Univerzita Pardubice
Fakulta elektrotechniky a informatiky

# Regulace s využitím PLC Siemens S-1200

Bc. Patrick Horáček

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

# ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Patrick Horáček** |
| Osobní číslo: | **I16171** |
| Studijní program: | **N2612 Elektrotechnika a informatika** |
| Studijní obor: | **Komunikační a řídicí technologie** |
| Téma práce: | **Regulace s využitím PLC Siemens S-1200** |
| Zadávající katedra: | **Katedra elektrotechniky** |

## Zásady pro vypracování

Cíl: Cílem je vytvořit řídicí aplikaci pro PLC Siemens S-1200. Pro návrh regulátoru bude použita vybraná metoda vycházející z modelu řízené soustavy.
Obsah teoretické části: Teorie pro návrh regulátoru a logického řízení. Programovatelné logické automaty a inteligentní relé – hardware a programování. HMI interface.
Obsah implementační části: Konfigurace PLC, vytvoření programu pro PLC i HMI panel, který umožní v manuálním režimu ovládání soustavy a v režimu automat bude probíhat regulace pomocí PID regulátoru. Popis metody návrhu regulátoru. Regulační experimenty.

Rozsah pracovní zprávy: **40 stran A4**
Rozsah grafických prací: **15**
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

L. Šmejkal, M. Martinásková, PLC a automatizace 1, Základní pojmy, úvod do programování, BEN – technická literatura, Praha 1999
L. Šmejkal, PLC a automatizace 2, Sekvenční logické systémy a základy fuzzy logiky, BEN = technická literatura, Praha 2005
M. Martinásková, Programovací jazyky pro PLC, Automatizace, ročník 47, číslo 6, strana 380, 2004

Vedoucí diplomové práce: **Ing. Daniel Honc, Ph.D.**
Katedra řízení procesů

Datum zadání diplomové práce: **15. října 2018**
Termín odevzdání diplomové práce: **23. srpna 2019**

L.S.

**Ing. Zdeněk Němec, Ph.D.**
děkan

**Ing. Jan Pidanič, Ph.D.**
vedoucí katedry

V Pardubicích dne 15. listopadu 2018

Prohlašuji:

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

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

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

V Pardubicích dne 30. 8. 2019

Bc. Patrick Horáček

# PODĚKOVÁNÍ

Děkuji vedoucímu práce panu Ing. Danielu Honcovi, Ph.D., za poskytnuté informace, ochotu a především za čas, který mi věnoval při tvorbě této diplomové práce. Dále bych chtěl poděkovat mé rodině za podporu během mého studia.

**ANOTACE**

Tato diplomová práce se zaměřuje na návrh PID regulátoru pomocí metody Ziegler-Nichols, experimentální metody „pokus-omyl“, Kuhnovy metody a jeho realizaci v PLC. V teoretické části jsou popsané použité metody, konfigurace PLC a programování PLC. V praktické části byl vytvořen program v PLC pro řízení zadané soustavy společně s vizuální podobou v HMI. Vytvořená aplikace má možnost přepínání mezi dvěma režimy pro automatické a manuální řízení.

**KLÍČOVÁ SLOVA**

PLC, PID, Regulátor, TIA Portal, HMI

**TITLE**

Control with PLC Siemens S-1200

**ANNOTATION**

This diploma thesis is focused on tuning PID controller by using Ziegler-Nichol method, experimental method, Kuhn’s method and its implementation in PLC. Used methods, configuration of PLC and programming PLC are described in theoretical part. In practical part, program has been created in PLC with graphical user interface developed in HMI to control system. Created application can switch between two modes according to automatic or manual control.

**KEYWORDS**

PLC, PID, Controller, TIA Portal, HMI

# OBSAH

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# SEZNAM ZKRATEK

| | |
|---|---|
| PID | Proporcionálně integračně derivační |
| D/A | Digitálně-analogový |
| URO | Uzavřený regulační obvod |
| ORO | Otevřený regulační obvod |
| FCH | Frekvenční charakteristika |
| ZN | Ziegler-Nicholson |
| LED | Světelná dioda |
| AND | Logický součin |
| OR | Logický součet |
| XOR | Exklusivní logický součet |
| NAND | Negovaný logický součin |
| NOR | Negovaný logický součet |
| TTL | Logika digitálních integrovaných obvodů |
| LO | Logické obvody |
| SLO | Sekvenční logický obvod |
| PLC | Programovatelný logický automat |
| LAD | Ladder (název programovacího jazyka) |
| FBD | Function blok diagram (název programovacího jazyka) |
| SCL | Structured control language (název programovacího jazyka) |
| OB | Organizační blok |
| FC | Funkce |
| FB | Funkční blok |
| DB | Datový blok |
| STL | Statement list |
| HMI | Human machine interface |
| VB | Visual Basic (název programovacího jazyka |
| USB | Universal serial bus |
| DC | Stejnosměrné napětí |
| AUTO | Automatický režim |
| MAN | Manuální režim |

# ÚVOD

Tato práce se zaobírá návrhem PID regulátoru pro zadanou soustavu třetího řádu. PID regulátory jsou nejčastěji používanými regulátory v průmyslu. Bohužel velmi často jsou špatně nastavené nebo používají nastavení z výroby. Cílem práce je ověřit použitelnost teoretických metod pro nastavení PID regulátoru na základě znalosti přechodové charakteristiky a ověřit použitelnost návrhu aplikací v PLC.

V teoretické části této práce je popsán spojitý a diskrétní popis PID regulátoru a význam jednotlivých složek PID regulátoru. Dále budou popsány různé metody získání PID parametrů. Další kapitola je věnována Logickému řízení, tedy logickým funkcím a jejich minimalizaci. Následně bude popsán PLC a jeho konfigurace. Práce se dále zaobírá programovacími bloky a jazyky v prostředí aplikace TIA Portal od výrobce Siemens. V neposlední řadě je popsán návrh uživatelských obrazovek v HMI.

V praktické části, která navazuje na teoretickou, bude popsán použitý hardware a software. Následně v prostředí aplikace Matlab jsou demonstrovány zvolené metody za účelem získání PID parametrů. Pro regulace v prostředí Matlab je použito měřící karty Labjack pro analogové vstupy a výstupy. V poslední části bude vytvořen program PLC a HMI panel, který umožní v manuálním režimu ovládání soustavy a v režimu automat bude probíhat regulace pomocí PID regulátoru v PLC. Následně budou zobrazeny výsledky jednotlivých metod, jak v prostředí Matlab, tak i ve vytvořené aplikaci v PLC.

# 1 PID REGULÁTOR

Ačkoliv se nejedná o jedinou možnost, v praxi se nejčastěji využívá variant PID regulátoru, který akční zásah vyhodnocuje na základě regulační odchylky $e = w - y$ a jejího integrálu, případně derivace. Výhodami jsou:

- jednoduchost
- univerzálnost
- snadná realizovatelnost s elektronickými obvody

Tento přístup se osvědčil i pro regulaci složitých nelineárních systémů, třebaže v tomto případě kvalita výsledných regulačních průběhů není zaručena.

Obecný tvar PID regulátoru:

$$u(t) = r_0 e(t) + r_1 \int_0^t e(t)dt + r_2 e'(t)$$

kde $r_i$ jsou parametry. Regulátor tedy obsahuje tři složky:

- proporcionální (P)
- integrační (I)
- derivační (D)

PID regulátor je dynamický systém, jeho přenos je:

$$R(s) = r_0 + r_1 \frac{1}{s} + r_2 s$$

Často se využívají zjednodušené varianty PID regulátoru, které mají speciální označení:

P-regulátor (pouze zesílená záporná zpětná vazba) - $r_1 = r_2 = 0$

PD-regulátor (přidána derivační složka) - $r_1 = 0$

PI-regulátor (přidána integrační složka) - $r_2 = 0$

I-regulátor (samotná integrační složka) - $r_0 = r_2 = 0$

Případ $r_0 = r_1 = 0$ se nevyužívá (nerealizuje zpětnou vazbu).

V praxi se většinou pracuje s tvarem:

$$u(t) = r\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D e'(t)\right]$$

který je jednodušší pro praktickou realizaci. $T_I$ *a* $T_d$ jsou integrační a derivační časová konstanta. Přenos regulátoru je v tomto případě:

$$R(s) = r\left(1 + \frac{1}{T_I s} + T_D s\right)$$

kde položením $T_D = 0$ *a* $T_I \to \infty$ lze získat již zmiňovaný P-regulátor. [1]

## 1.1 Význam složek PID regulátoru

**P-složka**

P-složka slouží k zesílení zpětné vazby. Čím větší zesílení, tím je rychlejší regulační děj, ale pro příliš vysoké hodnoty je kmitavý a může být nestabilní a může být nestabilní (rozkmit výstupní hodnoty narůstá do nekonečna).

U statických soustav samotná P-složka nezaručí dosažení žádané hodnoty, neboť pro nenulovou hodnotu výstupní veličiny je nutný nenulový výstup regulátoru, tedy nenulová regulační odchylka. Odchylka je tím menší, čím je větší zesílení $r$.

**I-složka**

I-složka umožňuje dosáhnout nulové regulační odchylky i pro statické soustavy. Integrační složka, ale zvyšuje řád a prodlužuje regulační děj. Jestliže je přítomna integrační složka, regulátor obsahuje nulový pól, tj. je ve tvaru:

$$R(s) = \frac{P(s)}{s}$$

Ustálená hodnota regulační odchylky pro $\lim_{t\to\infty} w(t) = w_\infty$ pak je

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0} F_{ew}(s).w_\infty = \lim_{s\to 0} \frac{1}{1 + \frac{P(s)}{s}S(s)} w_\infty = \lim_{s\to 0} \frac{s}{s + P(s)S(s)} w_\infty = 0$$

U soustav s astatismem může být zřejmě integrační složka vynechána, protože nulový pól obsahuje už samotná soustava.

**D-složka**

D-složka urychluje regulační pochod, zvláště u soustav vyšších řádů, popřípadě soustav s dopravním zpožděním, kde se účinek akční veličiny projevuje až po uplynutí určité doby (např. dopravní pás). Umožňuje zpětné vazbě reagovat s určitým předstihem. V ustáleném stavu její vliv zmizí.

Dodáním derivační složky se sníží relativní řád přenosu otevřené smyčky $R(s)S(s)$. U systémů vyšších řádů se tím zmenší prodleva, než systém začne reagovat.

Derivační člen je ale idealizovaný a nejde přímo prakticky realizovat, protože jeho amplitudová charakteristika $A(\infty) \rightarrow \infty$ pro $\omega \rightarrow \infty$, a proto u praktické implementace nutně musí dojít ke zkreslení (na rozdíl od integrační složky, která naopak vyšší frekvence tlumí).

Nejběžnější praktická implementace derivační složky je založena na členu se setrvačností prvního řádu ve tvaru:

$$\frac{T_D s}{\alpha T_D s + 1}$$

Kde doručovaná hodnota $\alpha$ je cca v rozsahu 0,05-0,2. Setrvačný člen je spíše výhodou, protože potlačuje vysoké frekvence, které často odpovídají rušivým složkám v signálu. [1]

## 1.2 Číslicový PID regulátor

Při řízení počítačem se využívá tvaru regulátoru, který pracuje s posloupností hodnot. Počítač v pravidelných intervalech vzorkuje (tj. snímá hodnoty) výstupní veličiny, určuje akční zásah a výsledek zapisuje do D/A převodníku, který převádí číslicovou informaci na analogovou hodnotu, která je přivedena na vstup systému. Obvykle se předpokládá konstantní perioda vzorkování Δ a že doba zpracování výstupní veličiny i doba převodu na hodnotu akční veličiny jsou zanedbatelně malé. [1]

Číslicová verze vychází ze spojité verze – opět reaguje proporcionálně, integračně a derivačně vzhledem k aktuální regulační odchylce e(k). Akční veličina je kvantovaná podle bitové hloubky použitého digitálně-analogového převodníku a datového typu použitého pro výpočet regulačního zákona. Integrál a derivace v rovnici spojitého PID regulátoru

$$u(t) = k_p \left[ e(t) + \frac{1}{T_I} \int_0^t e(\tau) d\tau + T_D \frac{de(t)}{dt} \right]$$

jsou nahrazeny numerickým výpočtem (součtem ploch obdélníků u integrálu a podílem prvních diferencí u derivace)

$$\int_0^t e(\tau)d\tau \approx T_s \sum_{i=1}^{k} e(i), \qquad \frac{de(t)}{dt} \approx \frac{e(k)-e(k-1)}{T_s}$$

**Polohovací tvar číslicového PID regulátoru** je

$$u(k) = k_p \left[ e(k) + \frac{T_s}{T_I} \sum_{i=1}^{k} e(i) + T_D \frac{e(k)-e(k-1)}{T_s} \right]$$

Při výpočtu aktuálního akčního zásahu je potřeba mít informaci o sumě regulačních odchylek, aktuální a minulé regulační odchylce. Z praktických důvodů je v případě použití I složky vhodnější přírůstkový tvar PID regulátoru. Získáme jej tak, že napíšeme předcházející rovnici o jeden interval vzorkování posunutou do minulosti

$$u(k-1) = k_p \left[ e(k-1) + \frac{T_s}{T_I} \sum_{i=1}^{k-1} e(i) + T_D \frac{e(k-1)-e(k-2)}{T_s} \right]$$

Od této rovnice odečteme rovnici polohového tvaru číslicového PID regulátoru, čímž získáme

$$u(k)-u(k-1) = k_p \left[ e(k)-e(k-1) + \frac{T_s}{T_I} e(k) + T_D \frac{e(k)-2e(k-1)+e(k-2)}{T_s} \right]$$

Po úpravě získáme **přírůstkový tvar číslicového PID regulátoru**

$u(k) = u(k-1) + k_p \left(1 + \frac{T_s}{T_I} + \frac{T_D}{T_s}\right) e(k) + k_p \left(-1 + \frac{2T_D}{T_s}\right) e(k-1) + k_p \frac{T_D}{T_s} e(k-2)$, kde

$q_0 = k_p \left(1 + \frac{T_s}{T_I} + \frac{T_D}{T_s}\right), q_1 = k_p \left(-1 + \frac{2T_D}{T_s}\right), q_2 = k_p \frac{T_D}{T_s}$ , tedy

$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2)$, kde

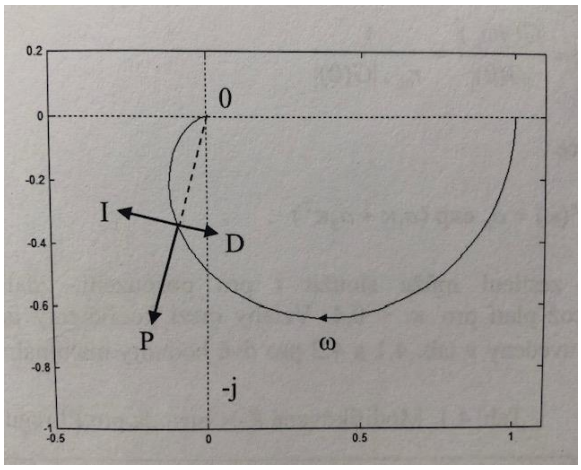
$\triangle u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2)$

Při výpočtu akčního zásahu použijeme akční zásah v minulém intervalu vzorkování a upravíme jej o hodnotu △u(k), která využívá informaci o aktuální, minulé a předminulé regulační odchylce. U tohoto tvaru regulátoru se jednoduše zajistí antiwindup, protože když dojde k omezení akční veličiny, v příštím kroku použijeme zrealizovaný akční zásah – nedochází k integraci v I složce. Volbou $T_D = 0$ získáme číslicový PI regulátor. Mohli bychom získat i P a PD regulátor, ale pro tyto varianty je polohový tvar vhodnější. [2]

# 2 NÁVRH REGULÁTORU

Dnes již existuje velké množství metod pro nastavení parametrů PID regulátorů a stále se publikují další. Některé metody vycházejí ze znalosti modelu regulované soustavy, jiné využívají jednoduché typy aproximačních přenosů (např. soustavu 1. řádu s dopravním zpožděním), další metody jsou založeny na jednoduchých experimentech (např. měření přechodové charakteristiky) atd. Jakou metodu tedy zvolit? Obecně je možné konstatovat:

- Úlohy automatického řízení jsou natolik rozmanité, že nelze vybrat ideální metodu, vhodnou pro všeobecné použití.
- Metody vyžadují různé počáteční znalosti o regulované soustavě. Z toho logicky vyplývá, že lze lépe nastavit regulátor pro soustavu, u které je více informací o jejím dynamickém chování.
- Na průběh regulačního pochodu jsou kladeny různé požadavky podle typu regulované soustavy, a proto jsou výhodné metody, které umožňují zadávat kvalitu regulačního pochodu.
- Různými kombinacemi parametrů regulátoru se docílí podobné chování URO. Odpovídá tomu i často uváděný fakt, že velké procento PID regulátorů v průmyslu má konstanty nastavené výrobcem. Obecně platí, že bod na FCH, vyjadřující chování ORO na zvolené frekvenci, může být posunut změnou parametrů PID regulátoru do požadovaného místa (viz. Obr. č. 1). Výsledný posun je dán vektorovým součtem působení jednotlivých složek, což má více možností řešení. Obvyklý praktický postup u jednoduchých soustav je, že se navrhne podle nějaké metody hrubé nastavení regulátoru a potom se regulátor v provozu doladí podle zkušeností obsluhy.




Obrázek 1: Posun bodu FCH ORO jednotlivými složkami PID regulátoru

Jako stále nejčastěji publikovaná a využívaná bude dále uvedena frekvenční Zieglerova-Nicholsova (ZN) metoda návrhu PID regulátoru pro neznámý model soustavy, založená na experimentálním zjišťování tzv. kritických hodnot. V další části bude popsána diskrétní metoda, u které se počítají kritické hodnoty z diskrétního modelu soustavy a kapitola bude zakončena Kuhnovou metodou. [3]

## 2.1 Frekvenční Zieglerova-Nicholsova metoda

Tato jednoduchá experimentální metoda sice patří ke klasickým metodám, ale její zmodernizovaná verze s relé ve zpětné vazbě je často používána i ve složitějších metodách, např. při decentralizovaném řízení vícerozměrových soustav. Ve čtyřicátých letech minulého století Ziegler s Nicholsem na základě experimentů doporučení pro nastavování konstant PID regulátoru. Metoda byla navržena buď po vyhodnocení přechodové charakteristiky, nebo pro měření meze stability. Druhý způsob spočívá v rozkmitání soustavy vhodně nastaveným P regulátorem. Zesílení regulátoru se postupně zvětšuje, až se obvod dostane na mez stability, tj. regulovaná veličina kmitá s konstantní amplitudou. Takto nalezené zesílení se nazývá kritické ($r_{0k}$) a současně se odečte kritická perioda kmitů ($T_k$). Pro nastavení PID a PI a P regulátoru byly doporučeny následující empirické vztahy (viz Tabulka č.1):

Tabulka 1: Nastavování podle ZN metody z ustálených kmitů

| Regulátor | $r_0$ | $T_i$ | $T_d$ |
|---|---|---|---|
| PID | $0{,}6\ r_{0k}$ | $0{,}5\ T_k$ | $0{,}125\ T_k$ |
| PI | $0{,}45\ r_{0k}$ | $0{,}85\ T_k$ | - |
| P | $0{,}5\ r_{0k}$ | - | - |

Nevýhodou ZN metody je, že regulační pochody mají větší překmity. Proto byly původní vztahy (Tabulka č. 1) na základě simulačních experimentů upraveny. Pro přenos regulátoru se využívá modifikovaná rovnice s parametrem $c = 0$. Zavádí se tzv. normalizované zesílení $\kappa$, což je poměr zesílení soustavy na kritické frekvenci k zesílení soustavy na nulové frekvenci:

$$\kappa = \frac{|G(j\omega_k)|}{|G(0)|} = \frac{1}{r_{0k}|G(0)|}$$

a pomocná funkce

$$f(\kappa) = a_0 \exp(a_1\kappa + a_2\kappa^2)$$

Normalizované zesílení může sloužit i pro posouzení, zdali bude soustava dobře regulovatelná, což platí pro $\kappa < 0{,}4$. Vztahy mezi koeficienty funkce $f(\kappa)$ a parametry PID regulátoru jsou uvedeny v Tabulce č. 2 pro dvě hodnoty maximální citlivosti $M_s$. [3]

Tabulka 2: Modifikovaná ZN metoda pro PID regulátor

| PID | $M_s = 1{,}4$ | | | $M_s = 2{,}0$ | | |
|---|---|---|---|---|---|---|
| | $a_0$ | $a_1$ | $a_2$ | $a_0$ | $a_1$ | $a_2$ |
| $r_0/r_{0k}$ | 0,33 | -0,31 | -1,0 | 0,72 | -1,6 | 1,2 |
| $T_i/T_k$ | 0,76 | -1,6 | -0,36 | 0,59 | -1,3 | 0,38 |
| $T_d/T_k$ | 0,17 | -0,46 | -2,1 | 0,15 | -1,4 | 0,56 |
| b | 0,58 | -1,3 | 3,5 | 0,25 | 0,56 | -0,12 |

## 2.2 Kritické hodnoty získané metodou relé ve zpětné vazbě

Jedním z důvodů, proč je ZN metoda stále používána, je možnost získat kritické hodnoty metodou relé ve zpětné vazbě. Hledání meze stability zkusmo je zdlouhavý proces spojený s nebezpečím, že se soustava rozkmitá. Relé ve zpětné vazbě, zapojené místo P složky, má amplitudu zvolené hodnotě $\pm M$ (obvykle 10 až 20 % rozsahu akční veličiny), takže regulovaná veličina nevybočuje z pracovní oblasti. Žádaná hodnota se nastaví na hodnotu odpovídající klidovému stavu. Pokud má regulovaná soustava fázový úhel větší než $-\pi$, vzniknou v URO kmity, které jsou na vstupu soustavy obdélníkové a na výstupu přibližně sinusové s amplitudou $A$. Podle Fourierova teorému ze každou periodickou funkci převést na součet harmonických složek. Pro obdélníkový průběh lze odvodit nekonečnou Fourierovu řadu, která obsahuje pouze liché sinusové členy:

$$u(t) = \frac{4M}{\pi}(sin\omega t + \frac{1}{3}sin3\omega t + \frac{1}{5}sin5\omega t + \cdots)$$

Projde-li tento signál soustavou, potlačí se podle typu soustavy více nebo méně složky vyšších frekvencí, takže na výstupu bude sinusový signál, který může být zkreslený zbytky vyšších harmonických. Zesílení soustavy na základní (kritické) frekvenci kmitů je poměr amplitudy výstupního a vstupního signálu, které jsou posunuté o $-\pi$:

$$|G(j\omega_k)| = \frac{A}{\frac{4M}{\pi}} = \frac{A\pi}{4M}$$

Pro kritické zesílení $r_{0k}$ prochází FCH ORO bodem (-1, j 0), takže $|G(j\omega_k)| * r_{0k} = 1$ a odtud

$$r_{0k} = \frac{4M}{A\pi}$$

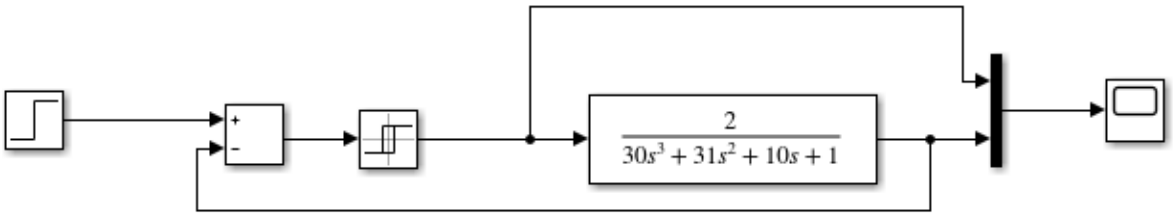
Kritická doba kmitu $T_k$ se odečte se odečte ze záznamu časových průběhů vstupní nebo výstupní veličiny. Kritická úhlová frekvence se vypočte ze známého vztahu:

$$\omega_k = \frac{2\pi}{T_k}$$

Vypočtené kritické hodnoty nemusí být zcela přesné, záleží především na tom, jak soustava filtruje vyšší harmonické. K největším chybám dochází u soustav 1. řádu s malým dopravním zpožděním, kde výstupní signál má pilovitý průběh. Naopak u soustav s velkým dopravním zpožděním je výstupní průběh spíše obdélníkový a potom je přesnější počítat kritické zesílení ze vztahu: $r_{0k} = \frac{M}{A}$

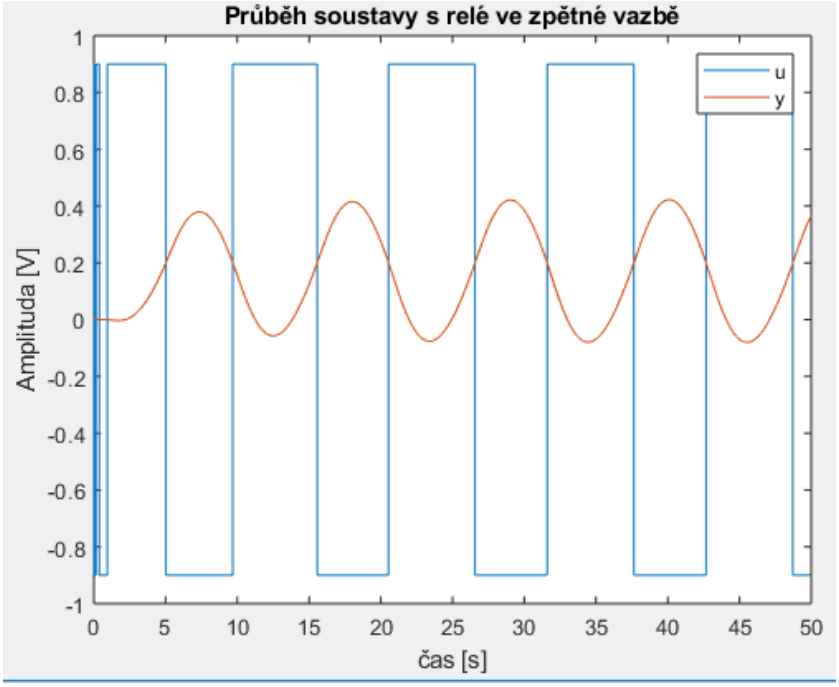
Přesnost kritických hodnot se také zhoršuje, pokud obdélníkový signál není symetrický (nemá stejnou délku kladné a záporné části).

V prostředí aplikace Matlab Simulink můžeme zmíněnou metodu nasimulovat pomocí skládání předdefinovaných bloků. V rámci simulace je potřeba za účelem rozkmitání soustavy umístit na vstup signál ve formě jednotkového skoku. Soustava třetího řádu je zvolena náhodně a relé ve zpětné vazbě je nastaveno na spínání hodnot $M = \pm 0{,}9$. Příklad možného zapojení bloků je zobrazeno na Obrázku č. 2) [3]




Obrázek 2: Simulace relé ve zpětné vazbě v Simulinku

Zobrazení výsledného průběhu (viz. Obrázek č. 3) akční veličiny $u$ a výstupní veličiny $y$ v závislosti na čase lze pomocí funkce „To Workspace“.

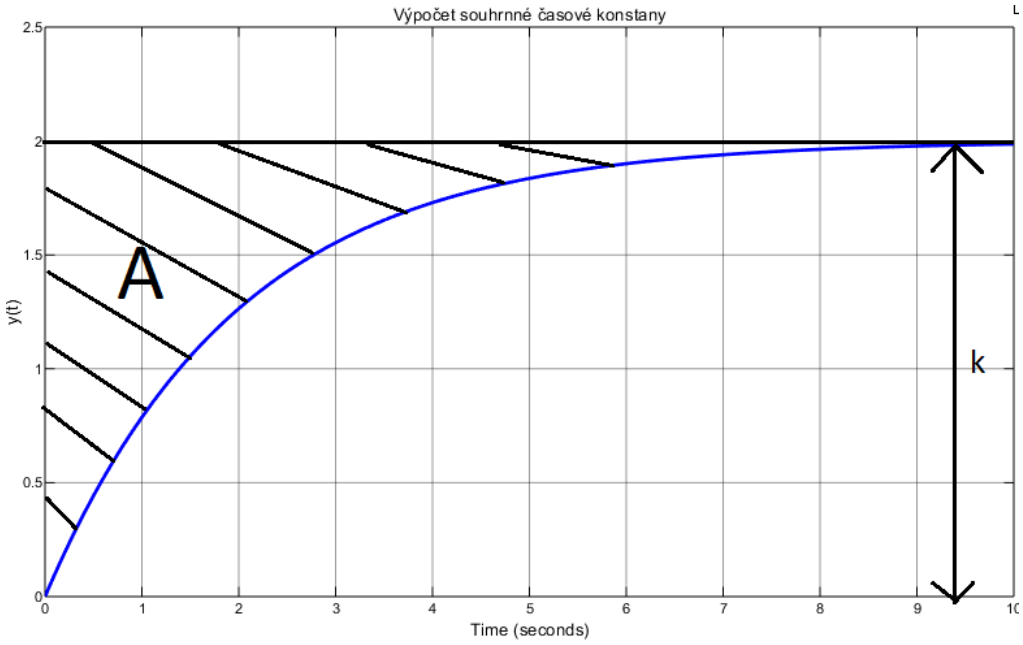



Obrázek 3: Průběh vstupní a výstupní veličiny v simulaci s relé ve zpětné vazbě

V diskrétním regulačním obvodu se kritické hodnoty mění s periodou vzorkování. Vzorkování lze aproximovat dopravním zpožděním o velikosti poloviny vzorkovacího intervalu, takže prodlužováním periody vzorkování se snižuje kritická frekvence (zvětšuje se doba kmitu) a snižuje se i kritické zesílení. Rozkmitat lze i soustavy, které nemají fázový posun větší než $-\pi$, jako např. soustavu 1. řádu. [3]

## 2.3 Kuhnova metoda

V roce 1995 byla odvozena metoda, kterou lze také dohledat pod názvem „pravidlo souhrnné časové konstanty“. Využití nalezneme u málo kmitavého regulačního obvodu, kde doba regulace je přibližně stejná při odezvě na změnu žádané hodnoty, jako na vstupní poruchu. Jedná se zejména pro regulátory PI a PID. Po vykreslení přechodové charakteristiky (viz. Obrázek č. 4) můžeme zjistit zesílení $k$ a plochu $A$, pomocí těchto parametrů můžeme poté vypočítat souhrnnou časovou konstantu $T_\Sigma$




Obrázek 4: Výpočet parametrů $k$ a $A$ z PCH

$$T_\Sigma = \frac{A}{k}$$

Pokud máme vypočítané všechny parametry, můžeme využít tabulek pro výpočet parametrů pro určité typy regulátorů. Kuhnova metoda se rozlišuje pro normální nastavení (viz. Tabulka č. 3) nebo pro rychlý regulační děj (Tabulka č. 4). [4]

Tabulka 3: Nastavení regulátoru Kuhnovou metodou - normálně

| Regulátor | $r_0$ | $T_i$ | $T_d$ |
|---|---|---|---|
| PID | $\frac{1}{k}$ | - | - |
| PI | $\frac{2}{k}$ | $0{,}7\ T_\Sigma$ | - |
| P | $\frac{2}{k}$ | $0{,}8\ T_\Sigma$ | $0{,}194\ T_\Sigma$ |

Tabulka 4: Nastavení regulátoru Kuhnovou metodou - rychle

| Regulátor | $r_0$ | $T_i$ | $T_d$ |
|---|---|---|---|
| PID | $\frac{1}{k}$ | - | - |
| PI | $\frac{0{,}5}{k}$ | $0{,}5\ T_\Sigma$ | - |
| P | $\frac{1}{k}$ | $0{,}66\ T_\Sigma$ | $0{,}167\ T_\Sigma$ |

# 3 LOGICKÉ ŘÍZENÍ

Logický řídcí systém pracuje s logickými veličinami (proměnnými) vázanými logickými vztahy. Logická veličina (proměnná) nabývá konečného počtu stavů – nejčastěji dvou vyjadřovaných hodnotami 0 a 1. Rozlišujeme tedy dva stavy regulované nebo akční veličiny. U spojitě proměnné veličiny rozhodujeme, zda je její hodnota pod nebo nad určitou referenční hodnotou (není/je dosaženo dané úrovně veličiny nebo jejího omezení). Nebo je to dáno dvouhodnotovou povahou dané veličiny (informace nespojitého měření nebo stavu akční veličiny – vypnuto/zapnuto). Logické řízení se realizuje pomocí logických obvodů, což jsou logické členy realizující logické funkce pracující s logickými veličinami.

Rozlišujeme dvě skupiny logických obvodů: kombinační (výstup je dán aktuální kombinací vstupů) a sekvenční (výstup závisí nejenom na aktuálních vstupech, ale také na sledu předchozích hodnot vstupů (stavu) – kombinační síť se zpětnou vazbou nebo se zpožděním ve zpětné vazbě (přechodná paměť) nebo použitím paměťových prvků, kterými jsou například klopné obvody. Logické obvody lze realizovat jako mechanické, elektrické nebo elektronické. [2]

## 3.1 Logické funkce

Logickou funkci je možné popsat slovně, například: pro rozsvícení LED diody na operačním panelu je nutné stisknout tlačítko F1 a F2 současně. Logický výraz popisující zmíněnou situaci lze vyjádřit jako funkci LED = F1.F2, kde „.“ znamená funkci AND neboli logický součin. Přehledně můžeme zmíněnou situaci zapsat do pravdivostní tabulky (viz. Tabulka č. 5)

Tabulka 5: Pravdivostní tabulka

| Vstup | Vstup | Výstup |
|---|---|---|
| F1 | F2 | LED |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Obecně má tabulka pro funkci N vstupních proměnných $2^N$ řádků (kombinací vstupů).

Karnaughova mapa je grafickým zápisem pravdivostní tabulky – každému řádku odpovídá určité políčko – má $2^N$ políček. U každého políčka je vyznačeno, zda patří k dané proměnné (políčko je označeno pruhem) nebo její negací (bez pruhu). Sousední políčka se liší pouze v hodnotě jediné proměnné.

Pro výše uvedenou funkci dvou proměnných (obvod se dvěma vstupy) má Karnaughova mapa následující tvar: (viz Tabulka č. 6).

Tabulka 6: Karnaughova mapa dvou proměnných

| F1 \ F2 | 0 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Toto políčko má vstup F1 hodnotu 1 a vstup F2 hodnotu 0, výstup má tedy hodnotu 0. Pro případ Karnaughovy mapy pro funkci tří proměnných $Y = A.B.C + A.\bar{B}.C$ má tvar:

Tabulka 7: Karnaughova mapa tří proměnných

| A \ B C | 0 | 0 | 0 | 0 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |

Pokud máme pouze jednu logickou proměnnou (označena x), která nabývá pouze dvou stavů 0 a 1, lze na tuto proměnnou aplikovat čtyři funkce (viz. Tabulka č. 8)

Tabulka 8: Logické funkce jedné proměnné

| x | Falsum | Aserce, identita | Negace, NOT | Verum |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

U dvou logických proměnných x a y je možných 16 kombinací. V pravdivostí tabulce č. 9 jsou uvedeny tři nejdůležitější z nich, kterými jsou logický součin AND, logický součet OR a výlučný logický součet XOR (NAND a NOR jsou negací AND a OR). [2]

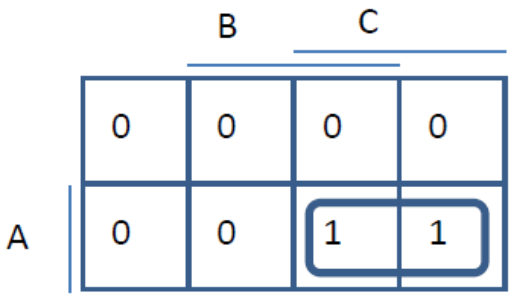
Tabulka 9: Vybrané logické funkce dvou proměnných

| x | y | Logický součin, konjunkce, AND | Logický součet, disjunkce, OR | Exklusivní disjunkce, XOR |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

## 3.2 Zjednodušování logických funkcí - minimalizace

Za účelem získání co nejjednoduššího výrazu (výsledná funkce musí odpovídat původní funkci), zjednodušujeme (minimalizujeme logickou funkci – ušetříme obvody, cenu nebo výpočetní výkon čas. Lze použít pravidla Booleovy algebry, Karnaughových map nebo specializovaného SW.

Funkci $Y = A.B.C + A.\bar{B}.C$, lze zjednodušit jako $Y = A.C.(B + \bar{B}) = A.C$, neboť $(B + \bar{B}) = 1$. Karnaughovou mapou zjednodušíme funkci tak, že najdeme smyčku s jedničkami a zapíšeme výsledek $Y = A.C$, protože na hodnotě B nezáleží (viz. Obrázek č. 5). [2]




Obrázek 5: Zjednodušení funkce pomocí Karnaughovy mapy [3]

## 3.3 Prvky logického řízení

Nejstaršími prvky pro logické operace jsou elektrické prvky – elektrická relé nebo stykače, které slouží k mechanickému spojení kontaktů. Relé jsou elektromagnetická nebo jazýčková (mají spínací, rozpínací kontakty). Stykače slouží ke spínání větších výkonů.

**Elektronické prvky** pro LO mohou být konstruovány jako diodové nebo tranzistorové. Bipolární tranzistorové obvody TTL používají jako základní prvek hradlo NAND nebo NOR. Obvody s unipolárními tranzistory mají menší příkon, velký rozsah pracovních teplot, napájecí napětí, malé zpoždění signálu, jednodušší výrobní technologie a tím nižší náklady (obvody typu MOS a CMOS).

U **kombinačních LO** závisí výstupy $y_i$ na aktuální kombinaci vstupů $x_i$ a nikoliv na jejich předchozím stavu.

U **sekvenčních LO** jsou výstupy $y_i$ funkcemi aktuálních vstupů $x_i$, ale i sledu (sekvence) předchozích hodnot vstupů a výstupů. Tyto informace se ukládají pomocí vnitřních proměnných $z_i$ – jejich kombinace určuje vnitřní stav SLO. Jinými slovy stejná kombinace vstupů nemusí vést na stejné výstupy. SLO musí tedy obsahovat paměťové členy uchovávající prostřednictvím vnitřních proměnných informaci o stavu LO. Typickým zástupcem SLO je řídící systém výtahu, kde řídící signál motoru výtahu závisí na tom, kde se kabina zrovna nachází. V situaci, že je pod námi, tak se motor roztočí na jednu stranu, pokud je nad námi, tak na druhou a pokud ve stejném patře, tak se rozsvítí světlo v kabině. Jiným příkladem SLO je řídicí systém automatické pračky nebo kávovaru.

Stavy paměťových členů buď samostatně (automat typu Moore) nebo spolu se vstupními proměnnými (automat typu Mealy) určují hodnoty výstupních proměnných. Kombinační LO jsou tedy určitým zjednodušením sekvenčních LO – bez vnitřních stavů.

Pokud nejsou přechody mezi vnitřními stavy řízeny, jedná se o tzv. **asynchronní SLO** (rychlé, ale je u nich riziko hazardních stavů způsobených zpožděním signálů). U **synchronních SLO** je povolena změna vnitřního stavu pouze v definovaném okamžiku zavedením synchronizačního signálu (taktovací, clock signál).

Funkci sekvenčního LO lze popsat slovně nebo logickou funkcí, ale používají se také časové nebo přechodové diagramy, tabulky přechodů nebo jazyky pro sekvenční programování (GRAFCET). U časového diagramu se zobrazují průběhy signálů včetně taktovacího signálu v závislosti na čase. U přechodového diagramu znamenají uzly vnitřní soustavy a hrany (orientované čáry) přechody mezi stavy popsané vstupními proměnnými (podmínka přechodu) a výstupními stavy. Sekvenční obvody jsou klopné obvody, registry, čítače, paměti nebo mikroprocesory. [2]

# 4 PROGRAMOVATELNÝ LOGICKÝ AUTOMAT (PLC)

Programovatelný automat je uživatelsky programovatelný řídící systém přizpůsobený pro řízení průmyslových a technologických procesů, mnohdy specializovaný na úlohy převážně logického typu (obzvláště u starších typů nebo u nejmenších systémů). Nejčastěji se označuje zkratkou PLC (Programmable Logic Controller), v německé literatuře se lze setkat s označením SPS (Speicherprogrammierbare Steuerung). Menší systémy bývají řešeny jako kompaktní PLC. [6]

Obrázek 6: Kompaktní PLC

Větší systémy bývají řešeny jako modulární (viz. Obrázek č. 7), kdy se jedná o systém s rozšiřitelnou konstrukcí, která poskytuje uživateli možnost vlastního individuálního sestavení automatizačního systému. Konstrukce může být doplněna například komunikačními moduly nebo bezpečnostními moduly. [6]

Obrázek 7: Modulární PLC

## 4.1 Konfigurace PLC

Skutečnou sestavu volí uživatel tak, aby co nejlépe přizpůsobil svůj PLC požadavkům řešené úlohy. V konkrétním případě mohou některé typy modulů chybět, jiné se mnohonásobně opakují. V krajním případě může být PLC vystavěn jako čistě binární (logický) systém, s výhradním použitím dvouhodnotových vstupů a výstupů, nebo naopak jako výhradně analogový (regulátor, měřící nebo monitorovací systém).

Existují i aplikace PLC bez fyzických vstupů a výstupů, kdy PLC funguje jen jako inteligentní a programovatelný komunikační adaptér (po připojení „cizího systému“ do sítě PLC, pro připojení operátorských panelů, snímačů čárového kódu a jiných identifikačních prvků). [6]

### 4.1.1 Kompaktní PLC

PLC v kompaktním provedení nabízejí určitou, i když omezenou variabilnost ve volbě konfigurace. Uživatel může k základnímu modulu připojit jeden nebo několik přídavných modulů z omezeného sortimentu s pevnou kombinací vstupů a výstupů, např. modul s 8 binárními vstupy a 8 binárními výstupy (tranzistorovými nebo reléovými), modul rychlých čítačů, analogový vstupní nebo výstupní modul apod. [6]

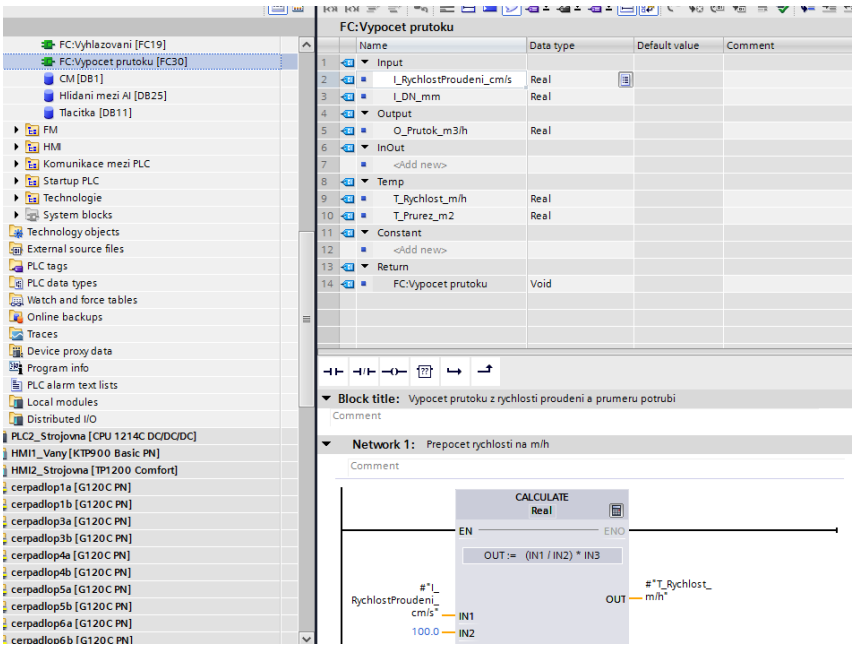
### 4.1.2 Modulární PLC

Nesrovnatelně větší volnost ve volbě konfigurace poskytují modulární programovatelné automaty, kdy lze zvolit jednu z variant plochého zadního rámu, do kterého lze zasouvat libovolné moduly. Rozšiřující moduly mohou být připojeny na vzdálenost stovek metrů. Místo rozšiřujících modulů mohou být připojeny podsystémy tak, že lze vytvářet různé strukturované distribuované systémy. [6]

# 5 PROGRAMOVACÍ BLOKY

Pomocí programovacích bloků je tvořena celá struktura a hlavní kostra celého programu v PLC. Mezi základní bloky patří organizační blok (OB), funkční blok (FB), funkce (FC) a datový blok (DB). Při vytváření nového bloku je velmi vhodné správné pojmenování, které souvisí s využitím dané části programu. Každý blok má své unikátní označení. Při vytváření organizačního bloku, funkčního bloku nebo funkce je nutné zvolit programovací jazyk ve kterém bude daný blok naprogramován. Počet použitých bloků je pouze na uživateli a na velikosti projektu.

Každý programovací blok má vlastní hlavičku, kde je nutné nadefinovat typy parametrů, které jsou v daném bloku programu využity. Definované parametry se rozdělí podle jejich účelu na vstupní či výstupní, vstupně-výstupní nebo pouze pomocná proměnná. Pomocná proměnná slouží například pro uložení mezi-výpočtu v daném skenu programu, ale po skončení daného skenu může nabývat jakékoli hodnoty, což musí mít daný programátor na paměti. Definování hlavičky u vzorové funkce, která slouží k výpočtu průtoku kapaliny potrubím o daném průřezu je naznačeno na obrázku č. 8. Vstupy do vzorové funkce jsou rychlost proudění kapaliny, která je měřena senzorem a průměr šířky potrubí. Výstupním parametrem je vypočtený hodinový průtok, který byl vypočten pomocí matematického vzorce v programu dané funkce.

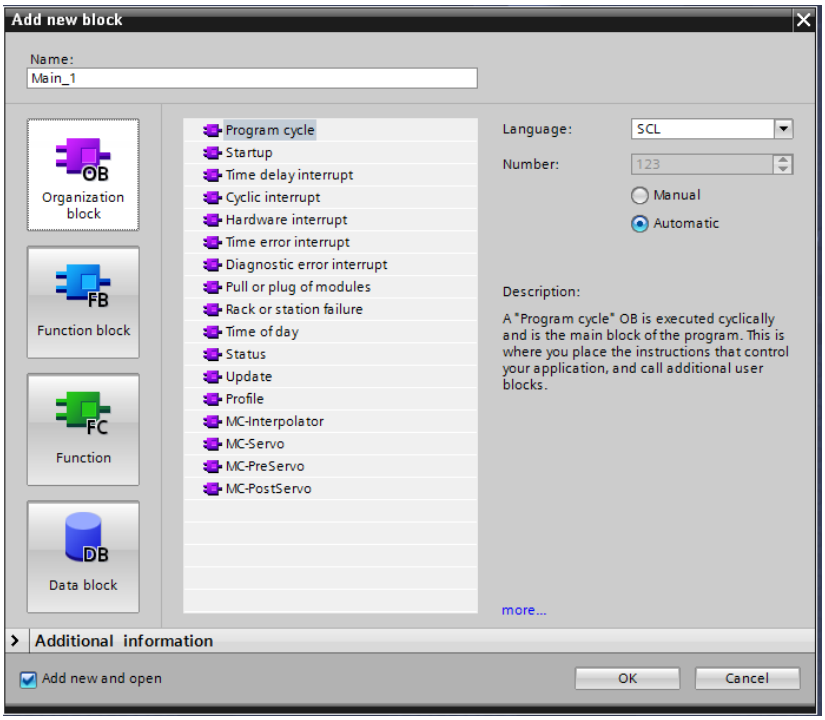



Obrázek 8: Definování hlavičky funkce

Uživatel má na výběr ze tří programovacích jazyků, kterými jsou LAD, FBD nebo SCL, více se těmito jazyky budeme zabývat v kapitole nazvané: „Programovací jazyky“.

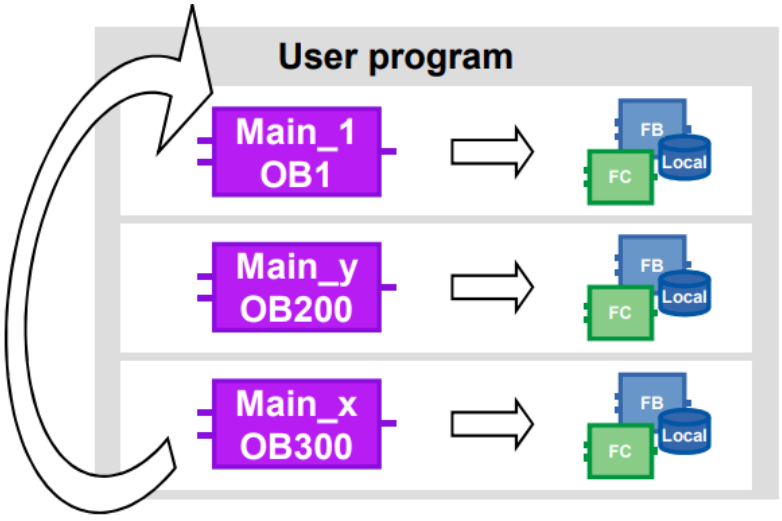
## 5.1 Organization block (OB)

Organizační bloky jsou prostředník mezi uživatelským programem a operačním systémem PLC. Uživatel má podle daného typu PLC na výběr ze skupiny předem definovaných bloků, které mají předem daný způsob využití. Přehled všech dostupných typů OB pro námi zvolené PLC Siemens řady S7-1200 je na obrázku č. 9




Obrázek 9: Přehled dostupných organizačních bloků

Organizační bloky jsou volány operačním systémem PLC a to tak, že sekvenčně v cyklu podle čísla bloku, tedy OB1 bude například voláno před OB200 (viz. Obr. č. 10).



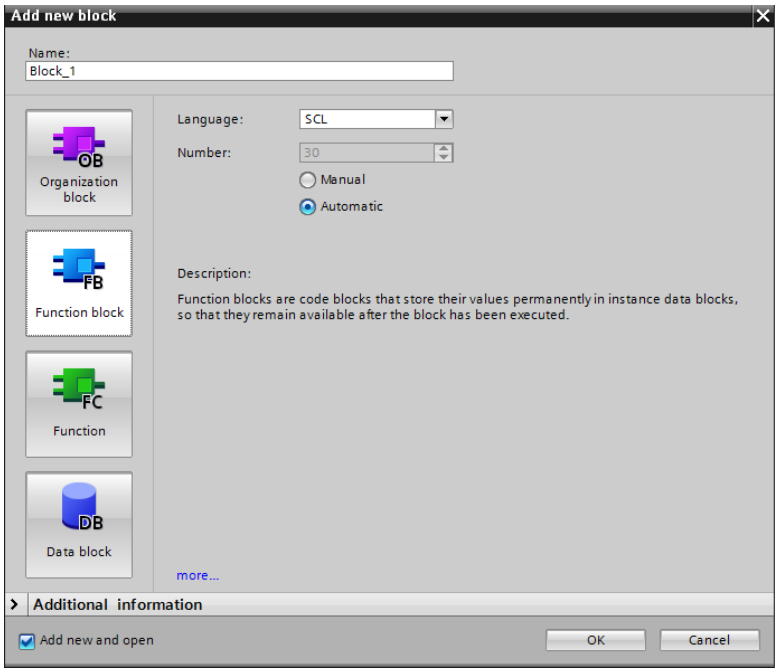

Obrázek 10: Sekvence volání organizačních bloků v PLC

Nejčastěji využívané organizační bloky prostředí TIA Portal jsou:

- **Main** – cyklicky vykonávaná hlavní část programu, kde jsou volány všechny funkční bloky a funkce, které používají datové bloky. Čas jednoho skenu toho programu se odvíjí na rychlosti procesoru a také na velikosti projektu.
- **Startup** – instrukce programu, které se mají vykonat při startu PLC.
- **Cyclic interrupt** – část programu, která se vykonává opakovaně s předem uživatelem definovanou periodou času, která je nezávislá na času jednoho skenu organizačního bloku Main. [7]

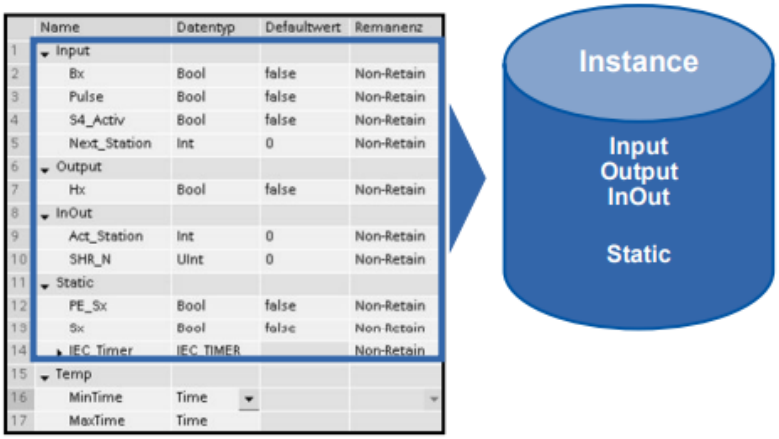
## 5.2 Function block (FB)

Funkční bloky jsou bloky s cyklickým ukládáním dat, jejichž hodnoty jsou permanentně uloženy. Cyklické ukládání dat je realizováno pomocí instance datového bloku, který má obvykle stejný název jako daný funkční blok. Použití funkčních bloků je za účelem vytvoření podprogramů a strukturování uživatelského programu. Uživatel má možnost daný funkční blok zavolat v programu na více místech, ale v tomto případě je doporučeno použít jiné instance, pokud možno multi-instance.




Obrázek 11: Vytvoření funkčního bloku

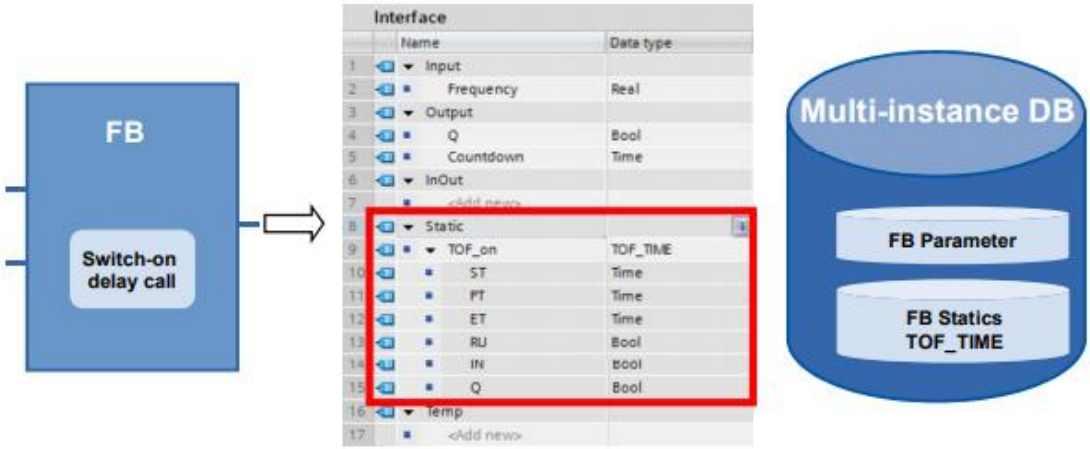
Zavolání funkčního bloku v programu je nazváno instance a jejíž data s kterými pracuje jsou uloženy v datovém bloku té dané instance. Datový blok je vytvořen podle definované hlavičky funkčního bloku, tedy nelze měnit parametry v rámci instance datového bloku.




Obrázek 12: Instance datového bloku

Instance datového bloku se skládá z permanentní paměti, kde lze definovat vstupy, výstupy, vstupně výstupní a statické proměnné. Volatilní paměti pomocné proměnné, které jsou platné pouze po dobu jednoho skenu programu. Uživatel může definovat statický tag, který nemění svou hodnotu během cyklického vykonávání funkčního bloku.

Multi-instance se používají při zachování dat instančního datového bloku nazvaného například A, pokud je příslušný funkční blok A volán v jiném funkčním bloku B, poté uloží tyto data instance A do instance B. Funkcionalita volaného funkčního bloku A je stále zachována. V následujícím Obrázku č. 13 lze vypozorovat, že určitý funkční blok využívající jiný funkční blok, který obsahuje v sekci statických proměnných časovač. Data z obou funkčních bloků jsou uložena v jedné multi-instanci datového bloku a tímto lze například docílit vytvoření datového bloku, který bude časově nezávislý, tedy například časový generátor. [7]




Obrázek 13: Multi-instance datového bloku

Výhody Multi-instance:

- Opakované použití
- Zavolání funkčních bloků na více místech v programu
- Přehlednější program s méně instancemi datových bloků
- Jednodušší kopírování částí programu
- Vytváření struktur v průběhu programování

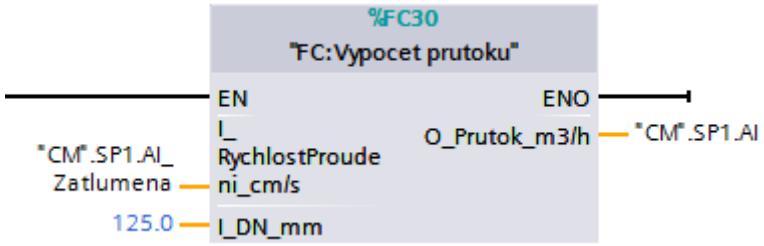
## 5.3 Function (FC)

Funkce jsou bloky bez cyklického ukládání dat a to je důvod, proč hodnoty parametrů v daném bloku funkce nemohou být uloženy před dalším zavoláním této funkce. Výsledkem funkce je návratová hodnota, jejíž datový typ se nastavuje v hlavičce dané funkce, pokud je zvolen typ Void, jedná se o funkci bez parametrů. V případě, že je zvolen typ návratové hodnoty např. Real, musí být výsledek dané funkce při jejím volání vložen také do proměnné typu Real.

Při volání funkce je nutné pracovat s aktuálními hodnotami parametrů. Uživatel musí dodržet způsob volání funkce, který se odvíjí od zvoleného programovacím jazyku. Po zavolání dané funkce je nutno tzv. namapovat tagy na definované parametry, které byly vytvořeny v hlavičce dané funkce (viz. druhý odstavec v kapitole Programovací bloky). Pro ukázku, jak se způsob volání liší je znázorněno volání vzorové funkce sloužící k výpočtu průtoku v potrubí, které lze vypozorovat na obrázku č. 14 volání v SCL a také volání v jazyce LAD (viz. Obrázek č. 15).

```
1 "FC:Vypocet prutoku"("I_RychlostProudeni_cm/s":="CM".SP1.AI_Zatlumena,
2                      I_DN_mm:=125.0,
3                      "O_Prutok_m3/h"=>"CM".SP1.AI);
```

Obrázek 14: Volání funkce v jazyce SCL

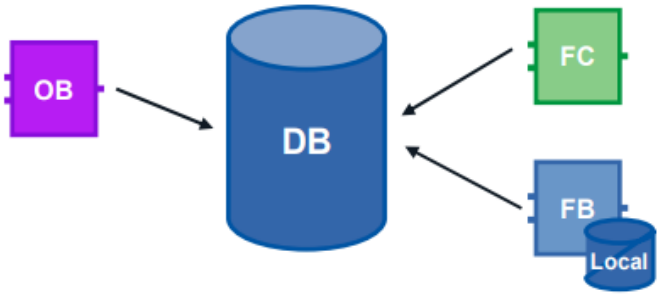



Obrázek 15: Volání funkce v jazyce LAD

Uživatel má možnost danou funkci zavolat v programu na více místech. V praxi se často využívá vnoření určité funkce do jiné funkce a následné zavolání této funkce do OB Main. [7]

## 5.4 Data block (DB)

V této části se jedná o globální datové bloky, jelikož funkční bloky mají své příslušné datové bloky, jak již bylo probráno v podkapitole nazvané „Funkční blok“. Proměnná data jsou uložena v globálním datovém bloku a jsou dostupná v celé šíři uživatelského programu (viz. Obrázek č. 16), tedy lze je používat v jakémkoli OB, FC nebo FB. Uživatel má k dispozici centrální datovou oblast, ke které je velice rychlý přístup. Struktura globálního datového bloku se může skládat z libovolného počtu proměnných, které jsou různého datového typu. [7]




Obrázek 16: Globální datový blok

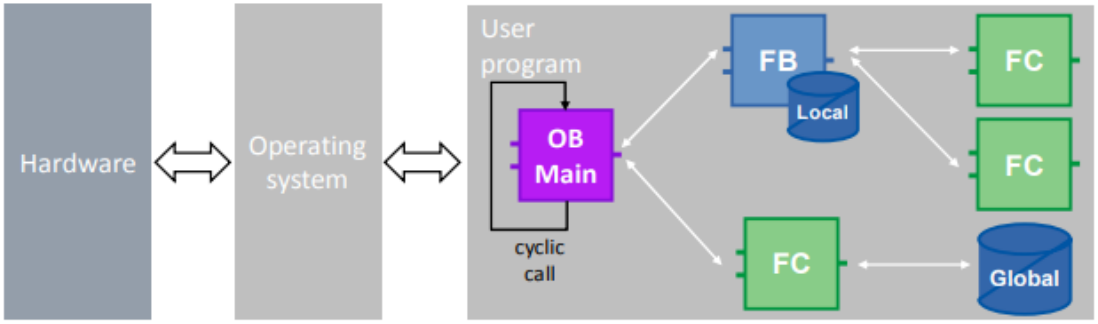
# 6 PROGRAMOVACÍ JAZYKY PLC

V této kapitole se budeme zabývat programovacími jazyky v prostředí aplikace TIA Portal. Programovací jazyky jsou prostředek k zápisu algoritmů, které jsou nezbytně nutné v oblasti průmyslu a automatizace. Simatic kontroléry se skládají z operačního systému a uživatelského programu. Uživatel má na výběr z několika různých programovacích jazyku. Každý programovací jazyk má své výhody, které lze využít na základě zvolené aplikace. Uživatel si může vytvořit každý blok v programu v jiném programovacím jazyku. Pro námi zvolené PLC řady S7-1200 nelze využít některých programovacích jazyků, jak lze vidět v Tabulce č. 6, nicméně nemá to na námi zvolenou aplikaci žádný vliv. [6]

Tabulka 10: Dostupnost programovacích jazyků

| Programovací jazyk | S7-1200 | S7-1500 |
|---|---|---|
| Ladder (LAD) | ✓ | ✓ |
| Function block diagram (FBD) | ✓ | ✓ |
| Structured control language (SCL) | ✓ | ✓ |
| Graph | × | ✓ |
| Statement list (STL) | × | ✓ |

Operační systém kontroléru organizuje všechny funkce a sekvence kontroléru, které nejsou připojeny se specifickou řídící úlohou, jako například volání uživatelského programu, chybové hlášení, správa paměti. Operační systém je integrovaná část kontroléru (viz. Obrázek č. 17).

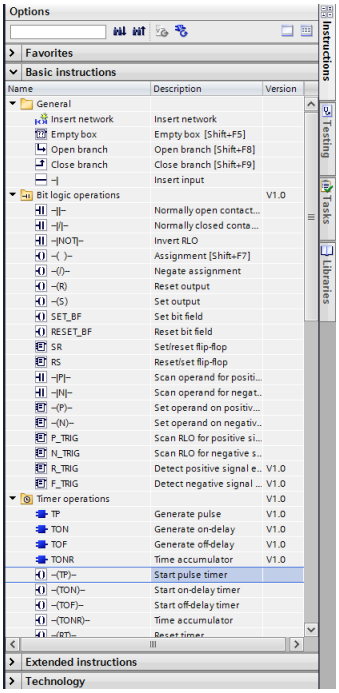



Obrázek 17: Operační systém v PLC

Program PLC je posloupnost instrukcí a příkazů jazyka. Typickým režimem jeho aktivace je cyklické vykonávání v programové smyčce. Na rozdíl od jiných programovatelných systémů se programátor PLC nemusí starat o to, aby na konci programu vrátil jeho vykonávání opět na začátek – zajistí to již systémový program. Naopak každé dlouhodobé setrvání programu v programové smyčce je „fatální chybou“ a systém jej hlásí jako „překročení doby cyklu“.

Vždy po vykonání poslední instrukce uživatelského programu je předáno řízení systémovému programu, který provede tzv. otočku cyklu. V ní nejprve aktualizuje hodnoty výstupů a vstupů: hodnoty uložené dosud v paměti jako obrazy výstupů (registry Y) přepíše do registrů výstupních periferních modulů a hodnoty ze vstupních modulů okopíruje do paměťových obrazů vstupů (registry X). Dále aktualizuje časové údaje pro časovače a systémové registry, ošetří komunikaci a provede ještě řadu režijních úkonů. Po otočce cyklu je opět předáno řízení prvé instrukci uživatelského programu.[7]

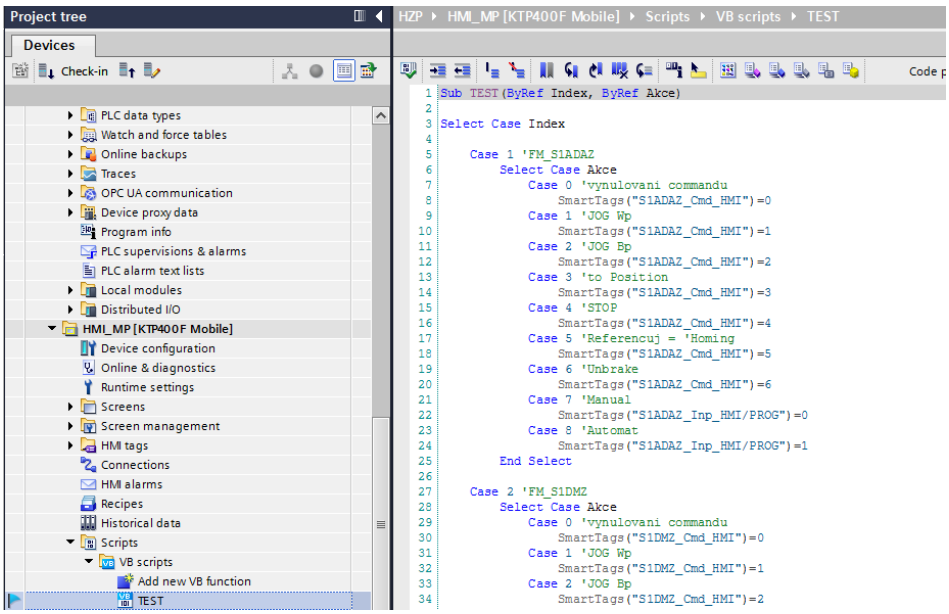
Uživatelský program zahrnuje všechny bloky, které jsou vyžadovány k vykonání specifické automatické činnosti. Bloky obsahují sekvence příkazů, které jsou cyklicky vykonávány. V prostředí TIA Portal je možné si zobrazit panel s instrukcemi (viz. Obrázek č. 18), které je možné použít k programování těchto bloků a případně si zobrazit nápovědu, která popisuje, jak daná instrukce funguje. Instrukce se mohou mírně lišit, podle zvoleného programovacího jazyka. Uživatelský program je naprogramovaný pomocí bloků v aplikaci TIA Portal a posléze nahrán do kontroléru.




Obrázek 18: Programovací instrukce v prostředí TIA Portal

Pokud je zvoleno novější PLC od výrobce Siemens řady S7-1200 popřípadě S7-1500, tak všechny programovací jazyky jsou kompilovány stejným způsobem, tedy převedením na strojový kód. TIA Portal optimalizuje stejný výkon aplikace pro jakýkoliv zvolený programovací jazyk a nedochází ke snížení výkonu z důvodu dalšího kompilování pokročilého programování při využití jazyka STL.

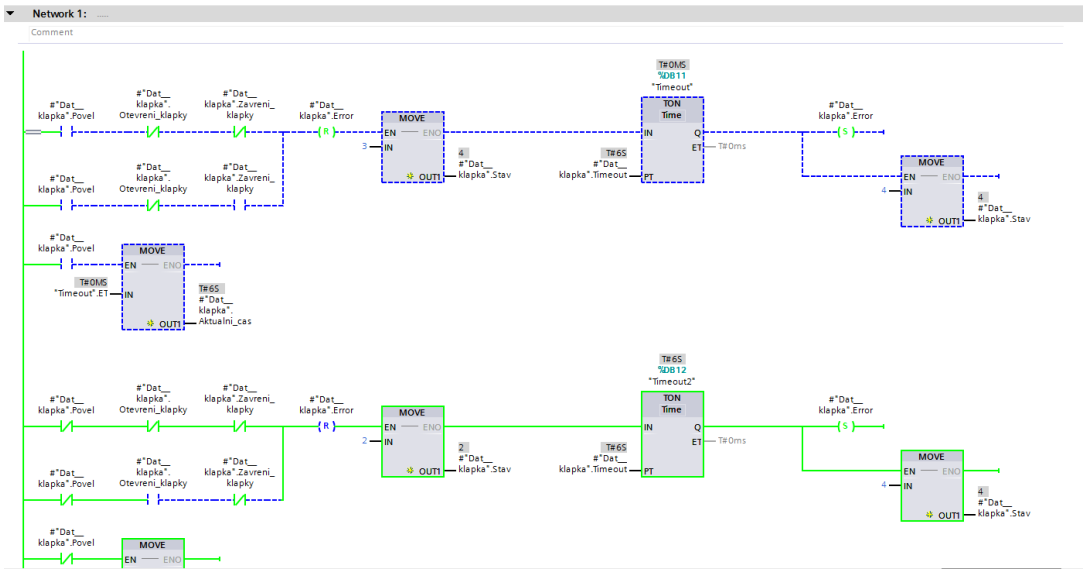
V oblasti HMI, kde uživatel vytváří vizuální část aplikace, lze vytvořit skript v programovacím jazyce Visual Basic (VB), který byl vytvořen společností Microsoft pro objektové modelování již v roce 1991. Visual basic je využívaný především, pokud uživatel není schopen docílit požadované akce pomocí předem definovaných funkcí či vlastností v oblasti HMI. Příklad VB skriptu je zobrazen na Obrázku č. 19).




Obrázek 19: Příklad HMI skriptu v programovacím jazyce VB

## 6.1 Ladder (LAD)

Programovací jazyk Ladder je velice oblíbený grafický jazyk pro začínající programátory, jelikož využívá tzv. „ranky“ popřípadě z anglického překladu „žebříky“ na které se vkládají instrukce ve formě objektů. Mezi největší výhody patří online monitorování programu (viz Obrázek č. 20), kterým je myšleno grafické sledování chování jednotlivých proměnných (objektů) v programu, je-li daný program nahrán do běžícího PLC. Uživatel má možnost program rozdělit do tzv. „networků“ za účelem zpřehlednění programu. Každý network může být nadepsán podle účelu dané části programu. [6]

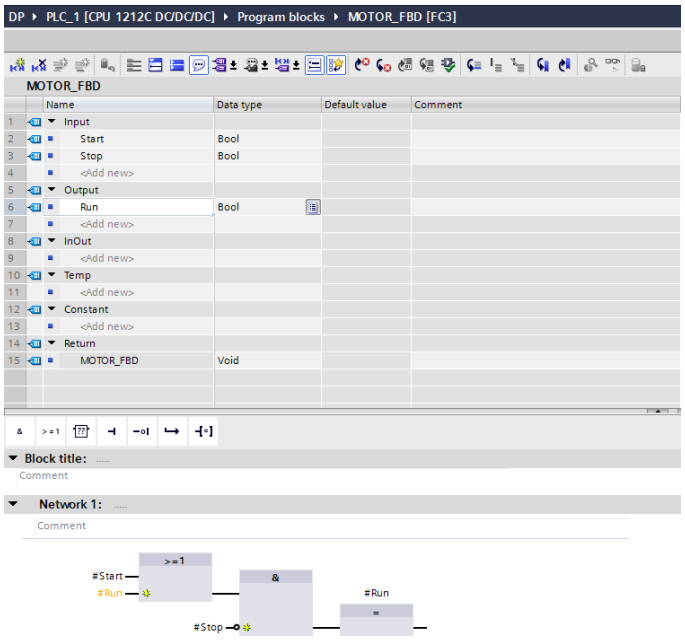



Obrázek 20: Monitorování programu v jazyce LAD

## 6.2 Function block diagram (FBD)

Jedná se o grafický jazyk funkčních bloků, jako v případě volby jazyka LAD. Základní logické operace popisuje obdélníkovými značkami. Výška značky je přizpůsobena počtu vstupů. Své značky mají i ucelené funkční bloky, například čítače, časovače, posuvné registry, paměťové členy, ale i aritmetické a paralelní logické instrukce. Vychází vstříc uživatelům zvyklým na kreslení logických schémat pro zařízení s integrovanými obvody. Obdobný, ale obecnější jazyk se využívá při popisu a programování systémů, zpracovávajících analogové proměnné, při programování regulačních a měřících úloh. [6]

Na obrázku č. 21 je znázorněn jednoduchý příklad programování v jazyce FBD pro zahájení startu motoru pomocí tlačítka umístěného na operačním panelu. Jednoduchá podmínka k odstartování motoru je založena na tom, že není stisknuto tlačítko Stop a zároveň motor je v zapnutém stavu.




Obrázek 21: Příklad programu v jazyce FBD

## 6.3 Structured control language (SCL)

Strukturovaný řídící jazyk SCL je vhodný zejména pro programování komplexních algoritmů a aritmetických funkcí nebo pro úlohy zpracování dat. SCL kombinuje více prvků z vyšších programovacích jazyků jako sériové smyčky, alternativních větvení programu a využívá rozšířené instrukce typické pro PLC (např.: adresování vstupů a výstupů, dotazování se na stav čítačů časovačů). SCL koresponduje k dalšímu strukturovanému programovacímu jazyku STL definovaném pro standard IEC 61131-3. [8]

## 6.4 Statement list (STL)

Je obdobou vyšších programovacích jazyků pro PC nebo mikrořadiče (např. Pascal nebo C). Umožňuje úsporný a názorný zápis algoritmů. Je oblíbený zejména u mladých absolventů odborných škol, pro které je nejpřirozenější.

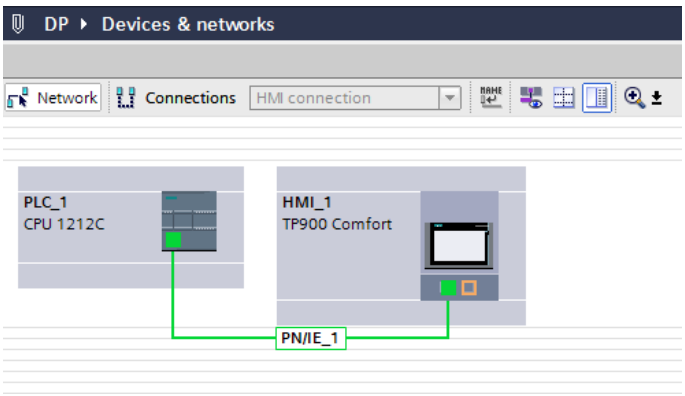
Jazyk sekvenčního programování je velmi názorný a podporuje systémový přístup k programování. Programátor má malý prostor k vytváření chaoticky neuspořádaných programů, je nucen zamyslet se nad podstatou problému, má možnost systematicky ji popsat a realizovat. Většina řízených technologií je svou podstatou sekvenční. Poměrně náročným sekvenčním problémem bývá vyhodnocení posloupnosti tlačítek a zásahů obsluhy. [6]

# 7 HUMAN MACHINE INTERFACE (HMI)

HMI používáme v průmyslu k řízení a monitorování strojů. Nejčastěji se s HMI setkáváme v podobě dotykových obrazovek, které mohou obsluhovat proškolení operátoři či údržba. Dotykové operační obrazovky slouží k zobrazení informací ohledně teploty, tlaku, kroku (stavu) ve kterém se stroj nachází, počtu vyrobených kusů, úroveň hladiny v nádrži nebo například přesnou pozici určitého dopravníku. Obsluha má tedy možnost nejen sledovat určité aktuální veličiny, ale zároveň pokud má dostatečné oprávnění může např. zapínat motory či pohybovat s pneumatickými válci atd. Různí výrobci operačních obrazovek používají různý software a programátor se podle toho musí zařídit.

## 7.1 Konfigurace HMI

Pro připojení HMI obrazovek k PLC je nutné, aby obě zařízení byla na stejné síti a komunikovala přes stejný komunikační protokol. Poté je nutné přidat operační panel podle jeho označení do hardwarové konfigurace (viz. Obrázek č. 22) v prostředí aplikace TIA Portal. Pokud máme zvolen správně panel, je nutné nastavit IP adresu danému panelu a následně propojit tyto zařízení pomocí čar (zařízení na stejné síti pod stejným komunikačním protokolem) v hardwarové konfiguraci. Po úspěšném propojení panelu s PLC již můžeme nahrát námi vytvořenou aplikaci, která obsahuje obrazovky s tlačítky, hodnotami, texty, obrázky a dalšími různými objekty podle daného projektu. Provede-li programátor grafickou změnu v HMI části, je poté povinen znovu nahrát novou verzi do operačního panelu, tedy změny se projeví až po opětovném nahrání. Mezi nejčastější průmyslové komunikační protokoly patří Modbus, Ethernet/IP, Profibus nebo Profinet.




Obrázek 22: Propojení DOP s PLC v hardwarové konfiguraci

# 8 PRAKTICKÁ ČÁST

V praktické části bylo cílem ověřit si získané teoretické znalosti, které již byly zmíněny v této diplomové práci. Hlavním cílem bylo vytvořit řídící aplikaci pro PLC Simatic značky Siemens s označením S7-1200. Tato aplikace se skládá z programové části PLC a vizuální části HMI, kde uživatel má možnost v manuálním režimu ovládat soustavu a po přepnutí do režimu automat probíhá regulace pomocí PID regulátoru.

## 8.1 Použitý hardware

Hlavní komponentou je v praxi často používané PLC Simatic S7-1200, které je kompaktní a již v základu poskytuje digitální a analogové vstupy, výstup na 24 V a také digitální výstupy. Zvolené PLC v základu bohužel neposkytuje analogové výstupy, proto musela být přidána další karta analogových výstupu Siemens SM1232 AQ, která se fyzicky připojí k PLC, ale také je nutné ji nakonfigurovat v softwaru. Dále byla použita měřící karta Labjack U12, kterou lze připojit do počítače pomocí USB. Za účelem simulování reálné soustavy bylo využito zařízení se dvěma vstupy a jedním výstupem, které simuluje soustavu třetího řádu. Pro napájení analogových vstupů, výstupů, PLC a karty analogových výstupů, bylo použito 24 V stejnosměrného napájecího zdroje od výrobce Schneider.

### 8.1.1 PLC Siemens Simatic S7-1200

Jedná se o volně programovatelný řídicí systém na bázi procesoru Simatic S7-1200. Hlavními specifikacemi tohoto kompaktního PLC (viz. Obrázek č. 23) jsou: [5]

- CPU 1212C
- 8 digitálních vstupů a 6 digitálních výstupů (24 V DC)
- 2 analogové vstupy (0-10 V DC)
- Stejnosměrné napájecí napětí 24 V DC
- Datová paměť (75 kB)
- Maximální proudový odběr s rozšířenými moduly – 1,2 A
- Podpora většiny průmyslových komunikačních protokolů
- 3 Diagnostické LED (RUN/STOP, ERROR, MAINT)


Obrázek 23: Zvolené PLC Simatic S7-1200

### 8.1.2 Siemens SM 1232 modul analogových výstupů

Jedná se o přídavný modul (viz. Obrázek č. 24) se čtyřmi analogovými výstupy, který lze fyzicky připojit k základnímu kompaktnímu PLC. Každý analogový výstup lze libovolně nastavit jako napěťový nebo proudový. Základní parametry tohoto modelu jsou: [10]

- 4 analogové výstupy
- Napěťové rozmezí ±10 V (14 bitové rozlišení)
- Proudové rozmezí 0-20 mA nebo 4-20 mA (13 bitové rozlišení)
- Modul napájen 24 V DC


Obrázek 24: Přídavný modul analogových výstupů

### 8.1.3 Labjack U12

Labjack je měřící periferie, kterou lze připojit pomocí USB do počítače. USB připojení slouží jako napájení a zároveň komunikace, tedy není potřeba žádné externí napájení. Uživatel má na výběr ze 30 šroubovacích terminálů, které mají využití jako:

- analogové vstupy (AI)
- analogové výstupy (AO)
- digitální vstupně-výstupní (IO)
- interní napájení (5V)
- uzemnění (GND)
- vstup pro 32 bitový čítač (CNT)
- Labjack  výstup operačního zesilovače pro referenční napětí 2,5 V (CAL)

Karta je vybavena konektorem pro připojení dalších 16 digitálních vstupně-výstupních pinů, které lze definovat programově, zda pracují jako vstup či výstup. Stav měřící karty zobrazuje status LED, která například po připojení do PC problikne 4x. Měření analogových vstupů je v rozsahu ±10 V s rozlišením 12 bitů. Na analogové výstupy aplikovat rozsah napětí 0-5 V s rozlišením 10 bitů.


Obrázek 25: Měřící karta Labjack U12

Měřící kartu Labjack lze implementovat do mnoha prostředí: MATLAB, LabVIEW, Python, Java, C++, Visual Basic, VBA Excel, Perl, .NET, a dalších. [2]

### 8.1.4 Soustava třetího řádu

Jedná se o zařízení se dvěma vstupy a jedním výstupem. Maximální napájecí napětí je 10 V. Toto zařízení simuluje reálné chování soustavy třetího řádu.


Obrázek 26: Zařízení simulující reálnou soustavu třetího řádu

## 8.2 Použitý software

Za účelem programování jakéhokoliv PLC je daný software nepostradatelný. Výběr softwaru pro tvorbu řídicí aplikace se volí podle výrobce řídicí jednotky, tedy pro námi zvolený typ PLC od výrobce Siemens, bylo zvoleno prostředí aplikace TIA Portal od stejného výrobce. Pro simulace s měřící kartou Labjack bylo zvoleno prostředí aplikace Matlab-Simulink, kde je možné pomocí bloků sestavit a simulovat libovolný regulační obvod.

## 8.3 Určení PID parametrů pomocí metod

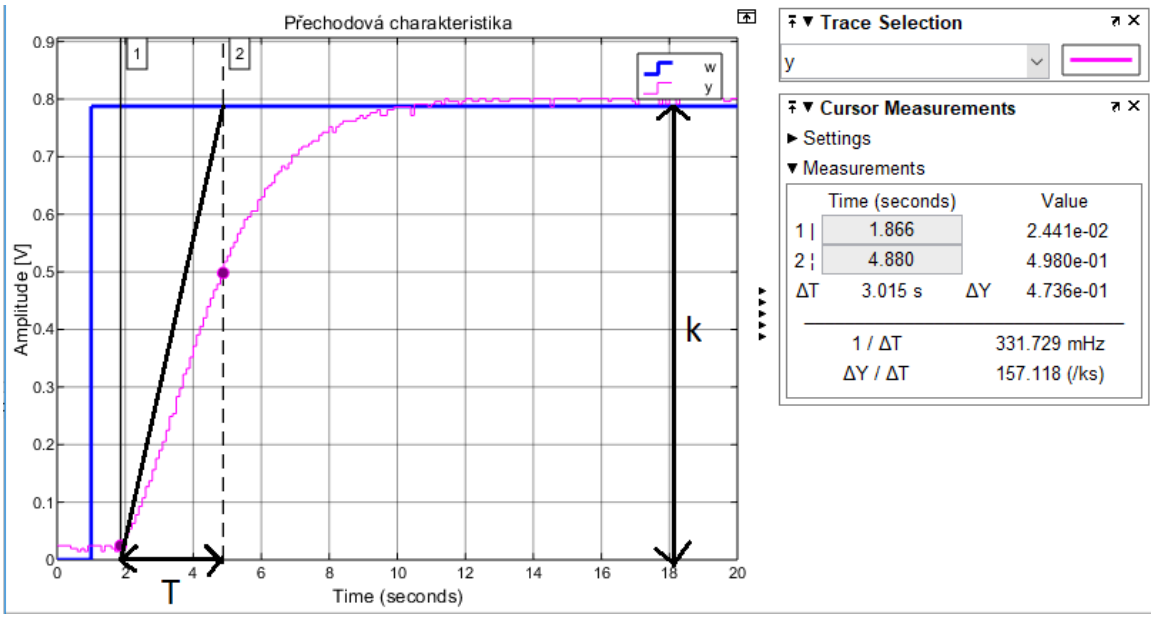
V této části budou postupně aplikovány různé metody pro získání PID parametrů, tyto metody již byly probrány v teoretické části této práce. Výsledné parametry té nejlepší metody budou poté aplikovány pro řízení soustavy pomocí PLC.

### 8.3.1 Určení časové konstanty z přechodové charakteristiky

Při identifikaci v prostředí aplikace Matlab bylo cílem určit časovou konstantu $T_I$ na základě odezvy na jednotkový skok. Přechodová charakteristika (viz. Obrázek č. 27) je reakcí soustavy na jednotkový skok na vstupu v našem případě v čase $t = 1$, při nulových počátečních podmínkách. Výstupní hodnota $y$ se ustálila na hodnotě 3,15, vstupní se ustálila na hodnotě $u = 4$ .Pokud máme ustálenou hodnotu vstupní a výstupní veličiny, poté můžeme vypočítat statické zesílení $K$ podle vzorce:

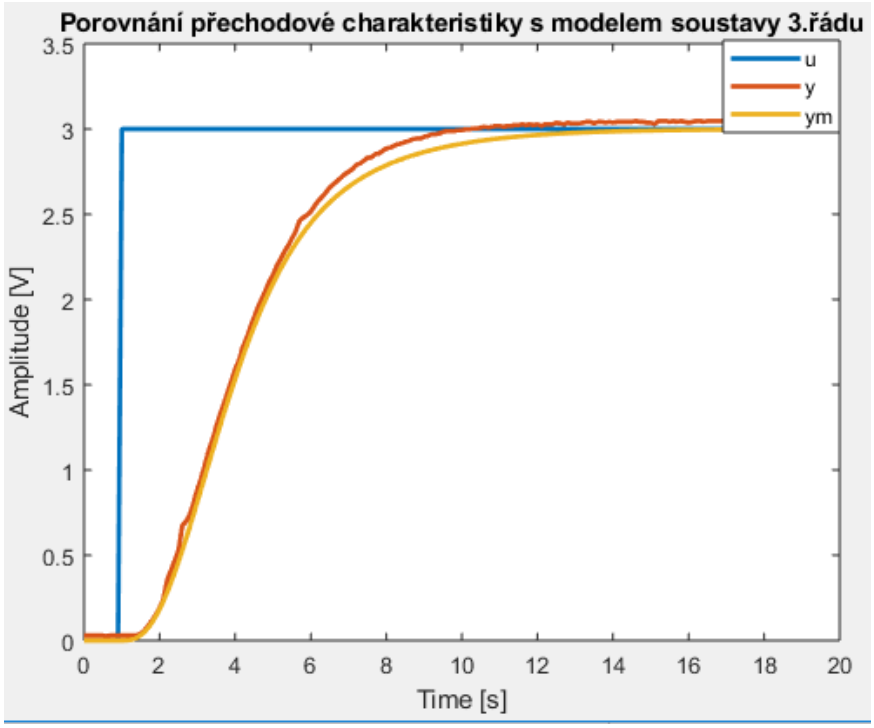
$$K = \frac{y(\infty)}{u(\infty)} = \frac{3{,}15}{4} = 0{,}79$$

Obrázek č. 27 zobrazuje přechodovou charakteristiku a určení časové konstanty $T$ a statického zesílení $K$, kde se tečna pro přesnější aproximaci volí v bodě, kde čas $T$ dosáhne 63,2% ustálené hodnot, která je v tomto případě v hodnotě $y(t) = 0{,}5$. Pomocí kurzorů v grafu jsem následně získal časovou konstantu $T = 3\ s$.




Obrázek 27: Přechodová charakteristika systému

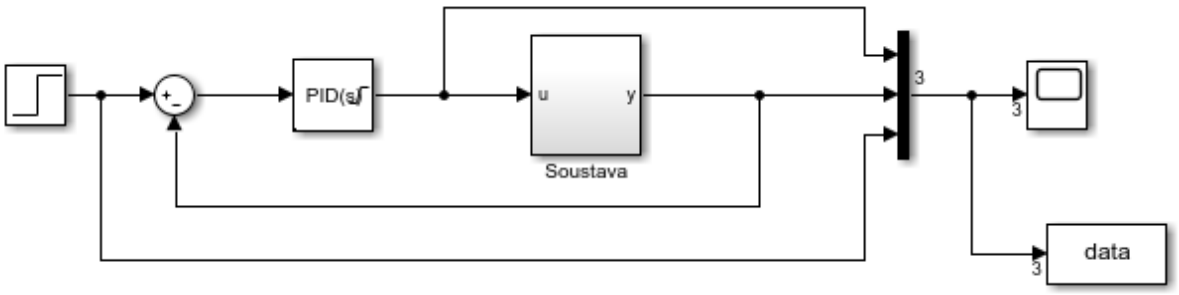
Předpokladem je soustava třetího řádu, tedy s přenosem $G(s) = \frac{1}{(Ts+1)^3}$, po dosazení, kdy jsem provedl simulaci odezvy funkcí lsim a postupnou změnou časové konstanty $T$ jsem získal výstupní průběh modelu $y_m$ a následně porovnal v jednom grafu s $y$ (viz. Obrázek č. 28).




Obrázek 28: Modifikovaný výstup soustavy třetího řádu

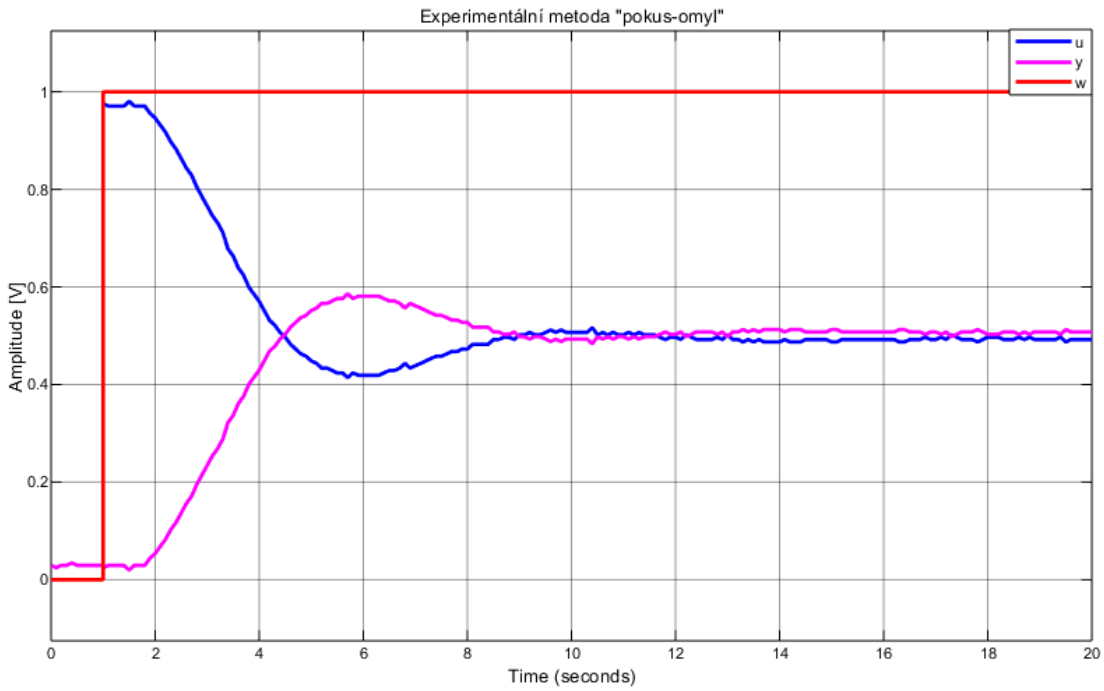
### 8.3.2 Experimentální metoda „pokus-omyl“

Jedná se o experimentální metodu pro seřizování regulátorů a to bez znalosti vlastností reálného regulačního obvodu, který můžeme simulovat. Schéma simulace tohoto obvodu byla provedena v prostředí Matlab a zapojení lze vyčíst z Obrázku č. 29. Jedná se o uzavřený regulovaný obvod s PID a soustavou třetího řádu.



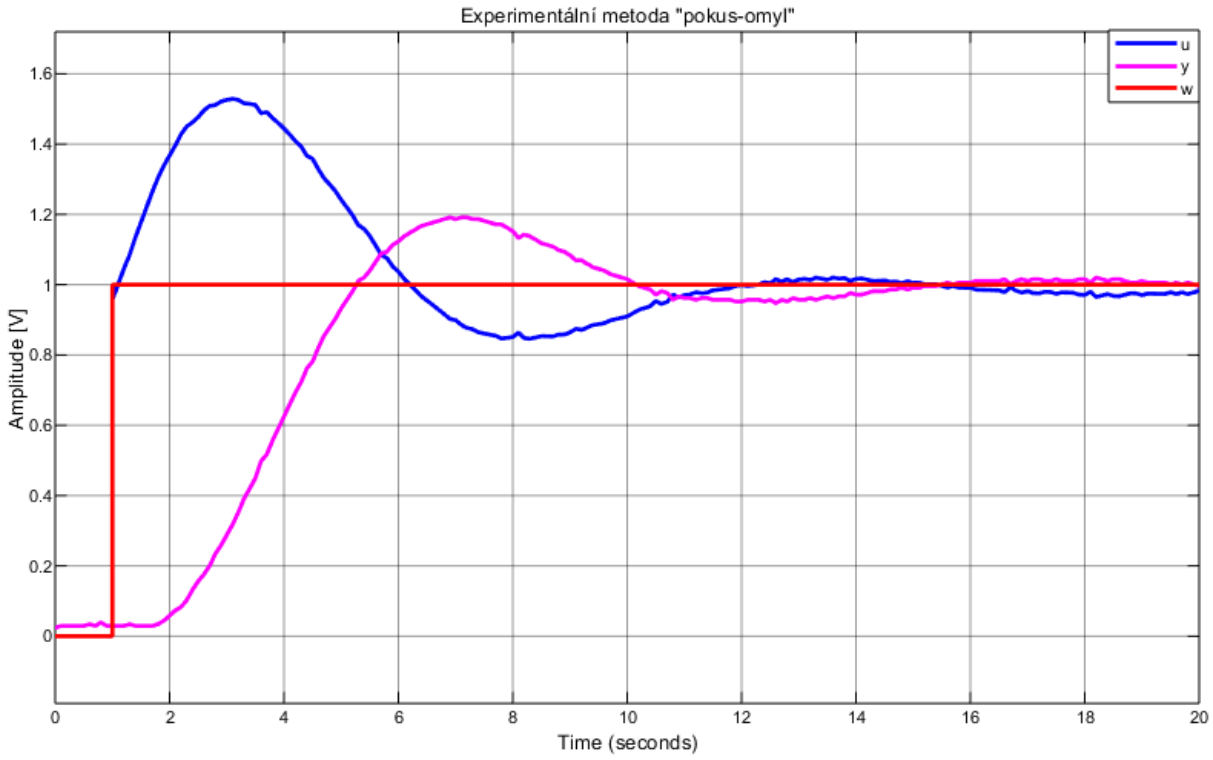

Obrázek 29: Schéma URO v Simulinku

Při experimentování jsem si nejdříve zobrazil akční veličinu $u$ při žádané hodnotě $w = 1$ a nastavením $r_0 = 1$. Akční veličina dosáhla svého maxima v hodnotě blízké jedné, poté se ustálila na hodnotě přibližně 0,5 (viz. Obrázek č. 30)




Obrázek 30: Regulační pochod bez I složky metodou „pokus omyl“

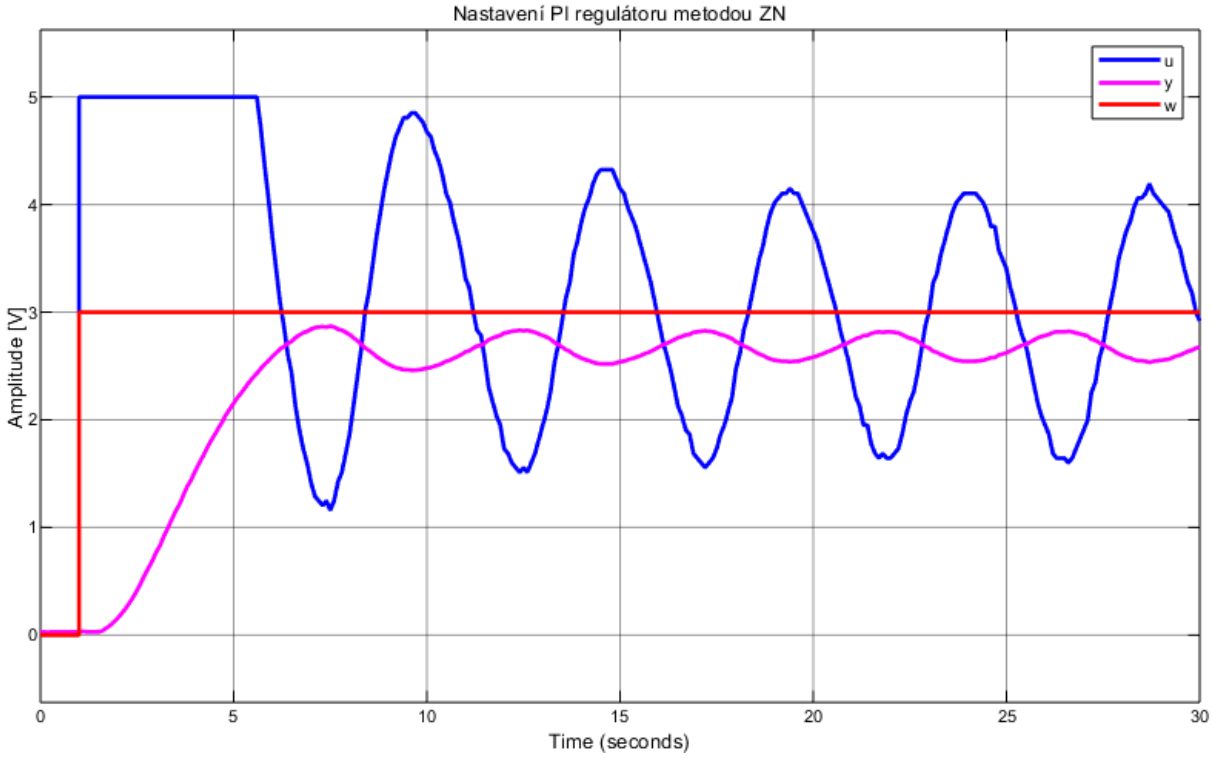
Následným experimentováním se snižováním počáteční integrační časové konstanty $T_I = 5$, jsem došel k závěru, že nejlepšími zvolenými parametry pro PI regulátor jsou hodnoty $r_0 = 1$ a $T_I = 2{,}2$ jejichž průběh jsem zaznamenal na Obrázku č. 31




Obrázek 31: Výsledek experimentální metody „pokus-omyl“

### 8.3.3 Nastavení PI regulátoru metodou Ziegler-Nicholson

Cílem bylo získání kritických parametrů $r_{0k}$ a $T_k$ pro uzavřený regulační obvod s PID regulátorem a soustavou třetího řádu (stejné schéma jako na Obrázku č. 28). Kritické parametry lze získat vyřazením integrační složky a to volbou maximální integrační časové konstanty $T_I$ a nastavením derivační složky tak, že derivační časovou konstantu položíme rovno nule $T_D = 0$. Postupným zvyšováním zesílení regulátoru $k_p$ přivedeme obvod na mez stability (viz. Obrázek č. 32).




Obrázek 32: Dosažení meze stability pro zadanou soustavu

Soustava dosáhla mezi stability, tedy regulovaná a akční veličina kmitá s konstantní amplitudou. Meze stability bylo dosaženo při nastavení $r_{0k} = 9, T_I = 0, T_D = 0$. Pomocí kurzorů v prostředí Matlab Simulink jsem si změřil tři periody $\Delta T = 14\ s$, jak lze vyčíst z Obrázku č. 28, tedy zjistil jsem hodnoty obou kritických parametrů:

$r_{0k} = 9$

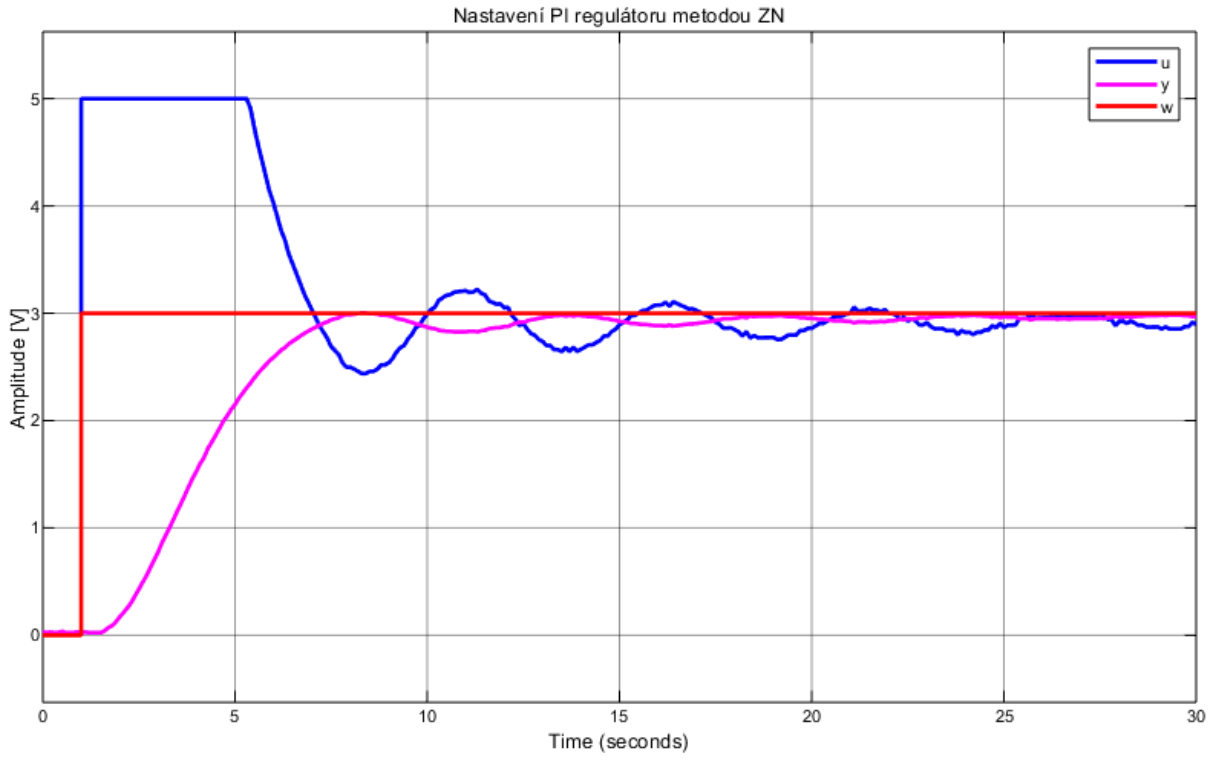
$T_k = 14/3 = 4{,}66$ s

Na základě zjištěných hodnoty kritických parametrů jsem poté mohl vypočítat parametry PI regulátoru podle zmíněné metody ZN, která je daná vztahy:

$r_0 = 0{,}45 * r_{0k} = 0{,}45 * 9 = 4{,}05$

$T_I = 0{,}85 * 4{,}66 = 3{,}9$

Tato metoda po dosazení vypočítaných hodnot do PI regulátoru bohužel nevedla k dobrým výsledkům. Ustáleného výstupu $y$ na žádanou hodnotu bylo docíleno až po 30 sekundách a akční veličina dosáhla maximálního zobrazitelného rozsahu, jelikož soustavu napájíme z 5 V je zde nastaven limit. Průběh lze sledovat na Obrázku č. 33




Obrázek 33: Výsledek ZN metody z kritických parametrů

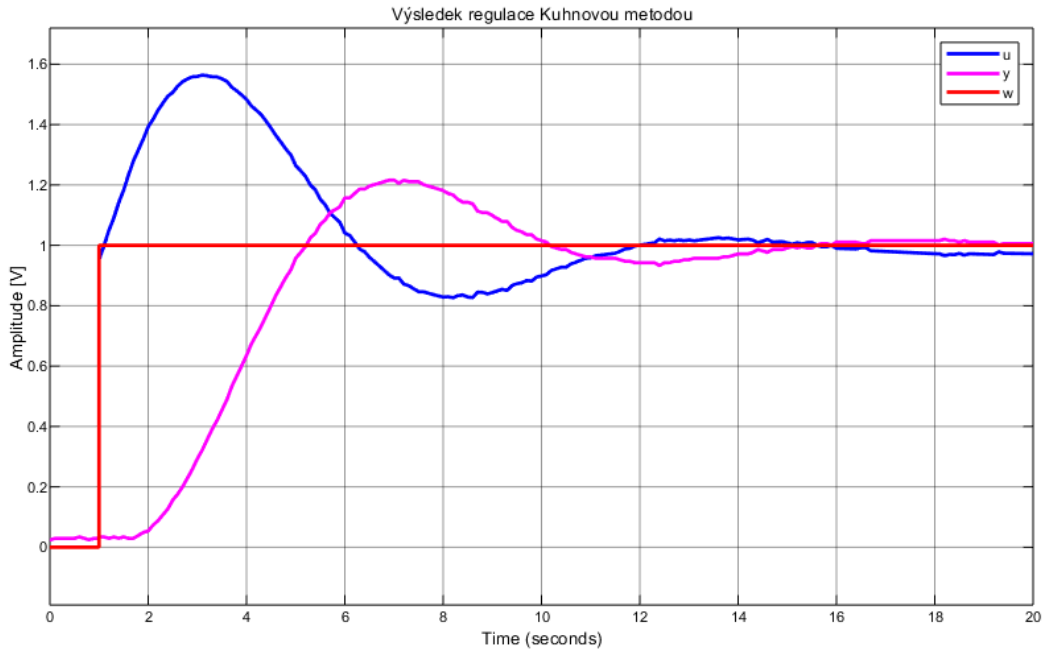
### 8.3.4 Nastavení pomocí násobné časové konstanty

Tato metoda je nazývána Kuhnova metoda, kdy na základě souhrnné časové konstanty $T_{\Sigma}$, kterou jsem již získal z přechodové charakteristiky soustavy na Obrázku č. 27, tedy $T_{\Sigma} = 3\ s$. Následně jsem pomocí vztahů v Tabulce č. 3, která se vyskytuje v teoretické části této práce, byl schopen vypočítat parametry PI regulátoru pro rychlý děj:

$$r_0 = \frac{1}{Z} = 1$$

$$T_I = 0{,}7 * T_{\Sigma} = \ 2{,}1$$

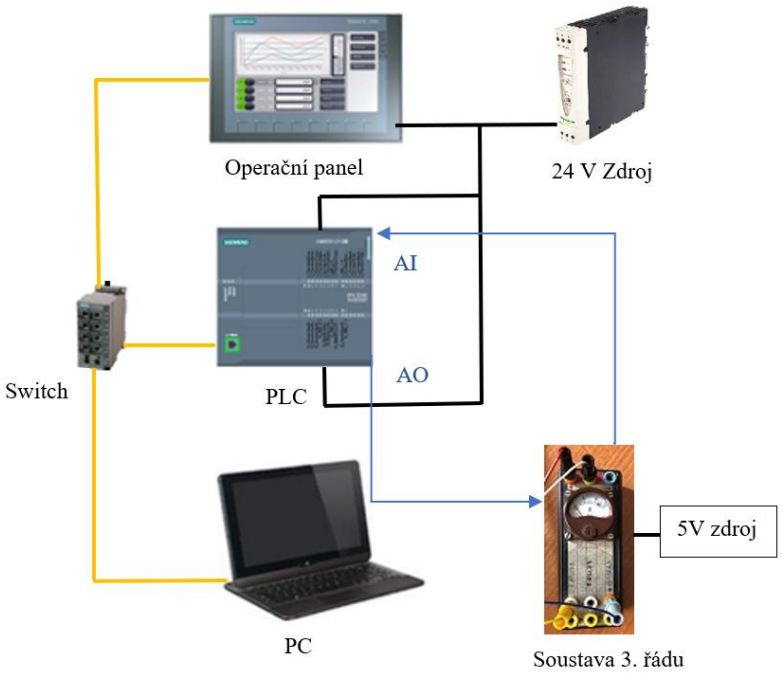
Výsledek je v podstatě skoro totožný s metodou „pokus-omyl“, kde byly experimentálně zjištěny skoro stejné parametry




Obrázek 34: Výsledek regulace pomocí Kuhnovy metody

## 8.4 Vytvoření PID regulátoru v PLC

Na základě experimentů a zjištěných vlastností soustavy třetího řádu jsem vytvořil v prostředí aplikace TIA Portal program, který reguluje zadanou soustavu. Schéma zapojení regulačního obvodu s PLC je na Obrázku č. 35.




Obrázek 35: Schéma zapojení regulačního obvodu

Program v PLC běží v organizačním bloku nazvaném „cyclic interrupt“, kde jsem měl jistotu opakovaného vykonávání dané části programu se stejným skenovacím časem, tím jsem tedy získal periodu vzorkování diskrétního PID regulátoru. Perioda vzorkování byla nastavena na hodnotu 500 ms. Výpočet jednotlivých proměnných diskrétního PID regulátoru byl popsán v kapitole 1.2. této práce. Programovacím jazykem byl zvolen jazyk SCL. V hardwarové konfiguraci jsem podle označení karty analogových výstupů vložil daný modul a definoval adresy k jednotlivým analogovým výstupům. Adresy analogových vstupů již byly defaultně nastaveny.

Na začátku programu v PLC probíhá čtení hodnot analogových vstupů, jejichž hodnota se musí znormovat a přeškálovat, jelikož se jedná o hodnoty v rozsahu 0-27648. Škálováním analogových hodnot na odpovídající hodnotu 0-10 V, tedy maximální rozsah 27648 odpovídá 10 V. To samé, ale obráceně probíhá naopak při výstupu z PID regulátoru, tedy vypočtenou hodnotu akční veličiny $u$ v rozmezí ±10 V je nutné přeškálovat na rozsah ±27648.

Na základě matematického výpočtu jednotlivých vstupů a výstupů, kdy např. hodnota $u(k)$ závisí v daném skenu na předchozí hodnotě $u(k-1)$, bylo nutné v každém skenu přepisovat hodnoty jednotlivých vstupů a výstupů a regulační odchylky do jiných proměnných (viz. Obrázek č. 36)

```
"PID"."y(k-2)" :="PID"."y(k-1)";
"PID"."y(k-1)" := "PID"."y(k)";

"PID"."u(k-2)" := "PID"."u(k-1)";
"PID"."u(k-1)" := "PID"."u(k)";

"PID"."y(k)" := "PID".b1 * "PID"."u(k-1)" + "PID".b2 * "PID"."u(k-2)" -"PID".a1 *"PID"."y(k-1)" -"PID".a2 * "PID"."y(k-2)";
//vypocet regulacni odchylky
"PID"."e(k-2)" := "PID"."e(k-1)";
"PID"."e(k-1)" := "PID"."e(k)";
"PID"."e(k)" := "PID"."w(k)" - "PID"."y(k)";
```

Obrázek 36: Přepisování vstupů, výstupů a regulačních odchylek

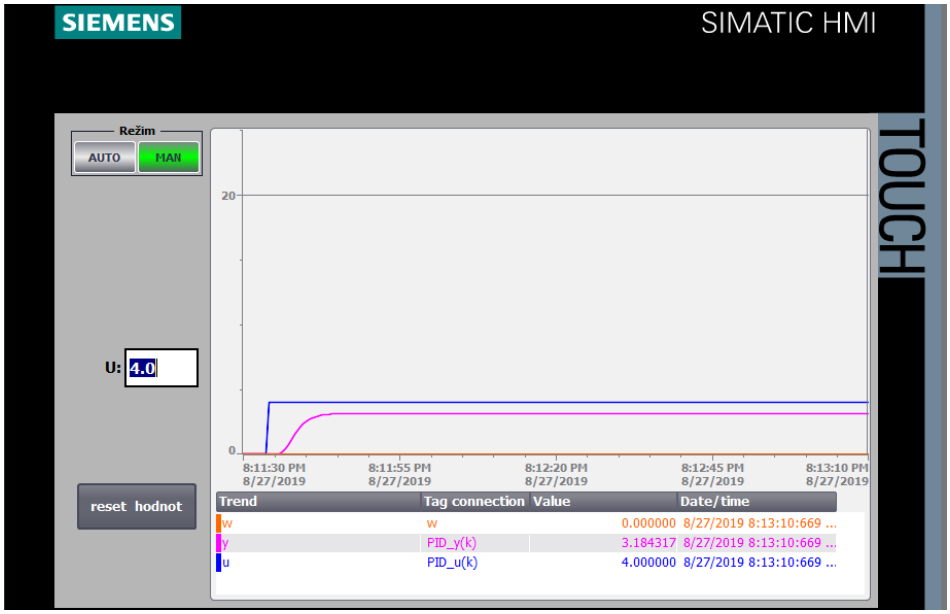
## 8.5 Vytvoření HMI

Pro zmíněný program byla vytvořena obrazovka s poli pro zadávání hodnot jednotlivých složek PID. HMI tagy slouží k tvorbě proměnných, které lze připojit k objektům na obrazovce. Následně jsou tyto HMI tagy namapovány na PLC tagy se kterými již pracuje program v PLC.

Uživatel má možnost přepínání mezi dvěma režimy, kterými jsou AUTO a MAN. Při volbě AUTO probíhá automatická regulace PID na základě zadaných hodnot jednotlivých složek a žádané hodnoty. Tlačítka fungují jako přepínač, nelze tedy zvolit oba režimy v jeden okamžik.

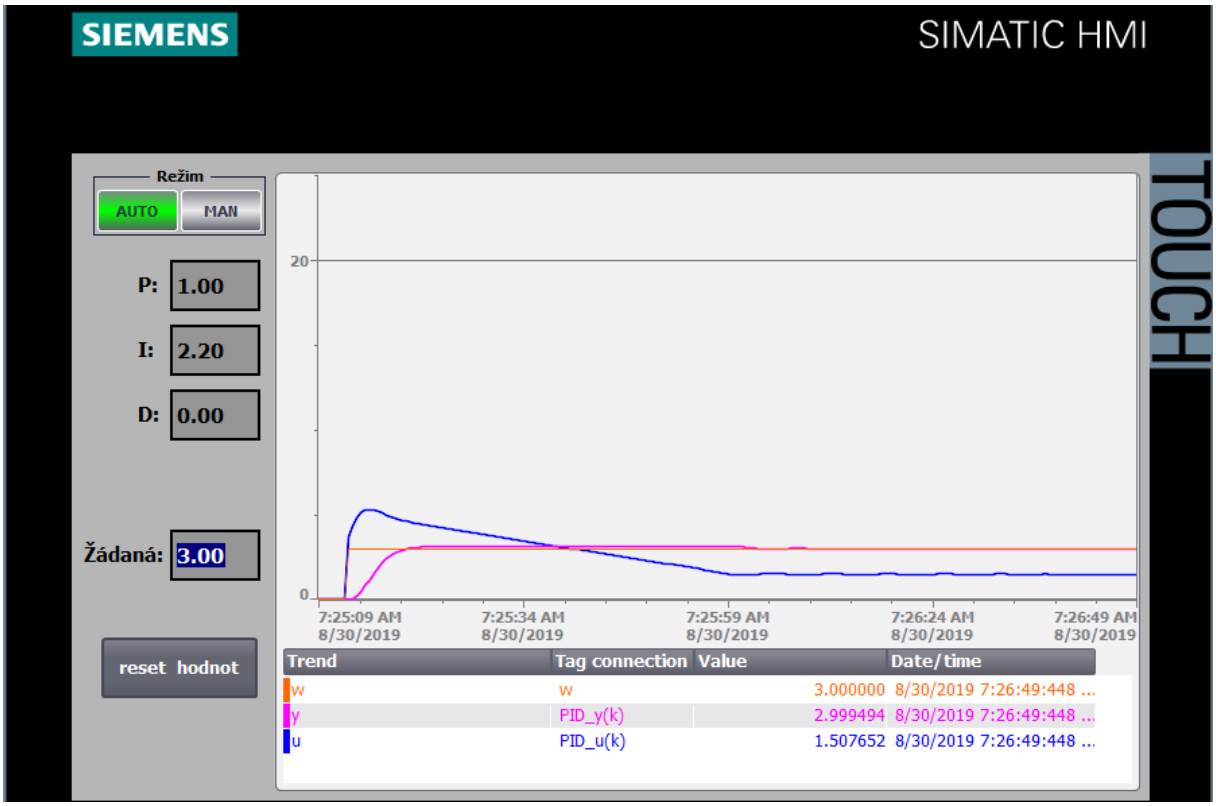
Dále je na obrazovce tlačítko pro reset hodnot, které vynuluje zadané hodnoty v jednotlivých zadávacích položek. Na obrazovce se nachází graf, který generuje průběh jednotlivých veličin, jejíchž hodnoty a barevné odlišení lze vypozorovat v legendě pod grafem.

Při volbě režimu MAN se uživateli zobrazí pouze pole pro zadání akční veličiny a program regulátoru PID je odpojen. V režimu MAN jsem pomocí PLC tedy získal přechodovou charakteristiku stejnou jako v případě zapojení v Matlab s měřící kartou Labjack. (viz Obrázek č. 37)




Obrázek 37: Přechodová charakteristika soustavy v manuálním režimu s PLC

Na základě získaných hodnot z experimentální metody „pokus omyl“ a pomocí Kuhnovy metody jsem při dosazení těchto parametrů PI získal v AUTO režimu výsledný průběh PI regulace, která je na Obrázku č. 38.




Obrázek 38: Výsledná PI regulace pomocí PLC

# ZÁVĚR

Výsledkem této práce je praktické ověření znalostí metod návrhu PID regulátoru, které byly popsány v teoretické části a tvorba aplikace pro PLC, která umožní ovládat a regulovat laboratorní soustavu. Pro ověření použitelnosti metod byly provedeny regulace v prostředí aplikace Matlab, a to na základě přechodové charakteristiky zadané soustavy třetího řádu. Pro získání parametrů PID byly použity metody Ziegler-Nichols, Kuhnova metoda a metoda „pokus-omyl“.

Na základě získaných výsledků jednotlivých metod návrhu PID jsem došel k závěru, že nejvhodnější metodou pro zadanou soustavu je Kuhnova metoda, ale také pomocí metody „pokus-omyl“ jsem dosáhl skoro totožných výsledků. Jako nevhodná metoda pro zadanou soustavu se ukázala metoda Ziegler-Nichols.

V praktické části byla vytvořena aplikace v prostředí TIA Portal, která realizuje PID regulátor v PLC a její vizuální podoba byla vytvořena v sekci HMI. Uživatel aplikace má možnost přepínání mezi automatickým a manuálním režimem řízení soustavy. Nastavením získaných parametrů pomocí Kuhnovy metody do vytvořené aplikace v PLC, bylo dosaženo téměř totožných výsledků, jako při regulacích soustavy pomocí měřící karty v Matlabu.

# POUŽITÁ LITERATURA


[1] CVEJN, Jan. *Elektronický studijní materiál k předmětu Regulace a automatizace* [online]. 2014 [cit. 2019-08-28]. Dostupné z: https://portal.upce.cz

[2] HONC, Daniel. *Elektronický studijní materiál k předmětu Automatizace* [online]. 2013 [cit. 2019-08-28]. Dostupné z: https://portal.upce.cz

[3] MACHÁČEK, Jiří. *Pokročilé metody řízení procesů*. Pardubice: Univerzita Pardubice, 2015. ISBN 978-80-7395-937-1.

[4] O'DWYER, Aidan. *Handbook of PI and PID controller tuning rules* [online]. 3rd ed. Hackensack, NJ: Distributed by World Scientific Pub., c2009 [cit. 2019-08-28]. ISBN 18-481-6242-1.

[5] *SIMATIC S7-1200 CPU 1212C compact* [online]. [cit. 2019-08-28]. Dostupné z: https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6ES7212-1AE40-0XB0

[6] ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. *PLC a automatizace*. Praha: BEN - technická literatura, 1999. ISBN 80-860-5658-9.

[7] *Programming Guideline for S7-1200/S7-1500* [online]. [cit. 2019-08-28]. Dostupné z: https://w3.siemens.com/mcms/sce/de/fortbildungen/ausbildungsunterlagen/tia-portal/tab-cardseiten/Documents/HW-Konfig-S7-1200/81318674_Programming_guide-line_DOKU_v13_en.pdf

[8] *SIMATIC S7-SCL* [online]. [cit. 2019-08-28]. Dostupné z: https://w3.siemens.com/mcms/simatic-controller-software/en/step7/simatic-s7-scl/pages/default.aspx

[9] ŠMEJKAL, Ladislav. *PLC a automatizace*. Praha: BEN - technická literatura, 2005. ISBN 80-730-0087-3.

[10] *SIMATIC S7-1200 Analog output module* [online]. [cit. 2019-08-28]. Dostupné z: https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6ES7232-4HD32-0XB0

# PŘÍLOHY

# PŘÍLOHA A – CD

- Zdrojové kódy (Matlab, TIA Portal)
- Práce v elektronické podobě