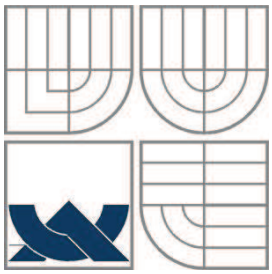
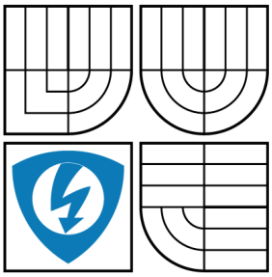


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A
KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

NADSTAVBA PROGRAMU WINCC
-PARAMETRIZACE PLC APLIKACE-

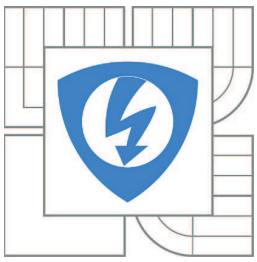
EXTENSION OF WINCC PROGRAM - PARAMETERISATION OF PLC APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — RADEK SYSEL
AUTHOR

VEDOUCÍ PRÁCE — ING. RADEK ŠTOHL PH.D.
SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Radek Sysel	***ID:***	133850
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Nadstavba programu WinCC - parametrizace PLC aplikace

POKYNY PRO VYPRACOVÁNÍ:

1. Seznámit se s problematikou řízení parních turbín firmy Siemens.
2. Navrhnout a realizovat sadu funkčních bloků ve vývojovém prostředí Simatic STEP7.
3. Realizovat grafické prvky v grafickém rozhraní WinCC pro ovládání parametrů parní turbíny.
4. Realizovat ukázkovou aplikaci v STEP7 a WinCC.
5. Ověřit funkčnost.

DOPORUČENÁ LITERATURA:

Dle vlastního literárního průzkumu a doporučení vedoucího práce.

Termín zadání:	11.2.2013	***Termín odevzdání:***	27.5.2013

Vedoucí práce: Ing. Radek Štohl, Ph.D.
Konzultanti bakalářské práce: ing. Peter Blaho

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cílem této bakalářské práce je navrhnout parametrizaci PLC aplikace a seznámit se se systémem řízení parních turbín vyráběných firmou Siemens v Brně. Orientovat se ve vizualizačním programu WinCC a propojit ho se Simatic STEP 7, který následně komunikuje s PLC. Následně na základě těchto poznatků vytvořit funkční aplikaci, která se začlení do jejich aplikace. Funkční aplikace a jejich aplikace budou spolu spolupracovat.

Po provedení teoretického návrhu jsem přistoupil k praktické realizaci dané aplikace. Nejdříve jsem si připravil funkční bloky ve STEP 7. Poté jsem přistoupil ke grafické realizaci v programu WinCC.

Na závěr jsem ověřil funkčnost a uvedl, jakým dalším směrem by se daná aplikace měla ubírat.

Klíčová slova

Simatic STEP 7, WinCC, vizualizace, PLC, zabezpečovací systém, jazyk C, funkční bloky, simulace, parní turbína.

Abstract

The aim of bachelor´s thesis is to propose parameters PLC applications and introduce a management system for steam turbines manufactured by Siemens in Brno. Be familiar with WinCC visualization program and link it with Simatic STEP7, which then communicates with the PLC. Subsequently, on the basis of this knowledge to create a functional application that is integrated into their applications. The functional application and their application will coordinate with each other.

After the theoretical design I came to the practical implementation of the application. First, I prepared the functional blocks in STEP 7. Then I went to the graphic's implementation in WinCC program.

In conclusion, I verified the functionality and I stated the other direction to which the application should take.

Keywords

Simatic STEP 7, WinCC, visualization, PLC, security system, programming language C, functional blocks, simulation, steam turbine.

Bibliografická citace:

SYSEL, R. *Nadstavba programu WinCC – parametrizace PLC aplikace*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 44s. Vedoucí bakalářské práce byl Ing. Radek Štohl, Ph.D.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Nadstavba programu WinCC – parametrizace PLC aplikace jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **24. května 2013**

……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Radku Štohlovi, Ph.D a celému oddělení řídících systémů firmy Siemens za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **24. května 2013**

……
podpis autora

Obsah

ÚVOD

Úkolem bakalářské práce je navrhnout nadstavbovou aplikaci pro stávající aplikaci firmy Siemens. Než jsem mohl cokoliv začít dělat, musel jsem si důkladně projít stávající aplikaci. A to nejen tu, která je naprogramována ve STEP 7, ale i tu ve WinCC. Bylo třeba pochopit jejich systém programování a dostat se blíže ke struktuře kódu.

V první kapitole stručně nastíním systém ovládání parní turbíny. V té další už se budu věnovat návrhu programu ve STEP 7 a popisovat systém předávání hodnot mezi PLC a programem.

Kapitola 3 je věnována pouze návrhu ve WinCC. Při návrhu ve WinCC jsem musel dbát na pozdější možné požadavky, to znamená dát programu možnost snadné úpravy o doplnění nových funkcí, aniž bych udělal zásadní zásah do samotné aplikace. Přitom aplikace musela být velice jednoduchá na ovládání.

Program se skládá z desítek vstupních/výstupních textových polí. Každý ovládá jiný parametr na parní turbíně. Ale požadavkem byly společně naprogramované funkce.

V dalších kapitolách již rozepisuji samotné důležité procesy probíhající v aplikaci. Problémem u práce bylo, že se nešlo o nic opřít. Probíhal tedy vývoj za pomocí poznatků od zkušených programátorů, analyzování a zkoušení různých cest, které by mě dovedly k cíli. Hledala se nejschůdnější cesta ke splnění požadavků. A to jak po stránce programové, tak po stránce grafické.

1 PARNÍ TURBÍNY SIEMENS

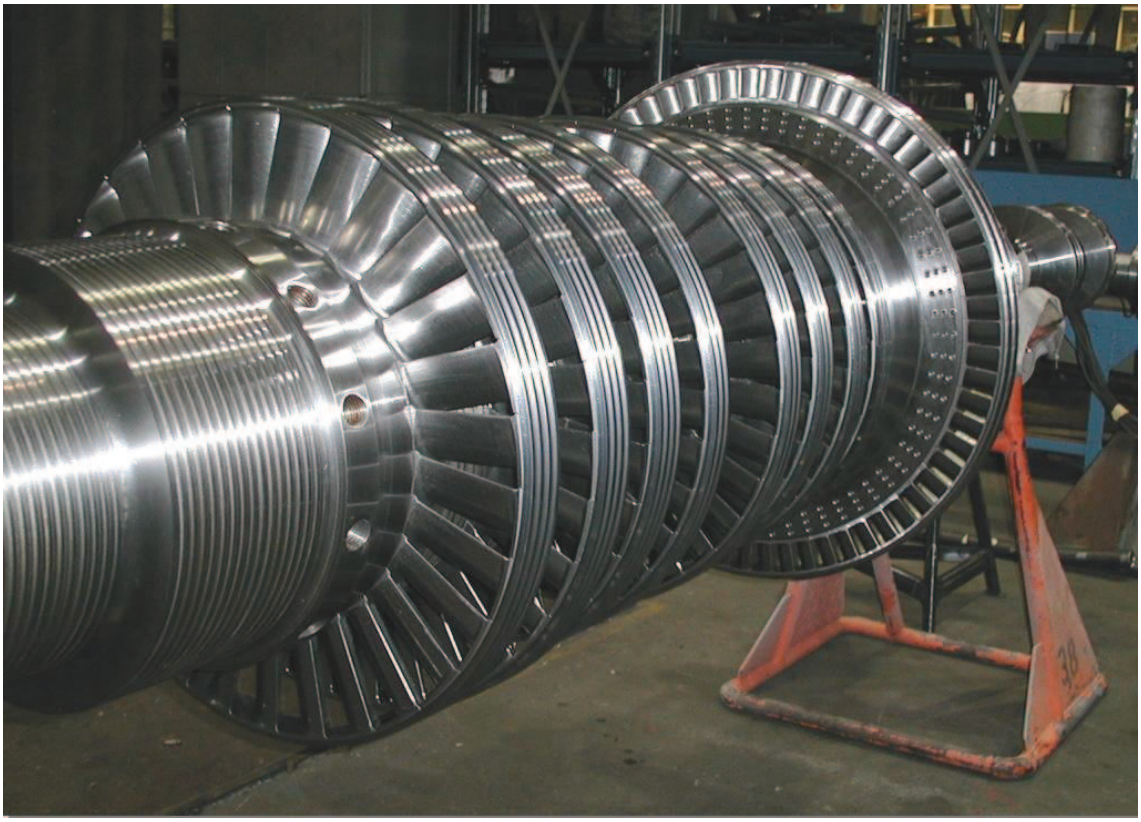
1.1 Parní turbína

Parní turbína převádí tepelnou energii páry na mechanickou. Za vynálezce bychom mohli považovat Brita Charlese A. Parsonse [4], který sestavil dvě skupiny patnácti za sebou řazených lopatkových stupňů kol do dvou skříní, jimiž pára prostupovala ze středu na obě strany, již v roce 1884.

Samotná výroba parních turbín v Brně má více než stoletou tradici. Vše odstartovala První brněnská strojírna roku 1900. Postupně se koncept parní turbíny zdokonaloval, nahradil zastaralý parní stroj, jenž měl velice malou účinnost.

Každý stupeň je tvořen pevnou rozváděcí lopatkovou mříží a oběžnou lopatkovou mříží umístěnou na rotoru (viz Obrázek 1). Rozváděcí lopatky tvoří řadu paralelních trysek. V závislosti na typu turbíny se mění i tvar lopatek.

V dnešní době se v Brně vyrábí parní turbíny na přehřátou páru. Vyrobí se jich přibližně 60 ročně a vyváží se do celého světa. Za zajímavé projekty stojí zmínka o polském městě Žerani, kam byla dodána jednotělesová turbína o výkonu 97 MW. Nebo parní turbína pro pontonovou elektrárnu, která je zakotvena u indického města Mangalore. Dále z českých projektů byla velice zajímavá parní protitlaková turbína s jedním regulovaným odběrem dodaná pro paroplynovou teplárnu Červený mlýn.



Obrázek 1: Rotor parní turbíny [5]

1.2 Rozdělení parních turbín

Turbíny se dělí podle několika kritérií [3]:

1) Tlak výstupní páry
   a) Kondenzační – Pára po průchodu turbínou kondenzuje v kondenzátoru.
   b) Protitlaké – Výstupní tlak je poměrně vysoký (0,11 až 0,6 MPa), takže se pára dá použít pro topné nebo technologické účely.
2) Odběry páry z turbíny
   a) Turbína s neregulovanými odběry – Pára se odebírá Na několika místech z turbíny a ohřívá napájecí vodu kotle, takže se zvětřuje účinnost tepleného oběhu.
   b) Turbína s regulovanými odběry – Pára se odebírá jedním nebo max. třemi odběry vhodným tlakem, kterými se dodává teplo spotřebitelům. Odběr páry se reguluje podle požadavků spotřeby.
3) Počet těles
   a) Jednotělesové – Pro menší výkony. Jedna skříň = jeden rotor.
   b) Vícetělesové – S částí vysokotlakou, nízkotlakou, popř.: středotlakou, pro větší výkony. Více skříní = více rotorů, vzájemně spojených do jednoho celku.

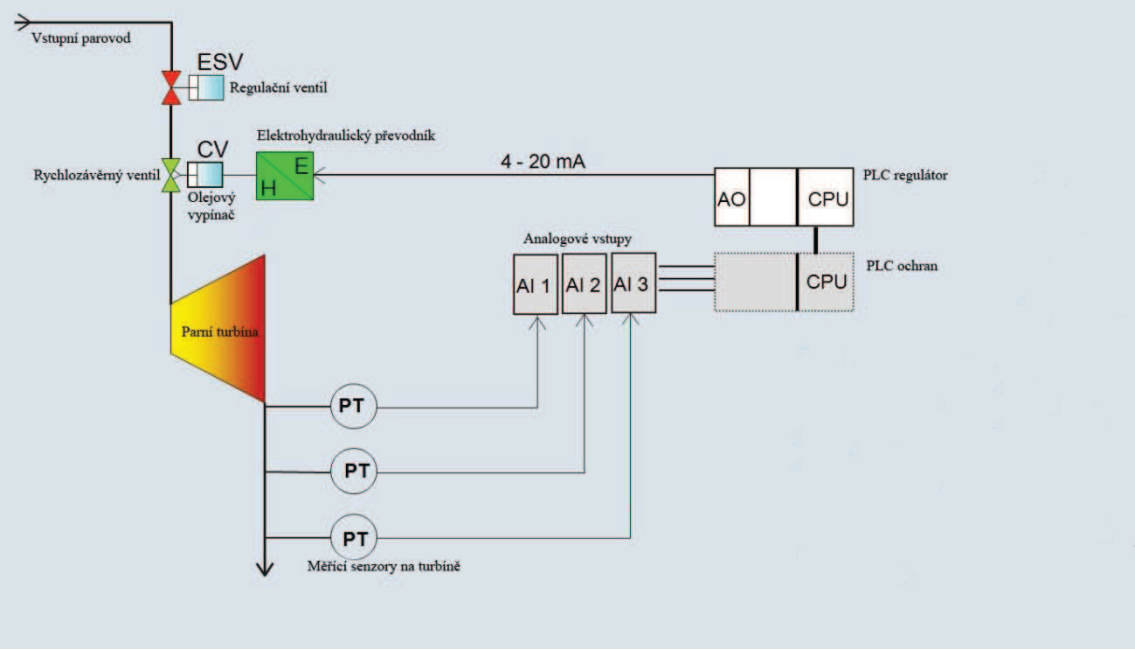
1.3 Princip řízení parních turbín

Vstupním parovodem jde pára do regulačního ventilu (Obrázek 5: Regulační ventil). Ten je ovládán v PLC regulátoru. Regulační ventil ovládá množství páry, která jde do turbíny.

Rychlozávěrný ventil (viz Obrázek 4) následuje hned po regulačním ventilu. Je ovládaný pomocí olejového vypínače (viz Obrázek 3), který je řízen PLC ochran. Dále hlavní funkcí ventilu je, aby v případě poruchy zastavil přívod páry do turbíny, která se nakonec zastaví.

Celý princip řízení turbíny spočívá v tom, že na turbíně jsou měření, ze kterých jdou informace do analogových vstupů a ty vedou do PLC ochran. Poté přes komunikaci do PLC regulátoru, kde se vše zpracuje a vyšle se signál na ovládání elektrohydraulického převodníku. V něm se patřičně vyreguluje turbína, následně se vyšle signál na regulační ventil, který podle něj se buď více otevře a zvedne se výkon turbíny, popřípadě zavře.

Celý cyklus můžete vidět zde (viz Obrázek 2).



Obrázek 2: Celý systém řízení [4]

Obrázek 5: Regulační ventil [2]

Obrázek 4: Rychlozávěrný ventil [2]

Obrázek 3: Olejový vypínač [2]

1.4 Důležitá měření na parní turbíně

Pokud je to možné, musí být k dispozici alespoň tyto základní přístroje [6]:

a) Tlaky
   - Na vstupní a přihřáté páře těsně před vysokotlakými a nízkotlakovými ventily a síty.
   - V odběrech u odběrových turbín.
   - V odběrech pro ohříváky napájecí vody.
   - Ve výstupech z každého tělesa.

— Ve zdroji mazacího oleje pro ložiska.
— Ve zdroji kapaliny pro řídící systém.

b) Teploty
— Na vstupní a přihřáté páře.
— Na páře na výstupu z vysokotlakového a středotlakového tělesa.
— V odběrech k ohřívákům napájecí vody.
— Na oleji na výstupu z chladiče.
— Na oleji vytékajícího z ložisek nebo v ložiskách v kovu.

c) Hladiny
— Hladina mazacího oleje v olejové nádrži.
— Hladina v nádrži regulační kapaliny.

Nyní bych rád některá zařízení, popřípadě měření, rozepsal blíže:

Digital Overspeed Protection Systém (DOPS) – Patří mezi nejdůležitější zařízení v řídicím systému parní turbíny. Slouží k měření otáček a k ochraně nadotáček. V případě, že dojde ke zvýšení otáček nad danou mez, dojde k odpojení napájení pro solenoidy v olejovém vypínači.

Regulace výkonu – Podle toho, jaký si operátor zvolí výkon, takový se bude držet. Dává signál regulátoru otáček (přidej/uber), aby se dosáhlo požadovaného výkonu. Začne otevírat, popřípadě uzavírat regulační ventily. Regulátor výkonu popřípadě nahrazuje regulátor vstupní páry (oba zároveň fungovat nemůžou).

Tlak páry na výstupu – Je-li na výstupu turbíny vyšší tlak než je přípustný, dochází k většímu namáhání lopatek. Při překročení povolené hodnoty se tedy odstaví parní turbína.

Tlak regulačního oleje – Tlak v olejovém vypínači. Když olej chybí, parní turbína se odstaví.

Tlak a teplota mazacího oleje – Pro tlak se používají čerpadla, která vytvoří tlak. Čerpadlo tlačí do ložisek olej, který chladí ložiska a vytvoří kluznou emulzi. Na ložiskách se vytvoří olejový film, rotor na něm plave. Měření tedy hlídá, aby bylo dostatek oleje pod určitou teplotou. Řídký teplý olej nebude dostatečně chladit a mazat ložiska. Opět by došlo k odstavení turbíny. Většinou se na turbínách nachází tři čerpadla.

2 FUNKČNÍ BLOKY V SIMATIC STEP7

2.1 PLC automat

Začátkem bych chtěl uvést, že všechny procesy budou fungovat na PLC automatech typu S7-400 (viz Obrázek 6). Hlavní znaky těchto automatů jsou:

- Výkonné automaty pro středně a vysoko náročné technologie
- Velký výběr typu CPU s narůstajícím výkonem
- Možnost rozšíření až na 300 modulů
- Komunikační sběrnice (bus) integrovaná do modulů
- Možnost propojení do sítě s – Multipoint Service (MPI)
  – PROFIBUS
  – Industrial Ethernet
- Centrální připojení na PG s přístupem do všech modulů
- Žádné omezení pro sloty
- Multiprocesorová činnost

Obrázek 6: Simatic S7-400 na projekt LUANDA REFINERY

2.2 Proměnné parametry funkčních bloků Siemens

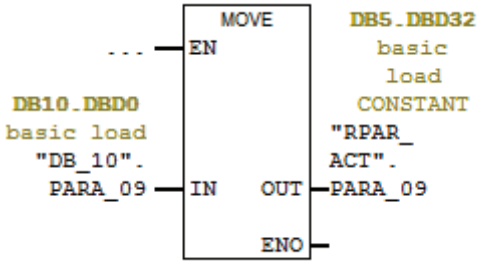
Soupis všech parametrů, které se budou měnit, najdete v příloze. Část jich ukážu v tabulce zde (viz Tabulka 1).

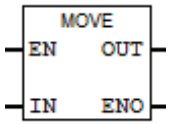
Všechny parametry, které jsou proměnné, původně proměnné nebyly, ale byly zadány jako konstanty. To značně ulehčovalo jejich implementaci do funkčního bloku daného spotřebiče, popřípadě regulátoru. Teď, když daná hodnota není konstantní, ale je posílána z vizualizačního programu WinCC jako určitá proměnná (záleží na typu parametru), je třeba zajistit, aby dorazila ve správném formátu do funkčního bloku. Toho docílíme pomocí bloků *Move* (viz. Obrázek 7).

S blokem *Move* můžeme zkopírovat do výstupu OUT všechny základní datové typy s délkou 8, 16 nebo 32 bitů.

Group	Název funkce	Název parametru	Popis parametru	data typ
SPEED	SPEED GOVERNOR	PARA_01	KP SPEED	REAL
FRQ SPEED	FREQUENCY CONTROLLER SPEED GOVERNOR	PARA_02	Nominal speed %	REAL
SPEED	SPEED GOVERNOR	PARA_03	DB speed freq ON	REAL
SPEED	SPEED GOVERNOR	PARA_04	DB speed freq OFF	REAL
SPEED	BASIC LOAD	PARA_09	basic load CONSTANT	REAL
SPEED	SPEED TREATMENT	PARA_16	speed measurement filter constant /synchro with grid	REAL
		PARA_17	Deadband at nominal speed	REAL
SPEED	SPEED SETPOINT RAMP	PARA_18	SP ramp turbine cold input SP integrator	REAL
FRQ	FREQUENCY CONTROLLER	PARA_19	KP gain frequency controller	REAL
FRQ	FREQUENCY CONTROLLER	PARA_20	deadband frequency controller	REAL
FRQ	FREQUENCY CONTROLLER	PARA_21	deadband-Hysterese frequency controller	REAL
SPEED	SPEED SETPOINT	PARA_24	MAX SPEED GBO	REAL
SPEED	SPEED SETPOINT	PARA_25	MAX SPEED GBO IMPULS	REAL
SPEED	SPEED SETPOINT	PARA_26	MAX SPEED GBO IMPULS ISLAND	REAL
SPEED	SPEED SETPOINT	PARA_27	MAX SPEED OSSPEED TEST	REAL
SPEED	SPEED SETPOINT	PARA_28	MAX SPEED NOT GBO	REAL
SPEED	ACCELERATION LIMTER	PARA_32	Kp Accel limiter	REAL
SPEED	ACCELERATION LIMTER	PARA_33	OFFSET Accel limiter	REAL
SPEED	ACCELERATION LIMTER	PARA_34	Tn1 Accel limiter	REAL
SPEED	ACCELERATION LIMTER	PARA_35	Tn2 Accel limiter	REAL
SPEED	ACCELERATION LIMTER	PARA_36	Tn switch Accel limiter	REAL
SPEED	SPEED SETPOINT RAMP	PARA_51	Autostop ramp	REAL
SPEED	VALVE POSITION	PARA_56	maximales Stellsignal vom Positionsregler FDRV 1	REAL

Tabulka 1: Proměnné parametry funkčních bloků





Obrázek 7: Blok Move

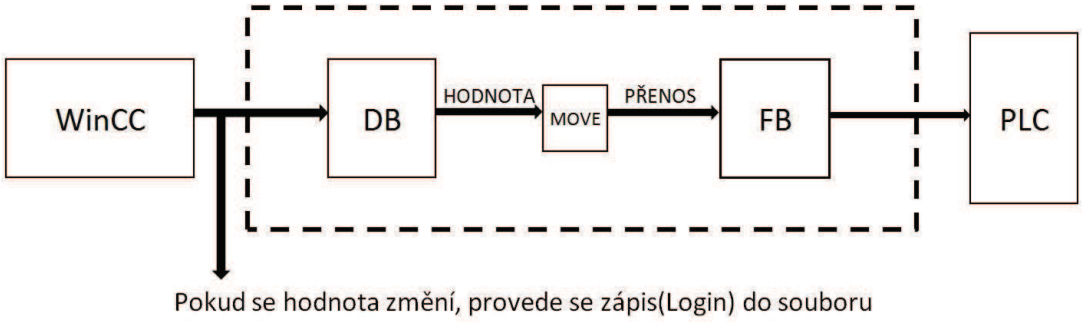
Obrázek 8: Použití bloku Move s opravdovými hodnoty

2.3 Vstupy a výstupy funkčních bloků

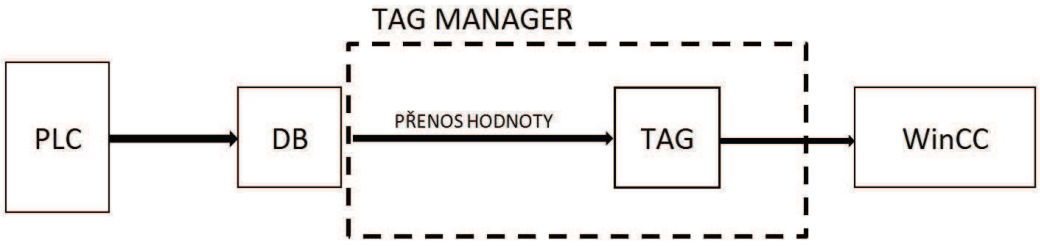
Funkční bloky samotné se měnit nebudou. To je velice důležité si uvědomit, protože odpadá starost s alarmovým hlášením jednotlivých bloků. To zůstává stále stejné a má parametrizace to neovlivní na tolik, aby se musely hlášení přenastavovat.

Lze tedy napsat, že vstupy i výstupy zůstanou funkčně stejné až na jediný vstupní parametr, kam se bude překlápět vyslaný signál z vizualizačního programu WinCC. Signál půjde napřed do Data bloku, kam se uloží, následně přes blok *Move* se přesune do proměnné, která bude vést přímo do vybraného funkčního bloku.

Schéma celého průběhu (viz Obrázek 9):



Obrázek 9: Schéma přenosu hodnoty z WinCC do PLC



Obrázek 10: Schéma přenosu hodnoty z PLC do WinCC

2.4 Funkční bloky Siemens

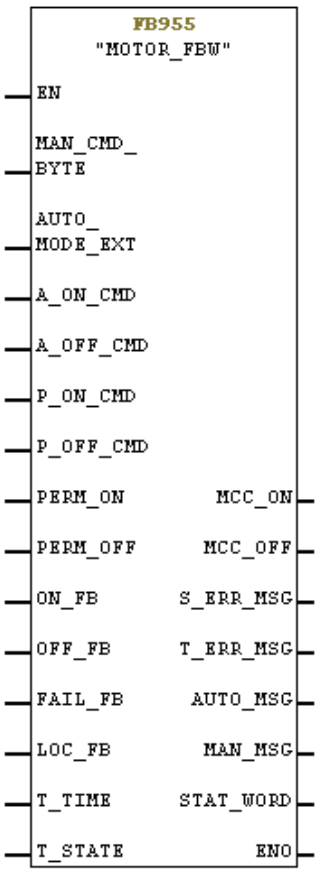
Funkční bloky firmy Siemens představují mechanismus regulátorů výkonu, otáček, různých motorů a jiných spotřebičů použitých na jejich turbínách. Každý funkční blok většinou představuje jeden z těchto prvků.

Vstupem jsou jeho původní parametry a výstupem stavové slovo, kam je zapisován stav daného prvku. Stavové slovo má pak úložné místo v paměti PLC a dle potřeby teda může být ukázán výpis. Daný prvek je ovládán přes povelové slovo, které představuje jeden byte. Jeden byte je sekvence 8 bitů, každý bit má jiný význam pro příkaz prvku. Pro ukázku zde uvedu celý funkční blok motoru (viz Obrázek 11), který slouží jako pomocné čerpadlo mazacího oleje, a jeho ovládání. Motor jsem si zvolil proto, že na něm budu moct nejlépe ukázat vlastnosti a princip ovládání funkčních bloků Siemens.

V předchozí kapitole jsme se bavili o proměnných parametrech, u našeho motoru se bude jednat o parametr čas prodlevy zapnutí po vyslání signálu na motor. Stejně jako všechny ostatní parametry jiných spotřebičů, i náš motor má daný parametr uložen v datovém bloku, kam se jeho hodnota překlápí z funkčního bloku, kde je umístěn.

Povelové slovo: Vstup je MAN_CMD_BYTE
Nyní se podíváme na jednotlivé bity povelového slova:
1.bit – Potvrzovací příkaz ->Slouží k provedení příkazů od 3.bitu výš
2.bit – Zrušení příkazu ->Slouží ke zrušení příkazů od 3.bitu výš
3.bit – Zapne motor
4.bit – Vypne motor
5.bit – Automatický mód motoru
6.bit – Manuální mód motoru

Dále si rozepíšeme jednotlivé vstupy funkčního bloku:
PERM_ON – Podmínka, kdy se může zapnout motor.
PERM_OFF – Podmínka, aby šel motor vůbec zapnout. Musí být TRUE.
ON_FB – Zpětná indikace, že motor běží.
OFF_FB – Zpětná indikace, že motor neběží.
FAIL_FB – Elektrická porucha. Měla by vše zablokovat.
LOC_FB – Motor nelze ovládat automaticky a ani manuálně. Ovládá se ze skříňky u turbíny.
T_TIME – Představuje čas, po který se čeká na potvrzení povelu
T_STATE – Nastavuje potřebný čas pro přepnutí z jednoho stavu motoru do druhého.
– Musí čekat a nenahlásit chybu.
Motor potřebuje svůj vlastní data blok, kam může ukládat své proměnné.



Obrázek 11: Funkční blok motoru

3 GRAFICKÉ ROZHRANÍ WINCC

3.1 Program WinCC

WinCC je vizualizační program, který slouží jako grafické rozhraní, pro posílání příkazů do programovatelného automatu a jeho celkové ovládání. Ačkoliv se to nezdá, je WinCC spíše nástroj pro programátora než pro inženýra. Výstup sice bude vždy v podobě obrázků a tlačítek, ale největší efektivnosti s ním dosáhneme až tím, když si danou událost na určitý prvek naprogramujeme.

Programováním můžeme dosáhnout lepší ovladatelnosti celého celku, a zároveň na jedno kliknutí ovlivnit více věcí. Je zde na výběr možnost programovat ve Visual Basic nebo v C. Osobně jsem si vybral jazyk C, protože s ním mám více zkušeností a zároveň i ve firmě Siemens ho používají ve WinCC jako hlavní programovací jazyk. Navíc je ve WinCC speciální prostředí pro programování v C (viz Obrázek 21). Prostředí umožňuje používat všechny funkce z knihovny C, syntaxe a způsob programování je také zachován.

WinCC má své knihovní funkce napsané v jazyku C pro ovládání vlastních grafických prvků. Dané funkce ulehčují samotné programování, protože dovolují přímý přístup ke grafickým prvkům na výstupu programu. Mají rovnou v sobě implementovány přístupové parametry grafických prvků.

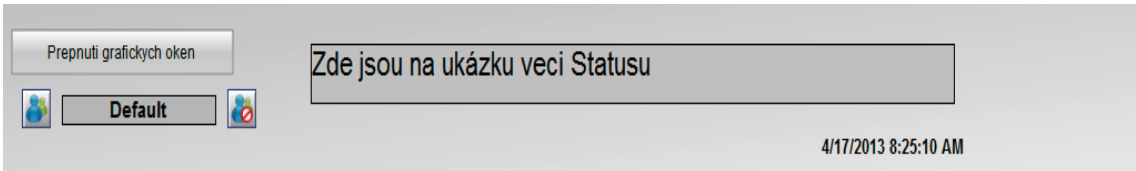
3.2 Simulační rozhraní

Dříve než se mohl provést samotný návrh parametrů, bylo třeba si vytvořit simulační prostředí, které by se chovalo stejně jako prostředí aplikace firmy Siemens, do které se pak bude aplikace mnou vytvořená přidávat. Simulační prostředí bylo také důležité z toho důvodu toho, že mi dávalo prostor vytvořit mé parametry.

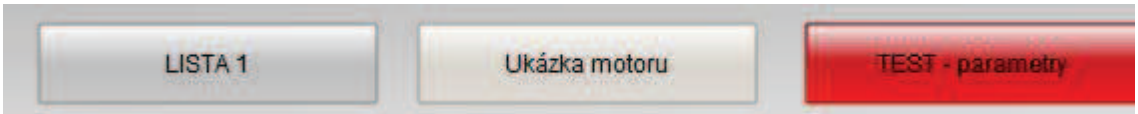
Simulační prostředí se skládá z následujících částí: Hlavní okno, které se nazývá *@Main.pdl*. Toto okno pak v sobě obsahuje následující části:

- @Status – Obsahuje okno na přihlašování a může obsahovat různá tlačítka, která umožňují specializovaná nastavení pro běh aplikace (viz Obrázek 12).
- @Toolbar.pdl – Obsahuje tlačítka menu (viz Obrázek 13). Pomocí nich se dostaneme do hlavních částí aplikace.
- Process.pdl – Zde se zobrazují samotné procesy aplikace (viz Obrázek 14).
- @Status_down.pdl – Doplňující informace pro operátora (viz Obrázek 15).
- @Login.pdl – Přihlašovací okno (viz Obrázek 16). Dovoluje nám napojení na zabezpečovací systém stávající aplikace.
- GSC Diagnostics – Velice důležité okno pro programátora (viz Obrázek 18). Dovoluje mu totiž odlaďovat případné chyby v programu.

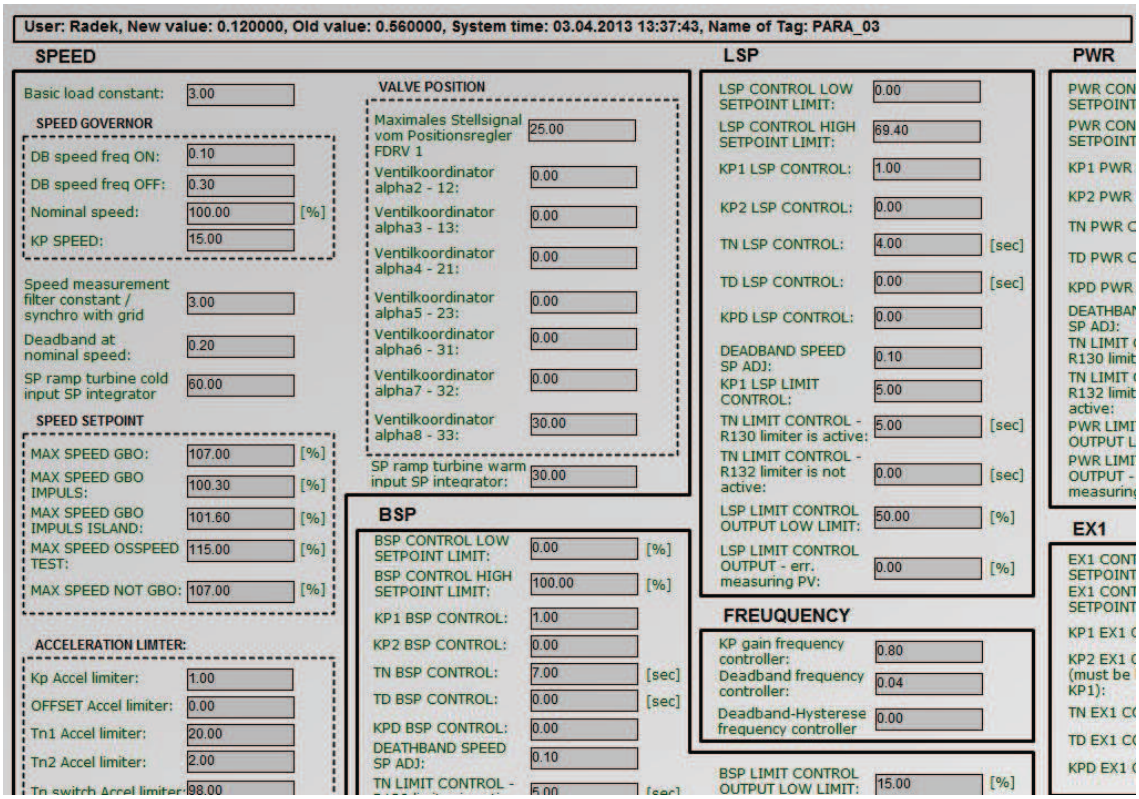
Procesní okna se navzájem překrývají. Zobrazuje se jen to, které jsme si zvolili v menu.



Obrázek 12: Okno status

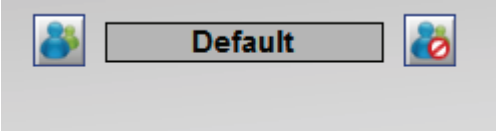


Obrázek 13: Toolbar menu

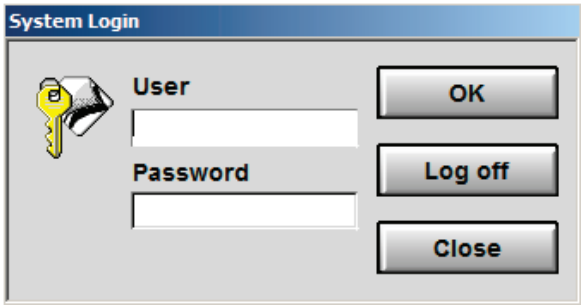


Obrázek 14: Process okno

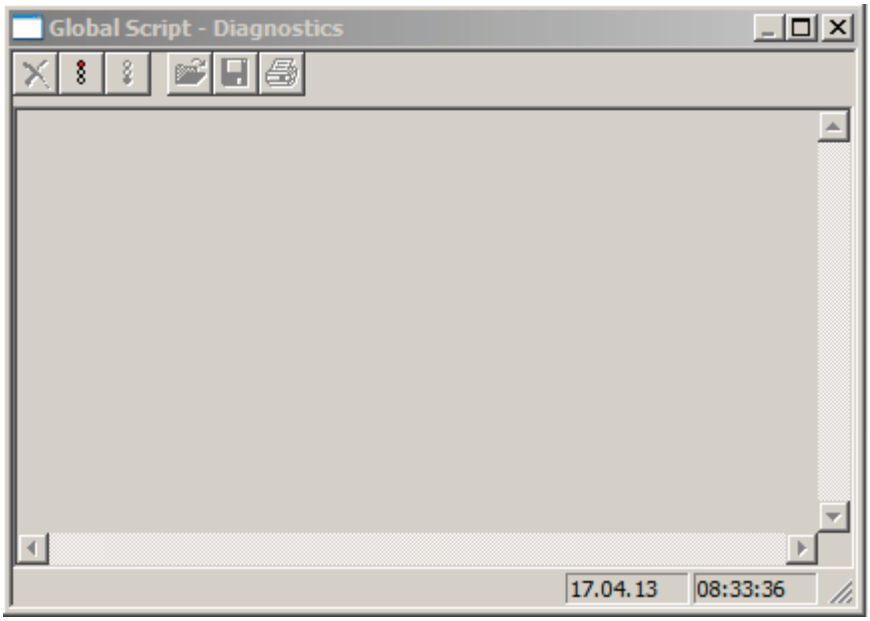
Obrázek 15: Status_down okno



Obrázek 16: Login do systému



Obrázek 17: Přihlašovací okno



Obrázek 18: Global Script - Diagnostics

3.3 Návrh parametrů

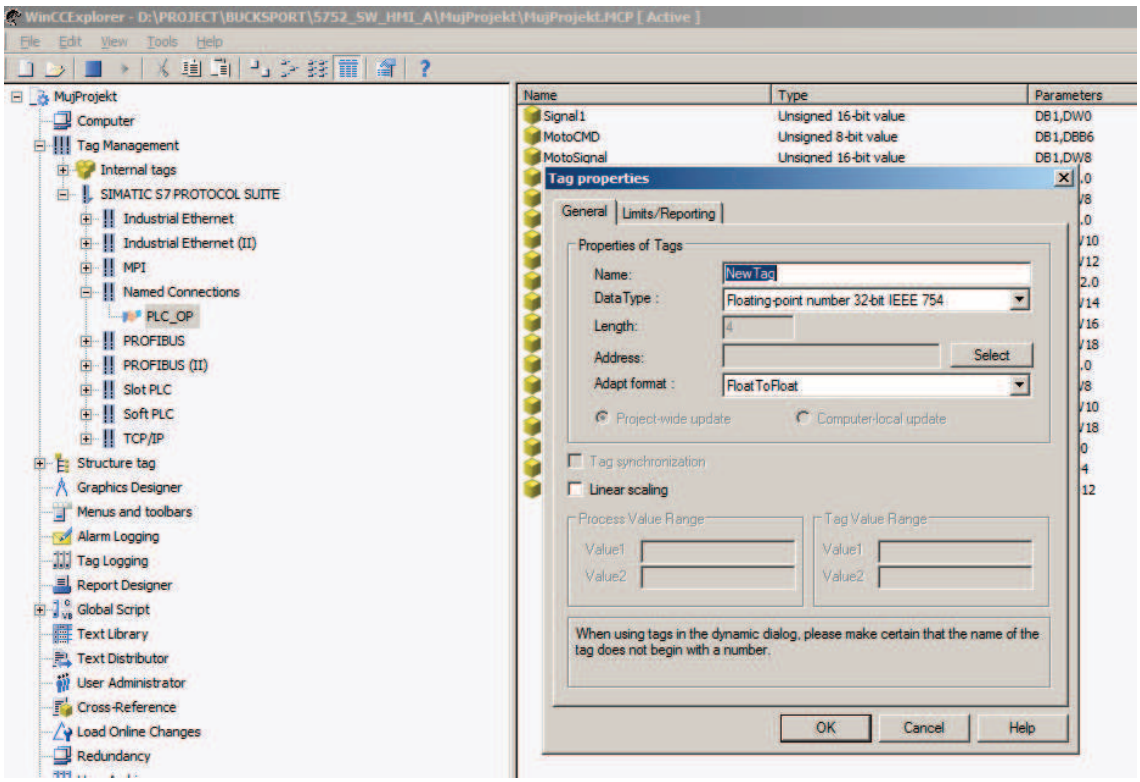
Při návrhu možnosti ovládání parametrů je třeba vycházet z toho, aby se nenarušila základní struktura ovládání parní turbíny, která je již vytvořena. Proto se musí vytvořit úplně nová stránka ve WinCC, kde budou všechny parametry zobrazeny a pomocí textových polí půjdou měnit. Do textových polí půjde vždy zapisovat ve formátu, v kterém je daný parametr nastaven. Do polí typu INT půjdou zapisovat pouze celá čísla, do polí typu REAL i čísla desetinná a podobně.

Každý parametr se bude ukládat do proměnné, zde se to nazývá TAG, přes ten se to pak přenese do místa v paměti PLC. TAGy se vytvářejí v Tag manažeru (viz Obrázek 19). Určí se vždy typ proměnné a místo v paměti PLC, kam se hodnota zapíše.

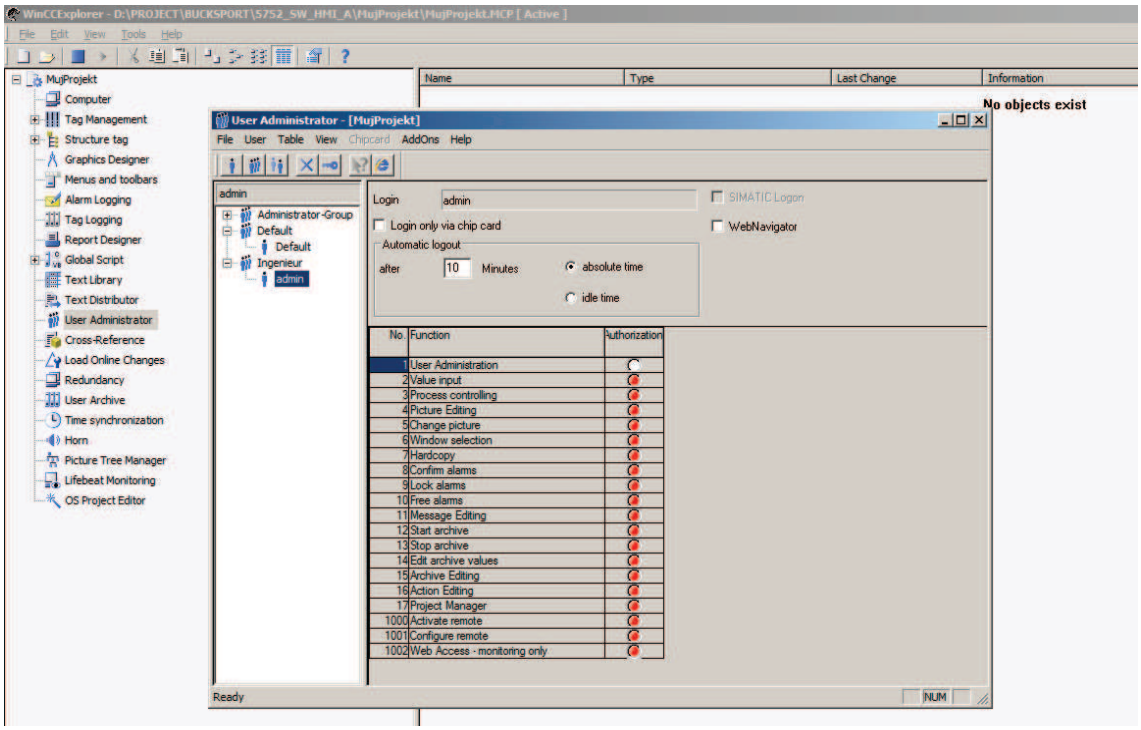
Dále je při návrhu parametrů důležité myslet na systém přihlašování, tedy na to, aby parametry mohli měnit jen určití uživatelé. Na to slouží aplikační část User Administrator (viz Obrázek 20), kde si můžeme vytvořit celé skupiny uživatelů

s určitými právy. Všichni uživatelé budou mít právo *Visible*, to znamená, že hodnoty parametrů si budou moct prohlédnout. Právo *Value*, tedy možnost do textového pole vepsat novou hodnotu, budou mít jen uživatelé s platným heslem.

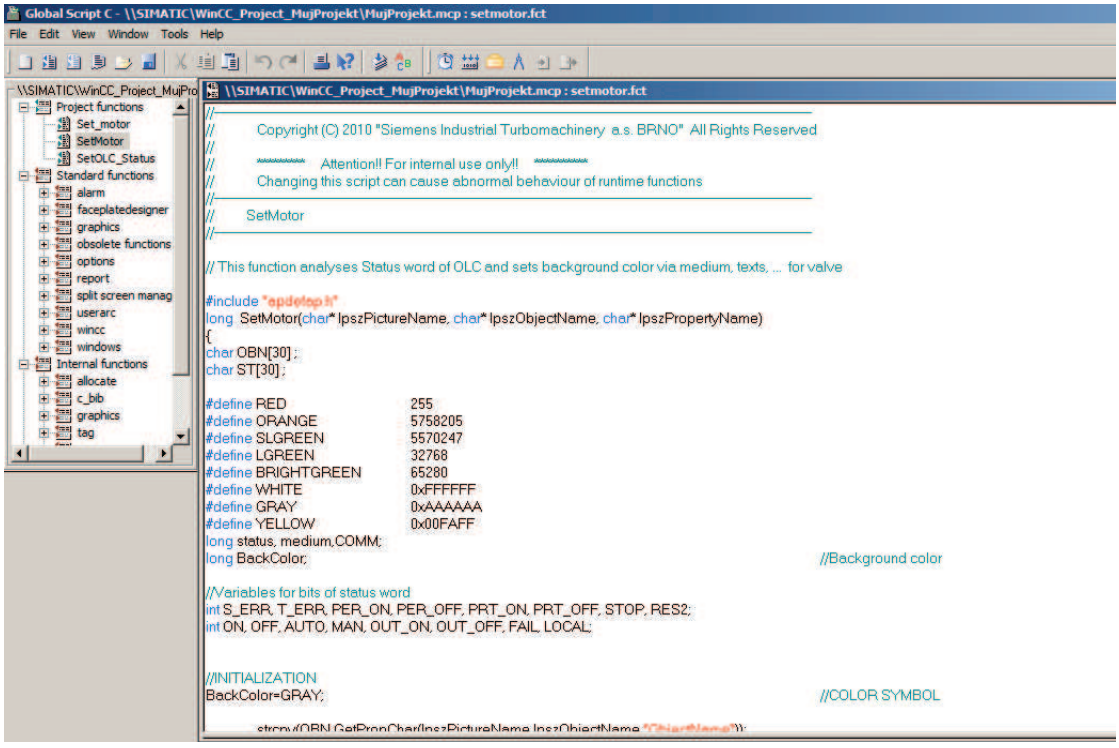
O hodinovém hesle se více dozvíme v kapitole HODINOVÉ HESLO, kde bude názorně popsáno.



Obrázek 19: Vytvoření nového TAGu v Tag Manageru



Obrázek 20: Okno User Administrator



Obrázek 21: Prostředí pro programování ve WinCC

3.4 Postup přidání prvku do WinCC

Program WinCC má stejně jako jiné grafické editory paletu nástrojů (viz Obrázek 24), pomocí kterých lze vytvářet jak nové tvary, tak upravovat stávající. Je zde možnost upravovat barvy vybraných prvků, popřípadě měnit jejich vlastnosti.

Všechny úpravy se provádí v grafickém editor (viz Obrázek 38). Pro nás je důležité, že jde přidat I/O pole, do kterého lze jak hodnotu vkládat, tak i naopak hodnotu zobrazit z vybraného TAGu. Naše I/O pole obsahuje několik vlastností, které si musíme upravit, aby celý náš systém fungoval. Na ty vlastnosti si napojíme naše vlastní funkce vytvořené v C. Funkce jsou napsány pro globální použití a to z toho důvodu, aby celý systém ovládání parametrů byl jednodušší na správu či případné změny ve struktuře.

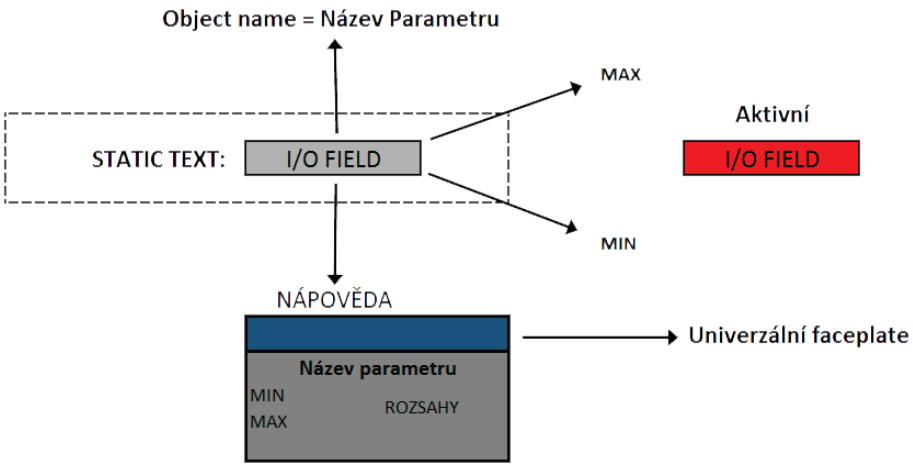
Rád bych zde vysvětlil ještě jeden pojem, se kterým se v následujících kapitolách budeme velice často setkávat. Je to pojem *faceplate* [číst "*fejsplejt"*]. Je to dialogové okno vztahující se vždy k určitému prvku (např. symbolu spotřebiče motoru). V aplikaci se toto dialogové okno zobrazuje na kliknutí pravého tlačítka myši. V našem případě bude *faceplate* nejčastěji zobrazovat nápovědu k parametrům. Ale *faceplate* může klidně sloužit jako dialogové okno pro vložení hesla nebo panel s řídícími prvky pro spotřebič. Možností je opravdu hodně.

Nejdříve si ve zkratce rozepíšeme, které naprogramované funkce se používají v I/O poli (bližší podrobnosti o nich jsou v dalších kapitolách):

- **SaveOld(char* lpszPictureName, char* lpszObjectName)** – Uloží do interního TAGu *oldValue* starou hodnotu parametru. Vstupními parametry jsou názvy obrázku, na kterém s objekt nachází a jméno objektu.
- **LoginNew(char* lpszPictureName, char* lpszObjectName)** – Po zavolání se provede kontrola, zda má uživatel právo zapisovat, poté se zkontroluje, zda je prvek aktivní a na závěr se zavolá funkce ZapisHodnotu, která zapíše veškeré změny. Ke zjištění práva uživatele se používá interní funkce WinCC, ta se skrývá pod názvem *PASSCheckLevelPermission("OS_low",2)*. V závorce je důležitá hodnota *2*, značí totiž, že se jedná o právo zapisovat. Pokud daná funkce vrátí hodnotu *TRUE*, má uživatel dané právo přiděleno. Daná funkce se volá z funkce *PrechodDoplnek(char* lpszPictureName, char* lpszObjectName)*.
- **ZapisHodnotu(char uzivatel[30],double oldValue,double newValue, char paramName[10], char functionName[10])** – Na základě předaných parametrů, zapíše do souboru jméno uživatele, starou hodnotu, novou hodnotu, jméno změněného parametru a čas změny.
- **VratCislo(char* lpszPictureName, char* lpszObjectName) –** Po předání názvu objektu parametru se do nového řetězce vloží daný název s posledním znakem názvu objektu navíc. Zavolá se na základě nového názvu daný TAG a vytáhne se hodnota, která bude návratovou hodnotou dané funkce.
- **show_FP(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName, char* PictWin, char*FacePlate) –** Otevírá faceplaty.

Vezme si souřadnice umístění myši a souřadnice objektu, na který jsme klikli. Následně zobrazí faceplate co nejblíž daných souřadnic.

- **LoadLimit(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName) –** Podle názvu objektu vyhledá v textovém poli jeho řádek a načte k parametru mezní hodnoty.
- **LoadLast() –** Vrací z logovacího souboru poslední provedenou změnu.
- **LoadAll() –** Vrací z logovacího soubory všechny provedené změny.
- **HelpFP(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName) –** Na základě názvu parametru se stará o to, aby se do nápovědy k danému parametru načetly odpovídající informace.
- **PrechodDoplnek(char* lpszPictureName, char* lpszObjectName) –** Velice důležitá funkce. Stará se o zápis nové hodnoty do databloku v PLC. Zajišťuje *beznárazový přechod* zvoleného parametru.



Obrázek 22: Grafický popis parametru

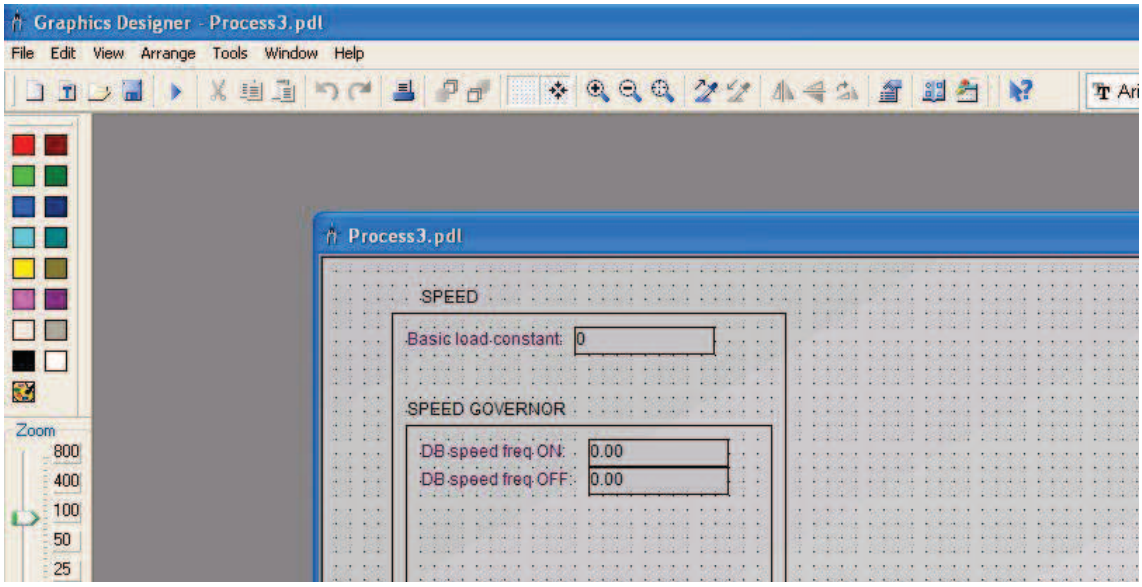
Dále se podíváme na vlastnosti samotného I/O pole, které budeme měnit:

- **Background Color –** Hlídá barvu I/O pole. Pokud je aktivní, pozadí pole je červené. Jinak šedé. O celou funkčnost se stará funkce *SetBackgroundIO()*.
- **Low Limit –** Dolní hranice pro zadání hodnoty do parametru.
- **High Limit –** Horní hranice pro zadání hodnoty do parametru.
- **I/O Field –** Umožňuje parametr udělat přípustným pro vkládání hodnoty, popřípadě ho zablokovat. Na tuto vlastnost je napojená funkce *IOCheck()*.
- **Input Value** – Zde zůstane hodnota prázdná
- **Output Value** – Je třeba zvolit *Tag*, odkud se bude hodnota načítat. To vše pomocí funkce *VratCislo()*.
- **Data Format** – Decimal (desítková soustava).

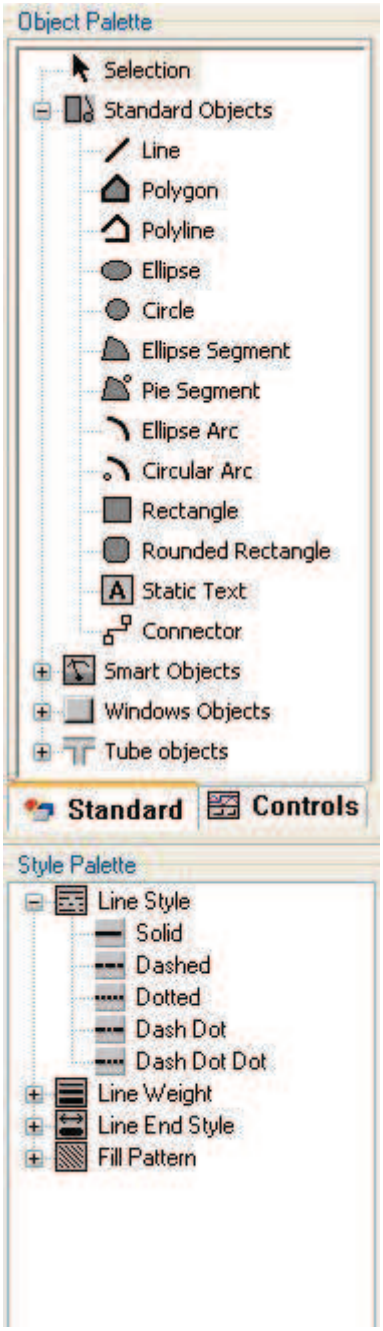
- **Apply on Full** – No (pozn. Anglicky). Hodnota bude vložena až po stisknutí klávesy *ENTER*.
- **Authorisation** – Hodnota nastavená na *Value*. Znamená, že do pole se může zapisovat jen s právy *Value*, je to ochrana WinCC.
- **Mouse Action** – Po kliknutí jakýmkoliv tlačítkem myši se zavolá funkce *LoadLimit()*.
- **Press Right** – Přímá akce na pravé tlačítko myši. Do Tagu *HelpTag* se odešle název parametru.
- **Release Right –** Na držení pravého tlačítka myši se zavolá funkce *show_FP()*.
- **Focus Change** – Provádí se zde zálohování staré hodnoty pro případ, že by byla hodnota změněna za novou. Na tuto vlastnost je napojena funkce *SaveOld*.
- **Property Topics**: **Input Value: Change –** Na jakoukoliv změnu vstupního pole se zavolá funkce *PrechodDoplnek()*.

Shrnutí: Po kliknutí na I/O pole se zaktivuje *Focus Change*, dojde k uložení staré hodnoty. Při přepsání hodnoty staré za novou (tzn., liší se) a stisknutí klávesy ENTER, dojde k zavolání funkce *PrechodDoplnek()*, která provede nejdříve kontrolu,zda je vše v pořádku, následně provede postupná zápis nové hodnoty do PLC. Poté se ve funkci zavolá funkce *LoginNew()*,která provede zápis změny do souboru.

Celý systém funguje velice spolehlivě.



Obrázek 38: Graphics Designer ve WinCC s ukázkou třech I/O polí parametrů



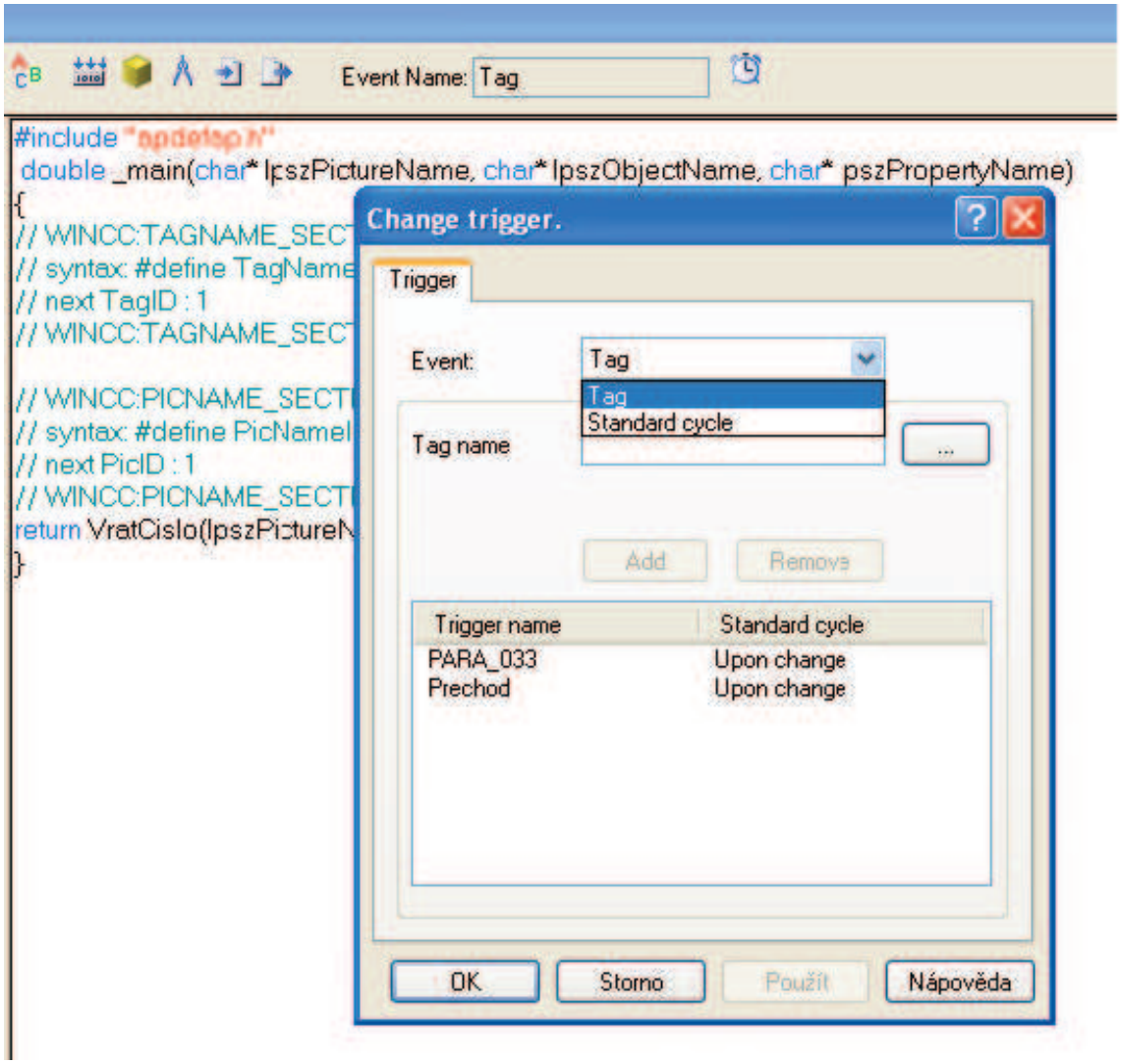
Obrázek 24: Paleta nástrojů ve WinCC

Paleta nástrojů obsahuje zajímavé prvky. Pomocí nabídky z *Windows Objects* si můžeme vkládat prvky, které nám velice zjednoduší práci s aplikací. Jedním z nich je tzv. *Windows picture*. To je pomocný obrázek, který můžeme nahrazovat za jiné obrázky. Mimo jiné se pomocí něj vyvolávají faceplaty ke všem spotřebičům, obecně ke všem objektům na obrazovce.

3.5 Trigger ve WinCC

Velice důležitou roli má ve WinCC tzv. trigger. Trigger slouží jako časovač, popřípadě spoušť určité události na základě předem daného podnětu. Když například prvek, který je napojený na číselný TAG, překročí určitou hodnotu, tak se může vyvolat na základě triggeru událost. Může to být zbarvení prvku, zmizení prvku, vyskakovací okno upozorňující na překročení hodnoty a podobně.

Jaké máme možnosti triggeru? Vybrat si můžeme mezi napojením na TAG nebo standardním cyklem (viz Obrázek 25).

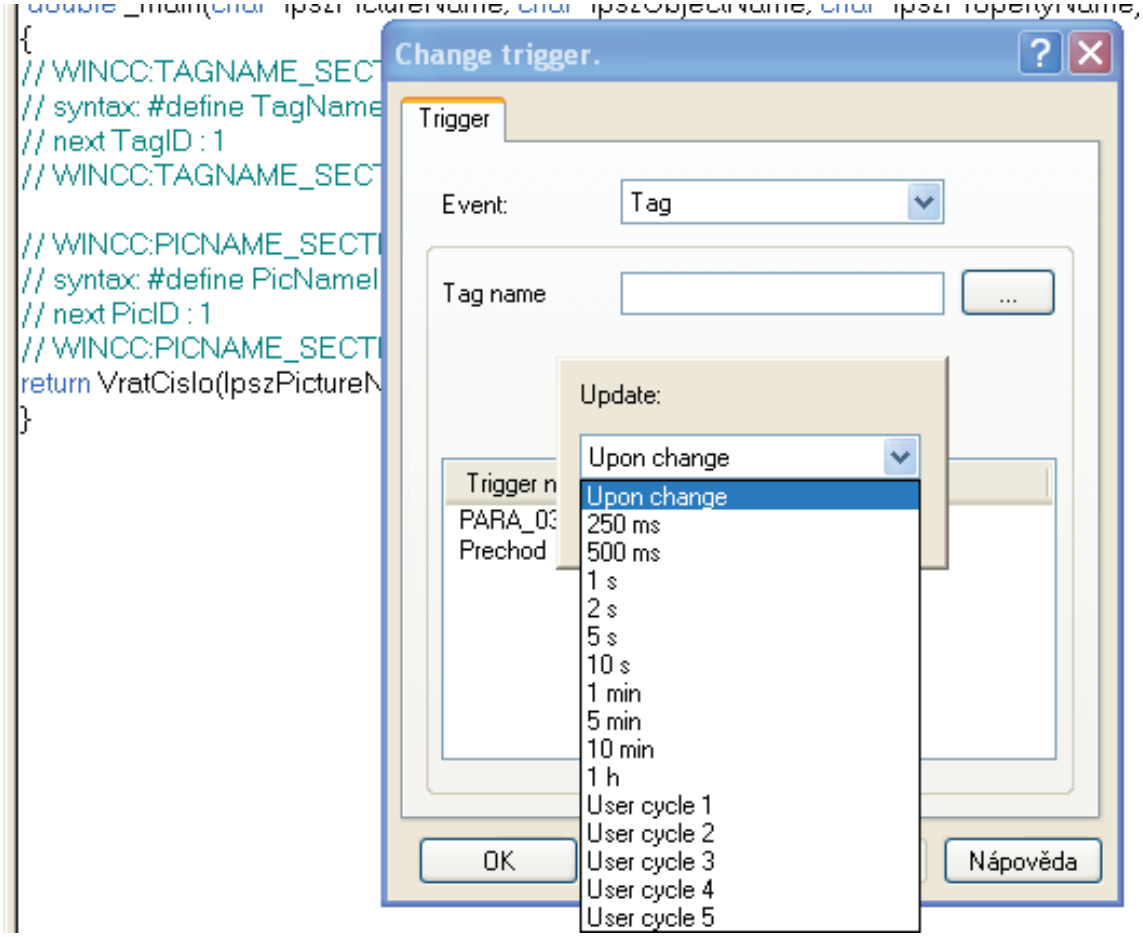


Obrázek 25: Možnost volby triggeru mezi TAGem nebo standardním cyklem

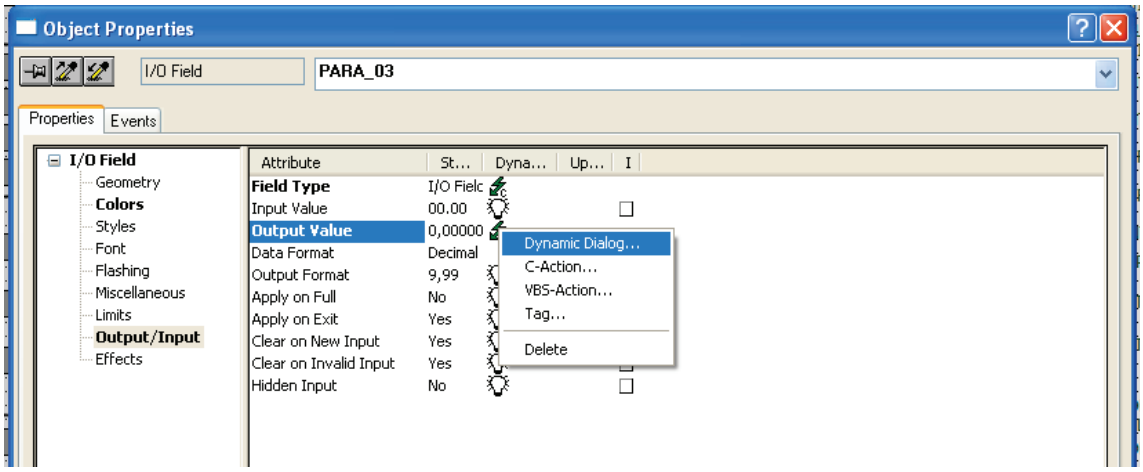
U možnosti s názvem TAG si můžeme nastavit s jakou rychlostí se daná změna provede (viz Obrázek 26). V nabídce máme Upon Change (změna TAGu – rychlost 250ms), 250ms, 500ms, 1s, 2s, 5s atd.. Celá funkcionalita spočívá v tom, že při změně hodnoty TAGu, proběhne změna vlastnosti prvku, ve které máme daný TAG napojen.

Druhou možností napojení je standardní cyklus. Zde si vymyslíme název cyklu, můžeme použít cokoliv za slova. Následně máme stejnou volbu výběru časů jako u TAGu. Ale standardní cyklus se liší tím, že nečeká na žádnou změnu hodnoty, běží neustále. Tedy neustále vyvolává skript, popřípadě jinou akci, na který je napojen.

Způsob nastavení (viz Obrázek 27), kdy a jak TAG změní hodnotu, si můžeme vytvořit pomocí funkce naprogramované v jazyce C, Visual Basicu, popřípadě tou nejjednodušší cestou a tou je přímá akce. U přímé akce pouze číselně nastavíme, při jaké hodnotě proběhne změna. Přímou akci například využívám v simulačním rozhraní pro rozpoznání, v jakém se nacházím menu obrazovky.



Obrázek 26: Rychlost změny provedení



Obrázek 27: Způsob volby změny, kam vložíme trigger

Díky triggeru a jeho správnému nastavení se mi povedlo, aby celá aplikace pracovala tzv. real-time. Každá změna se projeví ihned, po přepsání staré hodnoty se mi okamžitě ukáže nad parametry poslední provedená změna. Tak to funguje stejně s přihlašováním, popřípadě odhlášením, do systému. Aplikace hned zareaguje a upraví svou podobu, aby odpovídala nastavením a požadavkům, které si přeje uživatel.

Nastavení spouštění každé vlastnosti se muselo hodně zkoušet a postupně vychytávat všechny chyby.

4 HODINOVÉ HESLO

Původní záměr byl, aby hodinové heslo bylo opravdu hodinové, tedy aby se na základě mnou vytvořeného algoritmu měnilo každou hodinu. Z toho se ale muselo nakonec upustit, protože se zjistilo, že časové pásmo by nám dělalo velké problémy a mohlo by se nakonec stát, že do aplikace by se už zákazník nemusel dostat, jedině kdybychom mu ji celou odblokovali. To jsem samozřejmě nechtěl. Musel jsem tedy navrhnout nové řešení. To však muselo splňovat pár podmínek:

- Nesmí být uložené v programu WinCC. Tam je totiž lehce napadnutelné a hlavně změnitelné.
- Nebude už na základě algoritmu, ale bude konstantní.

To vedlo k tomu, abych dané heslo uložil do aplikace Step 7 (viz Obrázek 28), která je uložena v programovatelném automatu. Tam bude uzamčena v bloku chráněným heslem. To by mělo zabránit tomu, aby se s heslem jakkoliv manipulovalo. Heslo tedy bude konstantní, nicméně přívlastek hodinové už mu zůstal.

Na daný blok bude napojen ve WinCC TAG. To nám dovolí zprostředkování přístupu mezi WinCC a PLC. Teď bylo potřeba doladit, jak to celé bude probíhat v aplikaci WinCC.

Hodinové heslo je taková nadstavba zabezpečovacího systému, který již má firma Siemens vytvořený. Ale je součástí mé aplikace. Muselo tedy umět spolupracovat s jejich systémem. To se provedlo tak, že jsem doprogramoval do mého systému zabezpečení jednoduchou podmínku, která dovolovala se odkazovat na stejná práva uživatelů. Tím se vyřešil první problém.

Dalším novým požadavkem bylo, aby heslo platilo jen na dané obrazovce, kde se parametry nacházejí. Tím se myslí to, že když se klikne na jinou obrazovku aplikace, automaticky dojde k zablokování parametrů. K tomu stačila podmínka, zda se nacházíme na stránce s parametry. V mé aplikaci jsem danou podmínku zablokoval, aby se nemuselo neustále zadávat heslo. Na ostrém projektu danou podmínku zpřístupním.

Dostáváme se k tomu, že hodinové heslo dovoluje odblokování parametrů ovlivňujících chod parní turbíny. Ale je potřeba ho někam zadat. Za tímto účelem byl vytvořen faceplate (viz Obrázek 29) se vstupním textovým polem, kam zadané heslo můžeme zadat. Než si vysvětlíme funkčnost daného faceplatu, je třeba zmínit další požadavek na hodinové heslo, který je třeba splnit. Daný faceplate se mohl zobrazit pouze uživateli s právem *Value*. Tohle nám vyřešil už dřívější požadavek na heslo a tím mám na mysli ten, který požadoval napojení na původní zabezpečovací systém. Teď stačilo danou podmínku připojit k zobrazení faceplatu. Pro upřesnění: o celou funkčnost se stará funkce *Controller (lpszPictureName,lpszObjectName)*, ve které jsou naprogramovány všechny uvedené požadavky.

Nyní se vraťme k funkčnosti samotného faceplatu. Obsahuje vstupní textové pole, do kterého lze zadat heslo. Heslo bude ve stylu PASSWORD. To znamená, že když ho uživatel bude zadávat, uvidí pouze zástupné znaky (hvězdičky) a ne skutečné znaky.

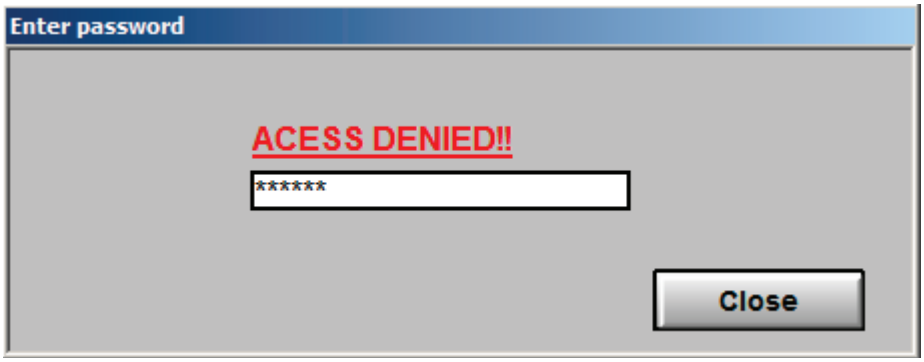
Pokud zadané heslo bude správně, faceplate zmizí, parametry se odblokují. Pokud bude zadané heslo špatně, bude na to uživatel upozorněn hláškou. Uživatel bude mít možnost zadat heslo znova, popřípadě může faceplate zavřít, nezadávat heslo,ale pouze si prohlédnout momentální nastavení parametrů.

Heslo je v celočíselném tvaru. V budoucnosti se počítá, že bude stejné jako heslo k zakázce, které dodává firma Siemens. Zamezí se tím komplikacím a případným záměnám hesel.

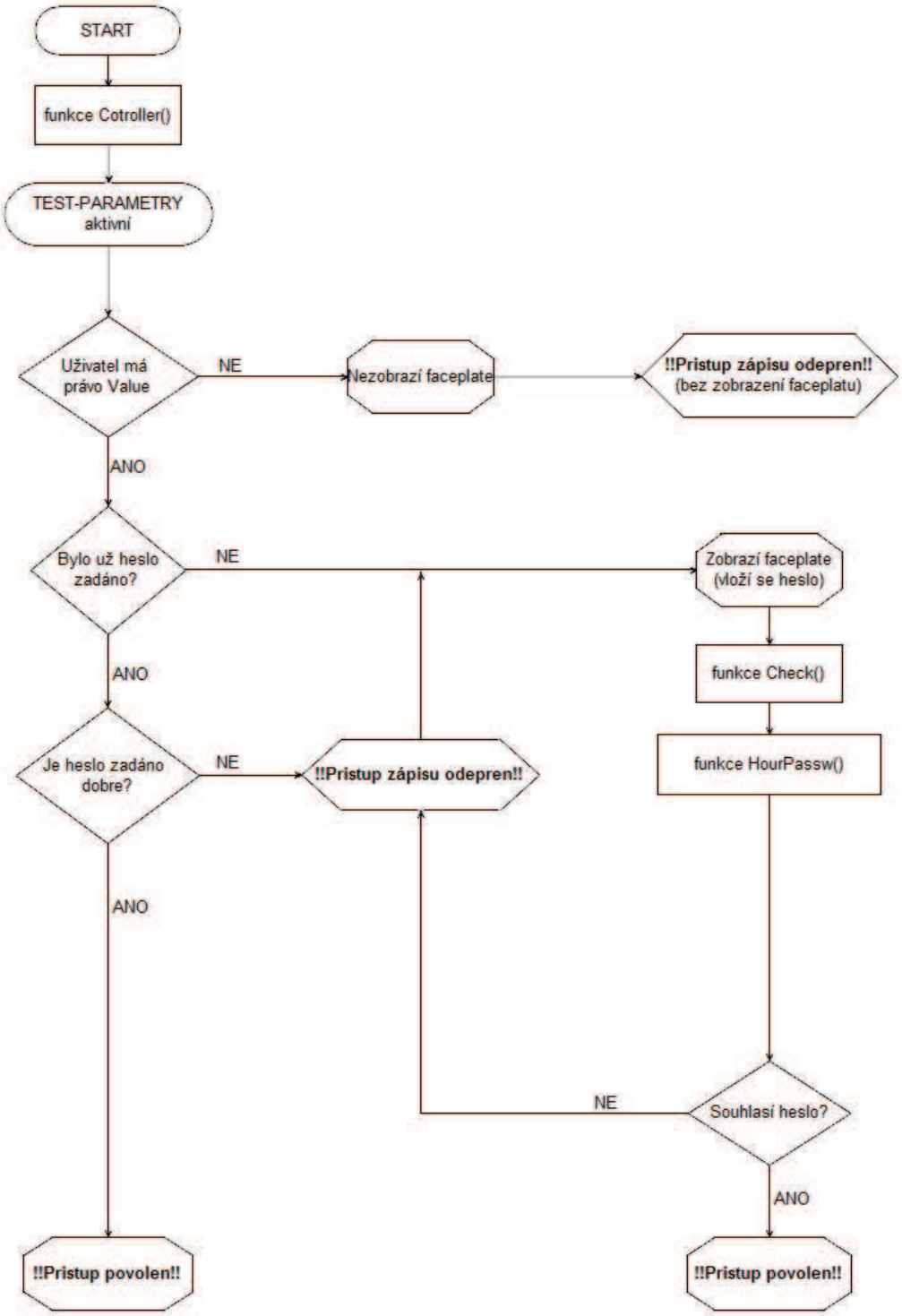
```
Network 1: Title:
Comment:

      L     100
      L     1000
      *I
      T     "DB_10".PASS                DB10.DBD312        -- PASSWORD
```

Obrázek 28: Ukázka kódu hesla ve STEP7



Obrázek 29: Faceplate pro zadání hodinového hesla



Obrázek 30: Vývojový diagram funkce Controller()

Vysvětlení pár pojmů k vývojovému diagramu:

- TEST-PARAMETRY – obrazovka, na které se nacházejí parametry

- Funkce Check() – funkce upozorní systém, že heslo bylo zadáno a aby faceplate znova nezobrazoval
- Právo Value – nastaveno u každého parametru a zároveň možnost nastavit u uživatele, zda tuto podmínku splňuje či ne. Ukazatel upozorňující na to, aby se faceplate zobrazil při splnění podmínky.

Hodinové heslo je naprogramováno, zabezpečení aplikace splněno. Na danou funkci byl kladen obzvlášť velký důraz. Zabezpečení každé aplikace patří k velice choulostivým věcem. Musí být uděláno správně a fungovat spolehlivě. Jako bonus je u mého zabezpečení automatické odhlášení po zhruba 10 minutách nečinnosti.

5 ZOBRAZENÍ NÁPOVĚDY

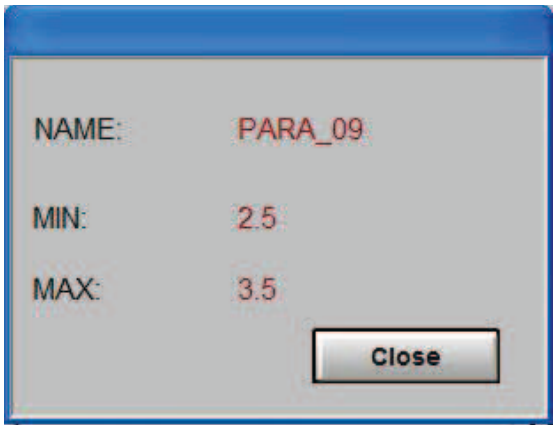
Aby uživatel věděl, co vlastně zadává do parametrů, bylo třeba vytvořit nápovědu. Ta se skládá z faceplatu, který se zobrazí po kliknutí pravého tlačítka myši na parametr. Nápověda bude obsahovat název parametru, MIN a MAX hodnoty. To je hlavně důležité z pohledu ochrany turbíny. Uživatel nesmí mít možnost zadat nesmyslné hodnoty. Mohly by totiž vést k odstavení, popřípadě k alarmovým hlášením na parní turbíně.

U parametrů jsou v souvislosti s nápovědou nastavené tři věci:

- Při jakémkoliv kliku tlačítka myši se načtou do parametru limity
- Držení pravého tlačítka myši odešle do TAGu *HelpTag* název parametru
- Zmáčknutí pravého tlačítka myši zavolá funkci *Show_FP()*

Po otevření faceplatu s nápovědou se načte funkce *HelpFP(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)*. Funkce funguje na základě jména parametru (*lpszObjectName*), u kterého chceme zobrazit nápovědu. Ten předá funkci svoje jméno a ona na jeho základě vyhledá v textovém poli help[][5] všechny požadované hodnoty: MIN, MAX. Následně je načte do limitů samotného parametru, nejenom do faceplatu nápovědy, kde se graficky zobrazí. Daný problém s limity se mohl také vyřešit tak, že by se u parametru dané rozsahy hodnot napojily na dva TAGy, ale to by bylo velice neefektivní. Parametrů je okolo 100. Muselo by se tedy vytvořit dalších 200 TAGů. Ty by zabíraly zbytečné místo navíc. Naše textové pole bude generováno z MS Excel, tedy jeho tvorba a i implementace bude velice rychlá. Mimochodem, WinCC má omezení na 2000 TAGů.

Obrázek 31:Parametr v aplikaci



Obrázek 32: Nápověda k parametru

Samozřejmostí je, že se nápověda zobrazí jen uživateli s potřebnými právy. Jinak zůstane skrytá a nepůjde nijak vyvolat. Další speciální vlastností nápovědy je umístění zobrazení nápovědy vůči parametru, na který jsme klikli. Použil jsem na zobrazení

nápovědy speciální funkci (byla již vytvořená programátorem panem Konečným) *Show_FP(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName, char* PictWin, char* FacePlate)*, kterou jsem si upravil pro své potřeby. Tato funkce vždy vezme rozlišení stránky, umístění parametru na stránce a podle toho zobrazí faceplate nápovědy tak, aby byl vždy viděn celý a co nejblíže parametru, ke kterému patří. Tím se zamezilo, aby se nápověda zobrazovala pouze na jednom místě pro všechny parametry, naopak se vždy zobrazí u parametru a nikdy se nezobrazí mimo obrazovku.

Zde vám část funkce Show_FP ukážu. Napřed ale vysvětlím, co proměnné v ní obsažené znamenají:

- objectLeft – odsazení objektu zleva.
- objectWidth - šířka objektu.
- FP_Width – šířka naší nápovědy.
- screenWidth – šířka obrazovky na které máme spuštěnou aplikaci.
- screenHeight - výška obrazovky na které máme spuštěnou aplikaci
- FP_PositionX – vypočítaná pozice X – souřadnice pro zobrazení nápovědy
- FP_PositionY - vypočítaná pozice Y – souřadnice pro zobrazení nápovědy
- objectTop - odsazení objektu ze shora.
- mainTop – automatické odsazení nápovědy ze shora. Nastaveno na 50pixelů.
- addTopFP – odsazení nápovědy od objektu ze shora. Nastaveno na 8px.
- FP_Height - výška naší nápovědy.

Ukázka kódu funkce Show_FP():

```
void show_FP(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName, char* PictWin, char*FacePlate)
{
#include "apdefap.h" // Vložíme knihovnu
#define MainPict "Process3.pdl"  // Definujeme procesní okno
if ((objectLeft + objectWidth + FP_Width) <= screenWidth) {
                FP_PositionX = objectLeft + objectWidth;
        } else {
                FP_PositionX= objectLeft - FP_Width;
        }
        SetLeft(MainPict, PictWin, FP_PositionX);
if ((objectTop  + mainTop +addTopFP + FP_Height) <= screenHeight) {
                FP_PositionY = objectTop + mainTop +addTopFP;
        } else {
                FP_PositionY = objectTop  - FP_Height +addTopFP+ mainTop;
        }
```

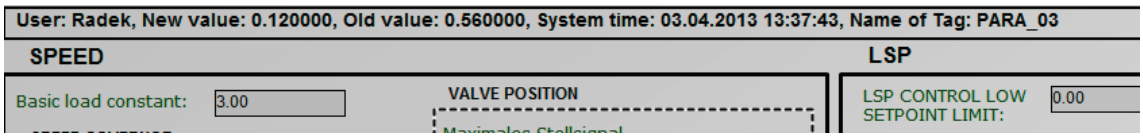
6 ZÁZNAM ZMĚN

Nejdůležitějším požadavkem byl zápis jakékoliv změny v aplikaci. Daný systém záznam změn musel být nezávislý na systému záznamu změn, který již byl vytvořen a používán v dosavadní aplikaci firmy Siemens. Ze záznamu nám muselo být jasné, kdo danou změnu provedl, datum a čas změny, stará hodnota, nová hodnota, název parametru.

Promyslel jsem návrh a zjistil jsem, že by bylo vhodné každou změnu zapisovat do textového souboru a následně pak načítat potřebné údaje zpět na obrazovku. Navrhl jsem k tomu funkci naprogramovanou v jazyku C, která se nazývá *LoginNew(char* lpszPictureName, char* lpszObjectName)*. Využívá dvou důležitých parametrů – názvu obrázku (lpszPictureName) a názvu objektu (*lpszObjectName*). Na základě těchto dvou parametrů si zjistí přesně, o jaký parametr se jedná a dohledá si zbývající údaje: Starou hodnotu, nově vloženou hodnotu, název parametru, který měníme. Ještě si navíc zjistí jméno uživatele, co danou změnu provedl.

Jazyk C se pro daný návrh velice hodil, protože obsahuje přímo funkce pro zapisování a načítání ze souboru. Samotný zápis do souboru se používá ve funkci *ZapisHodnotu(char uzivatel[30], double oldValue,double newValue, char paramName[10], char functionName[10])*, funkce *LoginNew* ji obsahuje v sobě a po provedení kontroly, zda uživatel má právo provést změnu a zda heslo bylo zadáno správně, zavolá se teprve funkce *ZapisHodnotu*. Funkci se předají argumenty: jméno uživatele (*uzivatel[30]*), stará hodnota (*oldValue*), nová hodnota (*newValue*), název parametru (*paramName[10]*). Případně jsem tam ještě připravil místo na poslední údaj a tím je funkční skupina (*functionName[10]*). Tento poslední údaj se zatím nevyužívá. Je pouze připravený na pozdější využití.

Když uživatel zadal do parametru novou hodnotu, provedl se zápis do souboru. Tam se řádek po řádku zapisovaly jednotlivé informace. Pak přišel požadavek, aby se poslední provedená změna ukazovala nad administrací parametrů. K tomu jsem si naprogramoval funkci *LoadLast()*. Funkce projede celý seznam, a poslední řádek načte do textového pole aplikace (Obrázek 33).



Obrázek 33: Poslední provedená změna

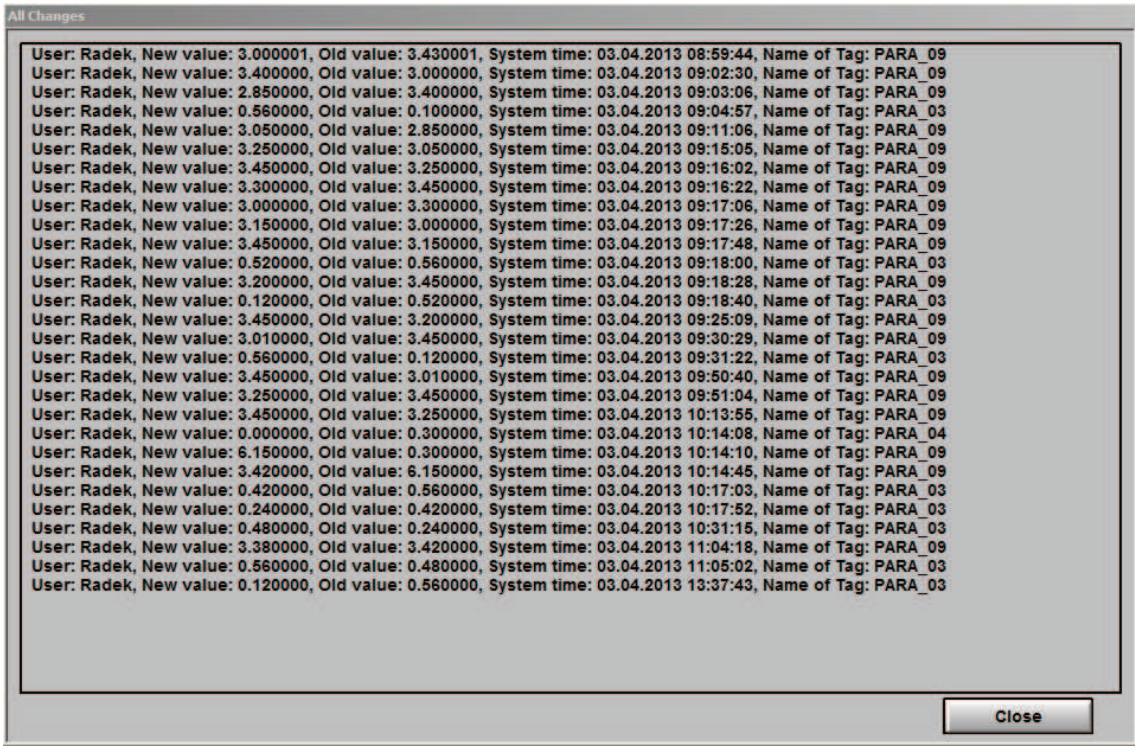
Uživatel hned po zadání hodnoty vidí, co zadal, případně se může ujistit, jaká poslední změna byla provedena.

Dále bylo vytvořeno tlačítko na zobrazení všech hodnot (viz Obrázek 34). Po kliknutí na něj vyjede okno (viz Obrázek 35), kde budou zobrazeny všechny provedené

změny. Jelikož se jedná o citlivé parametry tak se nepředpokládá, že by změn bylo hodně.

Na tlačítko je napojená funkce *LoadAll()*. Funkce otevře soubor, do kterého se zapisují změny. Řádek po řádku ho načte do textového pole od nejnovějšího po nejstarší. Momentálně je funkce nastavená na vypsání 25 nejnovějších zápisů.

Obrázek 34: Tlačítko k zobrazení všech změn



Obrázek 35: Okno zobrazující všechny změny

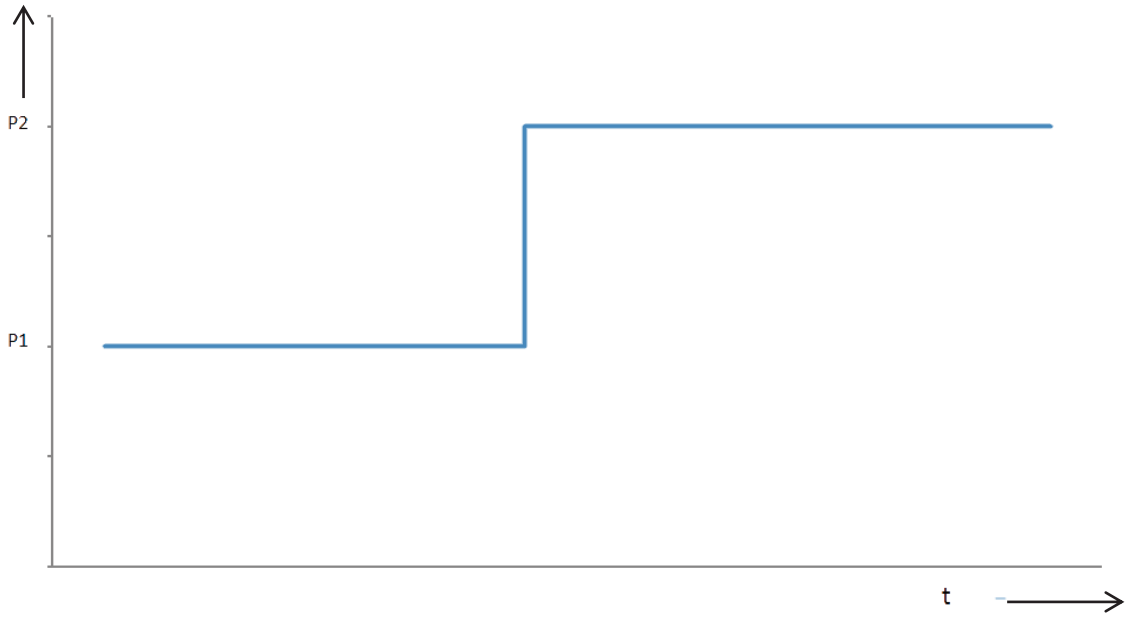
7 BEZNÁRAZOVÝ PŘECHOD

Tak jako u většiny spotřebičů, tak i zde bylo potřeba, aby se každá změna zapsala postupně a ne skokově (viz Obrázek 36). Skoková změna by mohla způsobit momentální přetížení spotřebiče, který by se nedokázal s větší změnou vyrovnat a mohl by přestat fungovat. Něco podobného by se mohlo stát i na samotné turbíně. S tím rozdílem, že by nám daná skoková změna s největší pravděpodobností odstavila celou turbínu. Proto jsem napsal funkci, která řeší daný problém. Řešení nám zajistí beznárazový přechod (viz Obrázek 37).

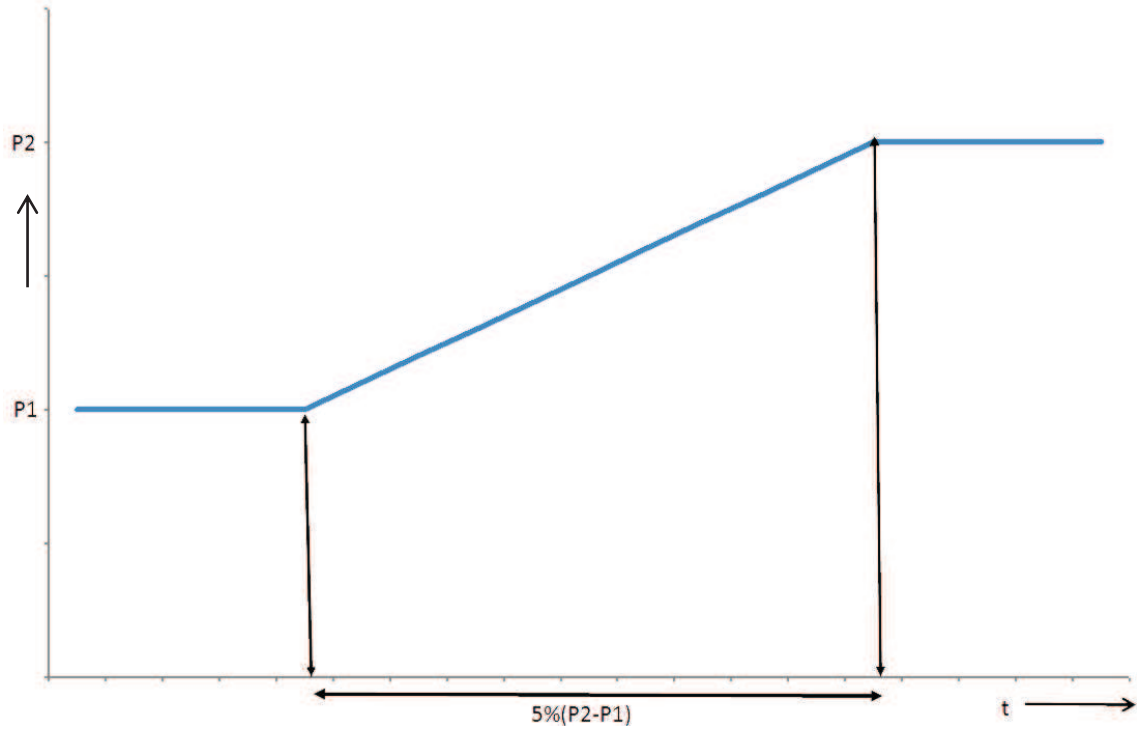
Jak funguje? Na základě jednoduché rovnice (viz Rovnice 1) bude po jedné vteřině zvyšovat, popřípadě snižovat, hodnotu z té původní až po deseti vteřinách dojde k hodnotě žádané. Byla k tomu speciálně naprogramována funkce *Prechod(char* lpszPictureName, char* lpszObjectName)*. Jak funkce funguje, si nejlépe ukážeme na vývojovém diagramu (viz Obrázek 38).

Daná funkce nám zajistí, že změna, kterou chce uživatel provést, bude provedena v pořádku a postupně, aby nedošlo k přetížení systému.

Vysvětlení k obrázkům:
P2 – žádaná hodnota
P1 – původní hodnota
t - čas



Obrázek 36: Skoková změna parametru



Obrázek 37: Beznárazový přechod parametru

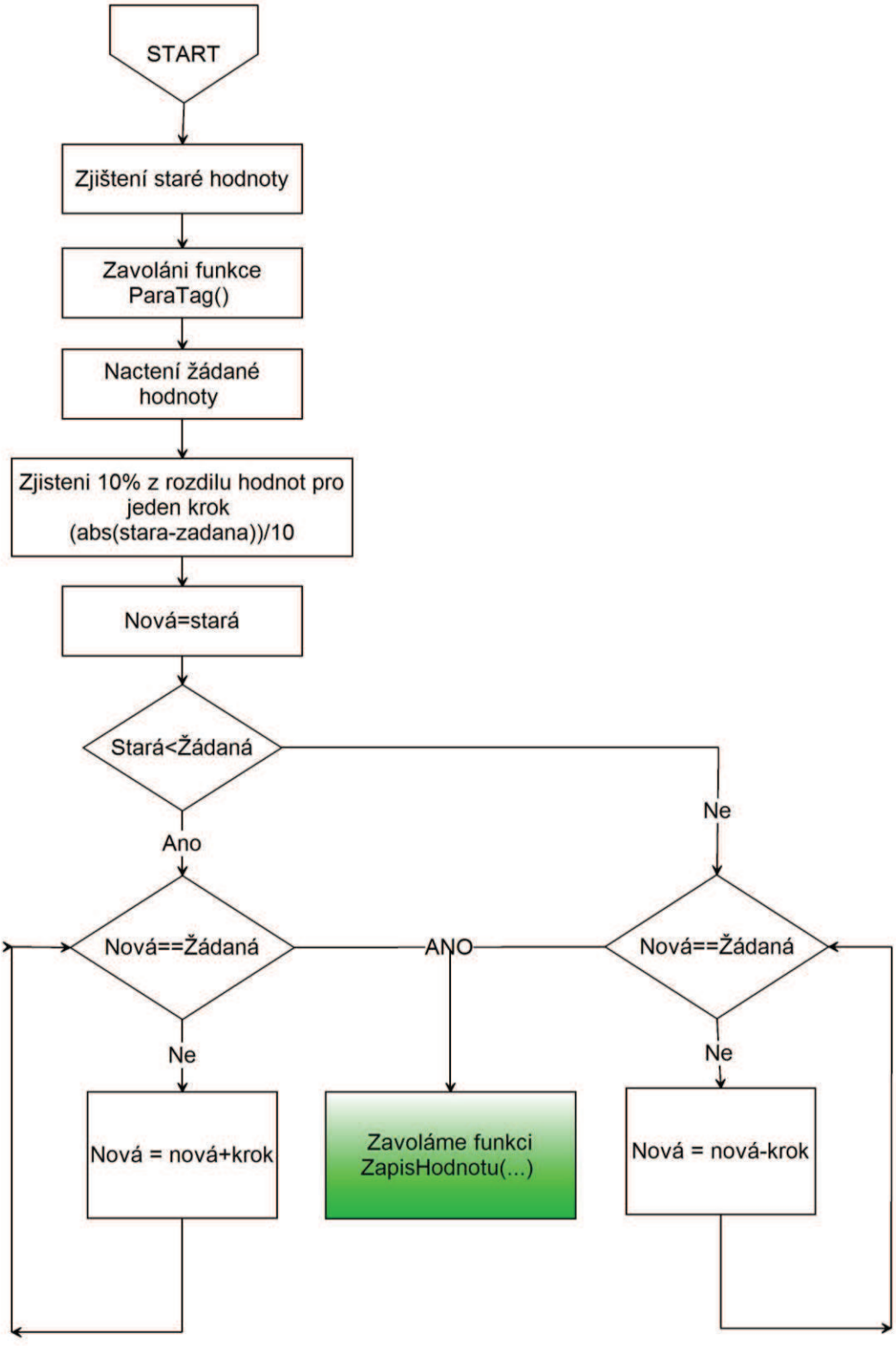
5% (P2-P1) – Případná velikost časového úseku by v budoucnosti mohla mít velikost 5% z rozdílu změn. Zatím je to pouze ve fázi jednání, zda by dané řešení mohlo opravdu být nasazeno.

Rovnice pro beznárazový přechod:

$$Krok = \frac{|stará - žádaná|}{10}$$

Rovnice 1:Výpočet jednoho kroku

Rovnice nám spočítá na základě nově vložené hodnoty krok, podle kterého se bude beznárazový přechod provádět. Nyní je pevně nastaveno, že hodnota se bude zvětšovat/zmenšovat o 10% z rozdílu hodnot po jedné vteřině.



Obrázek 38: Vývojový diagram funkce Prechod()

8 CO BUDE DÁL

Daná aplikace, kterou vypracovávám jako svou bakalářskou práci, bude v budoucnosti nasazena na ostrém projektu. Momentálně je dotažena do fáze, kdy funguje stabilní předávání parametrů, je chráněna systémem zabezpečení, má svou nápovědu.

Nicméně, pro ostré nasazení je ještě třeba vytvořit následující funkce:

- **Efektivita přenosu dat** – na stávajících programovatelných automatech dodávané firmou Siemens funguje připojení ProfiBus. Toto připojení má omezenou propustnost dat 244 bajtů. To by jejich a zároveň mou aplikaci nezvládlo. Proto je ještě třeba softwarově doladit danou funkcionalitu. V budoucnosti se plánuje nasazení spojení pomocí ProfiNet. Tam by daný problém zcela odpadl.
- **Záloha parametrů** – při změně kteréhokoliv parametru se provede záloha všech hodnot.
- **Obnova posledních dat** – může se stát, že dojde k restartu PLC. Potom se načtou do aplikace defaultně nastavené hodnoty. Ale ty už budou nežádoucí. Proto musí existovat v aplikaci možnost vrátit poslední uložené změny.
- **Dvojí úroveň parametrů** – parametrů zobrazujících se pro operátora bude méně, než parametrů zobrazujících se pro sytémového inženýra.
- **Grafická stylizace** – Než se aplikace nasadí, proběhne sjednocení velikosti písma, šířek I/O polí. Použití typových barev bude také stejné, jak v aplikaci firmy Siemens. Uživatel nesmí mít pocit, že najel do úplně jiné části aplikace.
- **Automatické generování celé nadstavby aplikace** – vrchol celé tvorby této práce. Naprogramovat skript ve Visual Basicu, který nám dovolí celou aplikaci zautomatizovat a vytvořit na jedno kliknutí tlačítka. Informace pro vytvoření parametrů by se načítaly z tabulky v MS EXCEL. Ta by obsahovala všechny potřebné údaje. Například jméno parametru, rozsahy hodnot, počet desetinných míst a podobně. Na daném skriptu se začne pracovat co nejdříve.

Úkoly jsou tedy zadány, dále ještě proběhne test v simulačním prostředí aplikace firmy Siemens. Ostré nasazení na zakázce by tedy mohlo být ke konci roku 2013.

ZÁVĚR

Vypracováním této bakalářské práce jsem se naučil pokročilejším základům programování v Simatic STEP 7, vizualizačnímu programu WinCC a jejich celkovému propojení s PLC. Pochopil jsem, jak simulovat celý projekt bez PLC, popřípadě s ním. Považuji za velice zajímavé, že jsem měl možnost nahlédnout mezi nejmodernější postupy návrhu ovládání řídicího systému parní turbíny.

Úkolem bakalářské práce byla praktická realizace parametrizace řídicího systému, tj. aby šly měnit vybrané parametry parní turbíny přímo z vizualizace přes operátorskou stanici, a ne přes programátora pomocí STEP 7. Po důkladném návrhu jsem začal s realizací. Po čase se ale zjistilo, že můj návrh, ač je spolehlivý, nedostačuje. Bylo nutné provést změny. Jedna z prvních byla, že na změnu jakékoliv hodnoty na turbíně je třeba naprogramovat lepší nástroj k zápisu hodnoty do PLC, než bylo původně zamýšleno. Proto mi byl od vedoucího podán návrh, zda bych dokázal naprogramovat beznárazový přechod. To znamenalo předělat celou aplikaci, protože už jsem každý parametr nemohl napojit přímo na jeho TAG. Musel jsem předávat hodnoty postupně, za pomoci jiného TAGu. Problém se samozřejmě povedlo vyřešit a hodnoty se zapisují do PLC postupně.

Další změnou, oproti původnímu návrhu, bylo hodinové heslo. Místo toho, aby se původně měnilo každou hodinu, je konstantní a zamčené ve funkčním bloku. Následně jsem vhodně vyřešil návrh vizualizace pro předávání hodnot z WinCC do Simatic STEP 7, aby můj návrh co nejméně zatížil už momentální řešení firmy Siemens pro řídicí systém parní turbíny.

U daného systému, který jsem vytvořil, byla velice důležitá ta část, v níž jsem celou aplikaci testoval. Musel jsem vyvolávat do aplikace chyby a zkoušet, jak se program zachová. Samotná funkčnost nestačila. Byla požadována i spolehlivost při nenadálé poruše. Proto taky docházelo k tomu, že jsem musel celou aplikaci dvakrát předělávat.

Na závěr bych ještě rád napsal, že mi bylo potěšením spolupracovat s týmem programátorů firmy Siemens a mít možnost se od nich učit. Uviděl jsem, jak programují profesionálové, jak probíhá návrh algoritmizace kódu do PLC a WinCC. Mnohem důležitější pro mě však bylo, že mi bylo ukázáno, jak se potom hledají různé chyby, testují a programově upravují.

Literatura

[1] SIEMENS INDUSTRIAL TURBOMACHINERY S.R.O. *Historie a současnost Parní turbíny v Brně*. 3. rozšířené a doplněné vydání. Brno: Trilabit, s. r. o., 2010. ISBN 978-80-902681-3-5

[2] ISA-5.5-1985. *Graphic Symbols for Process Displays*. North Carolina: Instrument Society of America, 1985.

[3] ZBYNĚK IBLER A KOL. *TECHNICKÝ PRŮVODCE ENERGETIKA*. Praha: BEN - technická literatura, 2002. ISBN 80-7300-026-1.

[4] SIEMENS INDUSTRIAL TURBOMACHINERY S.R.O. *Control & Instrumentation Overview*. 2005.

[5] SIEMENS INDUSTRIAL TURBOMACHINERY S.R.O. *Introduction and product portfolio*. 2005.

[6] ČSN EN 60045-1. *PARNÍ TURBÍNY: Část 1: Specifikace*. Praha: Gill, 1995.

Seznam použitých obrázků

Seznam příloh