kritické sekce (z té pod zámkem). Program bezpečný z hlediska vláken označujeme jako thread-safe. Lze toho dosáhnout hlavně pomocí zamykání a omezení vzájemné interakce mezi vlákny na minimum. Tato konstrukce je použita v metodě uvedené níže, která slouží pro čtení hodnot registrů ze zařízení. Stejným způsobem je ošetřena i metoda sloužící k zápisu do registrů zařízení. Tím je zajištěna vláknová bezpečnost aplikace.

V ukázce kódu metody si lze dále všimnout, že maximální počet registrů čtených najednou je omezen. Pak například při požadavku čtení všech registrů, je nutné rozdělit proces čtení do několika cyklů. To je zajištěno for cyklem s podmínkou dělitelnosti celkového počtu registrů maximálním povoleným počtem čtených registrů najednou (MaxRead). Následující podmínka slouží pro přečtení zbývajících registrů po ukončení cyklu. Parametr MaxRead je načítán z konfiguračního souboru zařízení. Obdobným způsobem je řešena i metoda pro zápis hodnot do registrů zařízení.

```
/// <summary>
/// Metoda pro čtení hodnot registrů ze zařízení do pole
/// </summary>
/// <param name="startingAdress">Počáteční adresa čtení. </param>
/// <param name="quantityOfReadRegisters">Počet čtených registrů. </param>
/// <returns>Pole s přečtenými hodnotami ze zařízení</returns>
public int[] ReadRegistersValue( int startingAdress, int quantityOfReadRegisters) {
int auxAdress = startingAdress;//Pomocná adresa
 int residue = quantityOfReadRegisters; //Počet zbývajících registrů pro čtení
 int[] reg = new int[MaxRead]; //Pole pro čtecí cyklus
 int[] regValue = new int[quantityOfReadRegisters];  //Celkové pole vracené metodou
 lock (MainWindow.locker) {//Zamknutí části programu pro ostatní vlákna po dobu
přístupu k sériové lince
//Počet čtených reg. v jednom cyklu je omezen. Proces musí být rozdělen na části.
       for (int i = 0; i < (int)Math.Ceiling((float)quantityOfReadRegisters /
MaxRead) - 1; i++){//Podmínka dělitelnosti celkového počtu registrů maximálním
povoleným počtem čtených registrů najednou
        reg = modbusClient.ReadHoldingRegisters(auxAdress, reg.Length);//čtení
         for (int j = 0; j < reg.Length; j++)//kopírování hodnot do celkového pole
             regValue[auxAdress - startingAdress + j] = reg[j];

         auxAdress += reg.Length;//Přičtení přečtených registrů do pom. adresy
         residue = residue - reg.Length; //Počet zbývajících registrů pro čtení
        }
       if (residue >0) {//Čtení zbývajících registrů
         reg = modbusClient.ReadHoldingRegisters(auxAdress, residue);//čtení hodnot
ze zařízení
         for (int i = 0; i < residue; i++)//kopírování hodnot do celkového pole
             regValue[auxAdress - startingAdress+ i] = reg[i];
        }
        return regValue; //Vrácení naplněného pole hodnotami registrů
        }
}
```

Jednotlivá okna aplikace jsou spuštěna v samostatných vláknech pomocí třídy Thread. Otevírá vlákna na úrovni operačního systému a umožňuje uživateli vlákno přerušit, pozastavit, sledovat stav a nastavit vlastnosti vlákna. Samostatná vlákna potřebují nezanedbatelné množství paměti a zatěžují procesor kontextovým přepínáním mezi jednotlivými vlákny. Jedná se o nejjednodušší možnost paralelního programování, ovšem snaha je vyhnout se tomuto způsobu. V některých případech neexistuje vhodná alternativa. Například v případě že uživatel chce spustit více složitých výpočtů na více procesorech. Hodí se tedy i pro dlouho trvající operace.

```
/// <summary>
/// Metoda onTick bude periodicky volána dokud nebude zrušeno pomocí tokenu.
/// </summary>
/// <param name="onTick"></param>
/// <param name="dueTime"></param>
/// <param name="interval"></param>
/// <param name="token"></param>
/// <returns></returns>//
private static async Task RunPeriodicAsync(Action onTick,TimeSpan dueTime,TimeSpan
interval, CancellationToken token)
     {
     // Počáteční čekací doba (doba než začne periodická smyčka)
     if (dueTime > TimeSpan.Zero)
        await Task.Delay(dueTime, token);
     // Opakuje smyčku do doby, dokud není token zrušen.
     while (!token.IsCancellationRequested)
         {
          // Volá funkci onTick, která implementuje kód samotného odměřování.
          onTick?.Invoke();
          // Čeká na další periodické opakování.
          if (interval > TimeSpan.Zero)
          await Task.Delay(interval, token);
            }
        }
```

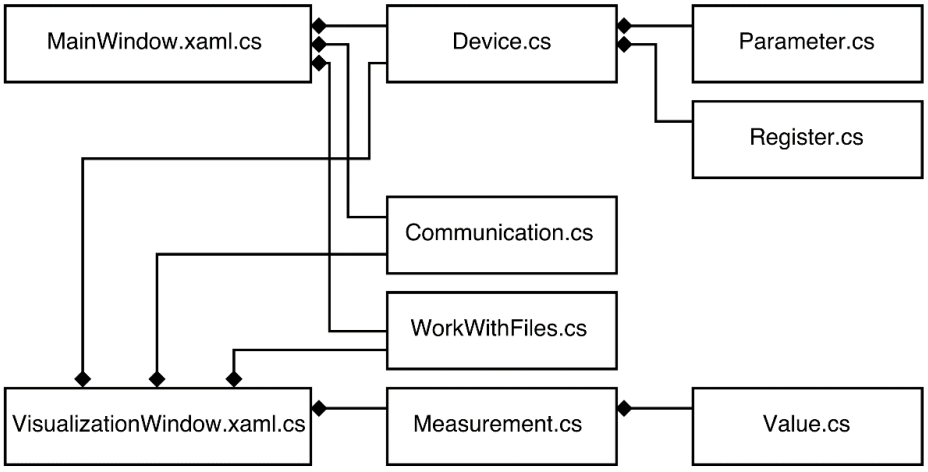
Periodické měření je implementováno metodou RunPeriodicAsync, která je instancí třídy Task. Metoda periodicky volá metodu onTick v časovém intervalu daném parametrem interval. Parametr dueTime udává dobu čekání před spuštěním periodického volání. V metodě onTick je implementováno samotné měření aktuální hodnoty. Parametr token slouží ke zrušení periodického volání metody v místě programu, kde má být měření ukončeno. Task kvůli každé úloze neotevírá nové vlákno, ale pracuje s tzv. fondem vláken (thread-pool). Z uživatelského hlediska je tato třída lepší, má lepší programové rozhraní a zamezuje plýtvání vlákny na úrovni operačního systému. Nenabízí ovšem plnou kontrolu nad spouštěnými vlákny. Třída se nehodí pro dlouho trvající operace, protože může dojít k zaplnění fondu vláken (Microsoft, 2017).

4.2.6 Komentáře a dokumentace

V ukázkách kódu v předchozím pododdíle si lze všimnout speciálních komentářů oddělených třemi lomítky (///). Tyto komentáře se zapisují pomocí XML tagů. Z komentářů je při kompilaci vygenerován XML soubor. Visual Studio poté z tohoto souboru načítá data pro Intellisense (kontextová nápověda při programování). Pomocí tohoto speciálního typu komentářů lze psát přímo do zdrojového kódu dokumentaci k třídám, jejich metodám a vlastnostem či dalším prvkům. Podpora pro psaní těchto komentářů je zabudována do samotného Visual Studia, které generuje kostru dokumentace automaticky. Dokumentace je nedílná součást programování. Slouží k usnadnění orientace a pochopení kódu pro další programátory, kteří chtějí využít dílčí části kódu (Microsoft, 2017).

4.3 STRUKTURA APLIKACE

Samotná aplikace je tvořena dvěma okny. Hlavním konfiguračním oknem a vizualizačním oknem, které je spouštěno tlačítkem z hlavního okna. V konstruktoru hlavního okna jsou vytvořeny instance tříd pro práci s externími soubory, komunikaci s připojeným zařízením a struktura tvořící samotné zařízení. Ke spuštění vizualizačního okna dojde v novém vlákně. A to z důvodu, aby byla obě okna aktivní a byla možná změna konfigurace zařízení při spuštěné vizualizaci. Vizualizačnímu oknu jsou předány reference instancí tříd vytvořených v hlavním okně a dále je vytvořena instance měření, obsahující strukturu sloužící pro vizualizaci a ukládání naměřených dat.



Obr. 4.2 – Struktura tříd aplikace

Hlavní třídou modelu aplikace je Device.cs. Její instance implementuje seznam instancí tříd Register.cs a Parameter.cs, které obsahují informace o registrech zařízení a parametry potřebné pro komunikaci. Třídy Communication.cs a WorkWithFiles.cs implementují metody ke komunikaci se zařízením a k práci se soubory mimo aplikaci. Instance třídy Measurement.cs implementuje seznam instancí třídy Value.cs, které obsahují naměřené hodnoty v jednotlivých časových okamžicích.

4.4 PRÁCE S DATY

Uložit data do zdrojového kódu samotné aplikace je možné, ale velice nepraktické. Zejména pro následné zpracování dat v jiném programu. Nebo z hlediska požadavku rozšíření o nová data, kde je vyžadován velký programátorský zásah. Proto je řešením externí zdroj dat. Z kterého aplikace načte a uloží data do paměti a nadále s nimi pracuje. Lze použít online nebo offline databázi. U online databáze je snadnější distribuce jejího rozšíření, ale za cenu připojení k internetové síti a přístupu k dané databázi. Naopak u offline řešení není potřeba připojení k síti, ale rozšíření databáze je nutné distribuovat aktualizacemi aplikace. Při použití online přístupu by bylo vhodné použít například SQL databázi. Vzhledem k tomu, že aplikace je určena pro přímou komunikaci se zařízením přes fyzické rozhraní EIA/TIA 485 je online databáze zbytečná. Proto je použita offline databáze, která se při vývoji aplikace dá snadno měnit.

4.4.1 Načítání a ukládání konfiguračních dat

Pro uložení konfigurace parametrů samotného zařízení a parametrů pro komunikaci bylo zvoleno externí uložení dat. Konfigurační soubor slouží aplikaci k získání parametrů připojeného zařízení. Pokud uživatel chce pracovat s jiným zařízením, než je regulátor použitý v této práci, lze pro něj vytvořit vlastní konfigurační soubor. Pokud bude zachována základní struktura souboru, tak není potřeba zásah do zdrojového kódu. Soubor může zároveň sloužit k uložení posledního nastavení regulátoru.

Data jsou uložena ve formátu XML. V aplikaci je s nimi pracováno pomocí nástrojů serializer a deserializer implementovaných v .NET. Soubor je strukturovaný a lze ho zobrazit jako stromovou strukturu, jako na obr. 5.2, která je po načtení dokumentu vytvořena jako požadovaný objekt, s kterým dále aplikace pracuje. Konfigurační soubor obsahuje objekt celého zařízení. Všechna data jsou tedy vnořeny v elementu Device. To je nezbytným požadavkem pro

správně strukturovaný XML dokument. Vnořeny jsou elementy Registers, ComParameters, Connected a GlobalDecimalPoint.

Obr. 4.3 – Stromová struktura souboru konfiguračních dat

Element Registers obsahuje seznam jednotlivých registrů zařízení spolu s upřesňujícími informacemi. Jedná se vlastně o přepis adresní tabulky registrů a dalších parametrů nezbytných pro správnou funkci programu z uživatelského manuálu zařízení. Každý z elementů Register obsahuje jeho jméno, adresu, aktuální hodnotu, možnou minimální a maximální hodnotu, nastavovanou hodnotu. Dále obsahuje informace o vlastnostech jako je pevná řadová čárka, případně o kolik desetinných míst je řádovou čárku možné posouvat a v poslední řadě, slouží-li registr pouze ke čtení, nebo i k zápisu. Adresní tabulka je u některých zařízení uvedena výrobcem v uživatelském manuálu. To ale není pravidlem.

Element ComParameters obsahuje nezbytné informace pro nastavení parametrů komunikace. Elementy Connected a GlobalDecimalPoint slouží jako pomocné proměnné.

4.4.2 Ukládání naměřených dat

Ukládání naměřených dat probíhá ve vizualizačním okně. Kvůli přehlednosti jsou data opět uložena ve formátu XML. Například pomocí programu Microsoft Excel je možné soubor otevřít a přehledně zobrazit jako tabulky pro další zpracování naměřených hodnot.

Tyto data jsou v aplikaci uloženy v objektu typu ArrayOfValue. Tento celý objekt je pomocí serializeru uložen do souboru. Ten obsahuje seřazené elementy Value vztažené vždy k časovému okamžiku určeného hodnotou elementu TimeDouble. Každý element Value obsahuje naměřenou hodnotu regulované veličiny, žádané veličiny, panelového výstupu

regulátoru a z něj odvozené binární výstupy PWM, alarmy 1,2 a indikaci funkce automatického hledání parametrů regulátoru (Kosek, 2000).

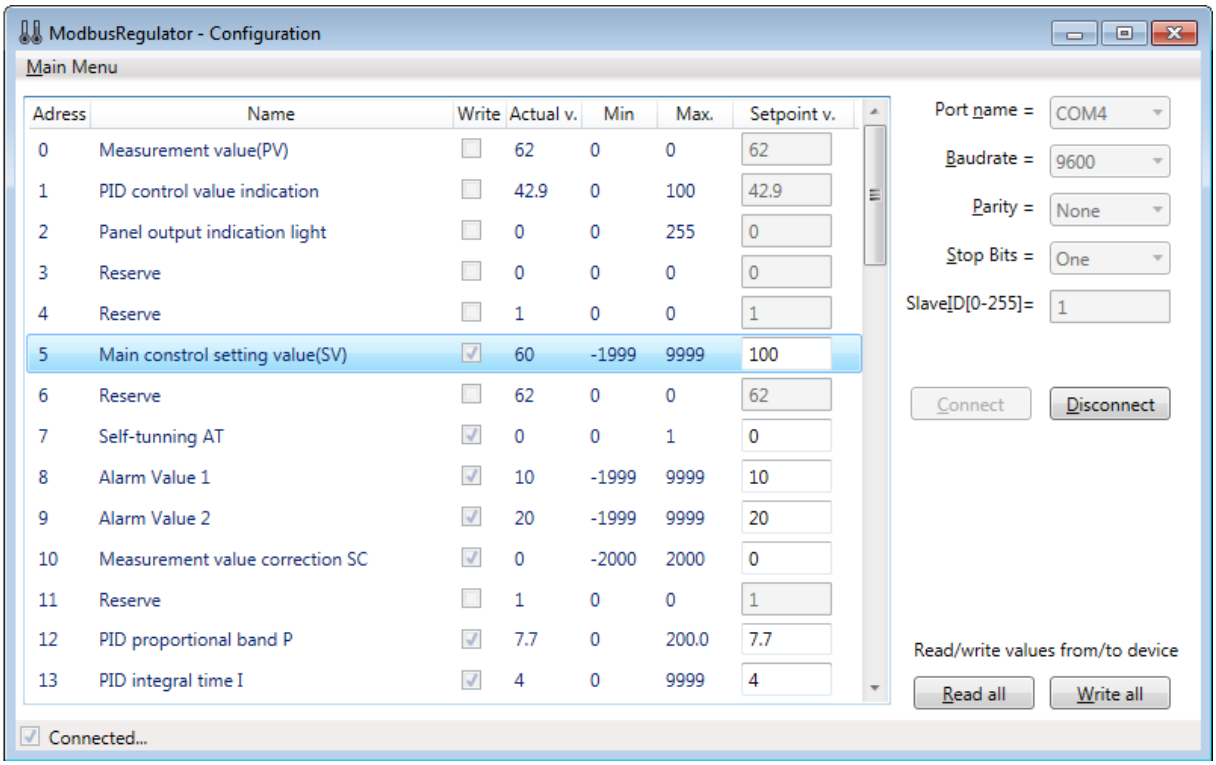
Obr. 4.4 – Stromová struktura souboru naměřených dat

4.5 OBSLUHA A GRAFICKÉ ZPRACOVÁNÍ APLIKACE

Aplikaci lze spustit pomocí souboru ModbusRegulator.exe v příloze A ve složce Aplikace\ModbusRegulatorSpustitelny. Nebo ji lze nainstalovat pomocí souboru setup.exe v příloze A ve složce Aplikace\ModbusRegulatorInstal. Pro správnou funkci aplikace je nutný .NET aplikační rámec ve verzi alespoň 4.5, který je implementován od Windows 8 a výše. Pokud není framework v PC přítomen, je při instalaci uživatel dotázán, zda ho chce doinstalovat. Při spuštění aplikace je vždy automaticky načítán konfigurační soubor ze složky C:\Users\UserName\AppData\Roaming\ModbusRegulator\defaultConfiguration.xml. Pokud se jedná o první spuštění programu a složka ani soubor neexistují, je uživatel vyzván k zadání cesty k souboru pomocí klasického dialogového okna pro otevírání souborů. Jedná se o přiložený soubor ConfigurationFileST828D.xml. Pokud tak uživatel udělá a jedná se o správný soubor, tak je soubor zkopírován na adresu uvedenou výše. Následně je z něj načtena konfigurační struktura. Při dalším spuštění aplikace je tento krok přeskočen a základní konfigurace je načtena ze zkopírovaného souboru. Aplikace byla odzkoušena na verzích operačního systému Windows 7 a Windows 10.

4.5.1 Hlavní konfigurační okno

Hlavní okno je otevřeno po spuštění aplikace. Slouží k nastavení a následnému otevření komunikačního kanálu. Poté je možné přes jeho rozhraní načíst a měnit konfiguraci připojeného zařízení.



Obr. 4.5 – Konfigurační okno

Při spuštění aplikace jsou aktivní pouze kontrolky panelu pro nastavení režimu komunikace sériového portu, který je umístěn v pravé horní části okna a tlačítko Connect pro otevření komunikačního kanálu. Rozbalovací seznam Port name zobrazuje všechny dostupné sériové porty na počítači. V případě, že žádný sériový port v době spuštění aplikace není dostupný, zůstane položka prázdná. Následující rozbalovací seznamy slouží pro nastavení dalších parametrů komunikace, jako je rychlost, parita, počet stop bitů. Kontrolka Slave ID umožňuje zadání adresy slave zařízení, které je připojeno ke sběrnici a uživatel s ním chce komunikovat. Po nastavení správných parametrů uživatel stiskem tlačítka Connect otevře komunikační kanál. Součástí otevření kanálu je zároveň kontrola odpovědi slave zařízení. Pokud je regulátor aktivní, připojen ke sběrnici a řádně odpoví na dotaz master uzlu v časovém intervalu, je vše v pořádku. V opačném případě dojde opět k uzavření kanálu a je nutné najít příčinu selhání. Ve stavovém řádku je znázorněno je-li komunikační kanál aktivní, či nikoliv.

Dominantou hlavního okna je výpis struktury zařízení ve formě adresní tabulky, který je umístěn v levé části okna a jeho obsah je načten z externího souboru. Po otevření komunikačního kanálu se výpis struktury spolu s tlačítkem Read all stane aktivním. V jednotlivých sloupcích výpisu je zobrazena adresa registrů, jejich názvy, slouží-li registry pouze ke čtení nebo i k zápisu, aktuální hodnoty registrů načtené ze zařízení, maximální a minimální možné hodnoty dle manuálu, nastavované hodnoty. Po zapnutí aplikace jsou jako aktuální a nastavované hodnoty zobrazena čísla uložena v konfiguračním souboru. Pokud je komunikační kanál otevřen, tak je možné pomocí tlačítka Read all načíst aktuální hodnoty ze zařízení do sloupce s aktuální a nastavovanou hodnotou. Poté je teprve aktivováno tlačítko Write all, které slouží k zápisu nastavované hodnoty z posledního sloupce do zařízení.

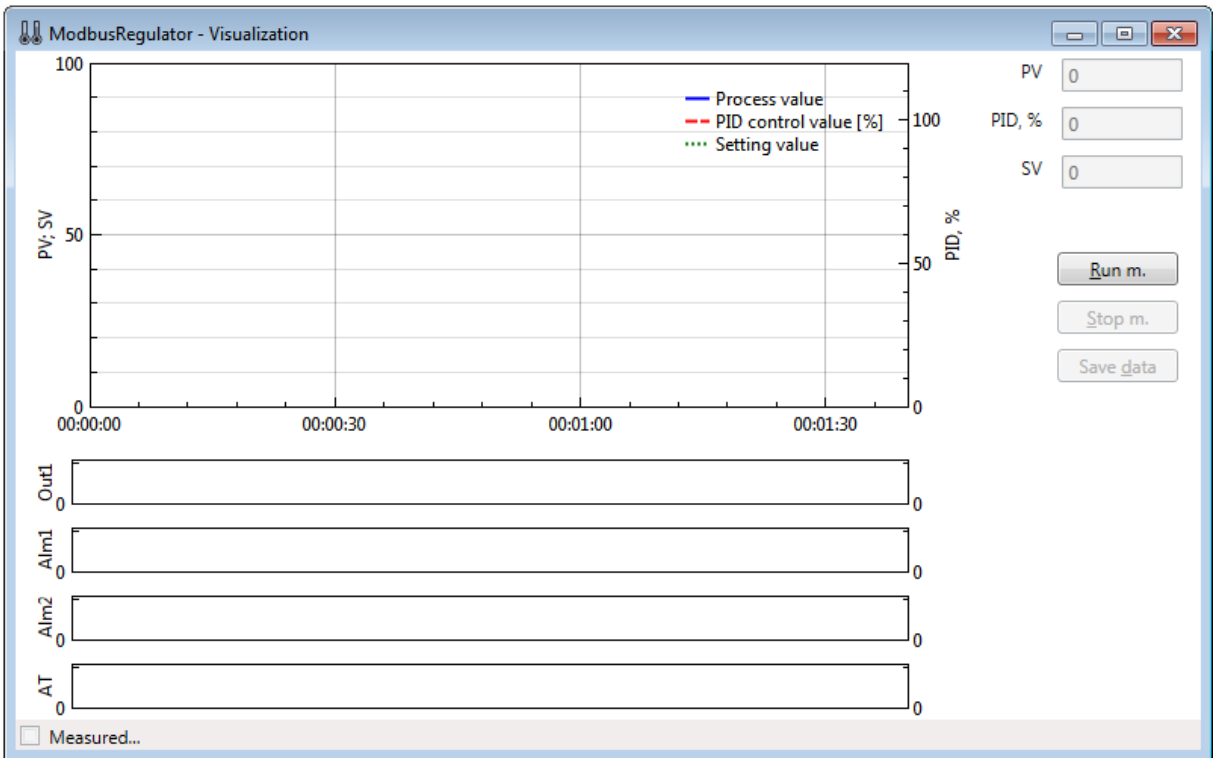
Samotné zadávání a změna hodnot probíhá v posledním sloupci, tedy Setpoint value. V tomto sloupci uživatel změní hodnoty požadovaných registrů a následně pomocí tlačítka Write all může zapsat obsah všech buněk v tomto sloupci do zařízení. Před využitím tohoto tlačítka a samotnou změnou hodnot ve sloupci je doporučeno načíst aktuální hodnoty tlačítkem Read all. Až poté provést změny hodnot a jejich zápis do zařízení. Pro změnu pouze jedné hodnoty může uživatel využít implementovanou vlastnost zápisu jednoho registru do zařízení pomocí klávesy Enter. Uživatel u vybraného registru změní nastavovanou hodnotu v posledním sloupci výpisu. Následně pomocí kliknutí levého tlačítka myši označí požadovaný řádek (na obr. 4.5 je označen registr s adresou 5) a stiskem klávesy enter potvrdí zápis vybrané hodnoty registru do zařízení.

Vlevo nahoře je umístěn panel hlavního menu. Jeho kontextové menu obsahuje položky Load configuration a Save configuration, které slouží k načtení jiné konfigurace zařízení, respektive k uložení aktuálního nastavení zařízení. Dále obsahuje položku Start visualization, která slouží ke spuštění vizualizačního okna v novém vlákně.

4.5.2 Vizualizační okno

Vizualizační okno slouží k odměřování dat a jejich vizualizaci. Je spuštěno z hlavního konfiguračního okna. V pravé horní části jsou umístěny kontrolky reprezentující číselné hodnoty regulované, akční a žádané veličiny. Pod nimi jsou umístěny tlačítka sloužící k obsluze měření a vizualizace. Při spuštění okna je aktivní pouze tlačítko Run m. Tím je spouštěno samotné periodické měření. K odečítání ze sběrnice dochází periodicky každých 0,1s. Dochází k pravidelnému odečítání prvních šesti registrů obsahujících informaci o hodnotě regulované veličiny, akční veličiny v %, registru reprezentujícího indikační panel, žádané veličiny a dvou

rezervních registrů. K odečítání dochází jedním dotazem na sběrnici. Skutečný čas odebrání vzorku je počítán jako rozdíl systémového aktuálního času v době odebírání vzorku a systémového času na začátku měření. Princip periodického odměřování je blíže popsán v pododdíle 4.2.5. Po ukončení měření tlačítkem Stop m. lze naměřená data uložit pomocí tlačítka Save data do externího souboru ve formátu XML.



Obr. 4.6 – Vizualizační okno

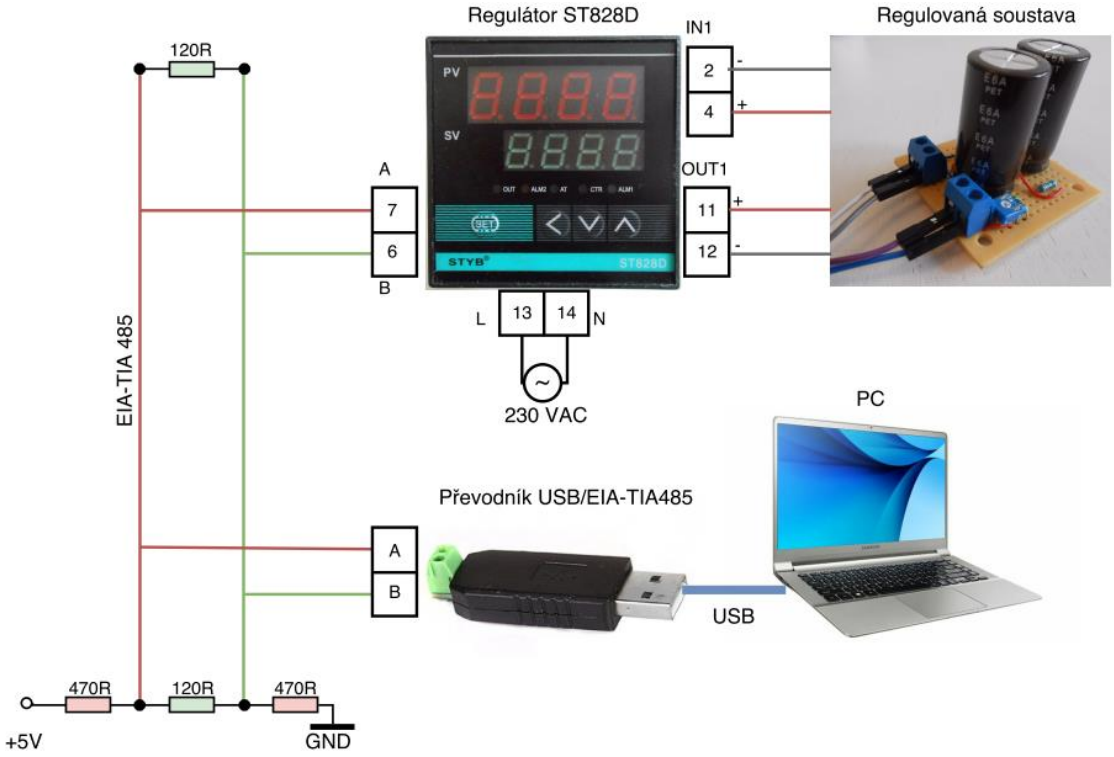
Zároveň s měřením probíhá vizualizace měřených dat. V levé části okna je k tomuto účelu umístěno pod sebou celkem 5 vykreslovacích oken. Jednotlivá vykreslovací okna jsou uzpůsobena pro vizualizaci dat v reálném čase a využívají tzv. XY grafy. Jejich výhodou je možnost formátování časové osy a zobrazení více průběhů do jednoho grafu. Všechny grafy na ose X zobrazují shodnou časovou osu. Horní největší okno na osách Y zobrazuje měřené spojité veličiny. Levá osa Y je vztažena k hodnotám žádané a regulované veličiny. Ty jsou označené PV a SV, shodně jako na panelu přístroje. Jednotky nejsou uvedeny, protože závisí na druhu měřené veličiny. Pravá osa Y je vztažena k hodnotě akční veličiny a je uvedena v procentech. V menších vykreslovacích oknech jsou zobrazeny jednotlivé binární průběhy. V pořadí, tak jak jsou umístěny okna fyzicky pod sebou, se jedná o výstup Out1 (zobrazuje četnost spínání reléového výstupu), alarmové výstupy Alm1 a Alm2 a signál AT signalizující probíhající funkci automatického ladění. Ve stavovém řádku je znázorněno je-li měření spuštěno nebo nikoliv.

5 TESTOVÁNÍ

V této kapitole je ověřena správnost funkce programu a samotného zařízení. Testování proběhlo na obou vybraných zařízeních (pododdíl 2.4.1). Jako předmět řízení byl sestaven elektrický model soustavy druhého řádu.

5.1 OŽIVENÍ PŘÍSTROJE

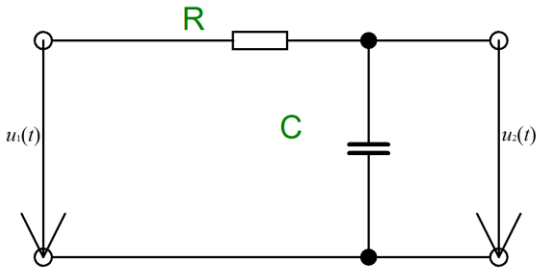
Před samotným vývojem aplikace muselo být zařízení oživeno. Minimální konfigurace zapojení pro ověření komunikace se zařízením je následující. Na svorky číslo 12 a 13 je nutné připojit napájení 230 VAC. Dále je na svorky 6 a 7 nutné připojit komunikační sběrnici přes převodník EIA/TIA485 / USB k PC. K základnímu otestování funkčnosti sběrnice byl použit program ModbusMaster. Jedná se o softwarový sériový terminál sloužící pro základní testování základního přenosu ASCII znaků (textu) přes sériové rozhraní s využitím protokolu Modbus ve verzi RTU nebo TCP. V tomto programu lze data zachytit, zobrazit i vysílat v podobě ASCII znaků. A to nezávisle na fyzickém rozhraní. Jedná se vlastně o vylepšený Hyperterminál, který je ve Windows historicky integrován. Aplikací tohoto druhu je nepřeberná řada. Vstup elektrického modelu regulované soustavy (viz oddíl 5.2) je připojen na výstup regulátoru na svorky 11 a 12. Výstup soustavy je připojen na vstup regulátoru na svorky 2 a 4.



Obr. 5.1 – Blokové schéma zapojení regulačního obvodu

5.2 ELEKTRICKÝ MODEL REGULOVANÉ SOUSTAVY

Pro ověření funkčnosti zařízení byla zvolena soustava 2. řádu. Tato soustava je realizována elektrickým hardwarovým modelem, který se skládá ze dvou integračních RC článků zařazených v sérii za sebou. Obrazový přenos regulované spojité soustavy je odvozen z diferenciální rovnice jednoduchého RC integračního článku (obr. 5.2).



Obr. 5.2 – Schéma jednoduchého RC integračního článku

Tento článek lze popsat diferenciální rovnicí 1.řádu ve tvaru

$$\tau \dot{u}_1(t) + u_2(t) = u_1(t), \tag{5.1}$$

kde $u_1(t)$ – vstupní signál RC článku, V,

$u_2(t)$ – výstupní signál RC článku, V,

τ – časová konstanta, s,

$$\tau = RC, \tag{5.2}$$

kde R – elektrický odpor, Ω,

C – elektrická kapacita, F.

Aplikací Laplaceovy transformace jsou získány rovnice ve tvaru

$$\tau U_2(s) \cdot s + U_2(s) = U_1(s), \tag{5.3}$$

$$U_2(s) \cdot (\tau s + 1) = U_1(s), \tag{5.4}$$

kde $U_1(s)$ – Laplaceův obraz vstupního signálu RC článku,

$U_2(s)$ – Laplaceův obraz výstupního signálu RC článku.

Obrazový přenos je definován jako podíl Laplaceova obrazu výstupního signálu U_2 k Laplaceovu obrazu vstupního signálu U_1. Výsledkem je rovnice obrazového přenosu jednoduchého integračního článku

$$F_1(s) = \frac{U_2(s)}{U_1(s)} = \frac{1}{\tau s + 1}. \tag{5.5}$$

Pokud jsou uvažovány dva sériově spojené přenosy jako na obr. 5.2, pak platí

$$U_2(s) = F_1(s)F_2(s)U_1(s), \tag{5.6}$$

kde $U_1(s)$ – Laplaceův obraz vstupního signálu modelu soustavy,

$U_2(s)$ – Laplaceův obraz výstupního signálu modelu soustavy,

$F_1(s)$ – obrazový přenos prvního RC článku,

$F_2(s)$ – obrazový přenos druhého RC článku.

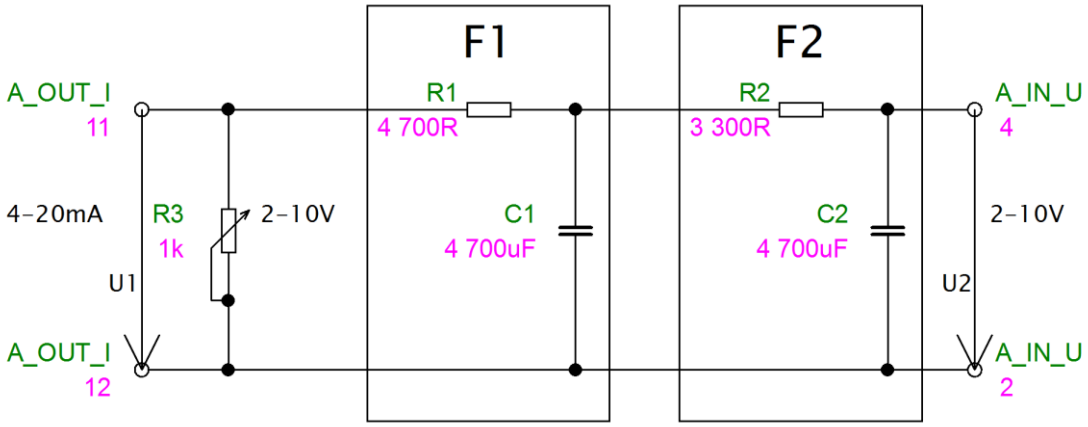
Z tohoto vztahu je odvozen vztah pro obrazový přenos celého modelu soustavy

$$F(s) = \frac{U_2(s)}{U_1(s)} = F_1(s)F_2(s) = \frac{1}{\tau_1 s + 1} \cdot \frac{1}{\tau_2 s + 1} = \frac{1}{(\tau_1 s + 1)(\tau_2 s + 1)}, \tag{5.7}$$

kde τ_1 – časová konstanta prvního RC článku, s,

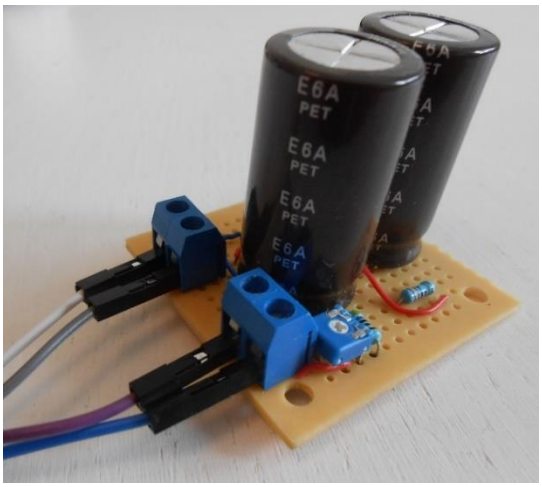
τ_2 – časová konstanta druhého RC článku, s,

$$\tau_1 = R_1 C_1, \quad \tau_2 = R_2 C_2. \tag{5.8}$$



Obr. 5.3 – Schéma zapojení modelu soustavy

První RC článek je fyzicky tvořen kondenzátorem C_1 = 4 700 µF a rezistorem R_1 = 4 700 Ω. Jeho časová konstanta je tedy přibližně τ_1 = 22,1 s. Druhý RC článek je tvořen kondenzátorem C_2 =4 700 µF a rezistorem R_2 = 3 300 Ω. Jeho časová konstanta je tedy přibližně τ_2 = 15,5 s. Hodnoty časových konstant jsou vzhledem k toleranci hodnot použitých součástek zaokrouhleny na první desetinné místo.



Obr. 5.4 – Fyzická realizace modelu soustavy

5.3 KALIBRACE VSTUPŮ A VÝSTUPŮ

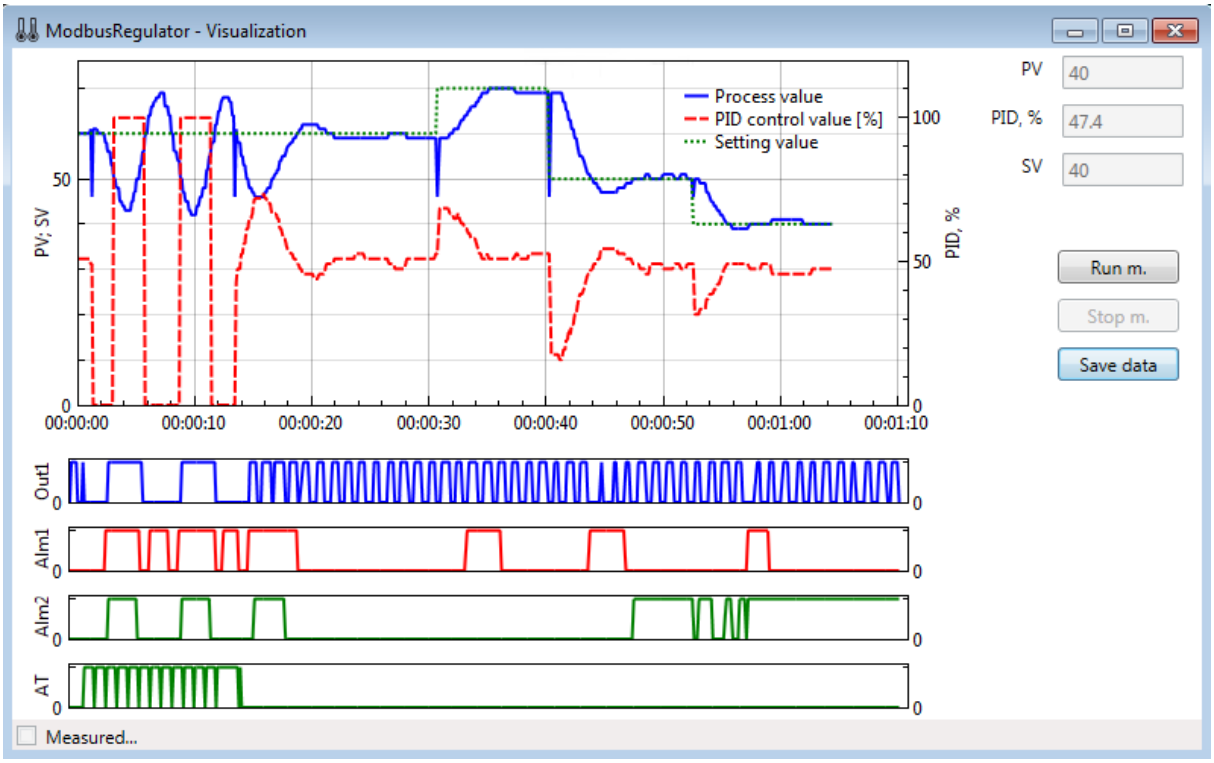
První zvolené zařízení disponuje výstupem s otevřeným kolektorem pro spínání relé v pevné fázi a reléovým výstupem. Reléový kontakt lze použít ke spínání integrovaného proudového výstupu 20 mA. Druhé zařízení je vybaveno proudovým spojitým výstupem s rozsahem od 4 mA do 20 mA. Takže v obou případech je nutné signál pomocí převodníku proud-napětí převést na napěťový signál o rozsahu od 0(2) V do 10 V. Pouze pro účel ověření funkčnosti aplikace a funkčnosti regulátoru je převodník proud-napětí řešen pouze jako odporový dělič bez teplotní kompenzace. Jedná se o trimer R3 na obr. 5.3 o jmenovité hodnotě 1 kΩ. Výsledná hodnota odporu převodníkového rezistoru je zhruba 500 Ω. To plyne z Ohmova zákona. Výstup regulátoru je nutné pomocí trimeru R3 zkalibrovat. Při odpojené soustavě a nastaveném maximálním vybuzení výstupu regulátoru musí být měřená hodnota na výstupu rovna 10 V.

Obě zařízení disponují nastavitelným vstupním obvodem. V této konkrétní aplikaci je použit napěťový vstup o rozsahu od 0 V do 10 V. Vstup má vnitřní elektrický odpor v řádu MΩ. Tento odpor je řádově vyšší než odpor soustavy a neovlivňuje tak znatelně měření.

Kalibrace napěťového vstupu regulátoru proběhla pomocí jednoduchého napěťového děliče s trimerem a pomocí multimetru. Chování vstupu lze ovlivnit nastavením příslušných registrů přes sběrnici nebo pomocí tlačítek přímo na zařízení. Jedná se o parametry ANL (nastavení posunu hodnoty na fyzickém vstupu při které bude zařízení ukazovat nulovou hodnotu), ANH (nastavení posunu hodnoty na fyzickém vstupu při které bude zařízení ukazovat maximální hodnotu) a SC (celkový posun měřené veličiny).

5.4 TESTOVÁNÍ APLIKACE A ZAŘÍZENÍ

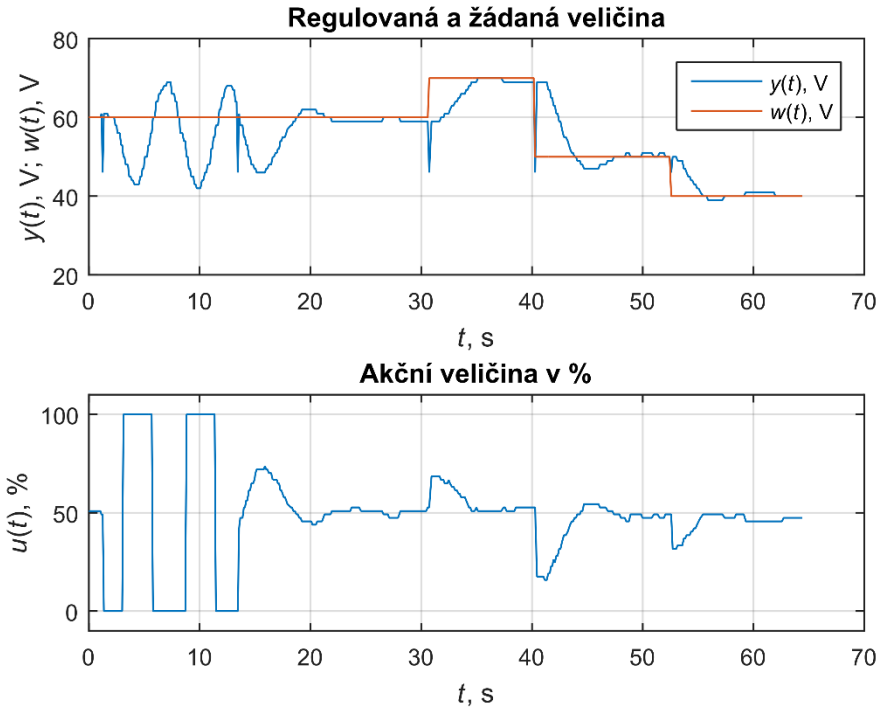
Testování aplikace probíhalo postupně během jejího vývoje. Správná funkčnost jednotlivých částí kódu aplikace byla postupně ověřována pouze s připojeným oživeným zařízením. Zjištěné chyby a nedostatky byly odstraněny. Ladění programu probíhalo tedy hlavně v rovině správnosti komunikace, práce se soubory atd. Dále byla aplikace testována postupně na obou zařízeních s připojeným modelem soustavy. Spolu s tímto testováním probíhalo odladění programu z hlediska obsluhy uživatelem.



Obr. 5.5 – Vizualizace v reálném čase s reléovým výstupem regulátoru

Na obr. 5.5 je zachycen průběh implementované funkce automatického ladění parametrů regulátoru. Výrobce blíže neupřesňuje použitou metodu uvnitř zařízení. Tato funkce proběhla v časovém intervalu od 2 s do 12 s. Nalezené parametry PID regulátoru jsou pásmo proporcionality $P_b = 5$, integrační časová konstantu sériové reprezentace $T_{IS} = 5$ a derivační časová konstantu sériové reprezentace $T_{DS} = 1$. V časech 31 s, 41 s a 53 s od začátku měření jsou zachyceny reakce regulačního obvodu na změny žádané veličiny. V tomto případě je použito první z vybraných zařízení (viz oddíl 2.4.1) s reléovým výstupem. Doba periody spínání relé je konfigurovatelná. Je nastavena hodnota na nejnižší možnou hodnotu 1 s. Zároveň je ověřena funkce alarmových výstupů. Ty jsou také plně konfigurovatelné. Jsou nastaveny následovně. První alarm přejde do stavu logické jedničky v případě, že regulační odchylka přesáhne

hodnotu 5. Druhý alarm je nastaven do logické jedničky v případě, že regulovaná veličina klesne pod absolutní hodnotu 50.

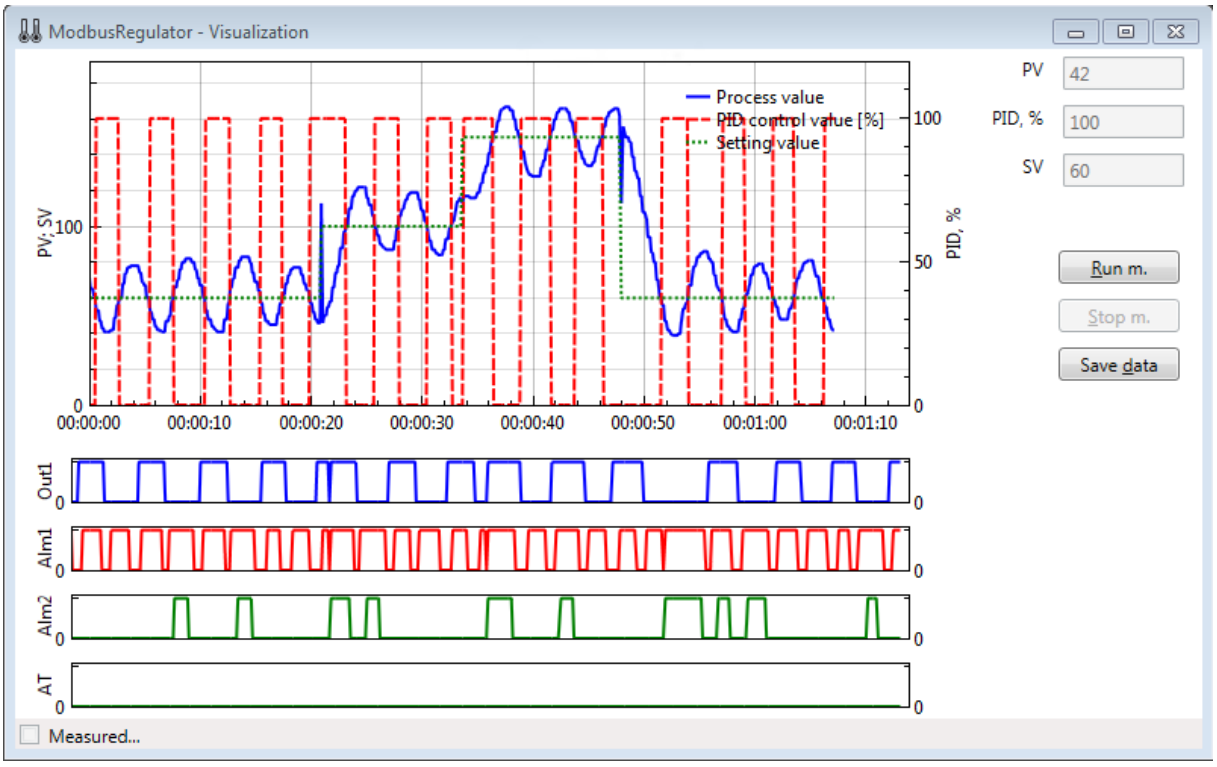


Obr. 5.6 – Naměřená data zpracovaná externě v Matlabu

Následně byla naměřená data uložena a pro demonstraci zpracována externě pomocí programu Matlab (obr. 5.6). Na obou obrázcích je zřetelné, že při změně žádané veličiny dojde vždy ke skoku regulované veličiny. Nejedná se však o výpočtovou chybu a nedochází k přechodovému ději. Tato skutečnost byla ověřena pomocí osciloskopu. Vždy při zápise do zařízení na jakoukoliv adresu dojde k nastavení prvního registru (regulovaná veličina) na odlišnou hodnotu. Při dalším čtení je již hodnota správná. Jedná se o chybu implementace v zařízení při zápise do jeho registrů ze sběrnice. Chyba je nejspíše způsobena tím, že komunikační rozhraní není prioritně určeno k pravidelnému odečítání dat, ale pouze pro konfiguraci nastavení regulátoru. Ovšem pro účely měření po omezenou dobu, například pro odměření regulačního pochodu, jsou vlastnosti zařízení dostačující.

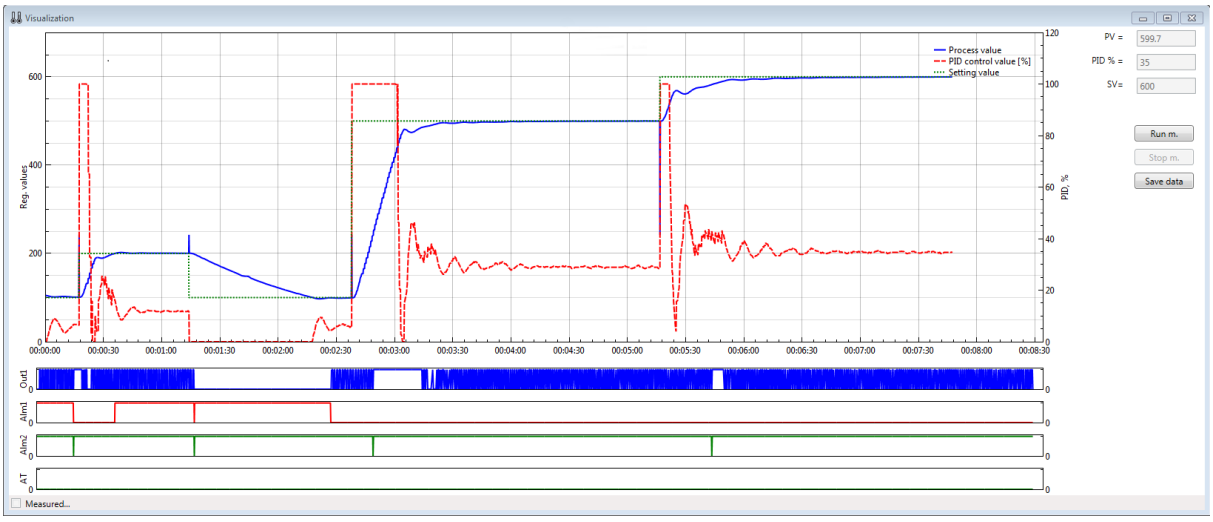
Na obr. 5.7 je zachycena vizualizace dvoustavové regulace v reálném čase. Měření je uskutečněno fyzicky na tom samém přístroji jako v předešlém případě. První alarm se nastaví do logické jedničky v případě, že regulační odchylka je větší než 10. Druhý alarm se nastaví do logické jedničky, pokud je regulační odchylka větší než 20. Soustava je pro tento způsob řízení relativně rychlá a regulovaná veličina značně kolísá. U skutečné tepelné soustavy bývají časové

konstanty regulované soustavy mnohem vyšší a tento režim lze využívat pro aplikace nenáročné na kvalitu řízení.



Obr. 5.7 – Vizualizace v reálném čase dvoustavová regulace

Na obr. 5.8 je zachycen regulační pochod s použitím druhého zařízení se spojitým výstupem regulátoru o rozsahu od 4 mA do 20 mA. Vlivem použití fyzicky jiného regulátoru a tím i jiného výstupu se změnilo chování regulačního obvodu. Parametry regulátoru vypočítané vestavěnou funkcí regulátoru mají hodnotu $P_b = 60$, $T_{IS} = 10$ a $T_{DS} = 1$.



Obr. 5.8 – Vizualizace v reálném čase se spojitým výstupem regulátoru

6 ZÁVĚR

Výstupem práce je funkční desktopová aplikace pro komunikaci PC s regulátorem ST828D včetně programové dokumentace, která je součástí kódu. Program umožnuje konfiguraci regulátoru, vizualizaci regulačního pochodu v reálném čase a umožňuje ukládání naměřených dat do externího souboru. Komunikace využívá protokol Modbus RTU provozovaný na sběrnici EIA-TIA 485. Konfigurační soubory pro komunikaci s různými zařízeními jsou uloženy v externích souborech. Hlavní možností rozšíření aplikace je vytvoření dalších konfiguračních souborů pro jiná zařízení a přizpůsobení zdrojovému kódu, aby s nimi program mohl pracovat. Dalšími možnými modifikacemi je například připojení více slave zařízení na sběrnici v jednom okamžiku nebo rozšíření o komunikační protokol Modbus TCP (implementován v EasyModbus).

Zvolená zařízení řady ST828 jsou určena především pro řízení pomalých tepelných soustav. To hlavně díky pevně dané vzorkovací periodě 0,5 s. Průběhy regulačních pochodů, zaznamenané během testování aplikace, jsou uspokojivé a odpovídají očekáváním. Funkčnost obou vybraných zařízení, včetně implementovaných funkcí (automatické ladění parametrů, alarmové výstupy atd.), byla potvrzena. Zařízení jsou pro jednoduché samostatné aplikace vzhledem k pořizovací ceně použitelné. Ve spojení s vyvinutým programem je nastavení regulátoru jednoduché a přehledné. Uživatel má během konfigurace parametrů regulátoru dobrou představu o chování regulačního obvodu i jednotlivých funkcí zařízení.

POUŽITÁ LITERATURA

ASTRÖM, K.; HÄGGLUND, T. 1995. *PID controllers: Theory,design and tunning. Second edition*. Research Triangle Park (USA): Instrument Society of America. 343 p. ISBN 1-55617-516-7.

ARDUINED.EU. 2017. *CH340G converter Windows 7 driver* [online]. [cit. 21. 4. 2017]. Available: http://www.arduined.eu/tag/ch340g/

BOBÁL, V.; BOHM, J.; FESSL, J.; MACHÁČEK, J. 2005. *Digital Self-tuning Controllers: Algorithms, Implementation and Application*. Londýn: Springer. 317 p. ISBN 978-1-85233-980-7

HLAVA, J. 2000. *Prostředky automatického řízení II: analogové a číselné regulátory, elektrické pohony, průmyslové komunikační systémy*. Praha: České vysoké učení technické. 160 s. ISBN 80-01-02221-8.

KOSEK, J 2000. *XML pro každého: podrobný průvodce*. Praha: Grada. 164 s. ISBN 80-7169-860-1.

MICROSOFT. 2017. .*NET Framework 4.6 a 4.5*[online]. [cit. 21. 4. 2017]. Dostupné z: *https://msdn.microsoft.com/cs-cz/library/w0x726c2(v=vs.110).aspx*

MODBUS.ORG. 2012. *Modbus application protocol specification V1.1.b3* [online]. Hopkinton: Modbus.org. 50 p. [cit. 21. 4. 2017]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

MODBUS.ORG. 2006. *Modbus over serial line specification and implementation guide V1.02* [online]. Hopkinton: Modbus.org. 44 p. [cit. 21. 4. 2017]. Available: http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf

MORNING ELETRONICS. 2016. *ST828 Series Manual and RS485 Communication Protocol* [email]. 18 p. [cit. 21. 4. 2017]

OXYPLOT. 2016. *Oxyplot documentation* [online]. [cit. 21. 4. 2017]. Available: http://docs.oxyplot.org/en/latest/

ROSSMANN-ENGINEERING. 2016. *EasymodbusTCP* [online]. [cit. 21. 4. 2017]. Available: http://easymodbustcp.net/easymodbustcp

PŘÍLOHY

A - CD

Příloha A

Příloha k diplomové práci

Software pro obsluhu a nastavování průmyslového regulátoru

Miroslav Musílek

CD

Obsah

1 Text diplomové práce ve formátu PDF

2 Úplný zdrojový kód aplikace (Visual Studio projekt)

3 Instalační balíček aplikace

4 Spustitelný soubor aplikace

5 Konfigurační soubor ConfigurationFileST828D.xml