ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA
STROJNÍ

DIPLOMOVÁ PRÁCE

SAMOLADICÍ REGULACE POMOCÍ PLC TECOMAT FOXTROT

2018

AUTOR:
BC. FRANTIŠEK HYLMAR
VEDOUCÍ PRÁCE:
PROF. ING. MILAN HOFREITER CSC.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

Dne:_____ *Podpis:_____*

Poděkování

Děkuji tímto panu prof. Ing. Milanu Hofreiterovi, CSc. za odborné vedení diplomové práce, za všechny cenné rady, připomínky a poskytnuté materiály. Především pak děkuji za všechen věnovaný čas a velkou trpělivost.

Děkuji také celé svojí rodinně a všem přátelům, za veškerou pomoc a podporu při tvorbě práce. Speciální poděkování patří zejména drahé kolegyni Bc. Alžbětě Hornychové.

Abstrakt

Práce se zabývá metodikou návrhu a realizace samoladících regulátorů. Po stručném úvodu do problematiky jsou v ní popsány tři vybrané metody pro automatické nastavení parametrů regulátoru. Při výběru je pak kladen důraz zejména na využití pro praxi. Následně je představen programovatelný automat PLC Tecomat Foxtrot a jeho softwarové možnosti pro programování řídících algoritmů. V další části práce je popsána praktická realizace vybraných metod samoladících regulátorů a jejich implementace do vybraného PLC v programovacím jazyce ST za použití prostředí Mosaic. V poslední části práce jsou pak všechny naprogramované samoladící regulátory otestovány na reálných laboratorních úlohách. Výsledky testování jsou zhodnoceny v závěru.

Abstract

The thesis deals with design and implementation methodology of self-tuning controllers. After a brief introduction three selected self-tuning methods are described. The selection of these methods is based mainly on their use for practice. Subsequently programmable PLC Tecomat Foxtrot is described also with its options in controller programming. The next part of the thesis describes implementation of selected self-tuning methods into selected PLC using programming lanuguage ST and programming tool Mosaic. In the last part of thesis, all programmed self-tuning controllers are tested on real laboratory tasks. The test results are evaluated in the conclusion.

Obsah

Úvod

V poslední době lze sledovat trend zvyšující se míry automatizace prakticky ve všech oblastech lidského života. Ať už se jedná o průmysl, výrobu, řízení budov nebo třeba zemědělství, s prvky automatizace se setkáváme stále častěji a častěji. Velký vliv na to má i současná dobrá ekonomická situace v naší zemi, kdy většinou problémem není nedostatek financí, ale lidských zdrojů. V mnoha případech je tak automatizace z dlouhodobého hlediska levnější variantou (například instalace samoobslužných pokladen namísto najmutí personálu do prodejny). V některých případech pak dokonce variantou jedinou. O tomto problému hovoří kupříkladu vyjádření Zemědělského svazu České republiky ze dne 24. 5. 2018: „Do deseti let odejde ze zemědělství z důvodu vysokého věku třetina pracovníků. Tento výpadek je podle Zemědělského svazu nezbytné řešit právě automatizací a robotizací.“ [1] Ve vyjádření se ZSČR odvolává také na provedenou studii, podle níž v posledních deseti letech investovalo do automatizace a robotizace 57 % zemědělských podniků. Nejvíce pak bylo investováno do navigačních systémů (70 % podniků) a do senzorů používaných v živočišné výrobě (52 %). [1]

Aby však automatizace mohla sloužit jako nahráda za lidskou práci, je třeba vysoká spolehlivost a robustnost řídících systému. V případě, že při nasazení automatizace dochází k příliš velké chybovosti dosažených výsledků, může její implementace a provoz zaměstnat více lidských pracovních sil, než dokáže sama nahradit. Proto je třeba klást při vývoji nových metod automatického řízení důraz především na funkčnost a využitelnost s přesah do praxe.

S rozšířením automatizace se do styku s jejími prostředky dostává stále větší počet lidí. Při návrhu nových systému je tak dobré brát v potaz i fakt, že ne všichni uživatelé prostředků automatizace mají patřičné technické vzdělání a zkušenosti. Vždy je tak třeba klást důraz na to, aby automatizace lidem život více usnadňovala, než komplikovala.

1. PID regulátory a nastavování jejich parametrů

K základním prvkům automatizace patří regulátory. Možností pro jejich využití je mnoho a jejich implementace není příliš náročná. Pro řízení analogových procesů je pak bezesporu nejznámější regulátor typu PID. Regulátory tohoto typu začaly v podobě, jak je známe dnes, vznikat již v letech 1915-1940 v důsledku zakládání velkých regulačních firem Bristol, Fisher, Foxboro, Honeywell, Leeds & Nortrup a Taylor Instrument. Mnohem dříve pak byly používány regulátory typu PI. První zmínky jsou již kolem roku 1750 kdy byl pro řízení otáček větrného mlýnu použit odstředivý regulátor. Na podobném principu pracoval také regulátor, který řídil otáčky Wattova parního stroje (1788). Pro pokrok ve vývoji regulátorů bylo zapotřebí pochopení významu jednotlivých částí regulátoru. Především pak bylo zásadní oddělení akčního členu od čidla. Konkrétní aplikací této myšlenky byl vývoj hydraulického čidla. První regulátor s derivační složkou byl sestrojen ve společnosti Taylor Instrument v roce 1935 jak pneumatický regulátor. S vývojem techniky přešel PID regulátor od pneumatiky až na současnou mikroprocesorovou technologii. Přesto princip regulátor zůstal prakticky beze změn. Řízení je stále realizováno pomocí rovnice, kde je akční veličina u výsledkem součtu proporcionální, integrační a derivační složky. [2]

Vzhledem k velkým výpočetním výkonům dnešních programovatelných automatů lze regulátory poměrně dobře upravovat a zdokonalovat jejich funkci. PID regulátory dovedou dobře řídit nejrůznější soustavy, ovšem pouze za předpokladu, že jsou správně nastaveny koeficienty u jednotlivých složek.

Existuje více metod, jak lze postupovat při nastavování parametrů (Ziegler a Nichols, Astrom a Hagglund apod.). K jejich realizaci je však zapotřebí jistá technická znalost a zkušenost uživatele. To do jisté míry omezuje rozšíření PID regulátorů. Řešení tohoto problému nabízí tzv. „samoladící“ regulátory. Jejich myšlenkou je automatické nastavení parametrů regulátoru s co možná nejnižším nutným zásahem uživatele. Tento přístup tak otevírá možnost práce s regulátory i pro méně kvalifikované uživatele. Těm kvalifikovaným pak usnadňuje rutinní práci při nastavování parametrů dle obvyklých metod.

V dalších kapitolách práce jsou popsány vybrané metody samoladících regulátorů.

2. Metody samoladících regulátorů

Při výběru metod byl kladen důraz zejména na jejich potencionální využitelnost v praxi. Existují totiž velmi sofistikované matematicky složité metody, které však své využití omezují pouze na simulované soustavy, popřípadě klade jejich realizace příliš velké nároky na znalosti a zkušenosti uživatele. Pro praxi je třeba využít co možná nejspolehlivějších a nejjednodušších metod, jejichž implementace a provoz neklade příliš velké nároky na uživatele.

2.1. Princip regulátoru dle J. Maršíka

Adaptivní regulátor s průběžným nastavováním parametrů dle J. Maršíka může pracovat se třemi strukturami regulátoru a to I, PI a PID. Parametry jsou vypočítávány průběžně a přímo. Mezi jeho hlavní přednosti patří zejména absence jakékoliv identifikační nebo inicializační fáze. Regulovaná soustava také může být libovolného řádu, lineární i nelineární. J. Maršík však dodává: „Předpokládá se, že je soustava sama o sobě stabilní a málo kmitavá, nejlépe s monotónní přechodovou charakteristikou.“ [3]

Jedinou veličinou, kterou algoritmus adaptivního regulátoru dle J. Maršíka sleduje, je regulační odchylka. Pro její vyhodnocení pak J. Maršík zavádí nové kritérium zvané „míra kmitavosti“. Toto kritérium je značeno jako κ a ve smyslu práce je definováno jako poměr mezi frekvencí průchodů nulou regulační odchylky a frekvencí průchodů nulou její první derivace.

Tento přístup k řízení je potencionálně výhodný zejména pro jeho nespornou univerzalitu při použití. Absence jakéhokoli specifičtějšího požadavku na typ řízené soustavy poskytuje této metodě široké spektrum využití.

„Míra kmitavosti“ je popsána vztahem

$$\kappa = \frac{f_e}{f_v} \tag{1}$$

kde κ je bezrozměrný parametr, f_e je frekvence průchodů nulou regulační odchylky a f_v je frekvence průchodů nulou její první derivace. Pro reálné průběhy pak platí, že:

$$f_v \geq f_e \geq 0 \tag{2}$$

Z této nerovnosti můžeme dále odvodit rozsah pro míru kmitavosti κ:

$$0 \leq \kappa \leq 1 \tag{3}$$

Je zřejmé, že při harmonickém signálu $f_v \rightarrow f_e$, a tak platí i $\kappa \rightarrow 1$. Velikost míry kmitavosti κ odráží také charakter regulačního procesu. Čím více je proces tlumený, tím větší je nerovnost $f_v > f_e$, a tím menší je hodnota κ. Naopak, čím je proces více kmitavý, tím je hodnota κ větší.

Nosným prvkem adaptivního regulátoru dle J. Maršíka je vztah míry kmitavosti κ a parametrů regulátoru v soustavě se zpětnou vazbou. Základním parametrem popsaným v [3] je pak společné zesílení s označením α. Při malých hodnotách κ jsou nízké i hodnoty α. Pokud roste zesílení α roste i míra kmitavosti κ až do chvíle, kdy dosáhneme meze stability. V tu chvíli nabývá své maximální hodnoty 1. Tato závislost ovšem nemá lineární charakter a závisí také na ostatních parametrech regulátoru.

Vzhledem k tomu, že rozsah (3) platí vždy, hledá Maršík hodnotu míry kmitavosti κ, při které regulační smyčka vykazuje co možná nejlepší charakter řízení. Jako odpověď uvádí, že pro většinu soustav je optimální hodnotou κ = 0,5. Tuto hodnotu opírá patrně o zkušenosti získané při vývoji metody. Zároveň však také dodává: „Toto pravidlo ovšem platí zhruba a má také své výjimky.“ [3]

Nastavování parametru zesílení regulátoru α lze vzhledem k průběhu κ(α) zajistit pomocí adaptační smyčky tak, aby platilo κ = 0,5. Optimální hodnota α by v ideálním případě lineární charakteristiky κ(α) ve tvaru:

$$\kappa(\alpha) = \frac{\alpha}{\alpha_{krit}} \tag{4}$$

kde α_{krit} odpovídá mezi stability, mohla být dosažena po prvním stanovení ustálené hodnoty κ. Hodnotu α lze v tomto případě získat dle úměry:

$$\alpha_{k+1} = \alpha_k * \frac{0{,}5}{\kappa_k} \tag{5}$$

S touto znalostí je možno sestavit rovnici (6), s jejíž pomocí můžeme upravovat společné zesílení α.

$$\alpha_{k+1} = \alpha_k * \frac{0{,}5\alpha_{krit}}{\alpha_k} = 0{,}5\ \alpha_{krit} \tag{6}$$

Zjištění, že optimální hodnota α vychází jako polovina hodnoty kritické kvituje J. Maršík s tím, že k podobnému závěru vede například pravidlo Zieglera a Nicholse nebo kritérium „maximální rezervy stability“ dle [3]. [3]

V další části své práce J. Maršík upozorňuje že: „Potřebná délka sledovaného regulačního procesu musí být dostatečná, abychom vyhodnocovali ustálený stav, a to nejen s ohledem na dynamiku obvodu, ale i statistický charakter náhodných poruch.“ [3] Pokud tedy měříme průběžně regulační odchylku *e* a její předchozí hodnoty ukládáme, můžeme detekcí jejích průchodů nulou N_e a počtu všech extrémů N_v zjistit míru kmitavosti dle vztahu:

$$\kappa = \frac{f_e}{f_v} \approx \frac{N_e}{N_v} \tag{7}$$

Podmínkou pro použití uvedené metody je však dostatečná filtrace vysokofrekvenčních šumů, jelikož na jejich výskyt není dle J. Maršíka popsaná regulační smyčka schopna reagovat. K řešení tohoto problému navrhuje J. Maršík zejména úpravu vzorkování. Pro příklad uvádí maximálně 5 vzorků na jednu půlvlnu překmitu regulované veličiny. [3]

V práci [3] je uveden i příklad algoritmu, pracujícího s PID regulátorem. Jeho strukturu tvoří několik rovnic, pomocí kterých jsou z regulační odchylky dopočítány parametry pro regulátor. Celý návrh je pak popsán v několika krocích:

1. Ze žádané hodnoty *w* a regulované veličiny *x* je vypočtena regulační odchylka *e*

$$e(n) = w(n) - x(n) \tag{8}$$

2. Velikost regulační odchylky *e* je porovnána s jejím rozptylem a je rozhodnuto, zda regulátor vyžaduje úpravu parametrů.

$$e^2(n) < \sigma_e^2(n) \tag{9}$$

V případě, že je podmínka splněna, program vynechá tu část, kde jsou přepočítávány parametry PID regulátoru a přeskočí rovnou do fáze 10.

3. Výpočet časové konstanty filtru τ pro následné určení středních hodnot parametrů.

$$\tau(n) = 2\pi\sqrt{\frac{\overline{v^2(n-1)}}{\overline{a^2(n-1)}}} \tag{10}$$

4. Aktualizace rozptylu regulační odchylky σ_e^2

$$\sigma_e^2(n) = \frac{e^2(n) + 3\tau(n)\sigma_e^2(n-1)}{1 + 3\tau(n)} \tag{11}$$

5. Výpočet parametrů $\overline{e^2(n)}$, $\overline{v^2(n)}$, $\overline{a^2(n)}$

$$\overline{e^2(n)} = \frac{e^2(n) + \tau(n)\overline{e^2(n-1)}}{1+\tau(n)} \tag{12}$$

$$\overline{v^2(n)} = \frac{(\Delta e(n))^2 + \tau(n)\overline{v^2(n-1)}}{1+\tau(n)} \tag{13}$$

$$\overline{a^2(n)} = \frac{(\Delta^2 e(n))^2 + \tau(n)\overline{a^2(n-1)}}{1+\tau(n)} \tag{14}$$

6. Určení míry kmitavosti κ

$$\kappa(n) = \frac{\overline{v^2(n)}}{\sqrt{\overline{e^2(n)} * \overline{a^2(n)}}} \tag{15}$$

7. Výpočet společného zesílení regulátoru $\alpha(n)$

$$\Delta\alpha(n) = \alpha(n)\frac{1}{\tau(n)}\left(\frac{\kappa_{\check{z}\acute{a}d.}}{\kappa} - 1\right) \tag{16}$$

8. Výpočet koeficientu proporcionální složky regulátoru

$$\beta(n) = \sqrt{\frac{\overline{e^2(n)}}{\overline{v^2(n)}}} \tag{17}$$

9. Výpočet koeficientu diferenciální větve regulátoru

$$\gamma(n) = \sqrt{\frac{\overline{e^2(n)}}{\overline{a^2(n)}}} \tag{18}$$

10. Výpočet akční veličiny regulátoru u (dosazení všech vypočtených parametrů do rovnice regulátoru)

$$u(n+1) = \alpha(n)\gamma(n)\Delta e(n) + \alpha(n)\beta(n)e(n) + \sum_{i}^{n} \alpha(i)e(i) \tag{19}$$

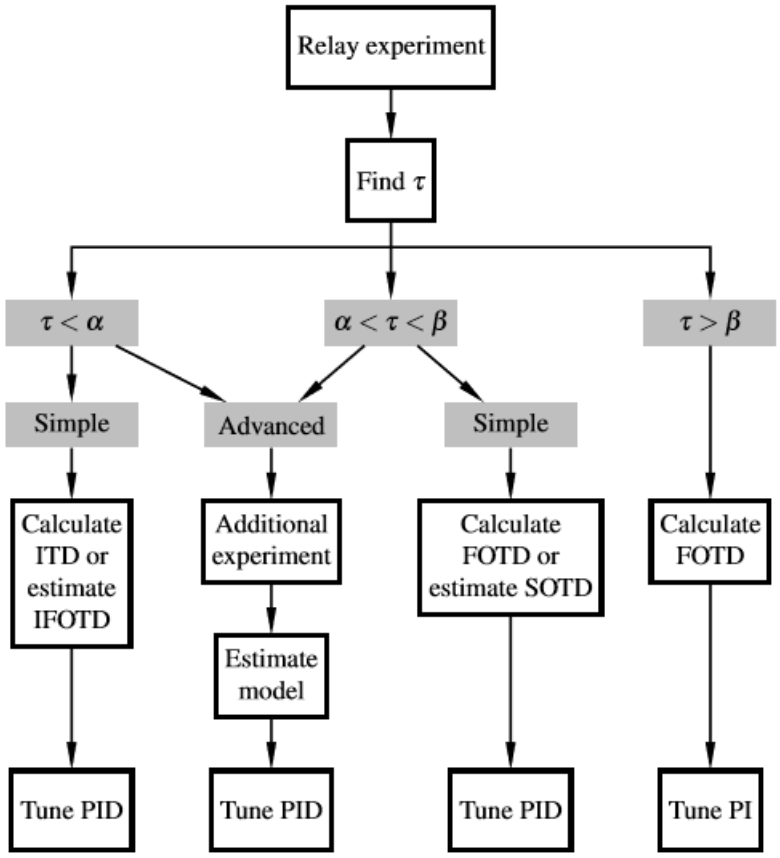
11. Po nastavení výstupu regulátoru je program ukončen a vrací se na začátek [3].

Odvození rovnic použitých v algoritmu je uvedeno v [3].

2.2. Princip regulátoru dle J. Berner

2.2.1. Úvod do metody

Základem regulátoru dle J. Berner je inicializační fáze, ve které jsou získány parametry pro model soustavy. Druh modelu soustavy je nutno zvolit a je třeba počítat s tím, že vybraný model nemusí uspokojivě popisovat reálné chování soustavy. Možností, jak zmírnit toto omezení a vylepšit tak algoritmus, je zavedení rozhodovacího parametru, na základě kterého můžeme vybírat z více modelů. Příkladem takovéto úpravy je i schéma (Obrázek 1), které pro výběr vhodného modelu popisuje ve své práci [4] i J. Berner.



Obrázek 1 - Rozhodovací schéma pro určení vhodného modelu soustavy dle dopravního zpoždění [4]

Rozhodovacím parametrem je v tomto případě normalizované dopravní zpoždění $\tau \in [0,1]$. Způsob jeho určení v metodě dle J. Berner je popsán v jedné z dalších kapitol (2.2.4). Jakmile je τ určeno, je možno ho porovnat s předem určenými parametry α a β. Tyto parametry představují meze intervalů hodnoty τ, ve kterých je pro identifikovanou soustavu vhodný přiřazený model. Pro nízké hodnoty $\tau < \alpha$ je dle schématu pro soustavu přiřazen model ITD – integrační s dopravním zpožděním. Pokud hodnota normalizovaného dopravního zpoždění leží v intervalu $\alpha < \tau < \beta$ jsou vypočteny parametry pro model FOTD – prvního řádu s dopravním zpožděním. Pro oba tyto intervaly lze provést ještě dodatečný experiment, pomocí kterého je rozhodovací algoritmus doplněn o dva další modely. V případě intervalu $\tau < \alpha$ jde o model IFOTD – integrační prvního řádu s dopravním zpožděním. V případě intervalu $\alpha < \tau < \beta$ pak o model SOTD – druhého řádu s dopravním zpožděním. U hodnot $\tau > \beta$ je pro popis soustavy použit model FOTD – prvního řádu s dopravním zpožděním. Pro poslední interval pak metoda dle J. Berner nezavádí žádné

doplňující rozhodovací pravidlo. Použité modely jsou pak definovány rovnicemi dle tabulky (Tabulka 1).

Typ modelu	Předpis
FOTD – prvního řádu s dopravním zpožděním	$P(s) = \frac{K_p}{1 + \mathrm{sT}} e^{-sL}$
SOTD – druhého řádu s dopravním zpožděním	$P(s) = \frac{K_p}{(1 + sT_1)(1 + sT_2)} e^{-sL}$
ITD – integrační s dopravním zpožděním	$P(s) = \frac{k_v}{s} e^{-sL}$
IFOTD – integrační prvního řádu s dopravním zpožděním	$P(s) = \frac{k_v}{s(1 + sT)} e^{-sL}$

Tabulka 1 - Definice matematických modelů soustav [4]

Každý z předpisů obsahuje několik parametrů určujících chování modelu. Jedná se o parametr udávající zesílení soustavy, které se nachází v čitateli předpisu (K_p) a parametr pro integrační systémy k_v. Dále o parametr dopravního zpoždění L. A nakonec časové konstanty označené T, T_1 a T_2, které jsou ve jmenovateli předpisu. Pokud určíme hodnoty těchto

konstant, lze pomocí nich dopočítat parametry PID regulátoru. V případě metody dle J. Berner pak dle tabulky (Tabulka 2).

Model	PID parameters
FOTD	$K = \frac{0.2L + 0.45T}{K_p L}$ $T_i = \frac{0.4L + 0.8T}{L + 0.1T} L$ $T_d = \frac{0.5LT}{0.3L + T}$
ITD	$K = \frac{0.45}{k_v L}$ $T_i = 8L$ $T_d = 0.5L$
SOTD	$K = \frac{0.19}{K_p} + \frac{0.37T_1 + 0.18T_2}{K_p L} + \frac{0.02T_1 T_2}{K_p L^2}$ $k_i = \frac{0.48}{K_p L} + \frac{0.03T_1 - 0.0007T_2}{K_p L^2} + \frac{0.0012T_1 T_2}{K_p L^3}$ $k_d = \frac{T_1 + T_2}{K_p (T_1 + T_2 + L)} \left(0.29L + 0.16T_1 + 0.2T_2 + \frac{0.28T_1 T_2}{L} \right)$
IFOTD	$K = \frac{0.37}{k_v L} + \frac{0.02T}{k_v L^2}$ $k_i = \frac{0.03}{k_v L^2} + \frac{0.0012T}{k_v L^3}$ $k_d = \frac{0.16}{k_v} + \frac{0.28T}{k_v L}$

Tabulka 2- vztahy pro určení parametrů PID regulátoru z daných matematických modelů soustav [4]

Pro každý model jsou v (Tabulka 2) tabulce vztahy, pro výpočet jednotlivých složek PID regulátoru. Samotný regulátor je pak obecně popsán rovnicí:

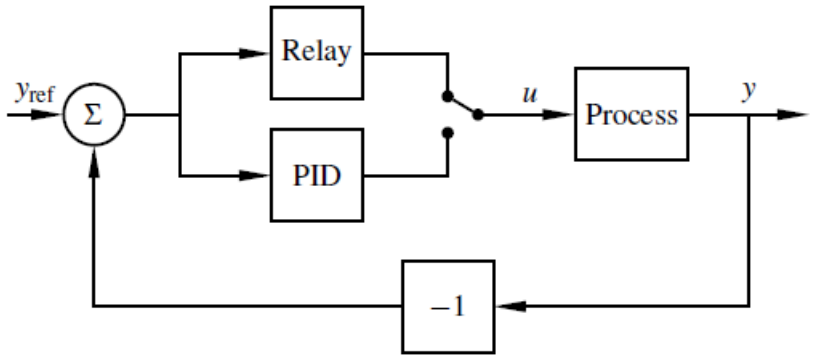
$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\theta)d\theta + T_d\frac{de(t)}{dt}\right) \tag{20}$$

Rovnice popisuje sumaci členů, ze kterých je složen výstup z regulátoru (akční veličina) *u(t)* v čase *t.* Výpočet je založen na aktuální hodnotě regulační odchylky *e(t)*, aktuální hodnotě její integrace a aktuální hodnotě její derivace. Míru, s jakou se jednotlivé složky regulátoru projeví ve výpočtu hodnoty *u(t)* ovlivňuje velikost jejich koeficientů. V uvedené formulaci PID regulátoru je stěžejním parametrem společné zesílení K. Tento koeficient určuje celkovou míru, s jakou se projeví změny v regulační odchylce *e(t)* na výstupu z regulátoru. Podíl integračního a derivačního členu regulátoru na hodnotě *u(t)* pak upravuje integrační časová konstanta T_i u integračního členu, respektive derivační časová konstanta T_d u členu derivačního.

V tabulce (Tabulka 2) můžeme kromě zmíněných koeficientů u modelů SOTD a IFOTD zaznamenat ještě koeficienty k_i a k_d. Jedná se o obdobu koeficientů T_i a T_d, pouze vztaženou k jiné formulaci rovnice PID regulátoru. Pro přepočet mezi nimi pak platí, že $T_i = K/k_i$ a $T_d = k_d/K$. K tomuto vyjádření se J. Berner dle svých slov uchyluje z praktických důvodů.

2.2.2. *Reléová identifikace*

Během inicializační fáze probíhá určení parametrů modelu pomocí reléové identifikace. Použití relé pro získání modelu soustavy bylo poprvé popsáno Astromem a



Obrázek 2 – Schéma zapojení zpětnovazební soustavy pro realizaci reléové identifikace

Hagglundem v roce 1984. Pod jejich vedením pak probíhá i práce Josefin Berner, která na tuto metodu navazuje a rozvíjí ji. Ideou tohoto postupu je automatizované nalezení kritického zesílení a kritické periody. Tyto parametry jsou užity pro naladění regulátoru a vycházejí z metody Zieglera a Nicholse z roku 1942.

Základem této metody je připojení relé namísto regulátoru v uzavřeném regulačním obvodu (Obrázek 2). Smyslem tohoto zapojení je uvedení soustavy do stavu řízeného a pravidelného kmitání. Pokud soustava do stavu přejde, jsme schopni určit z něj kritickou periodu ω_c a kritické zesílení k_c. Tyto parametry nám pak stačí pro určení parametrů potřebných pro řízení. Výhodou a značným zlepšením oproti starším metodám je zejména fakt, že soustava je během identifikační fáze po celou dobu řízena. To je důležité zejména pro procesy, u kterých existují omezení nebo jiné zvláštní požadavky na průběh regulované veličiny. U některých soustav může například aplikace metody Ziegler-Nichols ohrožovat životnost některých součástek stroje nebo dokonce celkovou bezpečnost procesu. J. Berner také uvádí: „Aplikace této metody je snadná a pro její použití nejsou zapotřebí žádné apriorní informace o řízeném procesu.“[4]

Samoladící regulátory využívající reléové identifikace se po svém uvedení na trh rozšířili napříč celým průmyslovým odvětvím zejména zásluhou již zmíněných výhod oproti starším metodám. Během let od představení původního regulátoru tohoto typu bylo popsáno mnoho různých modifikací. Jednou z nejnovějších je i metoda popsaná v této práci.

2.2.3. Popis metody

Metoda popsaná v [4] pracuje s asymetrickým relé s hysterezí. Pásmo hystereze (21) se nachází v okolí pracovního bodu y_0 a jeho velikost závisí a velikosti šumu regulované veličiny v ustáleném stavu soustavy. Hystereze h je zavedena za účelem odstranění možných nežádoucích přepnutí relé v reakci na poruchové signály a šumy. K přepínání relé s hysterezí dochází dle předpisu

$$u(t) = \begin{cases} u_{\text{on}}, & y(t) < y_0 - h, \\ u_{\text{on}}, & y(t) < y_0 + h, \quad u(t^-) = u_{\text{on}}, \\ u_{\text{off}}, & y(t) > y_0 - h, \quad u(t^-) = u_{\text{off}}, \\ u_{\text{off}}, & y(t) > y_0 + h, \end{cases} \tag{21}$$

kde $u(t)$ je hodnota akčního zásahu v čase t a $y(t)$ regulovaná veličina v témže čase. Parametry u_{on} a u_{off} jsou pak horní, respektive dolní mezí relé. Z předpisu je jasné, že k přepnutí relé dochází až ve chvíli, kdy regulovaná veličina dosáhne mezní hodnoty pásma hystereze.

Pro úspěšné provedení metody je zapotřebí splnit několik podmínek, jež definují podobu parametrů relé. Abychom zajistili asymetrii relé, musíme dodržet podmínku (22) pro stupeň asymetrie relé γ

$$\gamma = \frac{\max(d_1, d_2)}{\min(d_1, d_2)} > 1 \tag{22}$$

Jedním z parametrů potřebných pro sestavení modelu je i normalizované dopravní zpoždění $\tau \in [0,1]$. Pro jeho určení vycházíme z poměru časů, kdy je relé v horní, respektive dolní poloze. Tento poměr značíme ρ a určíme ho dle vztahu

$$\rho = \frac{\max(t_{on}, t_{off})}{\min(t_{on}, t_{off})} \tag{23}$$

kde t_{on} představuje dobu, po kterou je relé v horní poloze (na výstupu z regulátoru je u_{on}) a t_{off} dobu, kdy je relé v poloze dolní (na výstupu z regulátoru je u_{off}). Normalizované dopravní zpoždění τ pak můžeme spočítat pomocí vztahu:

$$\tau(\rho, \gamma) = \frac{\gamma - \rho}{(\gamma - 1)(0.35\rho + 0.65)} \tag{24}$$

Pomocí τ pak můžeme například rozhodnout o modelu, který pro soustavu použijeme, jak již bylo zmíněno v předchozí kapitole (2.2.1).

Pro určení parametrů modelu z dat získaných při připojení soustavy na relé existuje více metod. K výběru vhodného výpočetního postupu v této metodě J. Berner uvádí: „Při hledání jsme se soustředili na nalezení jednoduchých, intuitivních vztahů, které využívají naměřená data odolná vůči šumu.“ [4]

Výpočty, použité v této metodě, jsou založeny na čase, po který je relé připojené na vstup do soustavy v horní poloze t_{on}, respektive na čase, kdy je relé v poloze dolní t_{off} a na integraci výstupu ze soustavy I_y. Tuto integraci popisuje vztah (25).

$$I_y = \int_0^{t_p} (y(t) - y_0)dt \tag{25}$$

Hodnota y_0, kterou v rovnici odečítáme od hodnoty regulované veličiny $y(t)$, představuje hodnotu ustáleného výstupu ze soustavy v pracovním bodě. Jinými slovy jde o výchozí stav soustavy před započetím experimentu. Čas t_p, na kterém provádíme integraci, je časem periody kmitání soustavy a platí pro něj $t_p = t_{on} + t_{off}$. Jako potvrzení vhodnosti těchto parametrů vzhledem ke kritériím výběru dodává J. Berner: „Všechny použité parametry je jednoduché získat z naměřených dat a vykazují malou citlivost na ovlivnění šumem.“ [4]

K provedení vybrané výpočetní metody je také zapotřebí určit integrál z průběhu reléového vstupu do soustavy I_u. Tuto hodnotu popisuje obecně vztah (26) analogický k vztahu pro výpočet I_y.

$$I_u = \int_0^{t_p} (u(t) - u_0)dt \tag{26}$$

Vzhledem k tomu, že provádíme integraci obdélníkového signálu můžeme výpočet zjednodušit a získáme tak rovnici (27) ve tvaru:

$$I_u = u_{on}t_{on} + u_{off}t_{off} \tag{27}$$

Ze zmíněných vztahů jsme pak schopni dopočítat všechny potřebné parametry modelu soustavy.[4]

2.2.4. Model prvního řádu s dopravním zpožděním – FOTD

U FOTD modelu definovaného v (Tabulka 1) potřebujeme pro plné určení vypočítat tři parametry a sice K_p, T a L. Statické zesílení K_p lze při použití asymetrického relé určit za pomocí hodnot získaných integrací dle vztahu (28).

$$K_p = \frac{I_y}{I_u} \tag{28}$$

Při použití symetrického relé bychom ve jmenovateli dostali pokaždé nulu, protože plocha pod křivkou signálu vzhledem k hladině výchozího pracovnímu bodu se u symetrického relé rovná ploše nad křivkou vztažené k téže hodnotě.

Časové parametry T a L určíme dle vztahů odvozených v [4] příloze A.1. Pro normalizované dopravní τ zpoždění u systémů prvního řádu s dopravním zpožděním platí

$$\tau = \frac{L}{L + T} \tag{29}$$

z čehož můžeme vyvodit

$$\frac{L}{T} = \frac{\tau}{1 - \tau} \tag{30}$$

a následně pak

$$L = T\frac{\tau}{1-\tau}. \tag{31}$$

čímž získáváme vztah pro výpočet parametru dopravního zpoždění *L*. Pro časovou konstantu *T* pak J. Berner odvozuje vztah:

$$T = \frac{t_{on}}{ln\left(\frac{\frac{h}{|K_p|} - d_2 + e^{\frac{L}{T}}(d_1 + d_2)}{d_{1-}\frac{h}{|K_p|}}\right)} \tag{32}$$

V rovnici se kromě výše (2.2.1) popsaných hodnot vyskytuje ještě velikost pásma hystereze *h* a hodnota horní (d_1), respektive dolní (d_2), polohy relé. Při výpočtu pak jako první spočteme časovou konstantu *T* dosazením (31) do (32). Následně jsme schopni dopočítat i dopravní zpoždění *L* dosazením (32) do (31).

2.2.5. Integrační model s dopravním zpožděním – ITD

Model ITD popsaný v tabulce (číslo tabulky) a definovaný vztahem

$$P(s) = \frac{k_v}{s}e^{-sL} \tag{33}$$

může být také popsán diferenciální rovnicí

$$\dot{y}(t) = k_v u(t - L). \tag{34}$$

z toho pak J. Berner odvozuje vztahy pro parametry modelu. Pro zesílení k_v tak platí, že

$$k_v = \frac{2I_y}{t_{on}t_{off}(u_{on} + u_{off})} + \frac{2h}{u_{on}u_{off}} \tag{35}$$

a pro dopravní zpoždění *L* v tomto případě

$$L = \frac{u_{on}u_{of} - 2h/k_v}{u_{on} - u_{off}} \quad [4] \tag{36}$$

2.3. Princip regulátoru Foxboro EXACT

Adaptivní regulátor EXACT (Expert Adaptive Controller Tuning) byl představen již roku 1984 společností Foxboro [5] (od roku 2014 součást mezinárodní korporace Schneider Electric SE [6]). Principem adaptace je sledování dynamického chování soustavy při přechodech na odlišnou žádanou hodnotu nebo v odezvě na poruchu. K adaptaci pak dochází pouze v případě dostatečně velkých překmitů regulované veličiny při ustalování na žádané hodnotě. Vyhodnocovány jsou tři po sobě jdoucí maximální odchylky od žádané hodnoty e_1, e_2 a e_3. Dalším měřeným parametrem je ještě perioda kmitání regulované veličiny během ustalování T_p.

Z naměřených hodnot jsou pak pomocí vztahů

$$damping = \frac{e_3 - e_2}{e_1 - e_2} \tag{37}$$

$$overshoot = \left|\frac{e_2}{e_1}\right| \tag{38}$$

vypočítány parametry *damping* (tlumení) a *overshoot* (překročení). Dle velikosti těchto parametrů jsou následně pomocí vnitřních heuristických pravidel upraveny parametry PID regulátoru, jímž je řízena regulovaná veličina. [7]

2.3.1. Popis metody

Před spuštěním řídícího adaptačního algoritmu je zapotřebí nastavit parametry *PB*, T_i a T_d, které jsou v metodě použity pro nastavení PID regulátoru. Dále je třeba nastavit šířku pásma šumu *NB* (Noise Band). V případě, kdy v průběhu regulace překročí regulační odchylka po přeregulování nebo v reakci na poruchu dvojnásobek *NB*, dojde k dodatečné úpravě parametrů regulátoru pomocí vnitřních pravidel adaptačního algoritmu. Posledním vstupním parametrem je pak maximální čekací doba (W_{max}). Jedná se o maximální dobu, po kterou algoritmus čeká na výskyt druhého překmitu po prvním překročení dvojnásobku *NB*.

Pokud uživatel soustavu zná, může parametry nastavit dle předchozích zkušeností a znalostí. V opačném případě lze použít samoladící funkci a potřebné parametry vypočítat automaticky.

Principem samoladící funkce je určení parametrů modelu prvního řádu s dopravním zpožděním z přechodové charakteristiky soustavy. Nejprve program přejde do manuálního módu a v něm vygeneruje skok na vstupu do soustavy. Z odezvy soustavy na skok pak algoritmus určí parametry modelu a z nich pomocí vztahů (39, 40, 41) vypočte potřebné parametry *PB*, T_i a T_d.

$$PB = 120k_p \frac{L}{T} \tag{39}$$

$$T_i = 1{,}5L \tag{40}$$

$$T_d = \frac{T_i}{6} \tag{41}$$

Z přechodové charakteristiky soustavy je určena i maximální čekací doba W_{max} dle rovnice:

$$W_{max} = 5L \tag{42}$$

Šířku pásma šumu určí program na základě měření maximálních odchylek regulované veličiny při ustáleném vstupu do soustavy. Po odečtení potřebných hodnot z přechodové charakteristiky je na vstup do soustavy přivedena konstantní hodnota odpovídající stavu před skokovou změnou vstupní hodnoty. Poté je po určitou dobu měřena maximální regulační odchylka nad, respektive pod, žádanou hodnotou. Součtem obou maxim je určena hodnota šířky pásma šumu *NB*.

Parametr *NB* slouží kromě určení hranice pro adaptaci parametrů regulátoru ještě k dodatečné úpravě jeho derivační složky. Pokud soustava vykazuje velké zatížení šumem je vhodné zmenšit konstantu T_d, abychom zamezili kolísání signálu řídící veličiny [7]. Pro úpravu časové derivační konstanty T_d je vztahem

$$Z = \frac{(3 - 2 * NB)}{2{,}5} \tag{43}$$

definován rozhodovací parametr *Z*. Po jeho vypočtení se řídíme podle pravidel:

pokud $Z > 1$ pak $T_d = T_i / 6$

pokud $Z < 0$ pak $T_d = 0$

pokud $0 < Z < 1$ pak $T_d = Z * T_i / 6$

Kromě popsaných parametrů, které adaptivní regulátor vyžaduje pro svou funkci, lze nastavit ještě čtyři volitelně nastavitelné parametry. Konkrétně se jedná o maximální povolené hodnoty parametrů *damping* (tlumení = 0,3) a *overshoot* (překročení = 0,5). Dále o *Derivative factor* (derivační faktor = 1), jenž je doplňující konstantou u derivační složky PID regulátoru a jeho úpravou může uživatel dodatečně korigovat její velikost. A nakonec o *Change limit* (hranice změny = 10), jehož velikost určuje o kolik násobků výchozích hodnot parametrů regulátoru může adaptační funkce tyto parametry upravit. Pokud uživatel tyto parametry nenastaví, jsou ponechány na svých defaultních hodnotách (hodnoty v závorkách).

Ve chvíli, kdy má adaptivní regulátor nastavené všechny potřebné hodnoty, přechází do režimu řízení. V tomto režimu řídí pomocí PID regulátoru měřenou regulovanou veličinu soustavy na hodnotu žádanou, zadanou uživatelem. Pokud při změně žádané hodnoty nebo při zásahu poruchové veličiny dojde k překročení pásma šumu *NB* spustí regulátor adaptační sekvenci, ve které jsou z průběžně měřených hodnot *damping* a *overshoot* pomocí vnitřních pravidel regulátoru přepočteny parametry PID regulátoru. Přesná podoba pravidel adaptivního regulátoru Foxboro EXACT dosud nebyla zveřejněna. V práci [citace foxboro***] je uveden alespoň jejich příklad:

1. Pokud nejsou detekovány překmity regulované veličiny po změně žádané hodnoty, je snížena velikost parametrů *PB,* T_i a T_d.

2. Pokud jsou detekovány překmity a zároveň platí, že velikost parametrů *damping* a *overshoot* je menší než jejich maximum, je snížena velikost parametru *PB*. [7]

V práci [7] je zmíněno, že pravidla použitá při vývoji regulátoru Foxboro EXACT, jsou heuristické povahy. Dá se tedy předpokládat, že vychází ze zkušenosti tvůrců a většího množství naměřených dat.

3. Tecomat Foxtrot

Pro realizaci popsaných metod, byl vybrán průmyslově používaný programovatelný automat Tecomat Foxtrot od společnosti Teco a. s v provedení CP-1015. Hlavním důvodem pro tuto volbu pak byla zejména myšlenka práce, vyzkoušet funkčnost dostupných metod samoladících regulátorů na reálně využívaných systémech s přesahem do praxe.

PLC Tecomat Foxtrot (Obrázek 3) je kompaktní modulární řídicí a regulační systém pro malé a střední aplikace [8]. Disponuje šesti reléovými a dvěma analogovými výstupy a šesti přepínatelnými digitálními, respektive analogovými, vstupy. Analogové výstupy mají

Obrázek 3 - PLC Tecomat Foxtrot [8]

napěťový rozsah 0 – 10V. Digitální vstupy pak 0 – 24V. K navýšení vnitřní paměti dává Tecomat Foxtrot možnost využít slot na MMC/SD paměťovou kartu. Komunikaci s ostatními zařízeními zajišťuje rozhraní Ethernet a dva sériové kanály, z toho jeden pracující dle standardu RS-232 a druhý s pozicí pro volitelné submoduly. Pro připojení externích periferií slouží komunikační kanál s rozhraním CIB. [9]

3.1. Mosaic

Pro programování automatů nabízí společnost Teco a.s. nástroj Mosaic. Toto vývojové prostředí slouží nejenom pro návrh, ale i pro ladění programů na simulacích. Pomocí některých nástrojů pak umožňuje i sledovat a zasahovat do procesu pomocí vizualizací. Oficiální web společnosti pak uvádí, že: „Architektura prostředí i jeho jednotlivé nástroje ctí normu IEC61131-3.“ [10]

Norma IEC 61131-3 je třetí částí mezinárodního standardu, jež upravuje programování PLC. Obsahuje několik kapitol, které definují, jakým způsobem mají být vystavěny programovací nástroje pro automatizaci. Motivací pro její zavedení bylo usnadnění práce při nastavování PLC sjednocením podoby programovacích prostředí od různých výrobců a vývojářů.

Jednou z hlavních částí normy je výčet a popis programovacích jazyků. Konkrétně se jedná o textové jazyky: Structured text (ST), Insturction list (IL) a grafické jazyky: Function block diagram (FBD), Ladder diagram (LD), Sequenital function chart (SFC). Program Mosaic přitom využívá všechny jmenované kromě jazyka SFC. Místo něj zařazuje do nabídky programovacích jazyků ve smyslu IEC 61131-3 nenormovaný jazyk Continous flow chart (CFC), který je obdobou jazyka FBD s volnějšími zásadami pro kreslení blokových schémat [11].

3.1.1. Strukturovaný text (Structured text)

Vyšší programovací jazyk ST syntakticky připomíná jazyky Pascal nebo C, ze kterých vychází. Jedná se o jazyk objektově orientovaný, který využívá např. základní podmínky typu IF – THEN nebo umožňuje vytvořit sekvenci pomocí příkazu CASE. Vhodný je zejména při používání složitějších operací, matematických výrazů a rovnic.

Ze všech jmenovaných jazyků je to právě ST, jenž po zavedení normy zaznamenal nejvyšší nárůst zařazování do programovacích nástrojů. Jednou z jeho dalších výhod je také podobnost s běžně využívanými programovacími jazyky, ke kterým mají dnešní studenti vysokých škol často mnohem blíž než například k základům elektrických obvodů, ze kterých vychází jazyk LD [12].

3.1.2. Jazyk seznamu instrukcí (Instruction list)

Textový jazyk IL je nižším programovacím jazykem a oproti grafickým programovacím jazykům dokáže pracovat rychleji, díky jednodušší syntaxi. Je také velmi úsporný, co se týče velikosti kódu, a šetří proto místo v paměti PLC. Svou stavbou je podobný programovacímu jazyku assembler (ASM).

Navzdory svým výhodám je IL méně perspektivním jazykem. Výkon a kapacita paměti současných PLC již ve většině případů přesahuje požadavky uživatele a úspora IL proto není tak výrazným benefitem. [12]

3.1.3. Jayzk liniových schémat (Ladder diagram)

LD je grafický programovací jazyk, který je pravděpodobně nejvíce využívaným jazykem v PLC programování vůbec. Byl vyvinut tak aby nahradil relé obvody s pevnými vodiči [12]. Po vizuální stránce připomíná do série zapojené řídící obvody se spínacími a rozpínacími kontakty.

Jeho hlavní výhodou je jednoduchost, díky níž se jedná o jazyk přístupný i pro uživatele bez zkušeností s programováním, pouze se základními znalostmi s elektrickými obvody. Je ideálním nástrojem na jednoduché aplikace, které pracují pouze s binárními vstupy a výstupy, například pneumatické motory a jejich koncové snímače.

3.1.4. Jazyk blokových schémat (Function block diagram)

Dalším hojně využívaným grafickým programovacím jazykem je FBD. Ačkoli nárůst jeho implementace není tak velký jako např. u ST, stále se pravděpodobně jedná o druhý nejpoužívanější jazyk [12]. Jeho používání sebou nese výhody podobné jako u LD, nicméně nabízí uživateli širší paletu možností. Taktéž je to jeden z programů, u kterých je práce s nimi velmi intuitivní.

Vzhledově připomíná LD nicméně obsahuje místo spínacích kontaktů obdélníkové bloky, pracující buď jako funkce z knihovny nebo jako uživatelsky definované programové organizační jednotky.

3.1.5. Jazyk CFC (Continous flow chart)

Posledním jazykem z nabídky prostředí Mosaic je grafický jazyk CFC. Jedná se o jazyk podobný FBD. Oproti normovaným jazykům má ovšem volnější zásady pro kreslení blokových schémat [11]. Programátor tak má velkou volnost při vytváření blokových schémat a může volit strukturu programu dle vlastního uvážení. Programování v CFC je tak uživatelsky příjemnější a snadnější. Ačkoli není kodifikován normou IEC 61131-3, při implementaci často sdílí společné prvky s normovanými programovacími jazyky (datové typy proměnných používání POU apod.) [11].

Při tvorbě programů však CFC vzhledem k volnému umísťování bloků do prostoru přenáší zodpovědnost za přehlednost programu na tvůrce. Při nedodržování zásad pro přehlednou skladbu bloků programu, tak může dojít k vytvoření obtížně čitelných schémat.

Všechny jazyky dle normy IEC 61131-3 sdílejí datové typy proměnných i způsob volání funkcí. Díky tomu je možné je kombinovat a skrze vstupy a výstupy propojit do jednoho funkčního celku.

4. Postup při realizaci regulátorů

Realizace regulátorů probíhala v prostředí Mosaic. Z nabídky jazyků definovaných v normě IEC 61131–3 byl pro programování vybrán jazyk ST. Hlavním důvodem pro jeho zvolení byla možnost práce se složitějšími matematickými operacemi. V jazyku ST je také velká šíře možností pro vytváření složitějších sekvencí, které se dají využít při implementaci popsaných metod.

Nahrání programu do PLC bylo realizováno skrze školní síť pracující dle protokolu TCP/IP. K propojení regulátoru se sítí byl použit kabel s konektorem RJ – 45. Pro komunikaci a připojení k laboratorním soustavám byly použity analogové vstupy a výstupy na PLC.

Společným prvkem všech třech naprogramovaných regulátorů je úvodní deklarace globálních proměnných. Zde jsou propojeny fyzické vstupy a výstupy na PLC s virtuálními proměnnými. Všechny testované soustavy byly testovány v zapojení SISO. Proto deklarace obsahuje vždy jen regulovanou veličinu *y* (vstup do regulátoru) a akční veličiny *u* (výstup z regulátoru). Na obrázku (Obrázek 4) je příklad deklarace pro zapojení soustavy Teplovzdušný model popsané dále v práci v kapitole (5.1.1.).

```
VAR_GLOBAL
//vetrak + prutokomer
//(*
prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //výstup z PLC
   u AT ventilator : REAL; //akční zásah
  y AT prutokomer; //výstup soustavy
//*)
```

Obrázek 4 - propojeni fyzických vstupů a výstupů s proměnnými

4.1. Realizace regulátoru dle Maršíka

Jako první byl pro testování naprogramován regulátor dle J. Maršíka. Jelikož vyniká svou jednoduchostí, byla jeho implementace do PLC dobrá pro osvojení práce s programovacím prostředím Mosaic. Velkou pomocí při programování byl Maršíkův závěrečný příklad algoritmu popsaný v jednotlivých krocích. Navržená struktura byla použita jako jádro programu.

Při vývoji byl algoritmus průběžně testován na reálných laboratorních soustavách. Při práci tak byla k dispozici stálá zpětná vazba o funkčnosti programu. Díky tomu bylo možné lépe přizpůsobit zkoumanou metodu praktickému použití.

Pro zobrazení průběhu regulované veličiny *y*, žádané hodnoty *w* a akční veličiny *u* byl vytvořen graf pomocí nástroje GraphMaker jež je součástí programovacího prostředí Mosaic.

4.1.1. Popis programu

Na začátku programu je pomocí podmínkových funkcí IF – THEN omezen rozsah zadávané žádané hodnoty *w*. Uživatel může hodnotu měnit pomocí zadávacího pole vytvořeného nástrojem WebMaker. Hodnota je automaticky omezena na maximální možný

```
IF w > 10.0 THEN
w:=10.0;
END_IF;

IF w < 0.0 THEN
w:= 0.0;
END_IF;
```

Obrázek 5 - omezení hodnoty w

rozsah analogového vstupu do PLC (0 – 10 V) (Obrázek 5) Menší hodnoty jsou nastaveny na hodnotu 0 a větší na hodnotu 10. Následují kroky realizují algoritmus popsaný v [3].

Nejprve je vypočtena aktuální hodnota regulační odchylky *e* společně s její druhou mocninou a rozdílem oproti předchozí hodnotě. Naměřené hodnoty *e* jsou ukládány do pole *„e_pole[i]“*. Každá nová hodnota je pak uložena na první místo (*e_pole[1]*). Starší hodnoty jsou posunuty o jednu pozici dál. Dosud popsané operace se provádějí neustále okamžitě po spuštění programu.

Následující část je rozdělena na dvě větve podmínkovou funkcí IF - THEN. Tato podmínka kontroluje tři hodnoty. Zaprvé je zjištěno, zda je booleovská proměnná *„init“* na hodnotě TRUE. Její hodnota při úvodní deklaraci nastavena na hodnotu FALSE a na hodnotu TRUE se dostává po prvním provedení programu. Poté je zkontrolováno, zde není na hodnotě TRUE proměnné *„stop“* typu BOOL. Tato proměnná je navázána na tlačítko „STOP“, které je součástí ovládacího panelu programu a slouží pro jeho nouzové zastavení. Nakonec dojde ke kontrole podmínky pro adaptaci parametrů regulátoru.

Při prvním spuštění programu nebo při splnění podmínky pro přepočítání parametrů regulátoru přechází program do fáze adaptace (Obrázek 6).

```
//III. Casova konstanta
T             := 2.0*3.14*sqrt((v2)/(a2));

//IV. Rozptyl reg. odchylky
sigma1        := (e_2+(3.0*T*sigma2))/(1.0+(3.0*T));

//V. e, a, v
e1            := (e_2+(T*e2))/(1.0+T);

v1            := ((e_delta*e_delta)+(T*v2))/(1.0+T);

a1            := ((e_delta*(e_pole[1]-e_pole[3]))+(T*a2))/(1.0+T);

//VI. Filtrovana zadana hodnota
w_filtr1      := (w+(T*w_filtr2))/(1.0+T);

//VII. index kappa
kappa         := v1/sqrt(e1*a1);

//VIII. Spolecne zesileni reg. alfa
alfa_delta      := (alfa2*((kappa_zad/kappa)-1.0))/T;

//IX. Koeficient proporcialni vetve
beta          := sqrt((e1)/(v1));

//X. Koeficient diferencni vetve
gamma         := sqrt((e1)/(a1));

//XI. Akcni velicina regulatoru u

suma          := alfa_delta*e_pole[1] + suma2;
u             := ((alfa_delta*gamma*e_delta)+(alfa_delta*beta*e_pole[1])
                 + suma)/prepocet;
```

Obrázek 6 - adaptační algoritmus dle J. Maršíka

Uvedené rovnice jsou sestaveny dle rovnice v příkladu algoritmu z kapitoly (číslo kapitoly). Pokud již program zadaptoval regulátor a je v toleranci předepsané podmínky pro adaptaci pokračuje do fáze kde je pouze rovnice pro výpočet *u* odpovídající rovnici pro řízení z obrázku (Obrázek 6).

V obou případech je pak součástí kódu ještě omezovací podmínková funkce, která zajišťuje, aby hodnota akční veličiny *u* nepřesáhla maximální napěťový rozsah analogového výstupu z regulátoru tzn. 0 – 10 V. Tato funkce funguje obdobně jako (Obrázek 5).

Poslední částí programu (Obrázek 7) je potom uložení hodnot z předchozího kroku regulátoru do paměti PLC, aby mohly být provedeny výpočty potřebné pro adaptaci. Pod touto paměťovou funkcí je ještě podmínka zajišťující nulový výstup z regulátoru při stisknutí tlačítka STOP.

```
//Pamet do nasledujiciho cyklu

e_pole[5]     := e_pole[4];
e_pole[4]     := e_pole[3];
e_pole[3]     := e_pole[2];
e_pole[2]     := e_pole[1];

e2            := e1;
a2            := a1;
v2            := v1;
sigma2        := sigma1;
w_filtr2      := w_filtr1;
suma2         := suma;
alfa2         := alfa;
IF stop = TRUE THEN
u := 0.0;
END_IF;
```

Obrázek 7 - paměť do dalšího cyklu programu

4.2. Realizace regulátoru dle J. Berner

Při realizaci regulátoru dle J. Berner byl vyvíjený program neustále průběžně testován na reálných laboratorních úlohách, aby byla ověřena správná funkčnost jednotlivých segmentů. Implementace tohoto samoladícího regulátoru do knihovny Mosaic navíc byla motivována a podpořena firmou Teco a.s. I proto byl při vývoji brán zřetel především na co nejrobustnější řízení a co možná nejvíce univerzální identifikační funkci. Práce na regulátoru probíhala ve spolupráci s Bc. Alžbětou Hornychovou.

Během testování vyvíjeného programu bylo učiněno několik změn oproti metodě identifikace popsané v [4]. Ke změnám bylo přistupováno zejména v případech, kdy implementace navrhnutých postupů způsobovala výskyt nežádoucího chování reálných testovaných soustav. V jiných případech se pak jednalo pouze o optimalizaci úlohy pro lepší funkčnost na laboratorních úlohách. Konkrétně se pak největší zásahy týkaly nastavování poloh a přepínání relé.

První změnou byla úprava pásma hystereze. V práci [4] platí:

$$h_1 = h_2 = h = 2n_0, \tag{44}$$

Symetrické hranice hystereze však v kombinaci s použitým asymetrickým relé (jež navíc musí splňovat určité požadavky na stupeň asymetrie) občas vyvolaly stav, ve kterém se při poloze bližší pracovnímu bodu stávalo, že regulovaná veličina jen velmi obtížně přecházela přes hranici pro přepnutí. Problémy nastaly zejména v případě, kdy byl signál z PLC zatížen šumem a soustava tak okolo hranice několikrát překmitla. Z tohoto důvodu byla přijata změna:

$$h_1 \neq h_2, \tag{45}$$

Následně byly hranice pásma hystereze upraveny poměrně vzhledem k asymetrii relé:

$$h_1 = h = 2n_0, \tag{46}$$

$$h_2 = 1.4h_1, \tag{47}$$

Další změna se týká pravidel pro úpravu horní a dolní polohy relé. Zatímco v práci [4] je popsán algoritmus, pomocí kterého jsou parametry přepočítávány skrze po každé periodě přepočítávaný koeficient *„ratio“*, pro potřeby provedeného testování byl pro úpravu parametrů vytvořen soubor pravidel. Při nasazení algoritmu dle J. Berner totiž u rychlejších soustav docházelo k dramatickým zásahům do zejména dolních poloh relé. Mohlo se stát, že při dalším cyklu byla hodnota přepočtena tak, že regulovaná veličina nebyla schopna překročit pásmo hystereze a tím vyvolat přepnutí relé. Vytvořená pravidla vycházejí ze zkušenosti získané při práci na laboratorních úlohách. Stejně jako Algoritmus dle J. Berner jsou i použitá pravidla založena na hodnotách měřených při přepínání relé. Měřena je minimální (y_{min}), respektive maximální (y_{max}), hodnota regulované veličiny a čas, který je relé v poloze horní t_{on} a v poloze dolní t_{off}. Rozhodovacím parametrem, jenž určuje, zda a jak polohy relé přepočítat je hodnota normalizovaného dopravního zpoždění τ. K jeho výpočtu použijeme vztahy popsané v kapitole (číslo kapitoly ***), které k určení hodnoty τ využívají parametr stupně asymetrie relé γ a poměr časů ρ.

Pro použití vztahů z metody [4] jenž určují matematický model soustavy, je zapotřebí, aby hodnota τ byla kladná. Pokud tomu tak není, lze úpravou parametrů relé d_1 a d_2 docílit změny. Povaha změny pak závisí na naměřených hodnotách y_{min} a y_{max}. Pravidla pro úpravu poloh relé mají následující podobu:

Pokud $y_{min} > 1$ a $y_{max} > 1$ potom:

$$d_1 := u_0 + (d_1 - u_0) * 0.8$$
$$d_2 := u_0 + (d_2 - u_0) * 0.8$$

Pokud $1 > y_{min} > 0.5$ a $1 > y_{max} > 0.5$ potom:

$$d_1 := u_0 + (d_1 - u_0) * 0.9$$
$$d_2 := u_0 + (d_2 - u_0) * 0.9$$

Pokud $0.5 > y_{min} > 0.1$ a $0.5 > y_{max} > 0.1$ potom:

$$d_1 := u_0 + (d_1 - u_0) * 0.95$$
$$d_2 := u_0 + (d_2 - u_0) * 0.95$$

Pokud $y_{min} < 0.1$ a $y_{max} < 0.1$ potom:

$$d_1 := u_0 + (d_1 - u_0) * 1.1$$
$$d_2 := u_0 + (d_2 - u_0) * 1.1$$

Pokud $y_{min} < 0.1$ a $y_{max} < 0.1$ a navíc $d_1 = u_{max}$ potom:

$$d_2 := u_0 + (d_2 - u_0) * 0.9$$

Pokud $y_{min} < 0.1$ a $y_{max} < 0.1$ a navíc $d_2 = u_{min}$ potom:

$$d_1 := u_0 + (d_1 - u_0) * 0.9$$

Pokud pouze $y_{min} < 0.1$ potom:

$$d_1 := d_1 - 0.1$$
$$d_2 := d_2 - 0.1$$

Pokud pouze $y_{max} < 0.1$ potom:

$$d_1 := d_1 + 0.1$$
$$d_2 := d_2 + 0.1$$

Popsaná pravidla jsou aplikována do chvíle, než kontrolní výpočet hodnoty τ vyjde kladný. Testování pravidel pak ukázalo, že byla splněna základní premisa pro jejich použití a sice eliminace nežádoucích stavů způsobených nevhodně nastavenými parametry relé. Negativní vnesenou vlastností tohoto řešení je v některých případech delší čas potřebný pro dokončení úprav.

4.2.1. Popis programu

Základní strukturu programu tvoří tři fáze oddělené příkazem CASE OF. Program mezi nimi přechází změnou řídící proměnné *„faze“* typu INTEGER. Každá z fází pak má ještě svoji vnitřní strukturu členěnou obdobným způsobem. V programu jsou také využity některé funkční bloky z knihoven obsažených v prostředí Mosaic.

Nedílnou součástí programu je ovládací uživatelské rozhraní, vytvořené pomocí nástroje WebMaker, který je jedním z nástrojů programovacího prostředí Mosaic. Toto rozhraní pak slouží zejména pro nastavení hodnot potřebných pro funkci programu, ovládání programu a informování uživatele o tom, v jaké fázi se právě program nachází. Pomocí dalšího nástroje z nabídky Mosaic s názvem GraphMaker je vytvořen graf, který slouží pro zobrazení průběhu vstupů a výstupů z PLC a sběr dat.

Po nahrání programu do PLC a jeho spuštění přechází program automaticky do první fáze, ve které dochází k nastavení parametrů relé. Tato fáze je členěna pomocí příkazu CASE OF, jenž je řízen proměnnou *„stav init“*.

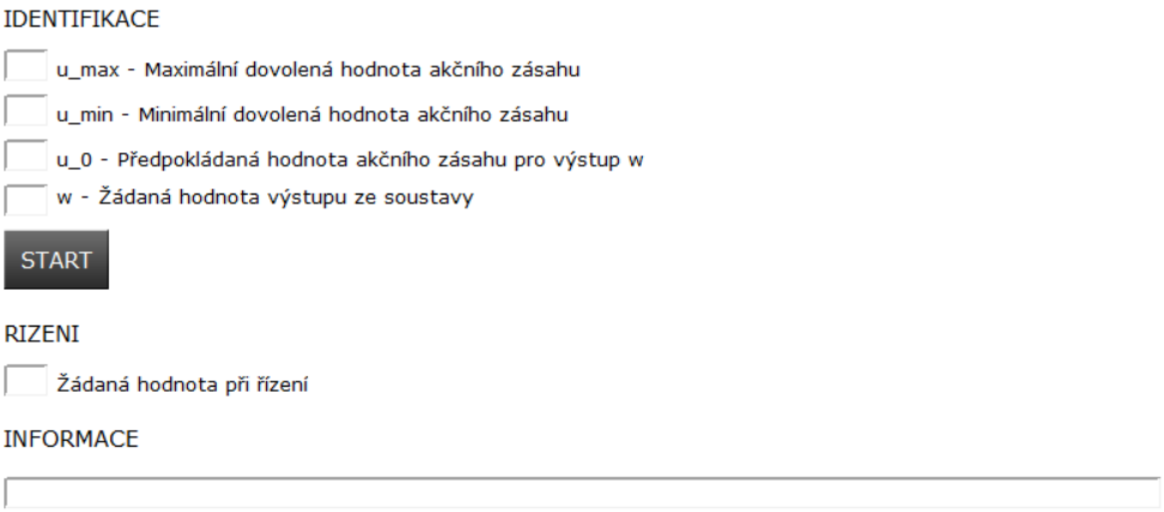
Výchozím stavem první fáze je podmínková funkce IF (Obrázek 8), ve které program čeká na změnu booleovské proměnné *„start“* na hodnotu 1.

```
CASE stav_init OF

  0:  // cekani na stisknuti tlacitka START
  IF start THEN
     stav_init := 1;
  END_IF;
```

Obrázek 8 - čekání na stisknutí tlačítka START

Tu může uživatel nastavit pomocí tlačítka „START“ na panelu (Obrázek 9) z nástroje WebMaker. Před spuštěním programu je však třeba nejprve v ovládacím panelu zadat všechny parametry potřebné pro správnou funkci programu.



Obrázek 9 - ovládací panel regulátoru dle J. Berner

Konkrétně se jedná o žádanou hodnotu (*w)*, tedy hodnotu, na které chceme, držet výstup (*y)* z řízené soustavy pomocí regulátoru. Dále počáteční akční zásah (u_0), kterýžto je odhadem vstupu do soustavy, při jehož nastavení se soustava ustálí někde v okolí žádané hodnoty. Posledními dvěma nastavitelnými parametry jsou maximální horní (u_{max}), respektive dolní (u_{min}), hranice akčního zásahu. Tyto parametry udávají rozsah akční veličiny *u*, ve kterém může soustava bezpečně pracovat.

Zadání parametrů *w*, u_{max} a u_{min} je nutnou podmínkou pro chod programu. Hodnota u_0 je z podstaty problematická, jelikož pro její správné nastavení je třeba předchozí znalost soustavy a zkušenost uživatele. Pro její určení je tak v programu vytvořena paralelní větev, při které je po spuštění algoritmu nejprve naměřena přibližná velikost akčního zásahu odpovídajícího zadané žádané hodnotě *w*. Tato hodnota je následně použita jako u_0. Vzhledem k tomu, že při měření a srovnávání regulátorů nebyla tato funkce použita, není dále popisována.

Dalším krokem inicializační fáze je kontrola zadaných parametrů (Obrázek 10).

```
1: //kontrola zadaných údajů

IF u_mins > u_maxs THEN
   hlaska := 'Hodnota u_maxs musi byt vetsi nez u_mins.';
   start:=FALSE;
   stav_init := 0;
   dlouhy_start := FALSE;
   ELSE
   IF dlouhy_start = FALSE THEN
      IF u_0 >= u_maxs OR u_0 <= u_mins THEN
         hlaska := 'Hodnota u_0 musi lezet mezi u_maxs a u_mins.';
         start:=FALSE;
         stav_init := 0;
         ELSE
         stav_init := 2;
         hlaska := '';
      END_IF;
      ELSE
      stav_init := 2;
      hlaska := '';
   END_IF;
```

Obrázek 10 - kontrola vložených parametrů

V této části se ověřuje, zda parametry zadané uživatelem neodporují své logice. Konkrétně musí platit, že hodnota u_{max} musí být větší než u_{min} a hodnota u_0 pak musí ležet někde v tomto rozsahu. V případě nedodržení některé z podmínek se program vrací zpět do předchozího stavu a vypíše na textový řádek v ovládacím panelu informaci o tom, jakým způsobem zadané parametry odporují svému smyslu. Uživatel tak dostane prostor pro úpravu parametrů do potřebné podoby a po opravě může program znovu spustit stisknutím tlačítka „START“. V případě kladného výsledku kontroly přechází program do dalšího kroku, kterým je čekání na ustálení regulované veličiny (Obrázek 11).

```
     u := u_0;
     hlaska:= 'Ceka se na ustaleny stav';

//----------------------------------------
   CASE klid OF
0:
     IF timer1.Q = FALSE THEN
        pocitadlo := pocitadlo +1.0;
        y_suma_k := y_suma_k + y;
     END_IF;

     timer1(IN:= casovac, PT :=T#20s);
     IF timer1.Q THEN
        y1_k := y_suma_k/pocitadlo;
        pocitadlo := 0.0;
        y_suma_k := 0.0;
        klid := 1;
        casovac := FALSE;
     END_IF;

     IF prepnuti THEN
        casovac := TRUE;
        prepnuti := FALSE;
        ELSE
            prepnuti := TRUE;
     END_IF;
```

Obrázek 11 - čekání na ustálení y

K určení ustáleného stavu slouží úsek programu, jenž je rozdělen na několik fází opět pomocí funkce CASE OF, řídící veličinou *„klid“*. Na začátku je nastavena hodnota akční veličiny na výchozí uživatelem zadaný odhad u_0. Poté přechází program do první fáze ustalování, kdy je spuštěn funkční blok *„timer1“*, z knihovny StdLib v2.1. Jedná se o časovač, který po svém spuštění čeká po dobu nastavenou při svém volání. Po uplynutí této doby změní svou výstupní booleovskou proměnnou *„timer1.Q“* na hodnotu TRUE. Než se tak stane je průběžně při každém čtení programu v PLC počítána suma *„y_suma_k“* z hodnot regulované veličiny *y*. Zároveň je pomocí proměnné *„pocitadlo“* zaznamenáván počet přičtených hodnot. V momentě, kdy časovač doběhne, je podělením sumy hodnot *y* počtem vzorků zjištěna střední hodnota regulované veličiny na měřeném úseku. Vypočtená hodnota je uložena do proměnné *„y1_k“* a hodnoty proměnných *„y_suma_k“* a *„pocitadlo“* jsou vynulovány. Nakonec program přechází do další fáze, která funguje na stejném principu jako fáze první. Po uložení nové střední hodnoty regulované veličiny do proměnné *„y2_k“*, se cyklus s časovačem opakuje ještě jednou v poslední měřící fázi, jejíž výstupem je opět střední hodnota měřeného úseku uložená do proměnné *„y3_k“*. Po skončení měření přechází program do kontrolní fáze (Obrázek 12), ve které se rozhodne, zda je výstup ze soustavy ustálen.

```
3:

    IF (y1_k > y2_k AND y2_k > y3_k) OR (y1_k < y2_k AND y2_k < y3_k) THEN
    y1_k := y2_k;
    y2_k := y3_k;
    klid := 2;
    ELSE
    klid := 4;
    casovac := FALSE;
    END_IF;
```

Obrázek 12 - podmínka pro ustálení

Rozhodování je realizováno pomocí podmínkové funkce IF – THEN. V případě, že pro střední hodnoty třech po sobě jdoucích úseků platí $y1\_k < y2\_k < y3\_k$ nebo $y1\_k > y2\_k > y3\_k$ tzn. regulovaná veličina stále stoupá nebo klesá, program uloží do první ($y1\_k$) měřené střední hodnoty hodnotu druhou ($y2\_k$) a do druhé ($y2\_k$) hodnotu třetí ($y3\_k$). Následně se vrátí o jednu fázi zpátky, opakuje měření a jeho výsledek vloží do poslední měřené proměnné

y3_k. Tato sekvence je opakována až do chvíle, kdy podmínka stoupání – klesání není splněna a výstup ze soustavy tudíž můžeme prohlásit za ustálený.

Po ustálení regulované veličiny přechází program do dalšího kroku, kterým je měření šumu (Obrázek 13)

```
4:
     pocet := pocet +1;
     suma_y := y +suma_y;

         IF y > max_y THEN
             max_y := y;
             END_IF;
             IF y < min_y THEN
             min_y := y;
             END_IF;
             pocet := pocet +1;
             suma_y := y +suma_y;

      static_tim4(IN:=casovac, PT :=T#10s);
      hlaska:= 'Meri se sum';
      IF prepnuti THEN
        casovac := TRUE;
        prepnuti := FALSE;
        ELSE
            prepnuti := TRUE;
     END_IF;
```

Obrázek 13 - měření šumu

Obdobně jako u předchozího kroku je zavolán časovač z knihovny StdLib v2.1 a po dobu definovanou při volání je do paměti zaznamenávána nejvyšší, respektive nejnižší, hodnota regulované veličiny *y*. Zároveň je měřena její průměrná hodnota obdobně jako u fáze ustalování podílem sumy *y* a počtu vzorků. Po doběhnutí časovače je z naměřených dat vypočtena hystereze *„h1“*, *„h2“* a také maximální, respektive minimální, dovolená odchylka *„y_maxdev“* a *„y_mindev* (Obrázek 14). Naměřená průměrná hodnota *y* je potom použita jako výchozí pracovní bod pro další chod programu.

```
IF static_tim4.Q THEN
  n0  := (max_y - min_y)/2.0;
   h   := a*n0;
   h1 := h*1.4;
   h2:=h;
   y_0 := (max_y + min_y)/2.0;
   y_mindev := ny * h;
   y_maxdev := 3.0*y_mindev;
   stav_init:=3;
   casovac := FALSE;

   IF dlouhy_start THEN
   w:=w;
   ELSE
   w:= suma_y/INT_TO_REAL(pocet);
   END_IF;
END_IF;
```

Obrázek 14 - měření průměrné hodnoty a hystereze

Následně přechází program do dalších kroků řízených proměnnou „*stav_init*“. V nich dochází k nastavení výchozích hodnot pro relé. Nejprve je jako hodnota horní polohy relé „*d1*“ nastavena uživatelem zadaná maximální hodnota akční veličiny. Stejným způsobem je pak nastavena i dolní poloha relé „*d2*“ z minimální povolené hodnoty akční veličiny (Obrázek 15).

```
3: //nastavení hranic relé, výpočet gamy

   d1:= u_maxs;
   d2:= u_mins;
   static_delta := 1.5*n0;
   gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
   stav_init:=4;
```

Obrázek 15 - prvotní nastavení relé

Pro získání počátečního odhadu obou hranic relé je dle metodiky popsané v [4] použit v čase exponenciálně narůstající akční zásah (Obrázek 16). Ten je zvyšován až do chvíle, kdy nastane jedna ze dvou situací. Buď hodnota akční veličiny dosáhne hodnoty *w + y_maxdev*

```
5: //exponencielní průběh
  cas_ted := GetRTC();
  cas_init := SUB_DT_DT(cas_ted,cas_start);
  k_exp:=1/3;
  IF u < u_maxs THEN
     IF y>=(y_maxdev+w) THEN
        stav_init :=6;
        d_p :=u;
        d_2p:=u_0-((d_p-u_0)*0.7);
        IF u_mins < d_2p THEN
        d2:= d_2p;
        END_IF;
        u:=d2;
        d1 := d_p;
        gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
     ELSE
         u:= EXP(k_exp*TIME_TO_REAL(cas_init)/10000.0)-1.0+u_0;
     END_IF;
  ELSE
      d_2p:=u_0-((d1-u_0)*0.7);
      IF u_mins < d_2p THEN
        d2:= d_2p;
      END_IF;
     stav_init :=6;
     u:=d2;
     spadova := 0;
     t_2_p:=GetRTC();
     t_1_p:=GetRTC();
     gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
  END_IF;
```

Obrázek 16 - exponenciální průběh

nebo uživatelem zadané maximální hranice akční veličiny *u_maxs*. V okamžiku, kdy k tomu dojde, je aktuální hodnota akční veličiny uložena jako horní hranice relé *d1*. Hodnota dolní hranice relé *d2* je pak vypočtena dle vztahu:

$$d_2 = u_0 - ((d_1 - u_0) * 0{,}7) \tag{48}$$

Před přechodem do dalšího kroku je ještě vypočtena hodnota stupně asymetrie relé γ v programu označená jako *„gama“*. Jedná se o poměr horní a dolní hranice relé, který je potřebný pro výpočet dalších parametrů modelu soustavy.

Jakmile jsou výchozí polohy relé nastaveny přechází program do fáze, kde dochází k jejich dodatečné úpravě. Tyto úpravy jsou nutné k zajištění dat použitelných pro následné výpočty určující matematický model soustavy. Pomocí přepínaní relé je soustava uvedena do stavu kmitání. K přepnutí relé pak dochází ve chvíli, kdy regulovaná veličina *y* dosáhne horní (*w + h1*), respektive dolní (*w – h2)* hranice pásma hystereze (Obrázek 17).

```
6: //kmitani rele a nastaveni jeho hodnot
  IF nabehova < 2 THEN
     IF u <> d2 THEN
       IF y >= w+h1 THEN
          u:= d2;

         t_on_p := SUB_DT_DT(t_2_p,t_1_p);
         t_1_p  := GetRTC();


          spadova := spadova +1;
       END_IF;
     END_IF;
     IF u <> d1 THEN
        IF y <= w-h2 THEN
           u:= d1;

           t_off_p := SUB_DT_DT(t_1_p,t_2_p);
           t_2_p   := GetRTC();
           nabehova := nabehova +1;
        END_IF;
     END_IF;
```

Obrázek 17 - přepínaní relé

V průběhu kmitání je potom průběžně měřena doba, po kterou je relé v horní poloze *„t_on_p“*, respektive doba, po kterou je relé v poloze dolní *„t_off_p“*. Zároveň je zaznamenávána maximální a minimální hodnota regulované veličiny *y* (Obrázek 18).

```
IF spadova = 1 AND nabehova = 1 THEN

   max_y_dev:= MAX(max_y_dev,ABS(y-w));
END_IF;

IF spadova = 0 AND nabehova = 1 THEN
   min_y_dev := MAX(min_y_dev,ABS(w-y));

END_IF;
```

Obrázek 18 - počítání maxima a minima y

Po každé periodě je z naměřených dat proveden kontrolní výpočet parametru γ *(„ro_p“)*, ze kterého je následně vypočtena kontrolní hodnota normalizovaného dopravního zpoždění τ *(„tau_p“).* Pokud platí *tau_p < 0,05* dojde k aplikaci pravidel popsaných v kapitole (4.2). Na obrázku č. (Obrázek 19) je část kódu aplikující poslední z uvedených pravidel. Ostatní

```
1:
 IF (max_y_dev-h1)>(min_y_dev-h2) THEN
     d1 := u_0 +(d1 - u_0)* 0.9;
     hlaskal:= 'chyba5';
     ELSE
     d2:= u_0 +(d2 - u_0)* 0.9;
     hlaskal:= 'chyba6';
  END_IF;
  gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
END_CASE;
```

Obrázek 19 - jedno z pravidel pro nastavené relé

úpravy jsou realizovány obdobným způsobem skrze podmínkovou funkci IF – THEN. Algoritmus provede vždy jen jednu z úprav. Po provedení jakékoli z nich je přepočítána proměnná *gama*. Zároveň jsou vynulovány veškeré proměnné a program opakuje měření s novými hodnotami parametrů relé. Tato sekvence pokračuje až do chvíle, kdy přestane být splněna podmínky pro úpravu.

```
static_tim2(IN:=ABS(y-y_0)< static_delta, PT :=T#20s);
IF ABS(y-y_0) >= static_delta THEN
   y_0 :=y;
END_IF;
```

Obrázek 20 - kontrola nežádoucího ustálení

Po celou dobu přepínání relé probíhá kontrola, zda se soustava nedostala do stavu, kdy po změně polohy relé není regulovaná veličina *y* schopna opustit pásmo hystereze (Obrázek 20). Toto opatření bylo zavedeno po zkušenostech při vývoji programu a testování na reálných soustavách.

Samotné programové úpravy parametrů relé jsou navrženy tak, aby k takovému stavu nedošlo. Uživatel však může nevhodným nastavením vstupních parametrů *w, u_maxs* a *u_mins* tento nežádoucí stav nechtěně vyvolat. Ve chvíli, kdy se regulovaná veličina po přepnutí relé příliš dlouho drží v pásmu hystereze, je měření zastaveno. Na ovládacím panelu je zobrazena definice chyby s doplňujícími pokyny pro uživatele a program se vrací na začátek, kde čeká na úpravu parametrů dle poskytnutých informací a opětovné spuštění (Obrázek 21).

```
IF static_tim2.Q THEN                                                    //při doběhnutí

   IF u = d1 THEN
      hlaska := 'Nelze seridit na zadanou hodnotu je treba snizit w nebo zvysit u_maxs.';
      u := u_mins;
      stav_init :=0;
      faze:=0;
      ustaleni :=0;
      start := FALSE;
   END_IF;

   IF u = d2 THEN
      hlaska := 'Nelze seridit na zadanou hodnotu je treba zvysit w nebo snizit u_mins.';
      u := u_mins;
      stav_init :=0;
      faze:=0;
      ustaleni :=0;
      start := FALSE;
   END_IF;

END_IF;
```

Obrázek 21 – výstup z kontroly

S vhodně nastavenými parametry relé přechází program do druhé ze třech základních fází – identifikace.

Fáze identifikace je dále členěna do několika dalších kroků příkazem CASE OF pomocí řídící proměnné *„stav“*. V této fázi dochází ke kmitání relé stejným způsobem jako při nastavování jeho hodnot. Opět je v průběhu kmitání relé měřen čas, jaký soustava stráví při dolní (*„t_off“)*, respektive při horní (*„t_on“*), nastavené poloze relé. Navíc je celý průběh vzorkován a jednotlivé datové body jsou ukládány do pole *„buffers[n,m]“*. Toto pole má tři sloupce, které slouží jako paměťové bloky pro ukládaní aktuální hodnoty *y*. Vzorkování probíhá dělením proměnné *„cas“*, což je doba, která uplyne od začátku periody, proměnnou *„iT“* – vzorkovací periodou numerické integrace.

```
buf_x := REAL_TO_UINT(CEIL(TIME_TO_REAL(cas)*0.001/iT))
IF buf_x < BUF_MAX_LEN THEN
   IF buf_x <> buf_x_old THEN  //čekání na určenou chví
      buf_x_old := buf_x;
      CASE buf_i OF
      0:
        IF buf_x <= buf_len[1] THEN
           difa:= difa + ABS(buffers[1,buf_x]-y);

        END_IF;
        IF buf_x <= buf_len[2] THEN
           difb:= difb + ABS(buffers[2,buf_x]-y);
        END_IF;
      1:
        IF buf_x <= buf_len[0] THEN
           difa:= difa + ABS(buffers[0,buf_x]-y);
        END_IF;
        IF buf_x <= buf_len[2] THEN
           difb:= difb +ABS(buffers[2,buf_x]-y);
        END_IF;
      2:
        IF buf_x <= buf_len[0] THEN
           difa:= difa +ABS(buffers[0,buf_x]-y);
        END_IF;
        IF buf_x <= buf_len[1] THEN
           difb:= difb +ABS(buffers[1,buf_x]-y);
        END_IF;
      END_CASE;
   END_IF;
   buffers[buf_i,buf_x] :=y; //uložit vzorek
   buf_len[buf_i] := buf_x;  //ulozeni delky bufferu
END_IF;
```

Obrázek 22 – paměť pro ukládání vzorků

Výsledek tohoto dělení je zaokrouhlen na celé číslo a výsledná hodnota je uložena do proměnné *„buf_x“*. Ta pak slouží jako index vzorku v paměti. Při prvním průchodu touto fází jsou vzorky ukládány do sloupce číslo 1 v poli *buffers[1, buf_x]*. Jakmile doběhne perioda relé, dojde k uložení aktuální hodnoty *buf_x* do pole *„buf_len[i]“,* které uchovává hodnotu množství vzorků v každém ze sloupců. Následně je proměnná *buf_x* vynulována a číslo sloupce pole je navýšeno o jedna. V další periodě jsou pak vzorky ukládány do dalšího sloupce (*buffers[2, buf_x]*). Tento postup zapisování je opakován stále dokola, přičemž pole *buffers[n,m]* uchovává hodnoty ze tří posledních period. Každá další započatá perioda přepisuje data z nejstaršího uloženého sloupce. Před tím, než je nový vzorek zapsán do pole, je ukládaná hodnota *y* porovnána s hodnotami se stejným indexem *buf_x* aktuálně uloženými ve dvou předchozích sloupcích. Z rozdílů *y* a uložených hodnot je pro oba sloupce zvlášť počítána suma, jejíž hodnota je ukládána do proměnné *„difa“,* respektive *„difb“*. Obě sumy jsou při přechodu do nové periody poděleny odpovídajícími délkami sloupců (*buf_len[i]*),

čímž jsou získány průměrné odchylky regulované veličiny *y* (*„dif01“, „dif02“, „dif12“)* ve stejném okamžiku naměřených period.

```
buf_i := buf_i +1; //posunutí polohy v paměti
      IF buf_i > 2 THEN
         buf_i:=0;
      END_IF;
buf_len[buf_i]:=0; //---------------------------

CASE buf_i OF     //vypocet prumernych odchylek
     0:
       dif01 := difa/UINT_TO_REAL(buf_len[1]);
       dif02 := difb/UINT_TO_REAL(buf_len[2]);
     1:
       dif01 := difa/UINT_TO_REAL(buf_len[0]);
       dif12 := difb/UINT_TO_REAL(buf_len[2]);
     2:
       dif02 := difa/UINT_TO_REAL(buf_len[0]);
       dif12 := difb/UINT_TO_REAL(buf_len[1]);
END_CASE;
```

Obrázek 23 – výpočet diferencí ze vzorků jednotlivých period

Popsaný postup je opakován až do chvíle, kdy dojde ke splnění ukončovací podmínky. Ta porovnává naměřené průměrné odchylky s maximální dovolenou chybou, která je uložená v proměnné *„chyba“* a pro potřeby práce má hodnotu 0,1. Tato podmínka má za účel zkontrolovat, zda jsou naměřená data získaná během ustáleného kmitání a bez nahodilých skokových poruch. Pokud ano, program může překročit do své poslední identifikační fáze.

```
IF dif01 < chyba AND  dif02 < chyba AND dif12 < chyba
stav:=2;
RETURN;
END_IF;
```

Obrázek 24 – podmínka pro opuštění měřicí fáze

V této části již má program uloženy všechny potřebné hodnoty a zahajuje výpočetní fázi. Nejprve je z parametrů *„iT“* a *buf_len[buf_i]* zpětně určen čas periody *„t_perioda“* a časy, ve kterých bylo relé v poloze horní (*t_on)*, respektive dolní (*t_off)*. Z nich jsou pak vypočteny parametry γ (*ro)* a τ (*tau)*.

```
2: //určení časů, výpočet Half-period ratio a normalized time delay
  t_perioda := REAL_TO_TIME(UINT_TO_REAL(buf_len[buf_i])*iT*1000.0);
  t_on := REAL_TO_TIME(UINT_TO_REAL(buf_u_switch[buf_i])*iT*1000.0);
  t_off := t_perioda - t_on;
  ro := MAX(TIME_TO_REAL(t_on),TIME_TO_REAL(t_off))/MIN(TIME_TO_REAL(t_on),TIME_TO_REAL(t_off));
  tau := (gama - ro)/((gama-1.0)*((0.35*ro)+0.65));
```

Obrázek 25 – výpočet časových parametrů

Následně je překontrolováno, zda výsledné tau, neleží mimo povolený rozsah $0.05 < \tau < 1.0$. Pokud ano, je rozhodnuto, který z časů *t_on* a *t_off* je delší. Jestliže je větší *t_on* je vzdálenost horní polohy relé od nulové hladiny snížena na 0.9 ze své původní hodnoty. V opačném případě je stejným způsobem snížena vzdálenost polohy dolní. Dodatečně je ještě přepočtena gama s novým parametrem pro relé a program se vrací do předchozí fáze měření vzorků. S pravidly pro nastavení parametrů relé je pravděpodobnost, že nastane takovýto stav minimalizována. Přesto jsou případy k tomu dojít může, což bylo důvodem pro zavedení popsané kontroly (Obrázek 25).

```
IF tau <= 1.0 AND tau >= 0.05 THEN                          //kontrola spravnosti
   stav :=3;
   hlaska := '';

   ELSE
        coef:= 0.9;
       IF (TIME_TO_REAL(t_on)-TIME_TO_REAL(t_off))<0.0 THEN
           d1 := u_0+((d1-u_0)*coef);
           gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
       END_IF;
       IF (TIME_TO_REAL(t_on)-TIME_TO_REAL(t_off))>0.0 THEN
          d2 := u_0+((d2-u_0)*coef);
          gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
       END_IF;
   stav :=0;
   hlaska := 'Mereni se opakuje, program prepocita meze rele';
   dif01 := 0.0;
   dif02 := 0.0;
   dif12 := 0.0;
END_IF;
```

Obrázek 25 – poslední kontrola po výpočtech časových konstant

Pokud τ v uvedeném rozsahu leží, program pokračuje ve výpočtech integrací akčních zásahů a uložených vzorků regulované veličiny. Nejprve je vypočten integrál I_u (Obrázek 26).

```
3: // integrál z u na Tp
   hlaska:= 'Probihaji vypocty parametru regulatoru';
   u := u_0;
   I_u := d1 * (UINT_TO_REAL(buf_u_switch[buf_i])*iT) +
   d2 * (UINT_TO_REAL(buf_len[buf_i]-buf_u_switch[buf_i])*iT);
   stav := 4;
```

Obrázek 26 – integrál Iu

Vzhledem k tomu, že máme obdélníkový řídící signál, stačí nám k tomu vynásobit hodnotu vzdálenosti polohy relé od nulové hladiny s časem, který byla poloha nastavena.

Pak je vypočten integrál I_y sečtením hodnot z pole *buffers[buf_i,i]*.

```
4: // integrál z y na Tp
   I_y := I_y + ((buffers[buf_i,i])+(buffers[buf_i,i+1]))/2.0*iT;
   i := i+1;
   IF i = buf_len[buf_i] THEN
      stav := 5;
   END_IF;
```

Obrázek 27 – integrál Iy

V tom okamžiku má program všechny potřebné hodnoty pro určení parametrů relé pomocí vzorců uvedených v kapitole (2.2.4) (Obrázek 28).

```
5: //výpočet parametrů modelu

   K_p := I_y/I_u;
   L_div_T :=tau/(1.0-tau);
   T := (UINT_TO_REAL(buf_u_switch[buf_i])*iT)/ LN(((h1+h2)/2.0/ABS(K_p)-d2+
   (d1+d2)* EXP(L_div_T ))/(d1-((h1+h2)/2.0/ABS(K_p))));
   L := T*L_div_T;
   stav := 6;
```

Obrázek 28 – výpočet parametrů modelu

Z parametrů modelu prvního řádu s dopravním zpožděním jsou pak přes vzorce uvedené v kapitole [číslo kapitoly] spočítány parametry pro PID regulátor (Obrázek 29).

```
6: //nastavení PID regulátoru

   K := (0.2*L+0.45*T)/(K_p*L);
   T_i := (0.4*L+0.8*T)*L/(L+0.1*T);
   T_d := (0.5*L*T)/(0.3*L+T);

   faze := 2;
   start:= FALSE;
   dlouhy_start:= FALSE;
   END_CASE;
```

Obrázek 29 – výpočet parametrů PID

Nakonec program opouští identifikační fázi a přechází do řízení soustavy pomocí funkčního bloku SimplePID z knihovny ModeLib v2.1 (Obrázek 30). Výchozí žádaná hodnota (*w)*

```
2: // RIZENI////////////////////////////////////////////////////////////////////

   hlaska:= 'Soustava je rizena';
   w:=w_pom;
   e := w-y;

   regulator(y:=y, w:=w, u_man := u_0, T := 0.01, max_u:=u_maxs, min_u:=u_mins,
             Gain := K, Ti:=T_i, Td:=T_d, u=>u, e=>e);

  END_CASE;

END_PROGRAM
```

Obrázek 30 – řízení pomocí PID regulátoru

regulátoru je nastavena na před startem zadanou hodnotu. Její změna je pak možná přes ovládací panel vytvořený v nástroji WebMaker.

4.3. Realizace regulátoru Foxboro EXACT

Stejně jako předchozí regulátory, byl i Foxboro EXACT při svém vývoji neustále testován na reálných laboratorních úlohách. Základním omezením při vývoji byla absence dosud nezveřejněných pravidel a rovnic pro dodatečnou úpravu parametrů regulátoru. V rámci práce tak byl realizován regulátor pouze v omezené úpravě založené na zveřejněném postupu pro počáteční určení parametrů z přechodové charakteristiky.

4.3.1. Popis programu

Program je rozčleněn do několika kroků pomocí funkce CASE OF. Řídící proměnnou je v tomto případě proměnná *„stav“* typu INTEGER. K ovládání programu je podobně jako u regulátoru dle J. Berner vytvořen ovládací panel pomocí nástroje WebMaker. Ke sledování stavu soustavy a ukládání dat je pak opět použit graf zobrazující průběh regulované veličiny *y*, akční veličiny *u* a žádané hodnoty *w*. Graf je vytvořen za použití nástroje GraphMaker.

V programu je použito několik bloků z knihoven prostředí Mosaic. Zároveň je použit jeden vlastní vytvořený funkční blok.

Program po nahrání do PLC a spuštění čeká ve své výchozí pozici na změnu proměnné *„start“* typu BOOL na hodnotu TRUE. Toho lze docílit stisknutím tlačítka „START“ na ovládacím panelu. Před spuštěním má ještě uživatel možnost definovat parametry používané při identifikaci programu a při řízení pomocí PLC. Jsou jimi stejně jako u regulátoru dle J. Berner žádaná hodnota w, počáteční odhad akčního zásahu u_0 a jeho maximální u_{max}, respektive minimální u_{min}, povolená hodnota. Zadávání probíhá stejně jako spouštění přes ovládací panel (Obrázek 31).

Obrázek 31 – ovládací panel regulátoru Foxboro EXACT

Po stisknutí tlačítka START probíhá kontrola zadaných parametrů, jež je obdobou kontroly použité v programu dle J. Berner (Obrázek 32) (popis principu viz kapitola č. 4.2.1).

```
1:
//Kontrola zadanych parametru
  IF u_mins > u_maxs THEN
   hlaska := 'Hodnota u_maxs musi byt vetsi nez u_mins.';
   start:=FALSE;
   stav := 0;
   ELSE
      IF u0 >= u_maxs OR u0 <= u_mins THEN
         hlaska := 'Hodnota u_0 musi lezet mezi u_maxs a u_mins.';
         start:=FALSE;
         stav := 0;
         ELSE

         hlaska := '';
         //Cekani na ustaleni systemu v pracovnim bode
         u := u0;

         ustaleni(vstup := y);
         IF ustaleni.ustaleno = TRUE THEN

         stav := 2;
         ustaleni.ustaleno := FALSE;

         END_IF;
      END_IF;
   END_IF;
```

Obrázek 32 – kontrola zadaných parametrů

Pokud jsou parametry správně zadány program nastavuje na vstup do soustavy *u* hodnotu u_0 a čeká na ustálení regulované veličiny *y*. Pro kontrolu ustálení je použit vytvořený funkční blok *„ustaleni“*. Tento blok je navržen dle části kódu použitého v programu identifikace regulátoru J. Berner a byl vyvinut ve spolupráci s Bc. Alžbětou Hornychovou. Pracuje na stejném principu (viz kapitola č. 4.2.1). Vstupem do něj je proměnná, kterou si přejeme sledovat (v tomto případě *y*). Výstupem je booleovská proměnná *„ustaleno“*. Na ní je navázána podmínková funkce, která určuje, co se má stát po dosažení ustáleného stavu. V tomto případě se jedná o přechod do dalšího kroku programu.

Po ustálení regulované veličiny je v dalším kroku naměřen její šum (Obrázek 33). Nejprve je spuštěn časovač *„timer1“* z knihovny StdLib v2.1 a po dobu nastavenou při jeho volání je měřena maximální *„max_y“*, respektive minimální *„min_y“*, hodnota regulované veličiny. Zároveň je sčítána suma všech hodnot *y* do proměnné *„suma_y“*. Počet sečtených hodnot je ukládán do proměnné *„pocitadlo“*. Při doběhnutí časovače je jeho výstupní proměnná *„timer1.Q“* nastavena na hodnotu TRUE. V té chvíli je dělením *suma_y / pocitadlo* získána průměrná hodnota *y* výchozího pracovního bodu (*„w_pom“*). Následně jsou proměnné *suma_y* a *pocitadlo* vynulovány a program přechází do dalšího kroku.

```
2:
//Mereni sumu a vypocet vychoziho bodu
timer1(IN:= casovac, PT :=T#10s);
suma_y := suma_y+y;
pocitadlo := pocitadlo + 1.0;
          IF y > max_y THEN
            max_y := y;
            END_IF;
            IF y < min_y THEN
            min_y := y;
            END_IF;

IF timer1.Q THEN
stav := 3;
w_pom := suma_y/pocitadlo;
suma_y := 0.0;
pocitadlo := 0.0;
END_IF;
```

Obrázek 33 – měření šumu

Po naměření šumu je navýšena hodnota akční veličiny *u* o desetinu zadaného rozsahu (*max_y – min_y)* (Obrázek 34). Do proměnné „*t_zacatek_L“* je uložena časová značka kdy

```
3:
//Skok o desetinu rozsahu
u := u0 + ((max_y-min_y)/10.0);
t_zacatek_L := GetRTC();
stav:= 4;
```

Obrázek 34 – skok o desetinu rozsahu

k navýšení došlo. Program pak přejde k dalšímu kroku, ve kterém čeká, až regulovaná veličina zareaguje na skokovou změnu *u*. K registraci změny je použita podmínková funkce IF – THEN. Ta je splněna ve chvíli, kdy je hodnota proměnné *y* větší než polovina pásma šumu (*max_y – min_y)* sečtená s průměrnou hodnotou pracovního bodu (*w_pom)* (Obrázek

```
4:

IF y > ((max_y-min_y)/2.0)+w_pom THEN
t_konec_L := GetRTC();
stav:= 5;
END_IF;
```

Obrázek 35 – kontrola odezvy na skok

35). Čas, ve kterém k tomu dojde je uložen do proměnné *„t_konec_L“.* Jeho hodnota pak bude použita pro výpočet dopravního zpoždění *L*. Další parametry potřebné pro zjištění parametrů modelu prvního řádu jsou vypočteny z přechodové charakteristiky.

Nejprve je spuštěn časovač *„timer3“*. Po uplynutí času vloženého při volání je časovač ukončen a je uložen aktuální čas do proměnné *„t_konec_T“*. Zároveň je uložena i hodnota *y* do proměnné *„y_pom_T“* (Obrázek 36). Změřené hodnoty jsou při závěrečných

```
5:

timer3(IN := casovac, PT:=T#5s);

IF timer3.Q THEN
w_pom_T := y;
t_konec_T := GetRTC();
stav:= 6;
END_IF;
```

Obrázek 36 – odečet hodnot pro výpočet směrnice

výpočtech použity pro určení směrnice tečny. Dalším krokem je čekání na ustálení na nové hodnotě regulované veličiny po provedeném skoku. K tomu je opět použit naprogramovaný funkční blok *ustaleni* (Obrázek 37). V momentě kdy je regulovaná veličina ustálena (*ustaleno = TRUE)*, je zahájen výpočet sumy hodnot *y*. Stejně jako v předchozím případě k tomu použijeme proměnné *suma_y* a *pocitadlo*. Výpočet probíhá po dobu nastavenou v časovači *„timer2“*. Výsledná hodnota po dělení (*suma_y / pocitadlo)* je uložena do proměnné *„w_skok“*. V ten moment má program načteny všechny hodnoty potřebné pro sestavení modelu a přechází do své výpočetní fáze.

```
6:
ustaleni(vstup := y);

IF ustaleni.ustaleno = TRUE THEN
suma_y := suma_y+y;
pocitadlo := pocitadlo + 1.0;
timer2(IN:= casovac, PT :=T#10s);

IF timer2.Q THEN
w_skok := suma_y/pocitadlo;
ustaleni.ustaleno := FALSE;

stav := 7;
END_IF;
END_IF;
```

Obrázek 37 – použití funkčního bloku ustálení

Výpočty jsou zahájeny určením statické citlivosti k_p. Ta je určena jako rozdíl hodnoty *y* po skoku (*w_skok)* a před skokem *(w_pom)* podělený velikostí provedeného skoku.

Dopravní zpoždění *L* je určeno jako rozdíl časových značek *t_konec_L* a *t_zacatek_L*. Jako poslední je vypočtena časová konstanta modelu *T*. Zde postupujeme dle doporučení pro

```
7:
//odecty z prubehu
kp := (w_skok - w_pom)/((max_y-min_y)/10.0);
L := TIME_TO_REAL(SUB_DT_DT(t_konec_L,t_zacatek_L))/1000.0;
T := (w_skok - w_pom)*(TIME_TO_REAL(SUB_DT_DT(t_konec_T,t_konec_L))/1000.0/(y_pom_T-w_pom));
stav := 8;
```

Obrázek 38 – výpočet parametrů modelu

výpočet časové konstanty z přechodové charakteristiky, jenž je uvedené v [7].

V poslední kroku výpočetní fáze (Obrázek 39) jsou z určeného modelu soustavy vypočteny parametry pro PID regulátor dle vzorců z [7] popsaných v kapitole (2.3.1).

```
8:
//vypocet parametru pro PID regulator

PB := 120.0*kp*L/T;
Ti := 1.5*L;
Td := Ti/6;
wmax := 5*L;
stav := 9;
```

Obrázek 39 – výpočet parametrů pro PID

Vypočtené parametry jsou následně dosazeny do funkčního bloku *„regulator“*, který je funkčním blokem *SimplePID* z knihovny ModeLib v2.1, a program pomocí něj začne řídit soustavu (Obrázek 40). Uživatel pak může soustavu řídit změnami žádané hodnoty *w* skrze ovládací panel.

```
9:
//Rizeni
 e := w-y;

 regulator(y:=y, w:=w, u_man := u0, max_u:=u_maxs, min_u:=u_mins,
 Gain := kp, Ti:=Ti, Td:=Td, u=>u, e=>e);

END_CASE;
```

Obrázek 40 – řízení PID regulátorem

5. Porovnání kvality řízení zvolených regulátorů

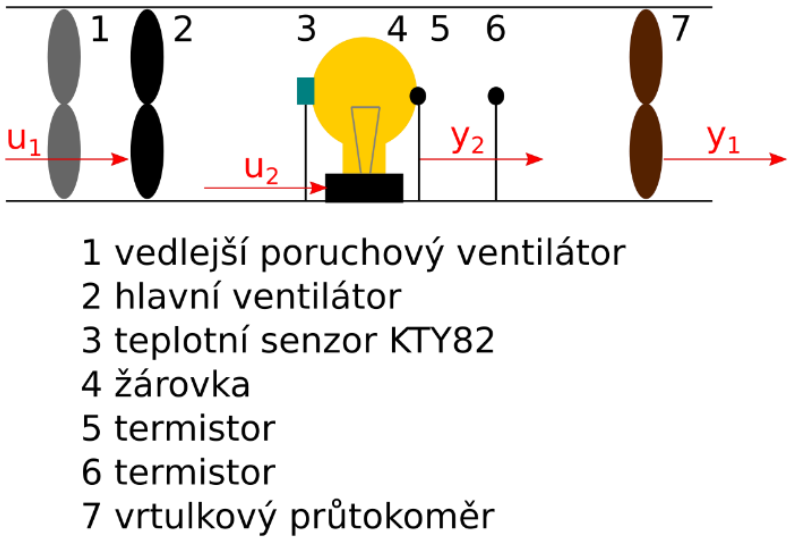
Hlavním předmětem práce bylo, kromě realizace regulátorů, jejich srovnání, co se kvality řízení týče. K porovnání byly použity laboratorní soustavy z laboratoře 111 na půdě FS ČVUT. Řízení soustav pak zajišťoval systém PLC Tecomat Foxtrot ve verzi CP-1015. K ovládání programu v PLC byly použity panely vytvořené nástrojem WebMaker. Pro vizualizaci řízení a sběr dat pak posloužil nástroj GraphMaker.

5.1. Soustavy použité pro testování regulátorů

Pro testování naprogramovaných regulátorů byly vybrány dvě laboratorní úlohy. Hlavním důvodem pro jejich volbu byla rozdílnost v chování obou úloh. Ačkoli obě víceméně odpovídají chování modelu prvního řádu jsou rozdílné v zašumění a dopravním zpoždění signálu.

5.1.1. Teplovzdušný model

Soustavu tvoří krytý tunel čtvercového průřezu, jenž je na jedné straně osazen dvěma ventilátory. Ventilátor vnitřní je hlavním ventilátorem soustavy a slouží pro zajištění proudění vzduchu tunelem. Vnější ventilátor tzv. poruchový, je posazen tak, aby působil proti ventilátoru hlavnímu. Tím pádem je možné ho využít pro simulaci vlivu poruchových veličin na soustavu. Na druhém konci tunelu je pak umístěn vrtulkový průtokoměr sloužící pro měření průtoku vzduchu tunelem. Uvnitř tunelu se nachází žárovka, která plní funkci tepelného zdroje pro ohřev vzduchu v tunelu. Pro měření teploty v okolí žárovky jsou na jejím povrchu umístěny dva senzory – termistor a teplotní senzor KTY82. Ve vzdálenosti 1 cm od žárovky je pak ještě druhý termistor, který je třetím teplotním senzorem soustavy. [13]

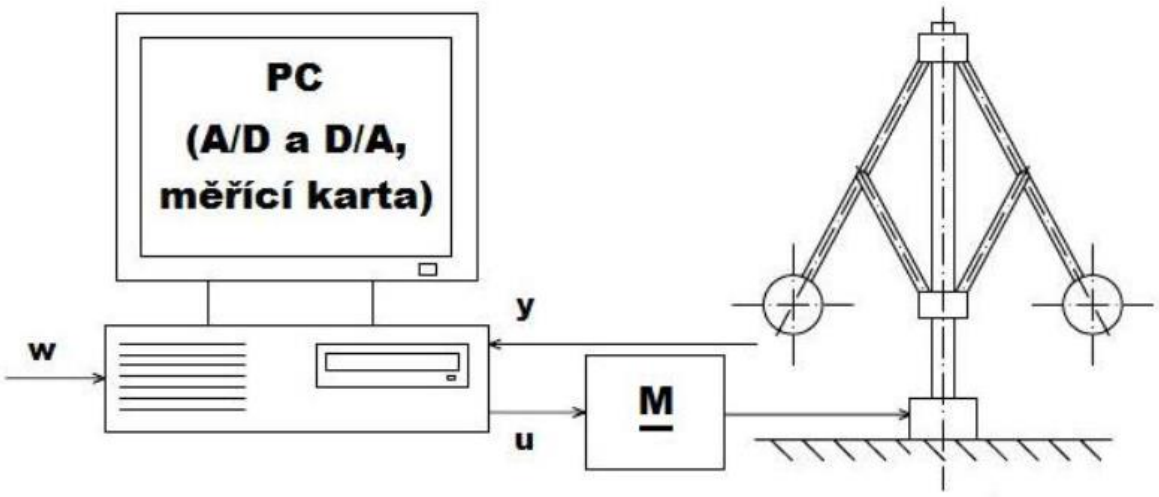


Obrázek 41 - Teplovzdušný model [14]

Soustava teplovzdušný model může pracovat v několika různých modifikacích, dle zvoleného zapojení. Metody naprogramovaných algoritmů pracují na principu řízení SISO. Z nabízených možností bylo pro testování úloh zvoleno zapojení, kdy je vrtulkový průtokoměr vstupem do PLC a hlavní ventilátor výstupem z něj.

5.1.2. Wattův roztěžník

„Tento laboratorní model je upravenou obdobou roztěžníku, který v 18. století navrhl James Watt jako součást proporcionálního odstředivého regulátoru pro regulaci tlaku páry ve svém parním stroji. Vlastní model roztěžníku je tvořen hřídelem, na kterém jsou rotačními vazbami připevněna dvě ramena s konci osazenými závažími. Jakmile je roztěžník roztočen zabudovaným elektromotorem na určité minimální otáčky (dané hmotností závaží), dojde působením odstředivé síly k překonání doposud převládající tíhové síly obou závaží a ramena se vychýlí z počáteční polohy - velikost vychýlení ramen je úměrná otáčkám roztěžníku. Otáčky roztěžníku *y* jsou měřeny otáčkoměrem a řízeny zvoleným typem regulátoru, resp. manuálně využitím akční veličiny *u*.“ [15]



Obrázek 42 - Schéma Wattova roztěžníku připojeného k PC [15]

5.2. Průběh testování

U všech regulátorů byl proveden stejný test kvality řízení, aby mohlo dojít k objektivnímu srovnání. Bylo zapotřebí přihlédnout také k metodě řízení jednotlivých regulátorů. Testování probíhalo ve formě několika po sobě jdoucích skokových změn žádané hodnoty v různých pásmech rozsahu měřené veličiny. Vzhledem k tomu, že soustava teplovzdušný tunel neměla lineární průběh statické charakteristiky, byl maximální rozsah regulované veličiny rozdělen na tři pásma. V nich pak bylo provedeno několik skoků v hodnotě 1 V nebo 2 V. Střídavě pak byla hodnota žádané veličiny snižována nebo zvyšována, aby bylo možné sledovat chování regulátoru v obou směrech. Při přechodech z pásma do pásma pak byl proveden také doplňující testovací skok o hodnotě 4 V. Zároveň bylo v průběhu testování několikrát pozorováno chování při ustáleném stavu.

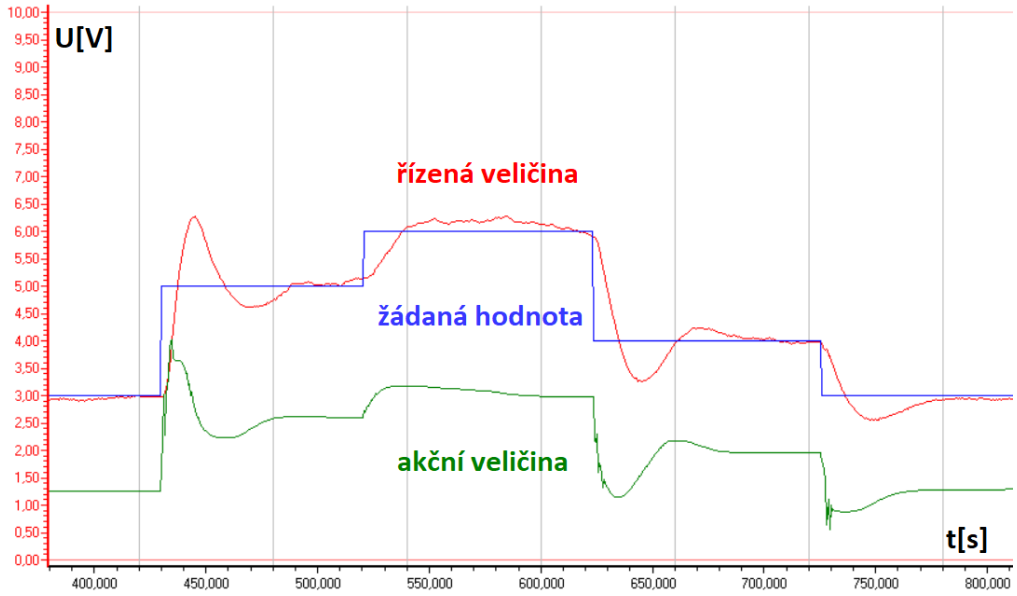
Kromě celkového charakteru řízení, patrného většinou již pouhým pohledem, byla dalším vyhodnocovaným parametrem velikost prvního překmitu po změně žádané hodnoty. Z hlediska řízení se jedná o důležitou vlastnost regulátoru, jelikož při příliš velkých překmitech může dojít k narušení funkce soustavy. Při častých změnách žádané veličiny pak mohou velké překmity regulované veličiny snížit celkovou životnost použitých zařízení. K vyhodnocení kvality řízení byla měřena ještě doba ustálení regulované veličiny na žádané

hodnotě. Doba potřebná k ustálení soustavy v požadovaném stavu je rozhodujícím parametrem zejména v případě, že regulovaná soustava je součástí většího automatizovaného celku (např. výrobní linka). Při zapojení regulátoru do vícekrokového procesu totiž hraje zásadní roli, jak dlouhou dobu potřebujeme pro získání požadované hodnoty regulované veličiny. Pokud je doba příliš dlouhá, je tím zpomalován celý automatizovaný systém. Další pracovní jednotky musí pro dodržení synchronizace čekat, dokud se regulovaná hodnota neustálí. V případě tohoto testování byla za ustálenou považována regulovaná veličina ve chvíli, kdy již hodnota regulační odchylky nepřesáhla hodnotu 5 % změny žádané veličiny.

Za směrodatný ukazatel kvality řízení byly použity průměrné hodnoty z více naměřených skoků. Testy byly opakovány vícekrát, aby byla snížena pravděpodobnost nežádoucích ovlivnění výsledků skrze poruchové veličiny na soustavách. Velké ovlivnění se potenciálně mohlo týkat zejména regulátorů dle J. Berner a Foxboro EXACT. Oba tyto regulátory totiž pro nastavení parametrů řízení využívají matematický model získaný z experimentální identifikace před zahájením řízení. Poruchy během této inicializační fáze tak mohly mít za důsledek sestavení modelu s chováním odlišným od reálné soustavy. Regulátor dle J. Maršíka pracuje pouze s aktuální regulační odchylkou a parametry přepočítává neustále. V jeho případě tak není nutné předpokládat větší rozdíly při opakování testování.

5.3. Testování teplovzdušného modelu

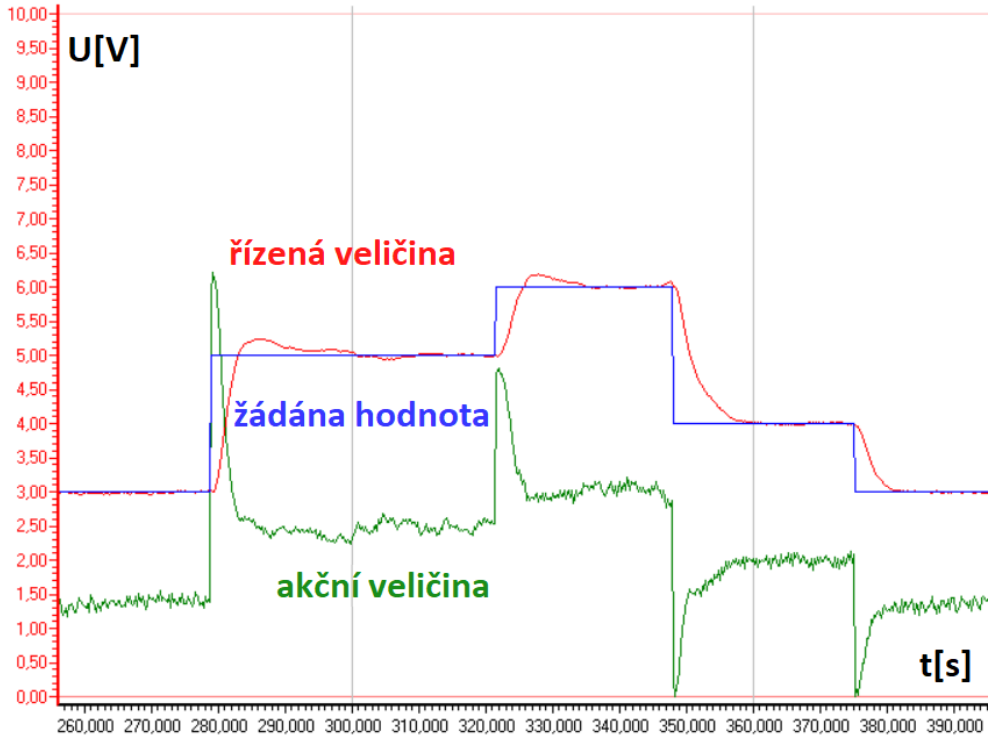
5.3.1. Výsledky testování regulátoru dle J. Maršíka



Graf 1- Ukázka řízení soustav teplovzdušný model regulátorem dle J. Maršíka

Charakter řízení soustavy teplovzdušný model regulátorem dle J. Maršíka je výrazný velkými prvními překmity, zejména po větších skocích. Na grafu (Graf 1) lze pozorovat i výskyt druhých, menších překmitů. Toto chování odpovídá popisu chování regulátoru při reakci na jednotkový skok v [Maršík citace ***] (případně doplnit přesnou citaci***). Po maximálně dvou překmitech pak regulovaná veličina příliš nekolísá. V jednom případě však můžeme sledovat nepříliš přesné ustálení na žádané hodnotě. Průběh řídicí veličiny vykazuje pozvolné přechody bez výraznějších skoků a kmitání. Průměrná procentuální hodnota prvního překmitu, vztažená k velikosti změny, jež překmit vyvolala, je 32,3 %. Tolerance této hodnoty záleží na aplikaci, přesto je patrné, že se jedná z pohledu automatizace o spíše vysoké číslo. Doba ustalování je v některých případech také poměrně dlouhá, v průměru dosahuje dokonce hodnoty 76 sekund. Obecně vzato lze však konstatovat, že regulátor dle J. Maršíka je schopen řídit testovanou úlohu.

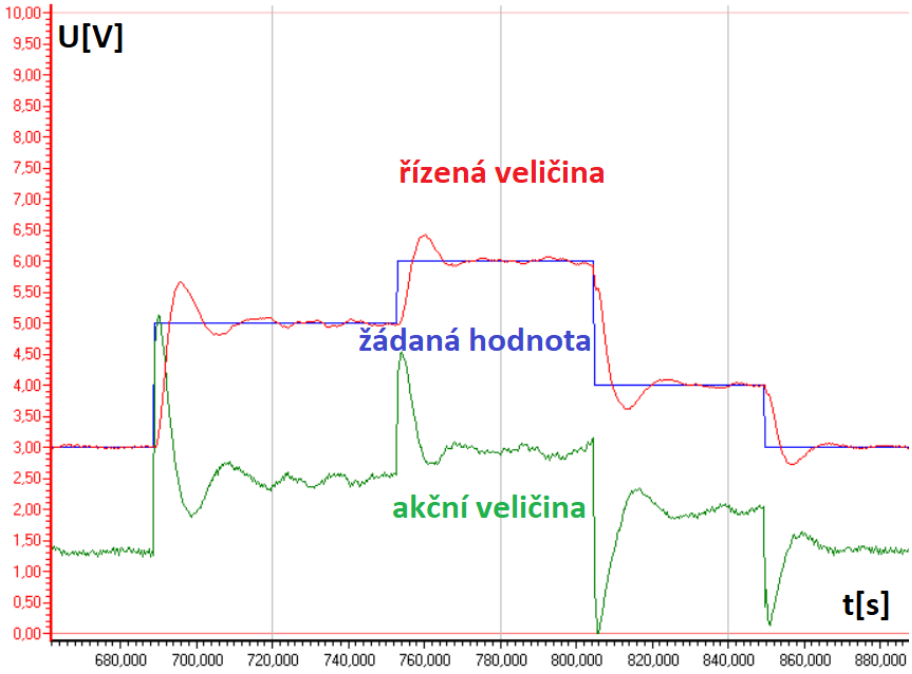
5.3.2. *Výsledky testování regulátoru dle J. Berner*



Graf 2 - Ukázka řízení soustavy teplovzdušný model regulátorem dle J. Berner

Řízení teplovzdušného modelu za pomoci regulátoru s využitím modelu soustavy získaného reléovou identifikací dle J. Berner, je na první pohled výrazné velmi malými nebo vůbec žádnými prvními překmity. Průměrnou hodnotou prvního překmitu je 4,0 %. Jak je vidět na grafu (Graf 2), tak v celém průběhu prakticky nejsou (kromě prvních) žádné další měřitelné překmity. Tato vlastnost se dobře promítá zejména do doby ustálení regulované veličiny, která je tak poměrně krátká. Průměrně dosahuje 11 sekund. Celkový vzhled řízení je pak charakteristický ostrými, agresivními zásahy řídící veličiny. Regulátor s takovýmto řízením je velmi odolný proti vnějším zásahům poruchových veličin. Řízení regulátorem dle J. Berner se na soustavě teplovzdušný model osvědčilo.

5.3.3. *Výsledky testování regulátoru Foxboro EXACT*



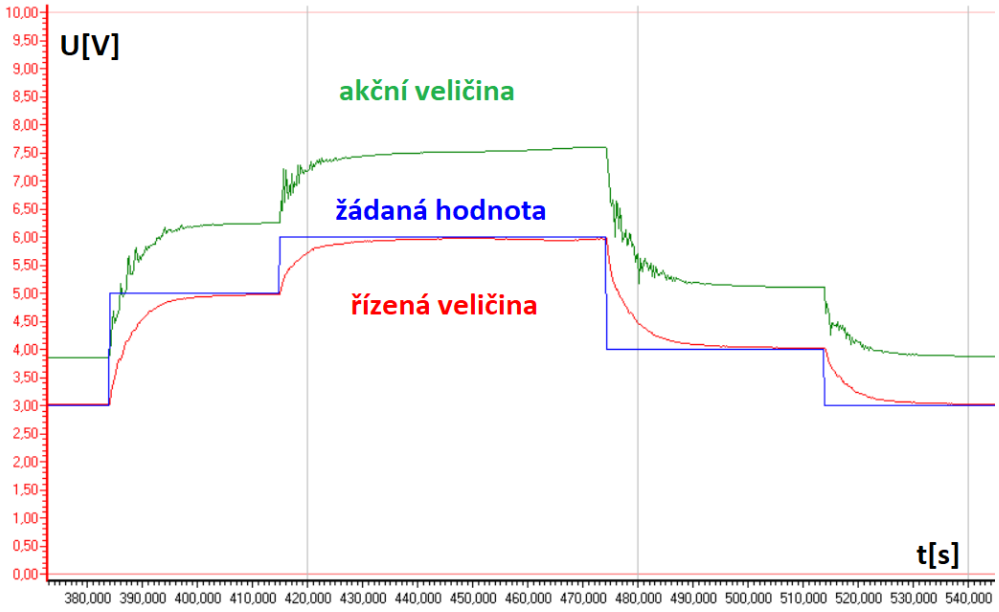
Graf 3 - Ukázka řízení soustavy teplovzdušný model regulátorem Foxboro EXACT

Řízení regulátorem Foxboro EXACT vykazuje na první pohled (Graf 3) patrné, vyvážené první překmity. Ty sice nedosahují velikosti např. překmitů u J. Maršíka, ovšem stále je jejich průměrná hodnota vyšší a to 19,5 %. Na průběhu lze pozorovat i druhé překmity, které však jsou na hranici definovaného ustáleného stavu. Díky tomu je pak také doba ustálení v průměru mezi dříve jmenovanými. Konkrétně odpovídá 17 sekundám. Průběh akční veličiny je také někde na pomezí mezi předchozími testovanými regulátory. Na jednu stranu můžeme pozorovat pozvolný průběh. Ten je však po celou dobu narušován mírnými ostrými kmity.

Obecně tak regulátor Foxboro EXACT svým charakterem řízení leží někde na pomezí mezi oběma dalšími. Je však třeba zmínit, že použitý algoritmus je sestaven pouze ze zveřejněné části metody a sice počátečního odhadu modelu. Další fáze metody, tedy průběžná adaptace není v algoritmu zavedena. Dá se tedy předpokládat, že naprogramovaný algoritmus zdaleka nedosahuje kvalit originálního regulátoru EXACT.

5.4. Testování Wattova roztěžníku

5.4.1. Výsledky testování regulátoru dle J. Maršíka

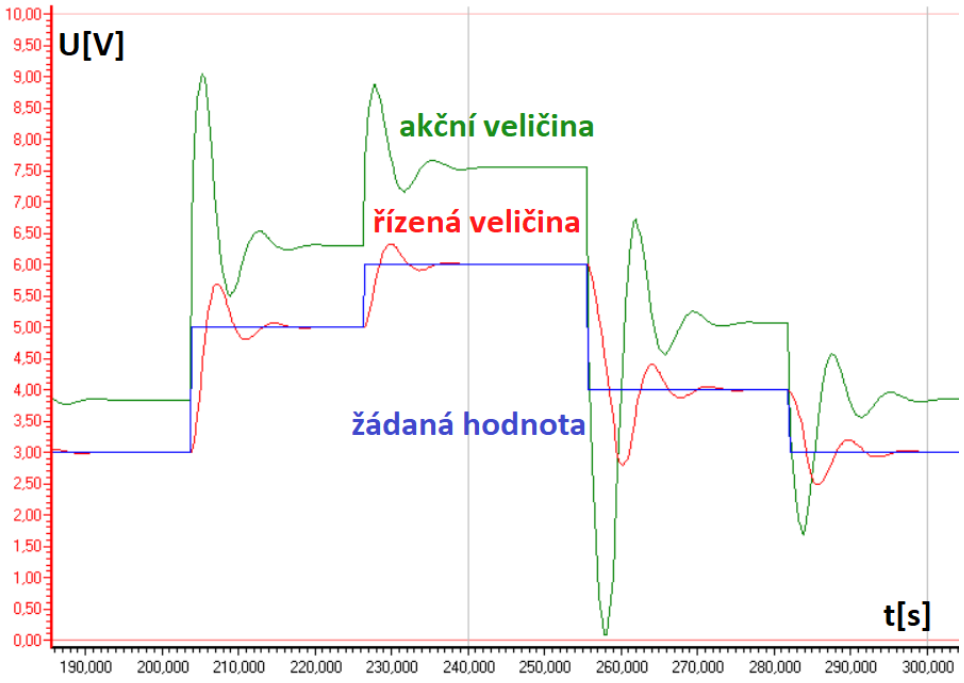


Graf 4 - ukázka řízení soustavy Wattův roztěžník regulátorem dle J. Maršíka

Z grafu (Graf 4) je patrný velmi odlišný průběh řízení regulátorem dle J. Maršíka soustavy Wattův roztěžník oproti průběhu v předchozím testu. Na první pohled si lze všimnout naprosté absence překmitů. Regulovaná veličina se po změně žádané hodnoty vždy pozvolně ustálí. Doba, kterou ustálení trvá je však v tomto případě asi největší slabinou řízení. Její průměrná hodná činí 17,5 sekundy. Při porovnání s ostatními regulátory se jedná o vysoké číslo. Akční veličina má zvláště při přeregulování kmitavý charakter, typický pro agresivní řízení. Čím blíže je však regulovaná veličina žádané hodnotě, tím více se průběh podobá předchozímu pozvolnému charakteru.

Řízení laboratorní úlohy Wattův roztěžník pomocí regulátoru dle J. Maršíka lze definovat jako bezpečné s pomalejším ustalováním. Ačkoli doby ustálení jsou dlouhé, regulátor soustavu spolehlivě řídí.

5.4.2. *Výsledky testování regulátoru dle J. Berner*

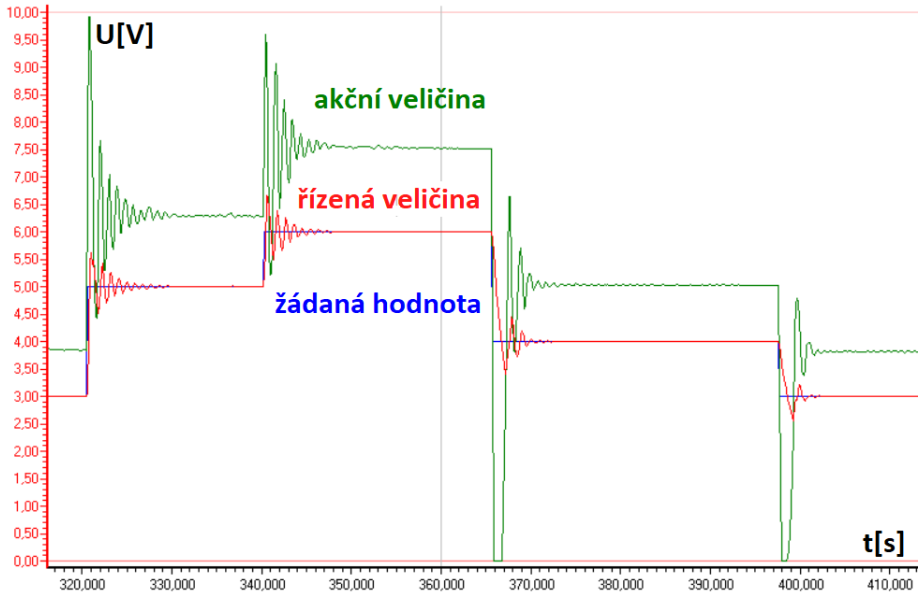


Graf 5 - ukázka řízení Wattova roztěžníku regulátored dle J. Berner

Charakter řízení Wattova roztěžníku regulátorem dle J. Berner je značně odlišný od průběhu předchozího testu. Výrazné první překmity jsou často doprovázeny i překmity druhými, potažmo třetími. V průměru dosahuji první překmity hodnoty 26,4 %. Doba ustálení je nicméně, překmitům navzdory, poměrně krátká. Její průměrná hodnota je 9,4 sekundy. Průběh akční veličiny nemá krom úvodního výrazného zásahu s předchozím testem žádné společné znaky. Je pozvolný bez výraznějšího kmitání.

Ačkoli se oproti přechozímu testu objevili výrazné překmity při ustalování na nové žádané hodnotě, regulátor dle J. Berner je schopen vcelku uspokojivě řídit i tuto testovanou soustavu.

5.4.3. *Výsledky testování regulátoru Foxboro EXACT*



Graf 6 - ukázka řízení soustavy Wattův roztěžník regulátorem Foxboro Exact

Řízení regulátorem Foxboro EXACT je odlišné od obou předchozích testovaných regulátorů. Po změně žádané hodnoty dochází k několika výrazným rychlým překmitům. Průměrná hodnota prvního z nich je pak dokonce 31,2 %. Kmitaní je pak výrazné zejména při navyšování žádané hodnoty. Při klesání je množství kmitů značně redukováno. Doba ustálení je nejnižší ze všech testovaných regulátorů. Její průměrná hodnota činí dokonce jen 4,2 sekundy. Průběh akční veličiny je výrazný ostrými kmitavými zásahy po ustálení je však víceméně stabilní.

Z dat získaných z průběhu řízení regulátorem Foxboro EXACT lze určit dvě vlastnosti. První je kmitavost při ustalování s velkými překmity, které mohou být potenciálně nežádoucí. Druhou je vysoká rychlost řízení s okamžitou odezvou. Navzdory kmitavé přechodové charakteristice je tak regulátor Foxboro EXACT schopen řídit testovanou úlohu.

6. Závěr

Všechny vybrané regulátory byly popsány a dle dostupné metodiky realizovány. K vývoji a testování byl použit systém PLC Tecomat Foxtrot s podporou software od firmy Teco a.s. Testy probíhaly na školních laboratorních úlohách na půdě Fakulty strojní ČVUT.

Shrnutí dosažených výsledků:

Teplovzdušný model	Regulátor dle J. Maršíka	Regulátor dle J. Berner	Regulátor Foxboro EXACT
Průměrná velikost prvního překmitu [%]	32,3	4,0	19,5
Průměrná doba ustálení [s]	76,3	11,2	16,5

Tabulka 3 - shrnutí výsledků získaných z dat naměřených na soustavě Teplovzdušný model

Wattův roztěžník	Regulátor dle J. Maršíka	Regulátor dle J. Berner	Regulátor Foxboro EXACT
Průměrná velikost prvního překmitu [%]	0,0	26,4	31,2
Průměrná doba ustálení [s]	17,5	9,4	4,2

Tabulka 4 - shrnutí výsledků získaných z dat naměřených na soustavě Wattův roztěžník

Na základě výsledného srovnání získaného z naměřených dat můžeme konstatovat, že každá z testovaných metod má své silné i slabší stránky. Rozdíly v průběhu řízení jsou dány především odlišným přístupem jednotlivých metod k návrhu samoladícího algoritmu.

Adaptivní regulátor dle J. Maršíka vykazoval při měření na soustavě Teplovzdušný model velké první překmity. Z hlediska bezpečnosti řízeného systému se tak v některých

případech může jednat o závažnou komplikaci. Na druhou stranu je třeba dodat, že přechody byly veskrze pozvolnějšího charakteru. Z hlediska doby ustálení se pak jednalo i vzhledem k pozvolným větším překmitům o vcelku pomalý řídící proces. Při testu na druhé úloze pak regulátor dle J. Maršíka ukázal své kvality. I když stálo šlo o relativně pomalejší řídící proces, jakékoliv překmity, výrazné zejména při prvním testování v tomto případě zcela absentovaly. Rozdílnost obou průběhů řízení lze vysvětlit ze zkušeností popsaných J. Maršíkem v [3]. Soustava teplovzdušný model disponuje výrazně zašuměným signálem řízené veličiny, což dle [3] působí negativně na řídící proces. Oproti tomu soustava Wattův roztěžník už ze své podstaty na šum a poruchové veličiny (při dostatečně robustně sestrojené pohyblivé části) náchylná není. Má také oproti Teplovzdušnému modelu mnohem rychlejší reakce na změnu akční veličiny. Algoritmus regulátoru založený na vyhodnocování *e* tak při pozvolném náběhu nestihne překmitnout. Velkou předností regulátoru dle J. Maršíka je bezesporu absence jakékoliv identifikační části. Při řízení tak i po opakovaných testech dosahuje konstantních výsledků. Jeho nasazeni také není vázáno na řád řízené soustavy. Lepších výsledků by regulátor mohl dosahovat zajištěním sofistikovanější filtrace signálu, při které by byl brán ohled na šum regulované veličiny.

Řízení regulátorem dle J. Berner využívajícího reléové identifikace dosahovalo velmi dobrých výsledků zejména na první testované úloze. Jelikož průběh identifikace je řízený, je velmi omezena možnost ovlivnění výpočetní algoritmu poruchovými veličinami. Před zahájením přepínaní je navíc měřen šum soustavy a s ohledem na něj je pak postupováno dále. Pro dosažené výsledky tak bylo velmi zajímavé sledovat průběh řízení na soustavě Wattův roztěžník. Zde se i přes ne tak špatné průměrné hodnoty hodnocených veličin ukázaly jisté limity identifikačních metod obecně. Velké překmity, které byly zaznamenány jsou patrně způsobeny metodou měření potřebných koeficientů pro PID regulátor. Zde je před samotným kmitáním nejprve naměřen šum regulované veličiny, na základě kterého je vypočtena hystereze. U soustavy Wattův roztěžník je však šum téměř nulový. Ve spojitosti s velmi rychlými reakcemi na změny akční veličiny tak vzniká situace, kdy se vyhodnocované kmitání relé pohybuje velmi blízko pracovnímu bodu. Tento stav je velmi náchylný na sebemenší chyby v průběhu měření. Je tak obtížné zajistit autentický odhad modelu soustavy. Regulátor dle J. Berner byl přesto nejspíš nejspolehlivějším z testovaných regulátorů. Kvalita jeho řízení v různých pásmech statické charakteristiky pak byla víceméně

podobná. Odlišnosti na soustavě Teplovzdušný tunel nebyly tak výrazné, aby byl znát rozdíl při naladění na různé pracovní body. Nutno však dodat, že obě testované úlohy byly svým charakterem blízké použitému modelu prvního řádu s dopravním zpožděním. Některé z možných metod na vylepšení tohoto regulátoru již byly zmíněny. Jednalo se zejména o nasazení více různých modelů soustav a typů řízení.

Druhý realizovaný regulátor pracující s identifikací - Foxboro EXACT byl při svém testování značně omezen zejména faktem, že jeho kompletní metodika není dosud zveřejněna. Testům tak byl podroben pouze prvotní, základní odhad parametrů PID regulátoru. Z tohoto důvodu nelze příliš objektivně hodnotit samotný regulátor Foxboro EXACT s kompletní adaptací. Realizovaná část programu však i tak vykazovala vcelku uspokojivé výsledky při řízení. Během testování na první úloze se kvalitativně pohyboval někde na průměru zbylých dvou metod. Ačkoli v obou případech řízení pomocí něj způsobovalo velké překmity regulované veličiny, doba ustálení byla pokaždé jen velmi krátká. Za potenciálně problematické lze považovat pouze kmitavé chování při testu na Wattově roztěžníku. Z dlouhodobého hlediska jsou při řízení soustav jakékoli pravidelné rychlé kmity zatěžující a mohou ohrozit životnost zařízení. Výhodou identifikace tohoto regulátoru je její jednoduchost a nízká časová náročnost. Vylepšením metody by jistě bylo její zkompletování přidáním adaptační části.

Ačkoli již delší dobu existuje a stále vzniká mnoho metod samoladících regulátorů, jejich implementace dosud naráží na svá technická omezení. Ne všechny limity jsou potenciálně řešitelné, ovšem stále je mnoho takových, které čekají na to, až budou prolomeny.

Seznam užitých bibliografických citací

[1] SIXTOVÁ, Jana a Eva FRUHWIRTOVÁ. V zemědělství chybí pracovníci. Musí je nahradit robotizace a automatizace. *ZSČR* [online]. **2018** [cit. 2018-06-15]. Dostupné z: https://www.zscr.cz/clanek/v-zemedelstvi-chybi-pracovnici-musi-je-nahradit-robotizace-a-automatizace-3735

[2] SCHLEGEL, Miloš. PRŮMYSLOVÉ PID REGULÁTORY: TUTORIAL. *REX Controls* [online]. **2010**, 24 [cit. 2018-06-15]. Dostupné z: http://matlab.fei.tuke.sk/raui/doc/PIDTutor_CZ.pdf

[3] MARŠÍK, J. *Jednoduché algoritmy číslicové adaptivní regulace.* Výzkumná zpráva ÚTIA ČSAV, č. 1106, 1981.

[4] BERNER, Josefin. *Automatic Tuning of PID Controllers based on Asymmetric Relay Feedback*. Lund, 2015. Disertační práce. Lund university, Lund Institute of Technology, Department of Automatic Control, 2015.

[5] BRISTOL, E.H. & KRAUS, T.W.. Life with Pattern Adaptation. Proceedings of the American Control Conference. 2. 888 - 893. 10.23919/ACC.1984.4788500, 1984.

[6] SCHNEIDER, Electrics. The history of Foxboro by Schneider Electric. *Schneider electric* [online]. 2018 [cit. 2018-06-15]. Dostupné z: https://www.schneider-electric.com/en/brands/foxboro/foxboro-history.jsp

[7] ÅSTRÖM, K.J., T. HÄGGLUND, C.C. HANG a W.K. HO. Automatic tuning and adaptation for PID controllers - a survey. *Control Engineering Practice*. 1993, **1**(4), 699-714. DOI: 10.1016/0967-0661(93)91394-C. ISSN 09670661. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pii/096706619391394C

[8] TECO A. S. PLC Tecomat Foxtrot. *Teco a. s.* [online]. 2017 [cit. 2018-06-15]. Dostupné z: https://www.tecomat.cz/products/cat/cz/plc-tecomat-foxtrot-3/

[9] TECO A. S. *Tecomat Foxtrot - PROGRAMOVATELNÉ AUTOMATY - ZÁKLADNÍ DOKUMENTACE MODULU CP-1015* [online]. 2008 [cit. 2018-06-15]. Dostupné z:

https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv11015_00_foxtrot_cp-1015_cz_en

[10] TECO A. S. Mosaic - pro vývoj PLC programu dle standardu IEC 61131-3. *Tecomat.cz* [online]. 2017 [cit. 2018-06-15]. Dostupné z: https://www.tecomat.cz/ke-stazeni/software/mosaic/

[11] ŠMEJKAL, Ladislav a Josef ČERNÝ. Esperanto programátorů PLC: programování podle normy IEC/EN 61131-3: časopis pro automatizační techniku. *AUTOMA*.

[12] THAYER, Ted. Speaking in Tongues: Understanding the IEC 61131-3 Programming Languages. *Control Engineering* [online]. Bosch Rexroth Electric Drives and Controls, 2009 [cit. 2018-06-15]. Dostupné z: https://www.controleng.com/single-article/speaking-in-tongues-understanding-the-iec-61131-3-programming-languages/afa606fff8fa23faf0ae3dfa1a49cec2.html

[13] Teplovzdušný model. Návody na laboratorní cvičení z automatického řízení. [online].[cit.19.3.2018]. Dostupné z: http://vlab.fs.cvut.cz/navody/les/tvm.pdf

[14] HORNYCHOVÁ, Alžběta. *Reléová identifikace v uzavřené regulační smyčce pomocí programovatelného automatu Tecomat Foxtort.* In *Studentská tvůrčí činnost: sborník 2018*. V Praze: Fakulta strojní ČVUT, 2018.

[15] Wattův roztěžník. Návody na laboratorní cvičení z automatického řízení. [online].[cit.19.5.2018]. Dostupné z: http://vlab.fs.cvut.cz/cz/ulohy/rozteznik.php

Příloha 1 – Adaptivní regulátor dle J. Maršíka - kompletní kód

```
VAR_GLOBAL
(*
termistor AT r0_p3_AI2.ENG : REAL; //vstup do PLC
zarovka   AT r0_p3_AO0.ENG : REAL; //výstup z PLC
 u AT zarovka : REAL; //akční zásah
 y AT termistor; //výstup soustavy
 // *)
  (*
prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
ventilator AT r0_p3_AO1.ENG : REAL; //výstup z PLC
u AT ventilator : REAL; //akční zásah
y AT prutokomer; //výstup soustavy
  //*)
//rozteznik
prutokomer AT r0_p3_AI0.ENG : REAL; //vstup do PLC
ventilator AT r0_p3_AO0.ENG : REAL; //výstup z PLC
u AT ventilator : REAL; //akční zásah
y AT prutokomer; //výstup soustavy

// Pracovni bod
e_Delta   : REAL :=0.0;
e_Delta2  : REAL :=0.0;
w_filtr1  : REAL :=2.5;
w_filtr2  : REAL :=0;
w         : REAL :=0.0;
kappa     : REAL :=1.0;
kappa_zad : REAL :=0.5;
alfa      : REAL := 0.05;
alfa2     : REAL := 0.05;
alfa_delta: REAL := 0.1;
e1        : REAL :=1.0;
e2        : REAL :=1.0;
a1        : REAL :=1.0;
a2        : REAL :=1.0;
v1        : REAL :=1.0;
v2        : REAL :=1.0;
e_2       : REAL :=0;
T         : REAL :=0.1;
sigma1    : REAL :=1;
sigma2    : REAL;
gamma     : REAL :=0;
beta      : REAL :=0;
e_pole    : ARRAY[0..5] OF REAL;
prepocet  : REAL := 1000.0;
index     : INT := 0;
suma      : REAL;
suma2     : REAL;
init      : BOOL := FALSE;
stop      : BOOL := FALSE;
END_VAR
PROGRAM prgMain
IF w > 10.0 THEN
w:=10.0;
END_IF;
IF w < 0.0 THEN
w:= 0.0;
END_IF;
```

```
//Adaptace
//I. reg odchylka
e_pole[1]    := w - y;
e_2          := e_pole[1]*e_pole[1];
e_delta      := e_pole[1]-e_pole[2];
//II. Analyza velikost reg. odchylky
IF init = TRUE AND (e_2) <= (0.05*sigma1) AND stop = FALSE THEN

//Regulace bez adaptace
suma         := alfa_delta*e_pole[1] + suma2;
u            := ((alfa_delta*gamma*e_delta)+(alfa_delta*beta*e_pole[1])+suma)/prepocet;
IF u > 10.0  AND stop = FALSE THEN
u := 10.0;
ELSE
             IF u < 0.0 AND stop = FALSE THEN
             u := 0.0;
             ELSE
                IF stop = FALSE THEN
                 u := u;
                 END_IF;
             END_IF;
END_IF;
//Adaptace
ELSE
//III. Casova konstanta
T            := 2.0*3.14*sqrt((v2)/(a2));
//IV. Rozptyl reg. odchylky
sigma1       := (e_2+(3.0*T*sigma2))/(1.0+(3.0*T));
//V. e, a, v
e1           := (e_2+(T*e2))/(1.0+T);
v1           := ((e_delta*e_delta)+(T*v2))/(1.0+T);
a1           := ((e_delta*(e_pole[1]-e_pole[3]))+(T*a2))/(1.0+T);
//VI. Filtrovana zadana hodnota
w_filtr1     := (w+(T*w_filtr2))/(1.0+T);
//VII. index kappa
kappa        := v1/sqrt(e1*a1);
//VIII. Spolecne zesileni reg. alfa
alfa_delta   := (alfa2*((kappa_zad/kappa)-1.0))/T;
//IX. Koeficient proporcialni vetve
beta         := sqrt((e1)/(v1));
//X. Koeficient diferencni vetve
gamma        := sqrt((e1)/(a1));
//XI. Akcni velicina regulatoru u
suma         := alfa_delta*e_pole[1] + suma2;
u            := ((alfa_delta*gamma*e_delta)+(alfa_delta*beta*e_pole[1])
              + suma)/prepocet;
IF u > 10.0 AND stop = FALSE THEN
u := 10.0;
ELSE
             IF u < 0.0 AND stop = FALSE THEN
             u := 0.0;
             ELSE
                IF stop = FALSE THEN
                 u := u;
                 END_IF;
             END_IF;
END_IF;
init := TRUE;
END_IF;
```

```
//Pamet do nasledujiciho cyklu
e_pole[5]    := e_pole[4];
e_pole[4]    := e_pole[3];
e_pole[3]    := e_pole[2];
e_pole[2]    := e_pole[1];
e2           := e1;
a2           := a1;
v2           := v1;
sigma2       := sigma1;
w_filtr2     := w_filtr1;
suma2        := suma;
alfa2        := alfa;
IF stop = TRUE THEN
u := 0.0;
END_IF;
END_PROGRAM
```

Příloha 2 – Regulátor dle J. Berner – kompletní kód

```
VAR_GLOBAL
//vetrak + prutokomer
//(*
prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //výstup z PLC
   u AT ventilator : REAL; //akční zásah
  y AT prutokomer; //výstup soustavy
//*)
//zarovka + termistor
(*
termistor AT r0_p3_AI2.ENG : REAL; //vstup do PLC
zarovka   AT r0_p3_AO0.ENG : REAL; //výstup z PLC
 u AT zarovka : REAL; //akční zásah
 y AT termistor; //výstup soustavy
*)
//Rozteznik
(*
prutokomer AT r0_p3_AI0.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO0.ENG : REAL; //výstup z PLC
   u AT ventilator : REAL; //akční zásah
  y AT prutokomer; //výstup soustavy
//*)
END_VAR
VAR_GLOBAL CONSTANT
  BUF_MAX_LEN : UINT:= 3000;
END_VAR
PROGRAM prgMain
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
  //INIT
  d1 : REAL;               //zadaná velikost akčního zásahu při horní poloze relé
  d2 : REAL;               //zadaná velikost akčního zásahu při dolní poloze relé
  gama : REAL;               //stupen asymetrie rele
  n0 : REAL;               //úroveň šumu
  nabehova : UINT :=0;       //pocitani nabehovych hran
  spadova : UINT :=0;        //pocitani spadovych hran
  max_y_dev : REAL := 0.0;   //maximalni vychylka nad zadanou hodnotou
  min_y_dev : REAL := 0.0;   //maximalni vychylka pod zadanou hodnotou
  min_y :REAL := 1020.0;     //minimalni hodnota vystupu pri konstantni hodnote vstupu
  max_y : REAL := 0.0;       //maximalni hodnota vystupu pri konstantni hodnote vstupu
  ny : REAL := 5.0;          //koeficient pro vypocet y_mindev
  a : REAL := 2.0;           //koeficient pro vypocet h
  ul_y_max : REAL;           //predchozi hodnota max_y_dev
  ul_y_min : REAL;           //predchozi hodnota min_y_dev
  cas_init : TIME;           //exponent
  cas_start : DT;            //čas začátku růstu exponenciely
  cas_ted : DT;              //aktuální čas
  stav_init : UINT := 0;     // stav v průběhu výpočtu
  u_maxs : REAL :=10.0;      // maximální dovolená hodnota vstupu
  u_mins : REAL := 0.0;      // minimalní dovolená hodnota vstupu
  static_tim2 : TON;         //časovač hlídající ustálení soustavy
  static_tim3 : TON;         //časovač hlídající ustálení soustavy
  static_tim4 : TON;         //časovač hlídající ustálení soustavy
  static_delta : REAL := 0.3;//přesnost ustálení soustavy
```

```
y_0: REAL;              //předchozí hodnota výstupu
y_maxdev : REAL;        //horni hranice pasma pro vystup soustavy pri zapojeni rele
y_mindev : REAL;        //dolni hranice pasma pro vystup soustavy pri zapojeni rele
u_last : REAL;          //predchozi hodnota akčního zásahu
y_last : REAL;          //předchozí hodnota výstupu soustavy
u_last_p : REAL;        //pomocně uložená hodnota předchozího akčního zásahu
u_0 : REAL := 2.0;      //zvolená výchozí hodnota u, nebo určená výpočtem
u_1 : REAL;             //hodnota akčního zásahu pro výpočet u_0
u_2 : REAL;             //hodnota akčního zásahu pro výpočet u_0
y_1 : REAL;             //hodnota výstupu soustavy pro výpočet u_0
y_2 : REAL;             //hodnota výstupu soustavy pro výpočet u_0
ustaleni : INT :=0;     //case
pocet : INT := 0;       //počítač vzorků pro výpočet průměru
suma_y : REAL;          //suma vzorků výstupů soustavy
y_last1 : REAL;         //předchozí hodnota výstupu soustavy
y_last_p : REAL;        //pomocná předchozí hodnota výstupu soustavy
uloz :BOOL := FALSE;    //pomocná proměnná pro načasování uložení vzorku
dlouhy_start : BOOL:=FALSE;//tlacitko pro nastavení nutnosti vypočíst u_0
w_pom : REAL;           //paměť pro uložení hodnoty výstupu
coef : REAL;            //koeficient pro přepočet poloh relé
podminky : INT := 0;    //case
k_exp : REAL := 1;      //koeficient pro úpravu doby náběhu exponenciely

//BERNER
stav : INT :=0;         //stádium výpočtu
w : REAL :=5.0;         //žádaná hodnota
h : REAL;               //odchylka od žádané hodnoty w
h1 : REAL;              //horní odchylka od žádané hodnoty w
h2 : REAL;              //dolní odchylka od žádané hodnoty w
t_on_p : TIME;          //doba horní polohy relé - pomocná
t_off_p : TIME;         //doba dolní polohy relé - pomocná
t_on : TIME;            //doba horní polohy relé
t_off : TIME;           //doba dolní polohy relé
t_perioda :TIME;        // doba trvání jedné periody
t_1 : DT;               //časová značka přepnutí na d1
t_2 : DT;               //časová značka přepnutí na d2
t_1_p : DT;             //časová značka přepnutí na d1 - pomocná
t_2_p : DT;             //časová značka přepnutí na d2 - pomocná
buf_i :UINT := 2;       //index pouzivaneho bufferu
buf_x :UINT;            //index v bufferu
buf_x_old : UINT;       //index predchoziho vzorku v bufferu
buf_len : ARRAY [0..2] OF UINT; // velikost bufferu
buf_u_switch : ARRAY [0..2] OF UINT; //index na kterém došlo k přepnutí relé
buffers : ARRAY [0..2,0..BUF_MAX_LEN-1] OF REAL; //prostor pro ukládání vzorků
hlaska : STRING;        //textove pole pro vypis informaci pro uzivatele
hlaska1 : STRING;       //textove pole pro vypis informaci pro vyvojare
start : BOOL:=FALSE;    //startovaci tlacitko
cas: TIME;              //čas od počátku periody
dif01 : REAL;           //prumerne rozdily mezi buffery 0 a 1
dif02 : REAL;           //prumerne rozdily mezi buffery 0 a 2
dif12 : REAL;           //prumerne rozdily mezi buffery 1 a 2
difa : REAL;            //suma rozdilu
difb : REAL;            //suma rozdilu
iT : REAL :=0.01;       //vzorkovací perioda numerické integrace
chyba : REAL := 0.1;    //povolené rozdíly naměřených průběhů při 1 periodě
ro : REAL;              //pomer casu po ktery je rele v horni respektive dolni poloze
tau : REAL;             //nomralizovane dopravni zpozdeni
ro_p : REAL;            //pomer casu po ktery je rele v horni respektive dolni poloze  - pomocný
tau_p : REAL;           //nomralizovane dopravni zpozdeni - pomocné
```

```
 I_u : REAL;              //integrál z u na periodě Tp
 I_y : REAL;              //integrál z y na periodě Tp
 i : UINT :=0;            //index pro vypocty numerickeho integralu
 K_p : REAL;              //statická citlivost
 //k_v : REAL;            //statická citlivost pro astaticky model
 L_div_T :REAL;           //pomer (tau)/(1-tau)
 T : REAL;                //časová konstanta
 L : REAL;                // dopravní zpoždění
 K : REAL;                //proporcionalni
 T_i :REAL;               //integracni
 T_d :REAL;               //derivacni
 faze : INT := 0;         //prechod mezi nastavenim rele, merenim a rizenim
 regulator: fbSimplePID;    //PID regulator z knihovny
 e : REAL;                //regulacni odchylka
 w_min : REAL;            //minimalni hodnota pro zobrazeni hodnot pro prepnuti rele
 w_max : REAL;            //maximalni hodnota pro zobrazeni hodnot pro prepnuti rele

 //kontrola ustaleni
  timer1 : TON;           //časovač pro kontrolu ustálení soustavy
  timer2 : TON;           //časovač pro kontrolu ustálení soustavy
  timer3 : TON;           //časovač pro kontrolu ustálení soustavy
  y1_k : REAL;            //průměrná hodnota výstupu po dobu časovače
  y2_k : REAL;            //průměrná hodnota výstupu po dobu časovače
  y3_k : REAL;            //průměrná hodnota výstupu po dobu časovače
  y_suma_k : REAL;        //suma výstupů soustavy po dobu časovače
  pocitadlo : REAL;       //počet výstupů soustavy po dobu časovače
  klid : INT;             //case
  casovac : BOOL := FALSE; //proměnná pro nulování časovače
  prepnuti : BOOL := FALSE;//proměnná pro nulování časovače
  d_p : REAL;
  d_2p :REAL;
  min_roz :REAL;
  max_roz :REAL;
 END_VAR
 VAR_TEMP
 END_VAR

//zobrazeni hranic do graphmakeru
  w_min := w-h2;
  w_max := w+h1;

  CASE faze OF

0://FAZE NASTAVENI RELE////////////////////////////////////////////////////////////////////////////////////////////////////////////

CASE stav_init OF

  0:  // cekani na stisknuti tlacitka START
  IF start THEN
     stav_init := 1;
  END_IF;

  1: //kontrola zadaných údajů

  IF u_mins > u_maxs THEN
     hlaska := 'Hodnota u_maxs musi byt vetsi nez u_mins.';
     start:=FALSE;
     stav_init := 0;
     dlouhy_start := FALSE;
```

```
    ELSE
    IF dlouhy_start = FALSE THEN
      IF u_0 >= u_maxs OR u_0 <= u_mins THEN
        hlaska := 'Hodnota u_0 musi lezet mezi u_maxs a u_mins.';
        start:=FALSE;
        stav_init := 0;
        ELSE
        stav_init := 2;
        hlaska := '';
      END_IF;
      ELSE
      stav_init := 2;
      hlaska := '';
    END_IF;

  END_IF;
  IF dlouhy_start THEN
    ustaleni:=0;
    hlaska:= 'Probiha urceni u_0';
    w_pom := w;
    ELSE
    ustaleni:=7;
    w_pom := w;
  END_IF;
7:
    u := u_0;
    hlaska:= 'Ceka se na ustaleny stav';

//---------------------------------------------------------------------------------------
  CASE klid OF
0:
    IF timer1.Q = FALSE THEN
      pocitadlo := pocitadlo +1.0;
      y_suma_k := y_suma_k + y;
    END_IF;

    timer1(IN:= casovac, PT :=T#20s);
    IF timer1.Q THEN
      y1_k := y_suma_k/pocitadlo;
      pocitadlo := 0.0;
      y_suma_k := 0.0;
      klid := 1;
      casovac := FALSE;
    END_IF;

    IF prepnuti THEN
      casovac := TRUE;
      prepnuti := FALSE;
      ELSE
        prepnuti := TRUE;
    END_IF;

1:
    IF timer2.Q=FALSE THEN
      pocitadlo := pocitadlo +1.0;
      y_suma_k := y_suma_k + y;
    END_IF;
    timer2(IN:= casovac, PT :=T#10s);
    IF timer2.Q THEN
```

```
      y2_k := y_suma_k/pocitadlo;
      pocitadlo := 0.0;
      y_suma_k := 0.0;
      klid := 2 ;
      casovac := FALSE;
    END_IF;

    IF prepnuti THEN
      casovac := TRUE;
      prepnuti := FALSE;
      ELSE
         prepnuti := TRUE;
    END_IF;

2:

    IF timer3.Q = FALSE THEN
      pocitadlo := pocitadlo +1.0;
      y_suma_k := y_suma_k + y;

    END_IF;
    timer3(IN:= casovac, PT :=T#10s);
    IF timer3.Q THEN
      y3_k := y_suma_k/pocitadlo;
      pocitadlo := 0.0;
      y_suma_k := 0.0;
      klid:=3;
      casovac := FALSE;
      timer3.Q := FALSE;

    END_IF;

    IF prepnuti THEN
      casovac := TRUE;
      prepnuti := FALSE;
      ELSE
         prepnuti := TRUE;
    END_IF;
3:

    IF (y1_k > y2_k AND y2_k > y3_k) OR (y1_k < y2_k AND y2_k < y3_k) THEN
    y1_k := y2_k;
    y2_k := y3_k;
    klid := 2;
    ELSE
    klid := 4;
    casovac := FALSE;
    END_IF;
//------------------------------------------------------------------------------------------ ustaleni
4:
    pocet := pocet +1;
    suma_y := y +suma_y;

       IF y > max_y THEN
          max_y := y;
          END_IF;
          IF y < min_y THEN
          min_y := y;
          END_IF;
```

```
            pocet := pocet +1;
            suma_y := y +suma_y;

     static_tim4(IN:=casovac, PT :=T#10s);    //časovač po který se měří velikost šumu
     hlaska:= 'Meri se sum';
     IF prepnuti THEN
       casovac := TRUE;
       prepnuti := FALSE;
       ELSE
          prepnuti := TRUE;
    END_IF;
    IF static_tim4.Q THEN
      n0  := (max_y - min_y)/2.0;
       h   := a*n0;
       h1 := h*1.4;
       h2:=h;
       y_0 := (max_y + min_y)/2.0;
       y_mindev := ny * h;
       y_maxdev := 3.0*y_mindev;
       stav_init:=3;
       casovac := FALSE;

       IF dlouhy_start THEN
       w:=w;
       ELSE
       w:= suma_y/INT_TO_REAL(pocet);
       END_IF;
    END_IF;
END_CASE;
    END_CASE;

  3: //nastavení hranic relé, výpočet gamy

    d1:= u_maxs;
    d2:= u_mins;
    static_delta := 1.5*n0;                                          // prepocteni max dovolene vychylky pri ustaleni
    gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));              // vypocet gamy
    stav_init:=4;

  4: //načtení času začátku měření
    hlaska:= 'Nastavuji se hodnoty pro polohy rele';                        //nacteni casu zacatku mereni
   cas_start := GetRTC();
   stav_init := 5;

  5: //exponencielní průběh
   cas_ted := GetRTC();                                             //nacteni aktualniho casu
   cas_init := SUB_DT_DT(cas_ted,cas_start);
   k_exp:=1/3;
   IF u < u_maxs THEN                                                  //kontrola, ze akcni velicina není vetsi nez povolena
      IF y>=(y_maxdev+w) THEN                                           //kontrola, ze vystupni velicina není vetsi nez zadana velicina s
maximalni dovolenou vychylkou
      //IF y>=((3*h1)+w) THEN
         stav_init :=6;
         d_p :=u;
         d_2p:=u_0-((d_p-u_0)*0.7);
         IF u_mins < d_2p THEN
         d2:= d_2p;
         END_IF;
         u:=d2;
```

```
      d1 := d_p;//-------------------------------------------------------------------------------
      gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
    ELSE
       u:= EXP(k_exp*TIME_TO_REAL(cas_init)/10000.0)-1.0+u_0;                    //exponencielni narust akcni veliciny
    END_IF;
  ELSE
     d_2p:=u_0-((d1-u_0)*0.7);
     IF u_mins < d_2p THEN
       d2:= d_2p;
     END_IF;
    stav_init :=6;
    u:=d2;
    spadova := 0;
    t_2_p:=GetRTC();
    t_1_p:=GetRTC();
    gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
  END_IF;

6: //kmitani rele a nastaveni jeho hodnot
  IF nabehova < 2 THEN
    IF u <> d2 THEN                                          //prechod do spodni urovne rele
      IF y >= w+h1 THEN
        u:= d2;

       t_on_p := SUB_DT_DT(t_2_p,t_1_p);
       t_1_p  := GetRTC();

        spadova := spadova +1;
      END_IF;
    END_IF;
    IF u <> d1 THEN                                          //prechod do dolni urovne rele
      IF y <= w-h2 THEN
        u:= d1;

        t_off_p := SUB_DT_DT(t_1_p,t_2_p);
        t_2_p   := GetRTC();
        nabehova := nabehova +1;
      END_IF;
    END_IF;

    IF spadova = 1 AND nabehova = 1 THEN                     //urceni maximalni vychylky vystupu nad zadanou velicinu

      max_y_dev:= MAX(max_y_dev,ABS(y-w));
    END_IF;

    IF spadova = 0 AND nabehova = 1 THEN                     //urceni maximalni vychylky vystupu nad zadanou velicinu
      min_y_dev := MAX(min_y_dev,ABS(w-y));

    END_IF;

  END_IF;

  static_tim2(IN:=ABS(y-y_0)< static_delta, PT :=T#20s);          //časovač pro kontrolu ustaleni - neprekmitnuti rele
  IF ABS(y-y_0) >= static_delta THEN
    y_0 :=y;
  END_IF;

  IF static_tim2.Q THEN                                      //při doběhnutí časovače přechod do dalšího stavu výpočtu/měření
```

```
      IF u = d1 THEN
        hlaska := 'Nelze seridit na zadanou hodnotu je treba snizit w nebo zvysit u_maxs.';
        u := u_mins;
        stav_init :=0;
        faze:=0;
        ustaleni :=0;
        start := FALSE;
      END_IF;

      IF u = d2 THEN
        hlaska := 'Nelze seridit na zadanou hodnotu je treba zvysit w nebo snizit u_mins.';
        u := u_mins;
        stav_init :=0;
        faze:=0;
        ustaleni :=0;
        start := FALSE;
      END_IF;

    END_IF;
                                                    //vypocet novych hodnot d1 a d2
    IF nabehova = 2  AND spadova = 1 THEN

      min_roz := min_y_dev - h2;
      max_roz := max_y_dev - h1;

      nabehova := 1;
      spadova  := 0;

      ro_p      :=      MAX(TIME_TO_REAL(t_on_p),TIME_TO_REAL(t_off_p))/MIN(TIME_TO_REAL(t_on_p),TIME_TO_REAL(t_off_p));
//výpočet kontrolního tau
      tau_p := (gama - ro_p)/((gama-1.0)*((0.35*ro_p)+0.65));

      y_last_p := tau_p;

      IF tau_p < 0.05 THEN    //uprava hranic pro kmitání relé při nevyhovujícím tau

        CASE podminky OF
        0:
        //IF max_y_dev -h1 > 1.0 AND  min_y_dev -h2 > 1.0 THEN
        IF max_y_dev -h2 > 1.0 AND  min_y_dev -h1 > 1.0 THEN
       ////////////////////ZMENA/////////////////////////////////////
          // d1:= u_0 +(d1 - u_0)* 0.9;
          // d2:= u_0 +(d2 - u_0)* 0.9;
           d1:= u_0 +(d1 - u_0)* 0.8;
           d2:= u_0 +(d2 - u_0)* 0.8;
           hlaska1:= 'chyba1';
           IF d1 > u_maxs THEN
             d1:= u_maxs;
           END_IF;
           IF d2 < u_mins THEN
             d2:= u_mins;
           END_IF;
        END_IF;
        IF max_y_dev -h1 < 0.1 AND  min_y_dev -h2 < 0.1 THEN
        //IF max_y_dev -h2 < 0.1 AND  min_y_dev -h1 < 0.1 THEN
        IF d1 = d_p  AND d2 = d_2p THEN

           podminky:= 1;
           RETURN;
```

```
      ELSE
      d1:= u_0 +(d1 - u_0)* 1.1;
      d2:= u_0 +(d2 - u_0)* 1.1;
      hlaska1:= 'chyba2';
      IF d1 > u_maxs THEN
        d1:= u_maxs;
      END_IF;
      IF d2 < u_mins THEN
        d2:= u_mins;
      END_IF;
    END_IF;
    ELSE
    //IF max_y_dev-h2 <0.1 THEN
    IF max_y_dev-h1 <0.1 THEN
      d1:= d1 + 0.1;
      d2:= d2 + 0.1;
      //d1:= d1 - 1.0;
      //d2:= d2 - 1.0;
      hlaska1:= 'chyba3';
      IF d1 > u_maxs THEN
        d1:= u_maxs;
      END_IF;
      IF d2 < u_mins THEN
        d2:= u_mins;
      END_IF;
    END_IF;
    //IF min_y_dev-h1 <0.1 THEN
    IF min_y_dev-h2 <0.1 THEN
      d1:= d1 - 0.1;
      d2:= d2 - 0.1;
      //d1:= d1 + 1.0;
      //d2:= d2 + 1.0;
      hlaska1:= 'chyba4';
      IF d1 > u_maxs THEN
        d1:= u_maxs;
      END_IF;
      IF d2 < u_mins THEN
        d2:= u_mins;
      END_IF;
    END_IF;
  END_IF;

  ////////////////////////////////////////ZMENA/////////////////////////////////////////////////////////////////////
   IF max_y_dev -h1 < 1.0 AND  min_y_dev -h2 < 1.0 AND  max_y_dev -h1 > 0.5 AND  min_y_dev -h2 > 0.5 THEN
      d1:= u_0 +(d1 - u_0)* 0.9;
      d2:= u_0 +(d2 - u_0)* 0.9;
   END_IF;
   IF max_y_dev -h1 < 0.5 AND  min_y_dev -h2 < 0.5 AND  max_y_dev -h1 > 0.1 AND  min_y_dev -h2 > 0.1 THEN
      d1:= u_0 +(d1 - u_0)* 0.95;
      d2:= u_0 +(d2 - u_0)* 0.95;
   END_IF;
  ////////////////////////////////////////ZMENA/////////////////////////////////////////////////////////////////////
 1:
 IF (max_y_dev-h1)>(min_y_dev-h2) THEN
    d1 := u_0 +(d1 - u_0)* 0.9;
    hlaska1:= 'chyba5';
    ELSE
    d2:= u_0 +(d2 - u_0)* 0.9;
    hlaska1:= 'chyba6';
```

```
    END_IF;
    gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
  END_CASE;

     u:=d1;
     tau_p := 0.0;
     t_on_p :=T#0s;
     t_off_p:=T#0s;
     ro_p:=0.0;
     u:=d1;
     stav_init :=6;
     ul_y_max := max_y_dev;
     max_y_dev := 0.0;
     ul_y_min := min_y_dev;
     min_y_dev := 0.0;
     gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
     ELSE
     podminky :=0;
     hlaska:= 'Probiha identifikace soustavy';
     faze :=1;
    END_IF;
    (*ul_y_max := max_y_dev;
    max_y_dev := 0.0;

    ul_y_min := min_y_dev;
    min_y_dev := 0.0; *)
  END_IF;

  END_CASE;
1: // FAZE IDENTIFIKACE ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  CASE stav OF
  0: //nastavení relé do horní polohy
    stav:=1;
    gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2)); //vypočtení stupně asymetrie relé
    u:=d1;
  1: //měření zapojení s relé
    IF w-y > h2 THEN
      IF u <> d1 THEN

          IF dif01 < chyba AND  dif02 < chyba AND dif12 < chyba AND dif01 > 0.0 AND  dif02 >0.0 AND dif12 >0.0 THEN
          stav:=2;
          RETURN;
          END_IF;

        t_on := SUB_DT_DT(t_2,t_1);
        t_1 := GetRTC();

        buf_i := buf_i +1; //posunutí polohy v paměti
            IF buf_i > 2 THEN
              buf_i:=0;
            END_IF;
        buf_len[buf_i]:=0; //----------------------------------------vynulování

        CASE buf_i OF     //vypocet prumernych odchylek hodnot vystupu
            0:
              dif01 := difa/UINT_TO_REAL(buf_len[1]);
              dif02 := difb/UINT_TO_REAL(buf_len[2]);
            1:
              dif01 := difa/UINT_TO_REAL(buf_len[0]);
```

```
            dif12 := difb/UINT_TO_REAL(buf_len[2]);
          2:
            dif02 := difa/UINT_TO_REAL(buf_len[0]);
            dif12 := difb/UINT_TO_REAL(buf_len[1]);
      END_CASE;

      difa := 0.0; //vymazání sumace rozdílů
      difb := 0.0;
      buf_x_old := 1; //vrácení indexu na začátek
    END_IF;
    u := d1;
  END_IF;

  IF w-y <(-h1) THEN
    IF u <> d2 THEN
      t_off := SUB_DT_DT(t_1,t_2);
      t_2 := GetRTC();
      buf_u_switch[buf_i] := REAL_TO_UINT(CEIL(TIME_TO_REAL(cas)*0.001/iT));    //cas prepnuti rele
    END_IF;
    u :=d2;
  END_IF;
  t_perioda :=t_on +t_off;
  cas := SUB_DT_DT(GetRTC(),t_1); //cas od pocatku periody

  buf_x := REAL_TO_UINT(CEIL(TIME_TO_REAL(cas)*0.001/iT)); //výpočet indexu  bufferu pro uložení vzorku
  IF buf_x < BUF_MAX_LEN THEN
    IF buf_x <> buf_x_old THEN  //čekání na určenou chvíli pro uložení vzorku
      buf_x_old := buf_x;
      CASE buf_i OF
      0:
        IF buf_x <= buf_len[1] THEN
          difa:= difa + ABS(buffers[1,buf_x]-y);      //vypocet sumy odychlek vzorku od aktualni hodnoty

        END_IF;
        IF buf_x <= buf_len[2] THEN
          difb:= difb + ABS(buffers[2,buf_x]-y);      //vypocet sumy odychlek vzorku od aktualni hodnoty
        END_IF;
      1:
        IF buf_x <= buf_len[0] THEN
          difa:= difa + ABS(buffers[0,buf_x]-y);      //vypocet sumy odychlek vzorku od aktualni hodnoty
        END_IF;
        IF buf_x <= buf_len[2] THEN
          difb:= difb +ABS(buffers[2,buf_x]-y);       //vypocet sumy odychlek vzorku od aktualni hodnoty
        END_IF;
      2:
        IF buf_x <= buf_len[0] THEN
          difa:= difa +ABS(buffers[0,buf_x]-y);       //vypocet sumy odychlek vzorku od aktualni hodnoty
        END_IF;
        IF buf_x <= buf_len[1] THEN
          difb:= difb +ABS(buffers[1,buf_x]-y);       //vypocet sumy odychlek vzorku od aktualni hodnoty
        END_IF;
      END_CASE;
    END_IF;
    buffers[buf_i,buf_x] :=y; //uložit vzorek
    buf_len[buf_i] := buf_x;  //ulozeni delky bufferu
  END_IF;

2: //určení časů, výpočet Half-period ratio a normalized time delay
  t_perioda := REAL_TO_TIME(UINT_TO_REAL(buf_len[buf_i])*iT*1000.0);
```

```
  t_on := REAL_TO_TIME(UINT_TO_REAL(buf_u_switch[buf_i])*iT*1000.0);
  t_off := t_perioda - t_on;
  ro := MAX(TIME_TO_REAL(t_on),TIME_TO_REAL(t_off))/MIN(TIME_TO_REAL(t_on),TIME_TO_REAL(t_off));
  tau := (gama - ro)/((gama-1.0)*((0.35*ro)+0.65));

  IF tau <= 1.0 AND tau >= 0.05 THEN                    //kontrola spravnosti tau
    stav :=3;
    hlaska := '';

    ELSE
      // IF tau<= 0.05 AND tau >=-0.05 THEN
        coef:= 0.9;
      //  END_IF;
      //  IF tau <= -0.05 AND tau >=-2.0 THEN
      //  coef:= 0.6;
      //  END_IF;
      //  IF tau <= -2.0 THEN
      //  coef:= 0.3;
      //  END_IF;
       IF (TIME_TO_REAL(t_on)-TIME_TO_REAL(t_off))<0.0 THEN
          d1 := u_0+((d1-u_0)*coef);
          gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
       END_IF;
       IF (TIME_TO_REAL(t_on)-TIME_TO_REAL(t_off))>0.0 THEN
         d2 := u_0+((d2-u_0)*coef);
         gama := MAX(ABS(d1-u_0), ABS(u_0-d2))/MIN(ABS(d1-u_0),ABS(u_0-d2));
       END_IF;
    stav :=0;
    hlaska := 'Mereni se opakuje, program prepocita meze rele';
    dif01 := 0.0;
    dif02 := 0.0;
    dif12 := 0.0;
  END_IF;

 3: // integrál z u na Tp
   hlaska:= 'Probihaji vypocty parametru regulatoru';
   u := u_0;
   I_u := d1 * (UINT_TO_REAL(buf_u_switch[buf_i])*iT) + d2 * (UINT_TO_REAL(buf_len[buf_i]-buf_u_switch[buf_i])*iT);
   stav := 4;

 4: // integrál z y na Tp
   I_y := I_y + ((buffers[buf_i,i])+(buffers[buf_i,i+1]))/2.0*iT;
   i := i+1;
   IF i = buf_len[buf_i] THEN
     stav := 5;
   END_IF;

 5: //výpočet parametrů modelu

   K_p := I_y/I_u;
   L_div_T :=tau/(1.0-tau);
   T    :=    (UINT_TO_REAL(buf_u_switch[buf_i])*iT)/    LN(((h1+h2)/2.0/ABS(K_p)-d2+    (d1+d2)*    EXP(L_div_T    ))/(d1-
((h1+h2)/2.0/ABS(K_p))));
   L := T*L_div_T;
   stav := 6;
  (* ELSE
   k_v                                                                                                                                                    :=
2*I_y/((TIME_TO_REAL(t_on)/1000.0)*(TIME_TO_REAL(t_off)/1000.0)*(d1+d2))+((h1+h2)/(d1*TIME_TO_REAL(t_on)/1000.0));
   L := (d1*(TIME_TO_REAL(t_on)/1000.0)-(h1+h2)/k_v)/(d1-d2);
```

```
    stav := 6;
    END_IF; *)

  6: //nastavení PID regulátoru
    K := (0.2*L+0.45*T)/(K_p*L);
    T_i := (0.4*L+0.8*T)*L/(L+0.1*T);
    T_d := (0.5*L*T)/(0.3*L+T);
    faze := 2;
    start:= FALSE;
    dlouhy_start:= FALSE;
    END_CASE;

2: // RIZENI//////////////////////////////////////////////////////////////////////////////////////////////////////////////
   hlaska:= 'Soustava je rizena';
   w:=w_pom;
   e := w-y;
   regulator(y:=y, w:=w, u_man := u_0, T := 0.01, max_u:=u_maxs, min_u:=u_mins, Gain := K, Ti:=T_i, Td:=T_d, u=>u, e=>e);
 END_CASE;
END_PROGRAM
```

Příloha 3 – regulátor Foxboro EXACT – kompletní kód

```
VAR_GLOBAL
//vetrak + prutokomer
//(*
prutokomer AT r0_p3_AI1.ENG : REAL; //vstup do PLC
  ventilator AT r0_p3_AO1.ENG : REAL; //výstup z PLC
   u AT ventilator : REAL; //akční zásah
  y AT prutokomer; //výstup soustavy
//*)
//zarovka + termistor
(*
termistor AT r0_p3_AI2.ENG : REAL; //vstup do PLC
zarovka   AT r0_p3_AO0.ENG : REAL; //výstup z PLC
 u AT zarovka : REAL; //akční zásah
 y AT termistor; //výstup soustavy
//*)
//rozteznik
(*
termistor AT r0_p3_AI0.ENG : REAL; //vstup do PLC
zarovka   AT r0_p3_AO0.ENG : REAL; //výstup z PLC
 u AT zarovka : REAL; //akční zásah
 y AT termistor; //výstup soustavy
//*)
END_VAR

PROGRAM prgMain
  VAR
  PB : REAL;
  Ti : REAL;
  Td : REAL;
  wmax : REAL;
  kp : REAL;
  L : REAL;
  T : REAL;
  w : REAL := 5.0;
  w_pom : REAL;
  y_pom_T : REAL;
  w_skok : REAL;
  e : REAL;
  u0 : REAL := 5.0;
  max_y : REAL := -1000.0;
  min_y : REAL := 1000.0;
  suma_y : REAL := 0.0;
  pocitadlo : REAL := 0.0;
  u_maxs : REAL := 10.0;
  u_mins : REAL := 0.0;
  t_zacatek_L : DT;
  t_konec_L : DT;
  t_konec_T : DT;
  stav : INT := 0;
  start : BOOL := FALSE;
  ustaleni :ustaleni1;
  regulator : fbSimplePID;
  casovac : BOOL := TRUE;
```

```
timer1 : TON;
timer2 : TON;
timer3 : TON;
hlaska : STRING := ' ';
END_VAR
CASE stav OF
0:
//Cekani na stisk tlacitka START
IF start = TRUE THEN
stav := 1;
END_IF;
1:
//Kontrola zadanych parametru
 IF u_mins > u_maxs THEN
  hlaska := 'Hodnota u_maxs musi byt vetsi nez u_mins.';
  start:=FALSE;
  stav := 0;
  ELSE
    IF u0 >= u_maxs OR u0 <= u_mins THEN
      hlaska := 'Hodnota u_0 musi lezet mezi u_maxs a u_mins.';
      start:=FALSE;
      stav := 0;
      ELSE
      hlaska := '';
      //Cekani na ustalení systemu v pracovním bode
      u := u0;
      ustaleni(vstup := y);
      IF ustaleni.ustaleno = TRUE THEN
      stav := 2;
      ustaleni.ustaleno := FALSE;
      END_IF;
    END_IF;
  END_IF;
2:
//Mereni sumu a vypocet vychoziho bodu
timer1(IN:= casovac, PT :=T#10s);
suma_y := suma_y+y;
pocitadlo := pocitadlo + 1.0;
       IF y > max_y THEN
         max_y := y;
         END_IF;
         IF y < min_y THEN
         min_y := y;
         END_IF;
IF timer1.Q THEN
stav := 3;
w_pom := suma_y/pocitadlo;
suma_y := 0.0;
pocitadlo := 0.0;
END_IF;

3:
//Skok o desetinu rozsahu
u := u0 + ((max_y-min_y)/10.0);
t_zacatek_L := GetRTC();
```

```
 stav:= 4;

 4:
 IF y > ((max_y-min_y)/2.0)+w_pom THEN
 t_konec_L := GetRTC();
 stav:= 5;
 END_IF;
 5:
 //mereni smernice tecny prechodove charakteristiky
 timer3(IN := casovac, PT:=T#5s);

 IF timer3.Q THEN
 y_pom_T := y;
 t_konec_T := GetRTC();
 stav:= 6;
 END_IF;
 6:
 ustaleni(vstup := y);
 IF ustaleni.ustaleno = TRUE THEN
 suma_y := suma_y+y;
 pocitadlo := pocitadlo + 1.0;
 timer2(IN:= casovac, PT :=T#10s);
 IF timer2.Q THEN
 w_skok := suma_y/pocitadlo;
 ustaleni.ustaleno := FALSE;
 stav := 7;
 END_IF;
 END_IF;
 7:
 //odecty z prubehu
 kp := (w_skok - w_pom)/((max_y-min_y)/10.0);
 L := TIME_TO_REAL(SUB_DT_DT(t_konec_L,t_zacatek_L))/1000.0;
 T := (w_skok - w_pom)*(TIME_TO_REAL(SUB_DT_DT(t_konec_T,t_konec_L))/1000.0/(y_pom_T-w_pom));
 stav := 8;

 8:
 //vypocet parametru pro PID regulator
 PB := 120.0*kp*L/T;
 Ti := 1.5*L;
 Td := Ti/6;
 wmax := 5*L;
 stav := 9;

 9:
 //Rizeni
  e := w-y;
  regulator(y:=y, w:=w, u_man := u0, max_u:=u_maxs, min_u:=u_mins,
  Gain := kp, Ti:=Ti, Td:=Td, u=>u, e=>e);
 END_CASE;
END_PROGRAM
```