VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

MPC-AUP

Návody do laboratorních cvičení

Garant předmětu:
doc. Ing. Václav Kaczmarczyk, Ph.D.

Autor textu:
doc. Ing. Václav Kaczmarczyk, Ph.D.
doc. Ing. Jakub Arm, Ph.D.
Ing. František Rusnák

Vytvořeno za podpory RP182302002, PPSŘ 2024

Brno 1. 9. 2025

Obsah

Úvod

Hlavním cílem těchto učebních textů je poskytnout podporu pro zpracování úloh v laboratorním cvičení předmětu Automatizace procesů.

Konkrétním cílem cvičení je zpracovat komplexní projekt na automatizaci technologického procesu, který bude splňovat požadavky a terminologii standardu ISA-S88.

Vzhledem k tomu, že dosud patrně nemáte předchozí zkušenosti s používanými nástroji nebo standardy, jsou v rámci tohoto návodu zpracovány i teoretické informace. Doporučujeme tyto části automaticky nepřeskakovat.

Právě čtete verzi dokumentu 2025-08.

1. Zařazení předmětu ve studijním programu

Předmět MPC-AUP je určen pro studenty posledního ročníku magisterského studia, tudíž se předpokládají výchozí znalosti o programování programovatelných automatů.

1.1. Předpokládané znalosti

Cvičení budou probíhat na pracovištích vybavených průmyslovými počítači BECKHOFF řady 9xxx. Základy práce s těmito automaty byly náplní kurzu BPC-PPA.

V případě, že jste ani jeden z těchto kurzů neabsolvovali, případně pokud máte pocit, že Vaše znalosti již nejsou nejčerstvější, doporučujeme přečíst si a mít během cvičení neustále k dispozici publikaci „Laboratorní cvičení BPPA“ kolektivu autorů (Štohl, Jirgl, Arm, Mišík).

1.2. Cíle laboratorních cvičení

- Technologický návrh zařízení
- Zpracování technické dokumentace
- Výběr komponent pro sestavení technologie
- Zpracování elektrotechnické dokumentace
- Vyšší programování PLC s důrazem na modularitu a standardizaci
- Programování SCADA/HMI
- Identifikace soustavy a návrh PID regulátoru
- Práce se systémem MES
- Zpracování projektové dokumentace

1.3. Hodnocení laboratorních cvičení

- Cvičení - max. 30 bodů
- Podmínka pro zápočet min. 15 bodů
- Hodnoceny budou jednotlivé úlohy

2. Systémy BECKHOFF

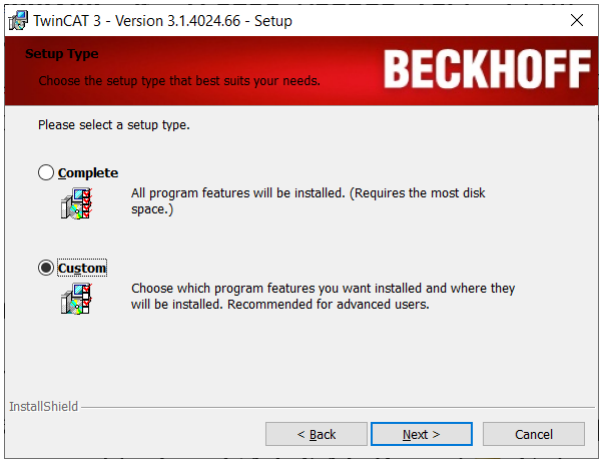
Následující kapitola popisuje možnosti, které se nabízejí pro programování průmyslových PC řady Beckhoff.

Pro programování a parametrizaci produktů firmy Beckhoff vyvinula společnost komplexní softwarový nástroj **TwinCAT** (The Windows Control and Automation Technology). V současné době je aktuální verze TwinCAT3 (TC3). Pomocí TC3 je tedy možné vytvořit program pro PLC, NC, CNC a robotické systémy. TC3 podporuje základní průmyslové komunikace (např. Modbus TCP, OPC UA) a také komunikační rozhraní počítače (síť, USB, PCI), na kterém je hostováno. Z hlediska programování PLC je podporován přístup dle standardu IEC 61131-3.

TC3 je postaven na prostředí Visual Studio, přičemž využívá jeho postupy pro založení, tvorbu a kompilaci vytvářeného projektu. V tomto prostředí implementuje strukturu projektu platnou pro všechny cílové platformy a také zpřístupňuje vlastní nastavení. Druhý způsob je integrace TC3 do stávající instalace Visual Studia, přičemž je možné integrovat i programovací jazyky C/C++, C#, VB.NET a dokonce i Matlab/Simulink.

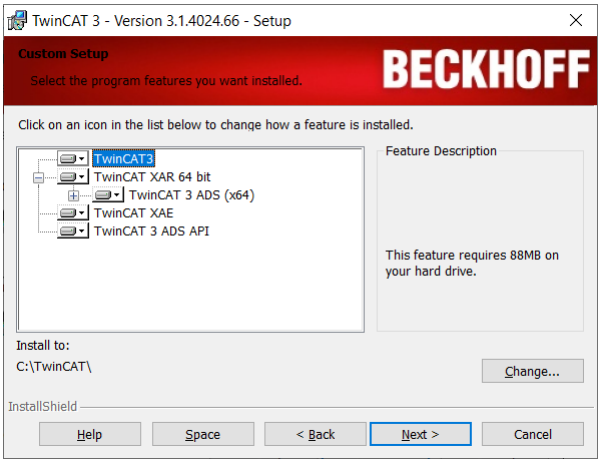
2.1. Instalace aplikace TwinCAT 3

Pro instalaci je nutné získat software označený TwinCAT 3 download | eXtended Automation Engineering (XAE) a TF6250 | TwinCAT 3 Modbus TCP ze stránek Beckhoff (viz odkaz v referencích). Po registraci a přihlášení lze nástroje bezplatně stáhnout – soubory **`TC31-FULL-Setup.x.x.xxxx.xx`** a **`TF6250-Modbus-TCP`**. První soubor obsahuje instalátor komplexního software obsahujícího nástroje XAE, XAR, sběrnice ADS a PLC. Druhý soubor obsahuje instalátor s doplňkem Modbus TCP Server. Po spuštění prvního instalátoru potvrdíme elevaci privilegií instalátoru a spustíme konfigurační část instalátoru. Zvolíme možnost vlastního nastavení:



Obrázek 1: Volba vlastního nastavení instalace

Poté vybereme všechny součásti k nainstalování:

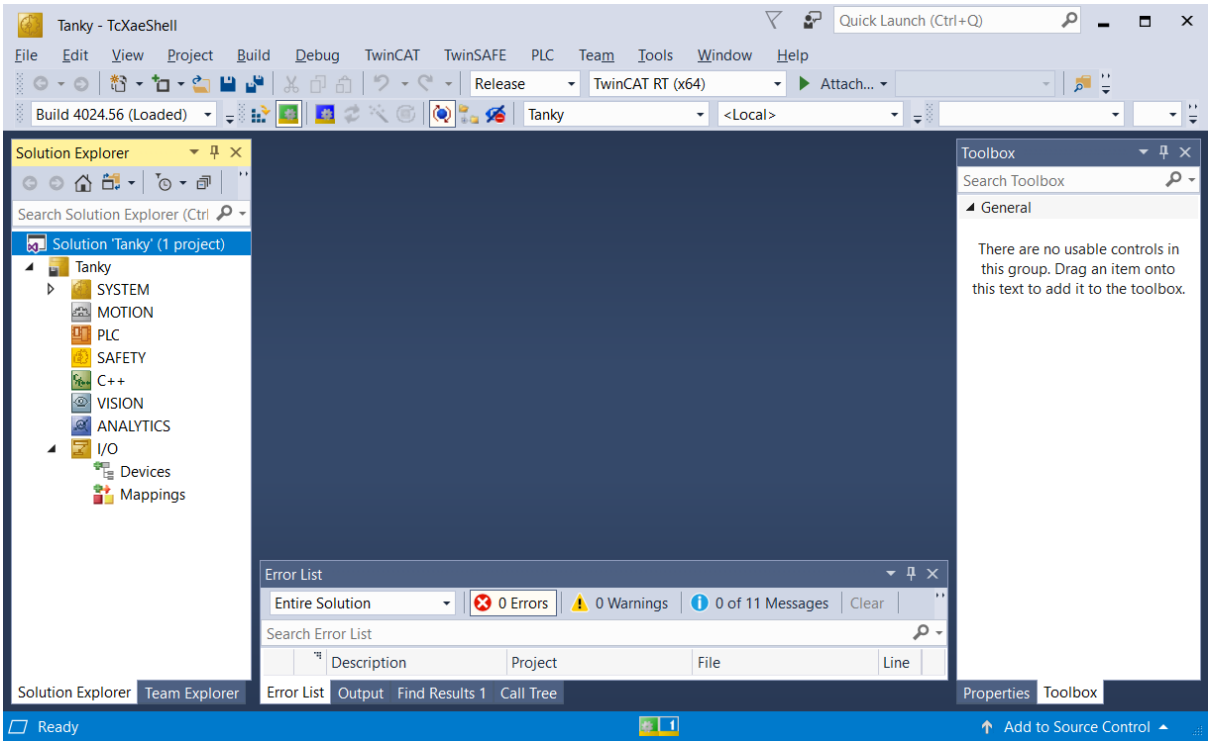


Obrázek 2: Volba součástí k instalování

Potvrzením konfigurace instalačního nástroje se softwarový nástroj začne instalovat. Po skončení je možné software XAE Shell spustit.

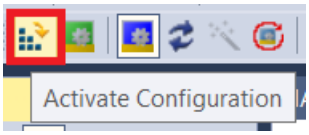
2.2. Spuštění aplikace TwinCAT

1. Spusťte aplikaci TWINCAT pomocí zástupce na ploše.
2. Založte nový projekt (File - New Project). Projekt umístěte na plochu do podadresáře s vaším loginem. Zvolte typ projektu TwinCAT XAE Project . Vyberte smysluplné jméno.



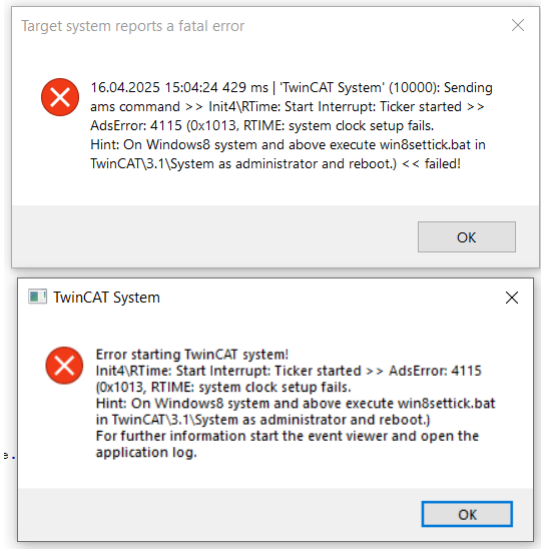
Obrázek 3 Aplikace TcXaeShell (TwinCAT)

Dále je potřeba před prvotním spuštěním runtime prostředí aktivovat konfiguraci runtime prostředí pomocí příkazu Activate Configuration, který potvrdíme pomocí zobrazeného dialogu.



Obrázek 4: Aktivování konfigurace

V tomto okamžiku se může stát, že software zahlásí chybu časovače:



Obrázek 5: Chyba časovače ve Windows

V tom případě je potřeba upravit nastavení časovače Windows pomocí souboru C:\TwinCAT\3.1\System\win8settick.bat. Tento soubor spustíme pomocí příkazové řádky v administrátorském režimu. Po úspěšném provedení je potřeba PC restartovat.

Z důvodu použití Modbus TCP musíme ještě doinstalovat doplněk TC3 Modbus TCP. Instalátor bude vyžadovat přerušení běhu software, a poté nainstaluje příslušnou službu. Před použitím musíme Modbus TCP konfigurovat.

NENÍ doporučeno zkoušet odinstalovat software v případě, že je runtime v RUN režimu. Tento proces může způsobit rozsáhlejší problémy. Vždy je nutné převést (pomocí ikon v toolbar nebo ikony v tray) runtime do režimu config před manipulací s instalací.

TwinCAT je založen na platformě Microsoft Visual Studio. Rozložení oken v aplikaci je možné libovolně přizpůsobovat. Kompletní struktura celého **řešení** je viditelná v okně **Solution Explorer**. Řešení může obsahovat více projektů, než jen jediný, v našem případě bude postačovat jeden projekt). V rámci projektu Tanky budeme potřebovat záložky

3. Systém pro konfiguraci systémových částí programu/hardwaru
4. PLC pro konfiguraci programového vybavení.

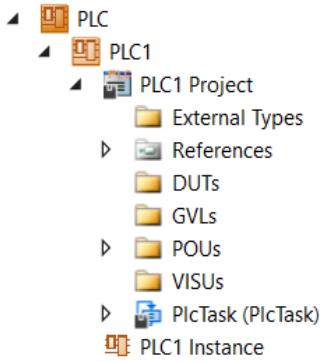
   Vpravo se nachází okna **Properties** a **Toolbox**, které se budou hodit při návrhu programů v jazyce LD, případně pro vývoj vizualizace.

V tuto chvíli není v projektu vytvořeno žádné PLC. Pravým klikem na položku PLC přidejte nový **Standard PLC Project** a pojmenujte jej.

2.3. Programové vybavení

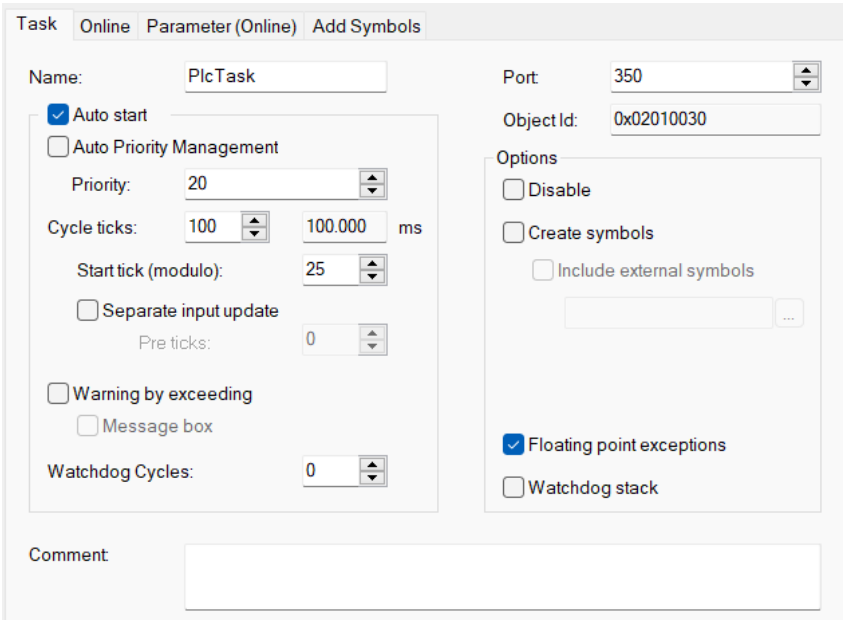
Struktura projektu

Po založení nového PLC projektu se tento zobrazí v příslušné záložce okna **Solution Explorer** tak, jak je zřejmé z následujícího obrázku. Objekt **PLC** v sobě může sdružovat více PLC. PLC1 je tedy jedno z několika možných PLC. **PLC1 Project** pak popisuje PLC aplikaci.



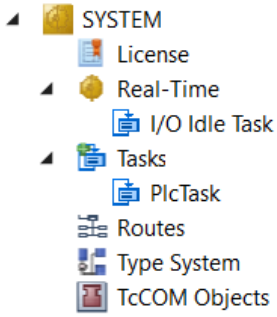
Pro naši práci budou důležité následující položky

- Složka **POUs**, do které budeme vkládat všechny programové kódy, a to jak ve formě programů (**PRG**), které budou přímo spouštěny systémem (o tom dále), tak i ve formě funkcí (**FC**), nebo funkčních bloků (**FB**), které budeme volat sami právě z programových bloků PRG.
- Složka **GVLs**, ve které můžeme vytvořit tzv. **Global Variable List** - seznam globálních proměnných, které můžeme následně používat v našich programových blocích. Globální proměnné by se však měly používat s rozmyslem a pouze pro uchování dat, která mají opravdu globální kontext.
- Složka **DUTs**, kam patří definice uživatelských datových typů. Mezi ty patří **struktury**, které pro vás budou důležité ve chvíli, kdy budete potřebovat efektivně ukládat a mezi funkcemi přenášet různá data. Dále je možné definovat **výčet** (enumeration), který zjednoduší definování několika pojmenovaných konstant, které spolu souvisejí. Mezi DUT patří také **union**, který umožní zpracování jedné datové struktury různými způsoby (jako strukturu, jako pole bajtů, atd…)
- Složka **VISUs**, ve které budete vytvářet vizualizace.
- Složka **References**, které obsahuje seznam referencovaných softwarových knihoven, které jsou důležité pro běh vašeho programu.
- Položka **PlcTask**, tedy definice úlohy, kterou bude PLC provádět. Poklepáním se dostanete na stejnou položku ve struktuře **System** (viz následující obrázek). Zde můžete nastavit parametry spuštění této úlohy v systému, mezi které patří priorita (nebudeme potřebovat nastavit), perioda spouštění (v ms), offset spuštění (start tick - modulo) a další.



Záložka SYSTEM

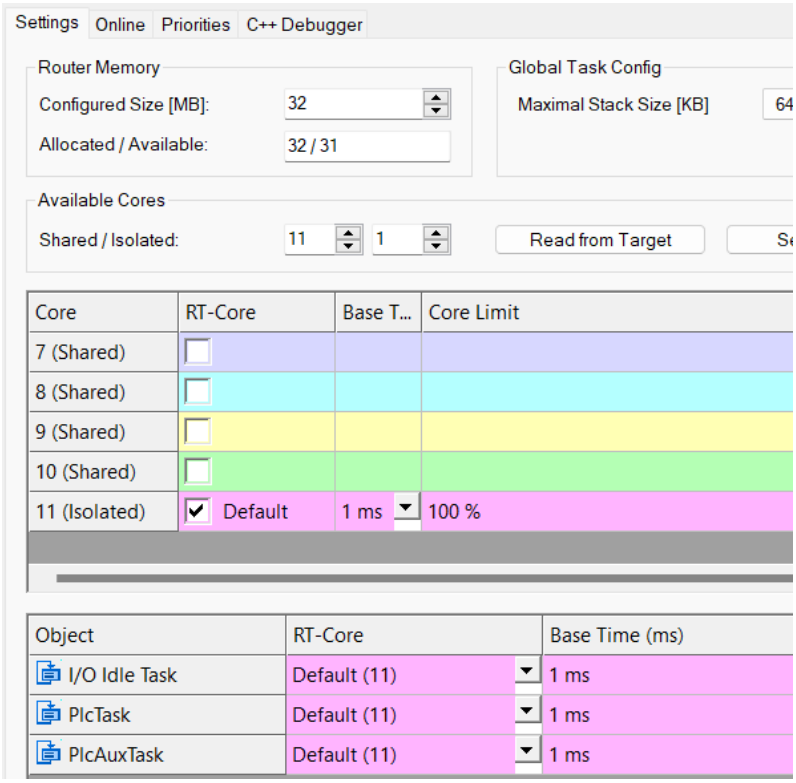
V této záložce se nastavují systémové parametry sloužící pro nastavení systému hostitelské platformy.



Pro nás bude důležitá položka **Licence**, pomocí které budeme schopni přidat do našeho projektu softwarové knihovny a vygenerovat pro ně dočasné licence.

Pomocí položky **Real Time**, můžeme definovat hardwarové prostředky (jádra PLC), které budou alokovány pro běh PLC úloh. Nejprve tlačítkem **Read from target** vyčteme informace propojeného PLC (nebo PC, na kterém běží TwinCat, záleží na nastavení). Pokud budeme využívat čistě softwarové PLC, můžeme jedno jádro CPU počítače izolovat (**isolated**) pouze pro běh SW PLC tak, jak je vidět na následujícím obrázku. V případě, že budeme využívat Beckhoff průmyslové PC, hodí se izolovat jádro pro běh časově kritických úloh (např. řízení pohonů). Záleží však vždy na počtu dostupných jader. V případě jedno či dvoujádrového IPC je vhodné alokovat jádra ve sdíleném (**shared**) režimu, který označuje, že jádro bude využíváno jak pro běh PLC úloh, tak pro běh systému Windows. Po změně počtu izolovaných jader je vždy nutno restartovat PC (IPC).

Doporučuji u PC v laboratoři nepoužívat izolaci jader!



Programujeme první blok

Kliknutím pravým tlačítkem myší na složku POUs a výběrem Add - POU můžete přidat nový programový blok. Zvolíme vhodné jméno a vybereme typ **Function Block** a jazyk ponecháme **Structured Text**.

Naším záměrem je vytvořit funkční blok, který na základě dvou vstupů určí výstup, a to tak, že nástupná hrana prvního vstupu bude zpožděná o 5 sekund. Výstup zpožďovače bude vstupem do logické funkce, která určí výsledek.

```
FUNCTION_BLOCK MyNewFunctionBlock
VAR_INPUT
    Inp1 : BOOL;
    Inp2 : BOOL;
END_VAR
VAR_OUTPUT
    Out : BOOL;
END_VAR
VAR
    Timer : TON;
END_VAR
```

```
Timer(IN := Inp1, PT := T#5S);
Out := Timer.Q AND NOT(Inp2);
```

Vstupní proměnné do funkčního bloku je nutno zadat v sekci **VAR_INPUT**, výstupní pak v sekci **VAR_OUTPUT**. Sekce VAR je pak vyhrazena pro lokální proměnné funkčního bloku. Zde si definujeme instanci funkčního bloku časovač (TON), který budeme používat pro zpoždění vstupního

signálu INP1. V této instanci časovače bude uložen jeho stav, tedy čas, který již uběhl od aktivace. Každý funkční blok, který budeme používat v programu, musíme takovým způsobem instanciovat.

Nyní můžeme vytvořený funkční blok zavolat z programového bloku. V bloku MAIN si definujeme čtyři proměnné - fb - opět budeme potřebovat instanci našeho vytvořeného bloku a vstupní a výstupní proměnné typu BOOL. Blok následně můžeme zavolat tak, jak je uvedeno na obrázku. Povšimněte si, jakým způsobem se zadávají vstupní proměnné, a jak se uloží výstupní proměnná.

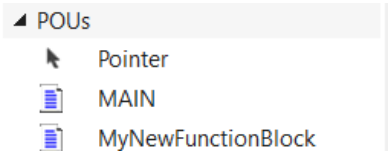
```
PROGRAM MAIN
VAR
    fb : MyNewFunctionBlock;
    in1 : BOOL;
    in2: BOOL;
    out : BOOL;
END_VAR
```

```
fb(Inp1 := in1, Inp2 := in2, Out => out);
```

Toto není jediný způsob. Blok můžeme volat také následovně:

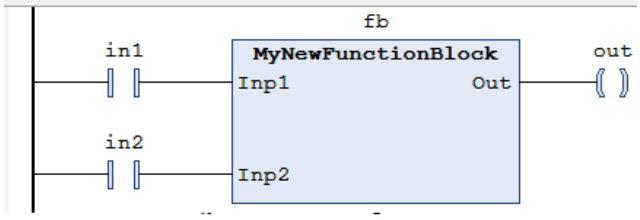
```
fb.Inp1 := in1;
fb.Inp2 := in2;
fb();
out := fb.Out;
```

Je také možné oba uvedené způsoby libovolně kombinovat. Další možností je volat funkční blok z programového bloku vytvořeného v LD diagramu. V tomto případě si v panelu Toolbox najdeme v sekci POUs námi vytvořený funkční blok, a ten přetáhneme do volného LD networku.

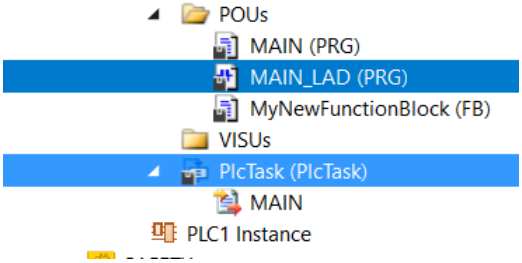


Funkční blok musíme opět “pojmenovat”, čímž se automaticky vytvoří instance daného typu jako lokální proměnná programového bloku. Nyní můžeme připojit vstupy a výstupy tak, jak je vidět na následujícím obrázku.

```
PROGRAM MAIN_LAD
VAR
    fb : MyNewFunctionBlock;
    in1 : BOOL;
    in2: BOOL;
    out : BOOL;
END_VAR
```



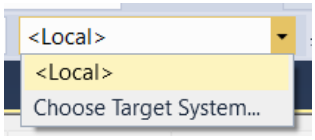
Ve chvíli, kdy jsme vytvořili nový programový blok (PRG) typu LD, je nutno říct systému, aby tento blok volal v rámci spouštění nějaké úlohy. V opačném případě blok nebude spuštěn. Nejjednodušší je v okně **Solution Explorer** přetáhnout daný blok MAIN_LAD na položku PlcTask.



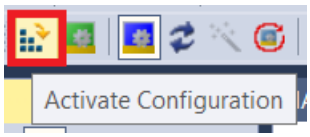
Poznamenejme, že můžete mít vytvořeno více PlcTask objektů, každý s různými parametry volání na úrovni systému.

Nahrání programu do SoftPLC

Jakmile máme program dokončen, zkusíme jej nahrát do PLC a spustit. Nejprve provedeme výběr systému, do kterého se bude nahrávat. Vzhledem k tomu, že chceme pracovat s lokálním softwarovým PLC, vybereme položku **<Local>**.



To provedeme stiskem tlačítka **Aktivovat konfiguraci**.



Po aktivaci by se měl systém přepnout z **konfiguračního režimu** (modrá ikonka aktivní) do režimu **běhu** (zelená ikonka aktivní)**.**

Následně potřebujeme PLC spustit. Toho dosáhneme stiskem ikony **Login**  a následně stiskem ikony **Start**  . Během přihlašování se testuje, zda program v PLC odpovídá aktuálnímu programu ve TwinCAT. Pokud tomu tak není, program nabídne nahrání aplikace (je nutno potvrdit v dialogovém okně).

Ladění programu

Po spuštění programu můžeme provádět jeho ladění. Můžeme si ve všech programových a funkčních blocích (které však alespoň jednou voláme), zobrazit hodnoty všech proměnných v tabulce, stejně tak v kódu (viz obrázek).

Expression	Type	Value	Prepared value	Address	Comment
fb	MyNewFunction...				
Inp1	BOOL	FALSE			
Inp2	BOOL	FALSE			
Out	BOOL	FALSE			
Timer	TON				
IN	BOOL	FALSE			starts timer with r...g edge, resets t...
PT	TIME	T#5s			time to pass, before Q is set
Q	BOOL	FALSE			gets TRUE, delay ... (PT) after a ri...
ET	TIME	T#0ms			elapsed time
M	BOOL	FALSE			
StartTime	TIME	T#0ms			
in1	BOOL	FALSE			
in2	BOOL	FALSE			
out	BOOL	FALSE			

```
1  fb(Inp1FALSE := in1FALSE, Inp2FALSE := in2FALSE, OutFALSE => outFALSE);
2
3
4  fb.Inp1FALSE := in1FALSE;
5  fb.Inp2FALSE := in2FALSE;
6  fb();
7  outFALSE := fb.OutFALSE;
```

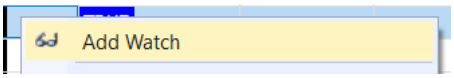
Zapamatujte si, že pokud nějaký funkční blok budete volat na více místech programu s různými instancemi datových bloků, budete při ladění vždy vyzvání k tomu, abyste si vybrali, kterou instanci chcete ladit.

Kliknutím do řádku ve sloupci **Prepared value** můžeme upravit hodnotu příslušné proměnné. Změna se však provede až po kliknutí na ikonu **Write values**  .

Expression	Type	Value	Prepared value	Address	Comment
fb	MyNewFunction...				
in1	BOOL	FALSE	TRUE		
in2	BOOL	FALSE			
out	BOOL	FALSE			

Takto si můžeme vyzkoušet, jestli náš funkční blok funguje podle předpokladů.

Pro ladění můžeme využít také **Watch** okno, do kterého můžeme vložit libovolnou proměnnou – kliknutím pravým tlačítkem myší a výběrem **Add watch**.



Toto okno se hodí při ladění komplexnějších algoritmů, kdy je potřeba najednou sledovat proměnné definované ve více různých funkčních blocích.

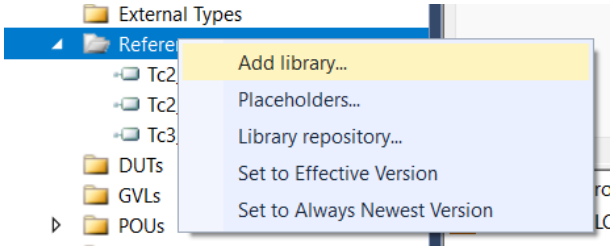
Stiskem klávesy F9 na libovolném řádku kódu můžeme vložit breakpoint.

```
1  fb(Inp1FALSE := in1 FALSE<TRUE>, Inp2FALSE := in2FALSE, OutFALSE => outFALSE);
```

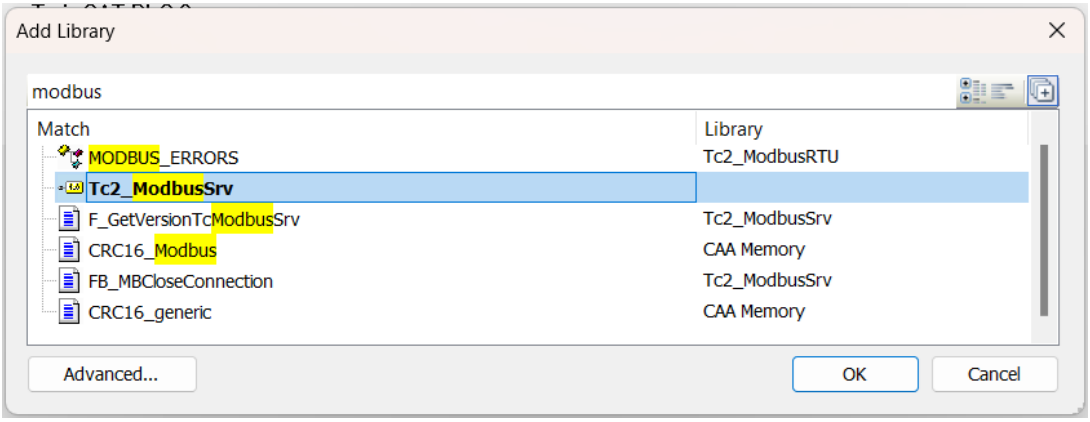
Následně můžeme provádět ladění tak, jak jsme zvyklí z Visual Studia, kdy F10 slouží pro vykonání aktuálně zobrazeného řádku, F11 pro skok dovnitř funkce/bloku, Shift + F11 pak pro opuštění aktuální funkce/bloku  (**Step Into, Step Over, Step Out**).

Instalace knihovny pro Modbus TCP (TF6250)

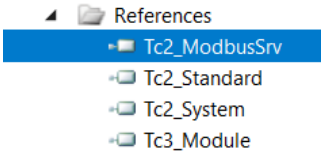
Pro práci ve cvičení budete potřebovat komunikovat prostřednictvím protokolu Modbus TCP se simulátorem technologického procesu. Detailně je tento simulátor komunikace s ním popsána v následujících kapitolách. Nyní si však ukážeme prerekvizitu, bez které komunikace není možná, a tou je instalace softwarové knihovny Modbus TCP klient/server. Pro instalaci knihovny vybereme v PLC projektu položku **Reference**, klikneme pravým tlačítkem myši a zvolíme možnost **Add library…**



V okně pro přidání knihovny můžeme do položky filtru vyplnit text **modbus** a následně vybereme položku **TC2_ModbusSrv.**



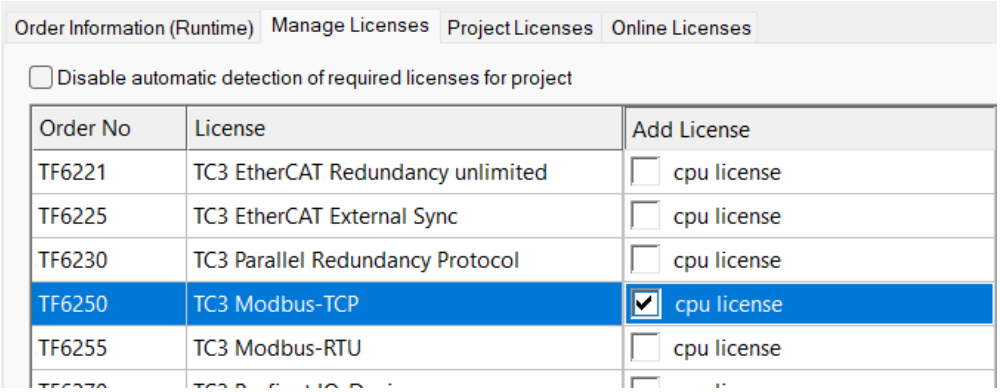
Po stisku OK dojde k přidání knihovny mezi reference



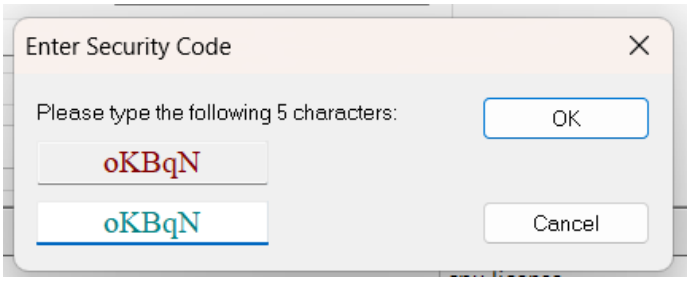
Pro využití knihovny je dále potřebná licence. V případě požadavku na dlouhodobý runtime provoz je nutné zakoupení komerční varianty této licence. Pro náš případ a provoz v Soft PLC postačí trial verze, která umožňuje nepřetržitý běh až 7 dní.

V okně **Solution Explorer** přejdeme do položky **System - License**, dvojklikem otevřeme licenční manažer.

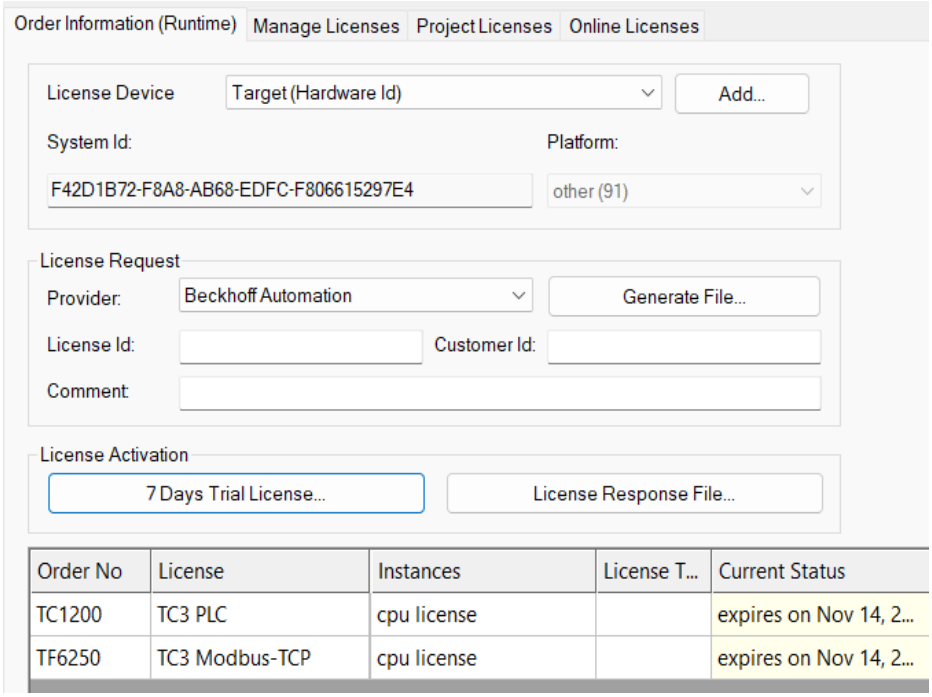
Na kartě **Manage Licenses** vyhledáme knihovnu TF6250 - TC3 Modbus-TCP a zaškrtneme checkbox **cpu license**.



Následně přejdeme na kartu **Order Information (Runtime)** a stiskneme tlačítko **7 Days Trial License…** Budeme vyzváni k opsání alfanumerického kódu. Po jeho opsání dojde k vygenerování sedmidenních licencí pro všechny používané softwarové knihovny.



Aby se změna projevila také na záložce **Order Information (Runtime)**, je nutno přejít na jinou záložku a následně zpět.



Tím bychom měli mít instalovanou a licencovanou knihovnu pro Modbus TCP. Následně již můžeme funkční bloky pro Modbus TCP komunikaci bez obav používat v našich programech. Připomeňme, že při změně parametrů komunikace bude vždy nutné znovu aktivovat konfiguraci PLC.

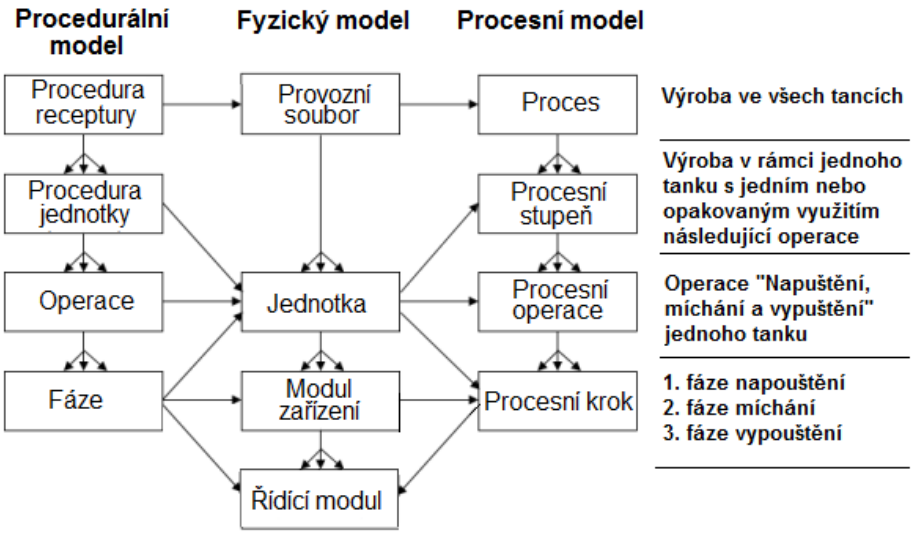
3. Základy programování v souladu s S88

Při tvorbě programu v PLC mnohdy okolnosti na první pohled svádí k myšlence vytvářet program prostým řazením instrukcí tak, jak udává funkční popis procesu. Tento popis je však jen jedním z požadavků, které musí program splnit. Dalšími požadavky jsou bezpečnostní, paměťová (kapacitní), časová a ekonomická hlediska nebo možnosti snadné modifikovatelnosti a další. Tyto požadavky a hlediska jsou zakotvena v normách, doporučeních, standardech a v neposlední řadě v představách investora.

Ve výkladu standardu ISA S88 v učebních textech pro přednášky se se dozvíte, že standard S88 je založen na třech modelech a jejich propojení. Jsou to modely:

- Procesní model – popisuje výrobní proces
- Fyzický model – popisuje fyzická zařízení procesu
- Procedurální model – popisuje recepturu (předpis) výroby

Vysvětlení těchto modelů a vazeb mezi nimi je zřejmé z následujícího obrázku.



Obrázek 6 Vazby mezi modely standardu S88

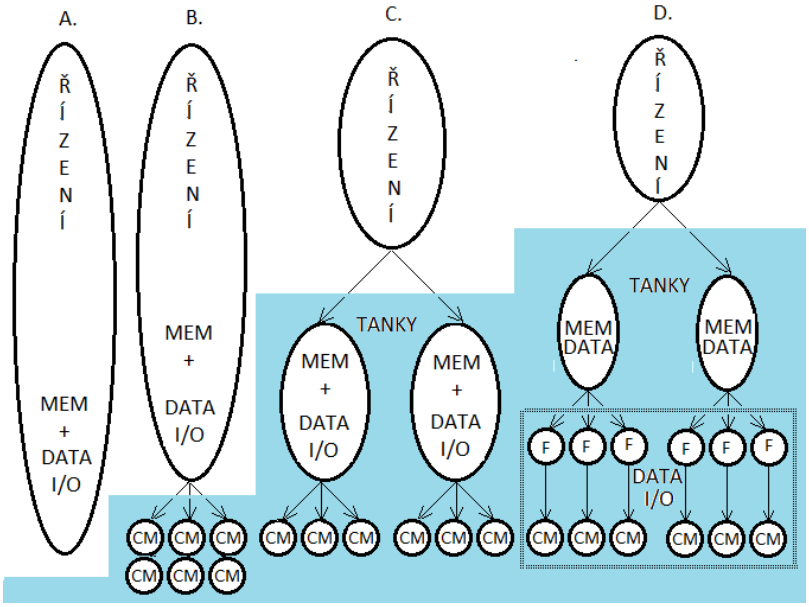
3.1. Požadavky na zařízení

Pro představu lze uvést základní provozní požadavky na řízení jakéhokoli výrobního zařízení:

Celý řídící program musí být provozovatelný v **automatickém** (AUT) a **ručním** (MAN) režimu. V automatickém režimu musí být respektovány **standardní požadavky na stavy, jako jsou např.** IDLE, RUN, HELD, DONE, STOPPED, ABORTED a **na povely**, kterými se tyto stavy navozují a řeší jako např. START, PAUSE, HOLD, RESUME, STOP. Jednotlivá zařízení a stroje musí být možno provozovat v ručním režimu, ale tak, aby jejich provoz byl bezpečný pro obsluhu i vlastní zařízení. Ruční provoz zařízení je možný i přes automatický režim procesu. Automatický provoz technologie je

možný i přesto, že nějaký modul zařízení je přepnut do manuálního režimu, v tom případě proces v určitém kroku čeká na přepnutí modulu zařízení do automatického režimu.

Mají-li být všechny požadavky na průmyslový řídicí systém splněny, není možné přistupovat k řešení programu jen z logiky funkce procesu, ale systémově, tak aby se nevytvářely kolizní vazby a nepřehledná křížení logických vazeb.



Obrázek 7 Míra opakovatelnosti v různých způsobech programování

Důsledkem všeho, co bylo jmenováno, je, že automatizační firmy si vytváří standardní moduly, ze kterých se pak programy skládají. Cílem je dosáhnout maximální možné opakovatelnosti v použití programových modulů. Ukázka takovéto opakovatelnosti je pro několik variant řízení zobrazena na předchozím obrázku, kde je její míra označena barevně.

3.1.1. Způsob A

V tomto způsobu jde o vytváření programu jako jednoho kompaktního celku, ve kterém jsou používána jak paměťová data MEM (markery a data z datových bloků), tak i I/O data z I/O karet PLC. Vzhledem ke kompaktnosti programu je opakovatelnost nulová a při zapracování nových změn do technologie se musí celý program modifikovat.

3.1.2. Způsob B

Tento způsob vychází z vyčlenění řídících modulů CM (pohony = ventily, motor, ohřev) z celkového programu. Každý modul CM (např. ventil) je implementován jako samostatná programová funkce, která se v programu opakovaně volá. Při modifikaci technologie se CM nových komponent pouze volají bez potřeby modifikace. Nicméně zásadní modifikaci je třeba provést v řídícím programu, ve kterém se opět vyskytují jak paměťová data MEM (markery a data z datových bloků), tak i I/O data z I/O karet PLC.

3.1.3. Způsob C

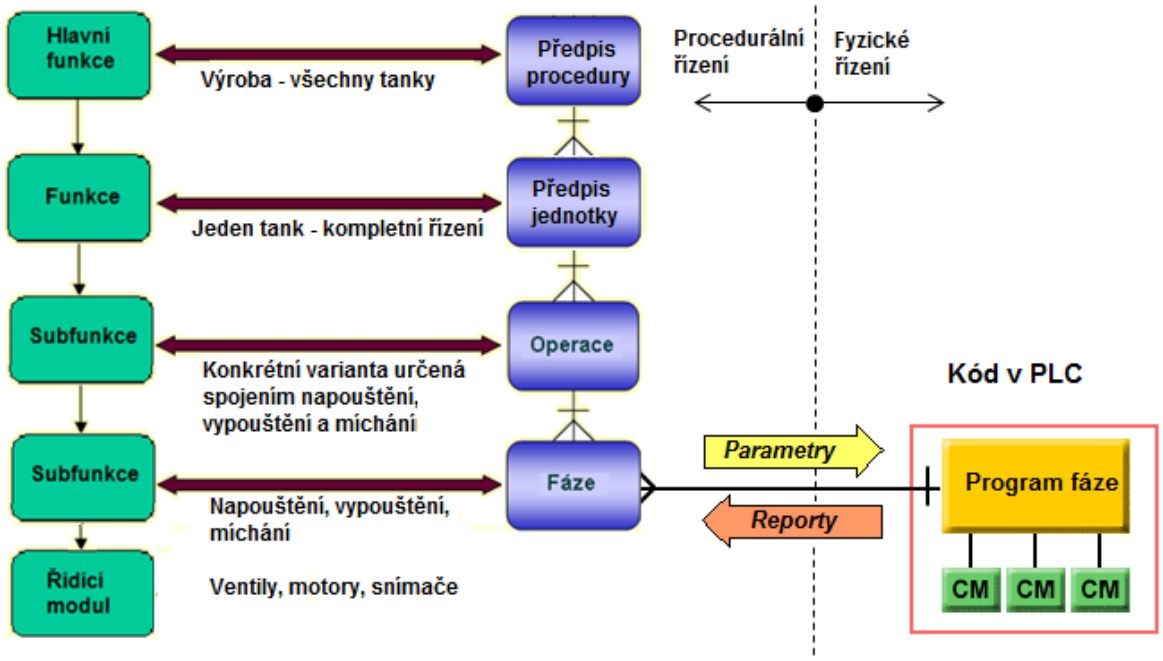
Tento způsob spočívá ve vyčlenění programu celého komplexního celku komponent (tento celek nazvěme jednotkou = unit), z řídicího programu. Část programu jednotky bude vytvořena opět jen jednou a bude opakovaně (pro více jednotek) volána. Zbývající program se nazývá hlavní funkce. Tím se opakovatelnost celého programu opět zvýší a případné doplnění technologie o novou jednotku stejného typu se projeví jen v počtu volaných řídících modulů a jednotek. Následné změny v hlavním řídícím programu už nebudou tak velké. I/O signály se však stále vyskytují nejen v řídících modulech, ale i v jednotkách (Tank). Hlavní funkce může mít několik jednotek.

3.1.4. Způsob D

Zásadní změnu přináší úprava ve způsobu D. Z jednotek se vyseparují části programového kódu obsahující návaznost na proces, tedy I/O data. Tyto nové části programu se nazývají fáze. Jednotka může mít několik fází. Takto upravená jednotka je zbavena I/O dat a obsahuje pouze paměťová data MEM (markery a data z datových bloků). Jak uvidíme dále, má tato vlastnost dalekosáhlý význam pro vznik programování typu BATCH.

3.2. Programování typu BATCH

Programování typu BATCH je předmětem jedné z přednášek kurzu MAUP a bude procvičováno v závěrečné části samostatného projektu. Souvislost mezi naším programem a programem typu BATCH, ke kterému v samostatném projektu směřujeme, je ilustrována na následujícím obrázku.



Obrázek 8 Vztah mezi hierarchickým programem PLC a programem BATCH

Zelenou barvou (vlevo) je na obrázku zobrazena hierarchie programu vytvořeného kompletně v PLC.

Struktura obsahuje:

- **Hlavní funkce** – řízení, volby, nastavení parametrů, množství apod. pro celý soubor jednotek
- **Funkce** – řízení jedné jednotky (program může být opakovaně použit pro více jednotek)
- **Subfunkce** – soubor několika fází například: napouštění, vypouštění, míchání
- **Řídící modul** – jsou to programové kódy pro motor, ventil, snímač apod., opakovatelně použitelné.

Modrou barvou je na obrázku nakreslena struktura programu, programovaného podle normy S88 BATCH, který je proveden v PC, nikoliv v PLC. Jak již bylo uvedeno, využije se zde toho, že v této části programu budou funkce, které obsahují pouze paměťová data MEM (markery a data z datových bloků) a ne prvky závislé na HW, proto mohou být programovány v PC. Části programového kódu obsahující návaznost na technologii/proces, tedy I/O data, budou naprogramovány v PLC a jsou ekvivalentní fázi, na obrázku jsou nazvány program fáze.

Silné šipky ukazují analogii modulů v obou způsobech programování.

3.3. Požadavky na řízení zařízení

Základním požadavkem funkce pro řízení zařízení je vytvoření interface mezi fyzickými I/O a požadavky řízení. To zjednodušeně znamená, že povely, které spouštějí/ovládají zařízení nejprve projdou přes tuto funkci a na základě stavu zařízení provede nebo neprovede akci. V této funkci je nutné ošetřit následující úkony:

1. ***Vyhodnocení poruchových stavů a přepnutí do bezpečného stavu*** (vypnutí výstupu, stav zařízení přepnout do poruchového stavu, indikovat poruchu)
    a. ***Pokud je poruchový signál na vstupu PLC*** (di…error) - Porucha signalizována speciálním výstupem zařízení, například termo-kontaktem motoru signalizujícím přehřátí.
    b. ***Pokud nedojde k rozběhu/otevření/zavření zařízení – pokud*** je k dispozici zpětná vazba od zařízení (u motoru – čidlo otáček, sepnutý stykač, snímač proudu apod., u ventilu koncové snímače), obslužný program ji očekává po sepnutí výstupu do definovaného času. Pokud se tak nestane, je nutné vyhlásit poruchu rozběhu. (nefunkční stykač, špatný koncový snímač, zadřený pohon apod.). Tuto poruchu je možné resetovat, a pokud je požadavek na spuštění stále aktivní, opět je možné sepnout výstup.
    c. ***Porucha zpětných vazeb –*** může nastat v případě, kdy by pohon neměl být aktivní, ale zpětná vazba signalizuje opak. V závislosti na typu zařízení je pak také vhodné vyhlásit poruchu. Nejčastěji se tyto problémy mohou vyskytnout u klapek, regulačních ventilů, ventilů atp., kdy jsou použity koncové snímače. Při špatném nastavení snímačů, nebo jejich poruše, může dojít k signalizaci obou koncových poloh a je nutné vyhlásit poruchový stav zařízení.
2. ***Přepínání manuálního a automatického režimu –*** každé řízené zařízení (ventil, pohon, ohřev) by mělo mít možnost přepínat mezi manuálním a automatickým režimem. Manuální režim je

především pro údržbu, testování a ladění (zařízení je v tomto režimu možno ovládat z HMI a není možné jej ovládat nadřazeným řídicím algoritmem). V automatickém režimu čeká zařízení na řízení nadřazeného prvku. Pokud jsou všechny potřebné náležitosti splněny, může se zařízení spustit. Manuální režim reaguje na hrany povelů, automatický režim na stav.

3. ***Implementace beznárazového přepínání*** - při přepnutí z automatického do manuálního režimu se řízené zařízení přepne do stavu, v jakém bylo před přechodem do ručního režimu. Při předání řízení zpátky do automatu již zařízení ovládá nadřazený algoritmus. Beznárazové přepnutí je nutné využívat hlavně při spínání pohonů vysokých výkonů, které mají většinou omezený počet sepnutí za určitou dobu. Také je vhodné řešit tento problém u pohonů řízených frekvenčním měničem nebo při jakémkoli spojitém řízení, kdy po přepnutí z automatického do manuálního režimu zůstane aktivní poslední stav s poslední hodnotou výstupu.

4. ***Počítání provozního času (motohodin apod.)*** - v praxi velmi důležitý parametr pro většinu zařízení (někdy se přidává i počítání servisních intervalů) není nutné je počítat u ventilů, servoklapek apod. Důležitá je možnost motohodiny nastavit, není vždy pravidlem, že s novým řízením se do provozu dostanou i nová zařízení.

3.4. **Modul zařízení**

Modul zařízení reaguje na způsob tvorby C z obrázku 9. Implementace modulu zařízení zpracovává vstupní signály ze zařízení a spíná výstupní signály pro zařízení (záleží na tom, ze kterého úhlu se na situaci podíváme, proto věnujte pozornost ne slovíčkům vstupní a výstupní, ale raději signály ze zařízení a signály pro zařízení). Modul zařízení vyhodnocuje stavy a požadavky na sepnutí. Je nutné definovat bezpečný stav (což je často vypnutý výstup v případě poruchy).

3.5. **Fáze**

Tento prvek je stěžejní pro způsob D z obrázku 9. V jednotce technologie zavedeme nový element procesního i procedurálního modelu, a tím je fáze.

Z pohledu elektrikáře je nejnižším prvkem v hierarchii řízení motor, ventil, snímač apod. Z pohledu technologa se takovýto ventil jeví jako „napouštění“, motor dopravníku jako „transport“, čidlo hladiny jako „horní hladina“ apod. Je to pohled procesní a z hlediska popisu funkce pohled procedurální. Fáze je tedy nejnižší a nejmenší element v procedurálním modelu. Fáze napouštění, transportu, horní hladiny apod. Fáze je určena pro automatické řízení řídícího modulu (motor, ventil).

Fáze slouží k řízení modulů zařízení v automatickém režimu. Je zde tedy podmínka, že fáze může ovládat modul, pouze pokud je sama fáze v aktivním stavu. Aby fáze mohla přímo ovládat zařízení pomocí svých parametrových bitů pro otevření/zavření ventilu nebo motoru, musí být modul zařízení v automatickém režimu.

Režimy fází jsou opět automatický a manuální. Proces, který je nadřazený fázím (většinou to bývá proces BATCH) se může spustit i přes manuální režim některých fází. Fázi v tomto případě pouze zapínáme a indikujeme její dokončení. U některých fází se fáze může ukončit, jakmile je splněna ukončovací podmínka (například dojde k napuštění na požadovanou hodnotu). Tato podmínka bude zadána jako žádaná hodnota ve SCADA systému.

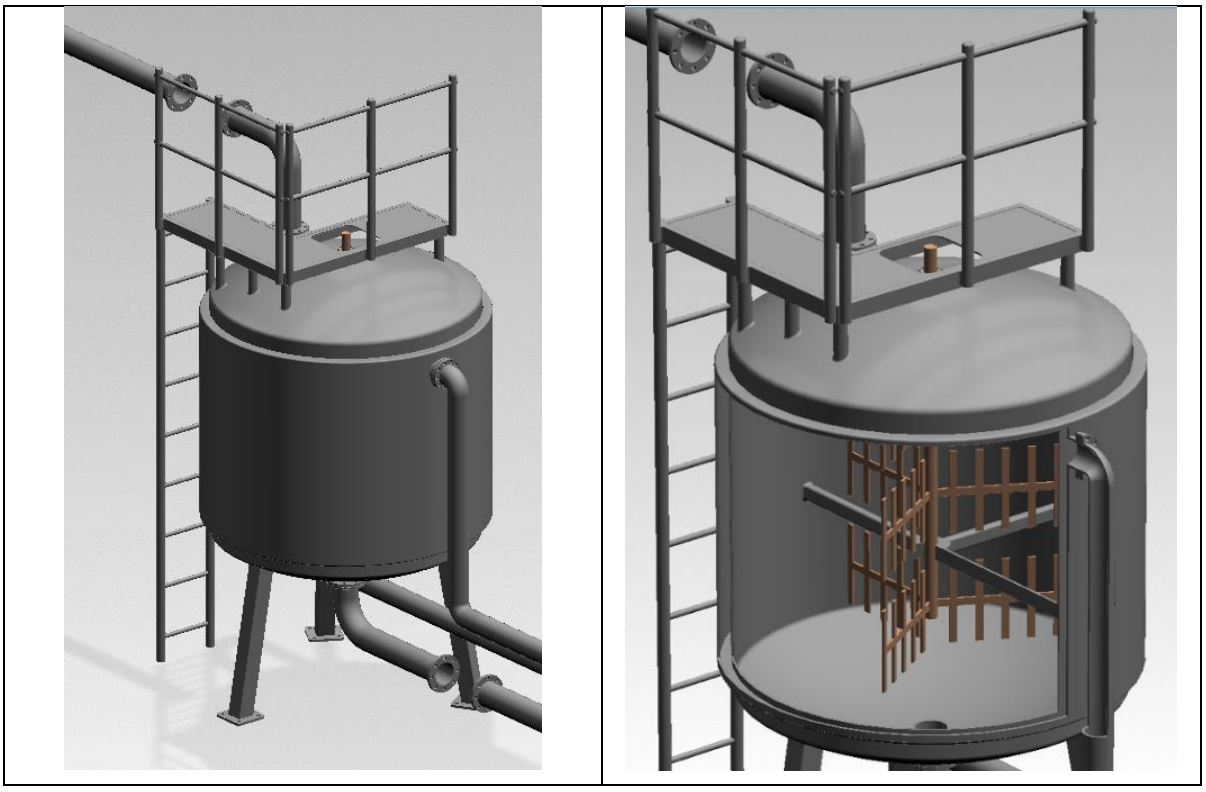
4. Projekt Tanky

Cílem celosemestrální laboratorní úlohy Tanky je zpracovat kompletní projekt pro automatizaci části technologického procesu, konkrétně pasterizační jednotky. Jednotlivými dílčími úkoly jsou:

Analýza zadání
Vytvoření P+I diagramu
Volba vhodné instrumentace
Vytvoření elektrotechnického schématu
Implementace řídicího SW pro PLC
Ladění regulátorů technologie
Tvorba vizualizace
Tvorba dávek pomocí BATCH systému
Funkční testy a protokol o testování.
Závěrečná zpráva

4.1. Technologický proces – pasterizační jednotka

Technologický proces sestává z nerezového dvouplášťového tanku (nádrže s vnějším ohřevem. Do tohoto tanku je potrubím přiváděna vstupní surovina. Dalším potrubím je z tanku odváděn hotový výrobek. Napouštění a vypouštění suroviny je řízeno ventily. V tanku se rovněž nachází mixovací jednotka, pomocí které jsou homogenizovány vlastnosti celého objemu kapaliny (míchání). Pro ohřev tanku je využívána externí cirkulační jednotka s plynule nastavitelným výkonem, která do dvoupláště nádrže vhání horké médium za účelem ohřevu kapaliny na požadovanou teplotu. Popisovaná technologie lze vidět na následujícím obrázku.



Obrázek 9 Konstrukce pasterizační jednotky

4.1.1. Popis technologie

Technologický proces slouží k pasterizaci kapalin. Nerezová nádrž je **vysoká 2000 mm** a její objem je přesně **2 m^3**. Pro přívod materiálu je využito vstupní a pro odvod výstupní potrubí. Vstupní potrubí o průměru **DN125** je konstantně tlakováno vstupním materiálem. Výstupní potrubí, rovněž o průměru **DN125** je přivedeno do zásobníků, které uchovávají výstupní produkt pro další zpracování. Technologie je vybavena mechanismem pro míchání materiálu uvnitř tanku (mixérem), jehož statický krouticí moment v okamžiku kdy je tank zcela plný je **380 N/m** a jehož maximální přípustná rychlost je **40 ot./min**. Tento mechanismus je vybaven převodovkou s **převodovým poměrem 38:1**. Pro ohřev je k technologii připojen tepelný okruh z přidružené výroby (jaderné elektrárny) s plynule regulovatelným jmenovitým výkonem **25MW**. Maximální přípustná teplota veškerých mechanických částí je **95 °C,** po jejímž překročení dojde k nenávratným škodám a technologie bude zničena.

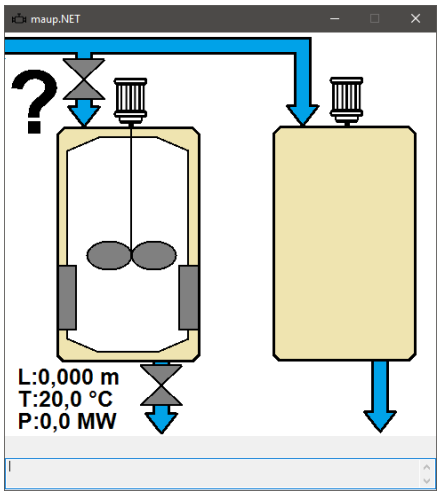
Popis funkce:

- Do tanku je napuštěno **požadované množství** vstupního materiálu
- Materiál je postupně zahříván na **požadovanou teplotu**
- V průběhu zahřívání je pro teplotu vyšší než **40 °C** nutné homogenizovat obsah tanku prostřednictvím mixeru.
- Po dosažení **požadované teploty** a jejím držení **požadovaný časový úsek,** je výsledný produkt odčerpán výstupním potrubím.

4.2. Simulátor technologie

Vzhledem k tomu že se na půdě UAMT nenachází dostatečný počet pasterizačních jednotek, který by bylo možné ničit studenty, je pro tvorbu studentských projektů vytvořen **simulátor technologie tanky**.

Tento simulátor je prostřednictvím komunikačního rozhraní virtuálně připojen na vstupy a výstupy PLC a zprostředkovává reálnou odezvu na akční zásahy. Stav simulátoru lze sledovat na projektoru v laboratoři. Pro každé pracoviště je spuštěna jedna instance simulace technologie s číselným označením odpovídajícím štítku, které naleznete na svém pracovišti.



Obrázek 10 Screenshot simulátoru technologie

V následující tabulce (Tabulka 1) jsou popsány významy jednotlivých signálů u jednotlivých komponent.

Tabulka 1 Mapování vstupů a výstupů technologie

Komponenta	Signál	Typ	Adresa
Napouštěcí ventil	Výstup aktivující otevírání ventilu (reaguje na stav).	Bool	QB0.0
	Výstup aktivující zavírání ventilu (reaguje na stav).	Bool	QB0.1
	Zpětná vazba udávající, zda je ventil zcela otevřen.	Bool	IB0.0
	Zpětná vazba udávající, zda je ventil zcela uzavřen.	Bool	IB0.1
	Indikace poruchového stavu ventilu.	Bool	IB0.2
Vypouštěcí ventil	Výstup aktivující otevírání ventilu (reaguje na stav).	Bool	QB0.2
	Výstup aktivující zavírání ventilu (reaguje na stav).	Bool	QB0.3
	Zpětná vazba udávající, zda je ventil zcela otevřen.	Bool	IB0.3
	Zpětná vazba udávající, zda je ventil zcela uzavřen.	Bool	IB0.4
	Indikace poruchového stavu vypouštěcího ventilu.	Bool	IB0.5
Míchadlo	Výstup aktivující motor míchadla (reaguje na stav).	Bool	QB0.4
	Zpětná vazba udávající, zda míchadlo běží.	Bool	IB0.6
	Indikace poruchového stavu míchadla.	Bool	IB0.7
Ohřev	Výstup aktivující ohřev.	Bool	QB0.5
	Výstup pro řízení výkonu ohřevu.	Int	QW0
Hladina	Vstup připojený ke snímači hladiny v tanku.	Int	IW0
Teplota	Vstup připojený ke snímači teploty tekutiny v tanku.	Int	IW2

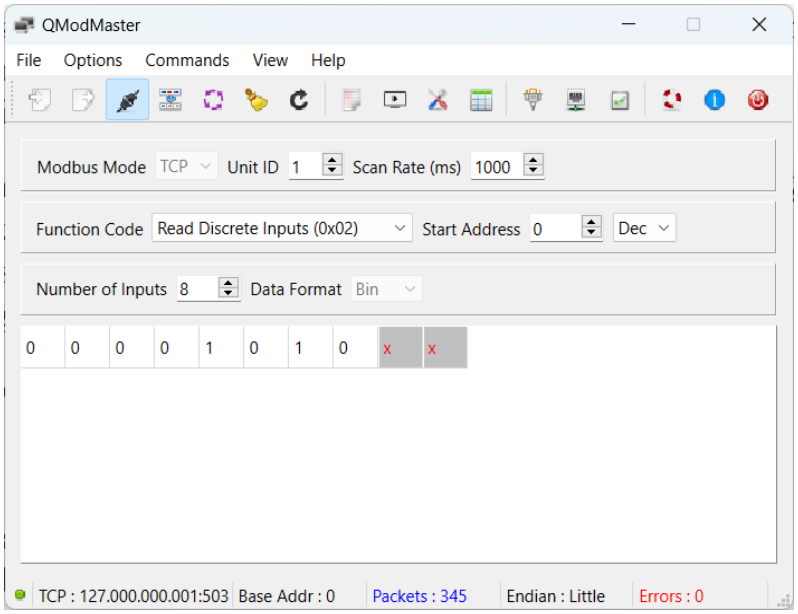
4.3. **Komunikace simulátoru s PLC**

PLC čte hodnoty vstupů a zapisuje hodnoty výstupů prostřednictvím protokolu Modbus TCP. Simulátor otevírá a naslouchá na portu 502 (Standardní port pro Modbus TCP). Komunikace je realizována prostřednictvím "spouštění" Modbus funkcí, které jsou

- Čtení diskrétních vstupů (Read Discrete Inputs - 0x02) - IB v tabulce výše
- Čtení vstupních registrů (Read Input Registers - 0x04) - IW
- Zápis více výstupních cívek - bitů (Write Multiple Coils - 0x0f) - QB

- Zápis více registrů (Write Multiple Registers - 0x10) - QW

Komunikaci je možno otestovat prostřednictvím programu qModMaster. V tomto programu je po otevření nutno zadat TCP port (Options - Modbus TCP), v případě, že je simulátor spuštěn na stejném počítači jako program, je možné ponechat lokální IP adresu (127.0.0.1). V hlavním okně programu je nutno nastavit Unit ID = 1, Start Address = 0 a ujistit se, že je zvolen Modbus Mode na "TCP". Pak pomocí třetí ikony zleva je možné aktivovat spojení a následující ikony slouží k realizaci jednoho cyklu či cyklické komunikace.



Obrázek 11 Test komunikace pomocí qModMaster

FAQ:

- Operační systém brání spuštění AUP.NET nebo program po spuštění po několika sekundách spadne
  - Zkuste program přesunout na plochu, nemějte více zanořených složek AUP.NET do sebe.
  - Stáhněte znovu program a zkuste spustit.
- Nefunguje komunikace
  - Zkontrolujte nastavení programu AUP.NET v souboru App.config, zejména mrkněte na nastavený ModbusTcpPort – musí samozřejmě souhlasit na klientovi i serveru. Můžete zkusit port změnit, samozřejmě na obou zařízeních.
  - Zkuste dočasně pozastavit firewall. Pokud to zabere, vložte do firewallu pravidlo, které povolí komunikaci na portu 502.

5. Úkol 1: Analýza zadání

Cílem cvičení je detailně se seznámit se zadáním a porozumět mu. Je výrazně doporučeno dělat si poznámky. ***Látka, která bude vysvětlena, je nezbytná k dokončení všech ostatních cvičení.***

5.1. Pokyny k vypracování:

1. Společně s vyučujícím se seznamte s technologickým procesem tanku.
2. Diskutujte použití různých komponent v procesu.
3. Sestavte společně technologické schéma (P + I diagram).
4. Seznamte se se simulací fyzikálního systému.

5.2. Výstup:

Výstupem, který bude použit pro další cvičení je vypracovaná úvodní část technické dokumentace obsahující

1. Popis technologie
2. Klíčové parametry technologie (odrážky)
3. Speciální požadavky na technologii
4. Popis výrobního procesu a jeho jednotlivých částí
5. Popis použitých akčních členů včetně jejich funkce při řízení procesu
6. Popis použitých senzorů včetně jejich funkce při řízení procesu
7. Popis a vysvětlení nežádoucích stavů, jejich příčin, způsobů detekce a zotavení
8. Ručně nakreslený P + I diagram

Dále pak je požadavkem

- Znalost funkčního popisu technologie
- Znalost simulátoru a jeho vstupů a výstupů.

5.3. Pravidla kreslení PI&D diagramů podle standardu ANSI/ISA-5.1-2009

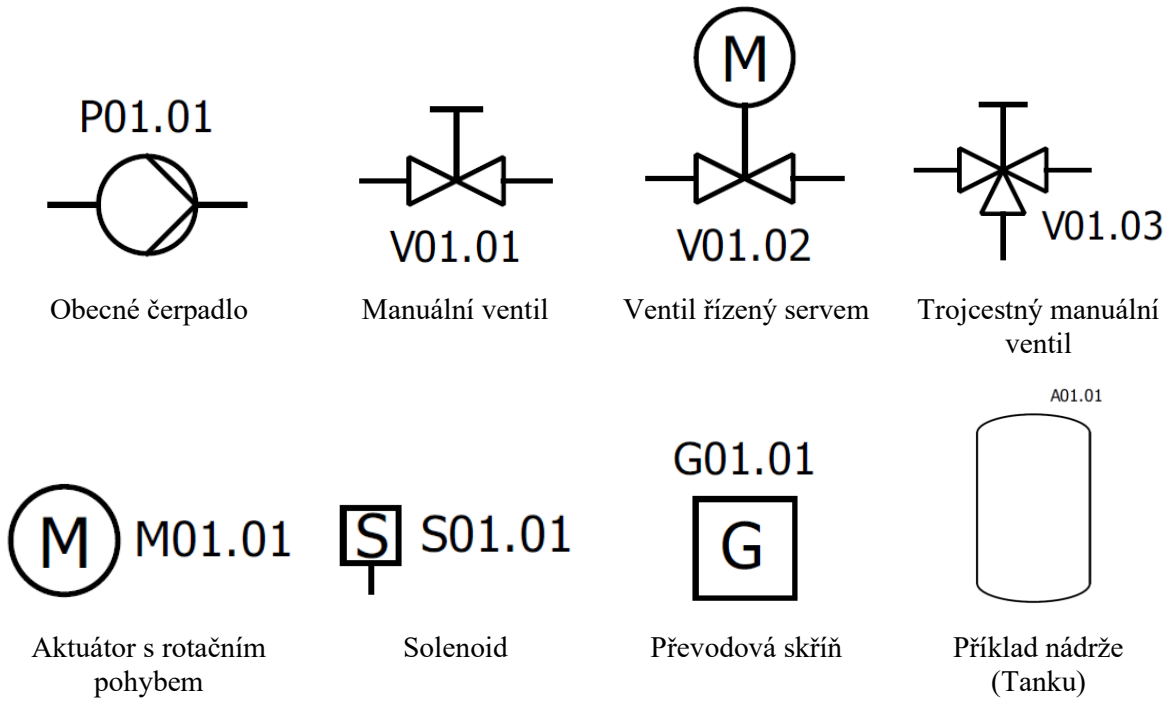
P+I (Pipes and instumentation) je schéma obsahující toky, akční členy, měřicí body apod. Schématické symboly vychází z normy ANSI/ISA-5.1-2009.

5.3.1. Symboly PI&D diagramů

Jednoduché schéma lze sestavit pomocí symbolů (Obrázek 12). Čerpadlo je využívané tam, kde je nutný přesun kapalin. Čerpadlo může zvýšit průtok, nebo v závislosti na typu ho přesně dávkovat. Pokud zadání projektu definuje buňku, která je uvažována jako součást rozsáhlejší technologie, je vhodné na výstupu použít čerpadlo pro překonání výškových rozdílů potrubí.

Ventily mají schopnost regulovat průtok materiálu v potrubí. Důležitý význam pro automatizaci technologie je typ aktuátoru, kterým je ventil řízen. Ventil vybavený servopohonem lze řídit elektronicky a zpětně vyčítat koncové polohy, nebo i úhel otevření. Ventily s manuálním ovládáním je vhodné umístit tak, aby umožňovaly výměnu hlavních ventilů a opravy technologie.

Aktuátor s rotačním pohybem a solenoid jsou zdrojem mechanického pohybu, který lze využít pro řízení ventilů. Úprava otáček a kroutivého momentu zajišťuje převodovka připojená například na asynchronní motor. Nádrž slouží jako prostor pro skladování, nebo zpracování materiálu. V závislosti na požadavcích technologie jsou z nádrže vyvedeny měřící body popsané v následující kapitole.



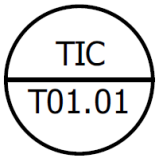
Obecné čerpadlo | Manuální ventil | Ventil řízený servem | Trojcestný manuální ventil

Aktuátor s rotačním pohybem | Solenoid | Převodová skříň | Příklad nádrže (Tanku)

Obrázek 12 Příklady symbolických značek PI&D diagramu

5.3.2. Měřící body (prvky instrumentace) a jejich značení

Symboly měřících bodů, nebo prvky instrumentace (Instrumentation device and function symbols) obsahují číslo bodu a popis jeho funkce. Funkce snímače je značena pomocí písmenného kódu, kde má každé písmeno v závislosti na pozici v kódu definovaný význam (Tabulka 3). Symbol měřícího bodu, který je připojen na ostatní symbol P+I diagramu udává, jaká veličina je snímána a jakým způsobem je využita. Například zkratka TIC (Obrázek 13) popisuje funkci snímání teploty, její zobrazení (pasivní) a řízení průběhu procesu (aktivní).



Obrázek 13 Značka měřícího bodu

Symboly prvků instrumentace mají různé variace podle rozlišení jejich umístění a typu (Tabulka 2). Umístění a typ udává, zda je měřená hodnota přístupná pouze lokálně v technologii, nebo zda je její hodnota dostupná centrálně, například prostřednictvím HMI.

Tabulka 2 Instrumentation device and functions symbols

No.	Shared display, Shared control (1)		C	D	Location & accessibility (6)
	A	B			
	Primary Choice or Basic Process Control System (2)	Alternate Choice or Safety Instrumented System (3)	Computer Systems and Software (4)	Discrete (5)	
1					• Located in field. • Not panel, cabinet, or console mounted. • Visible at field location. • Normally operator accessible.
2					• Located in or on front of central or main panel or console. • Visible on front of panel or on video display. • Normally operator accessible at panel front or console.
3					• Located in rear of central or main panel. • Located in cabinet behind panel. • Not visible on front of panel or on video display. • Not normally operator accessible at panel or console.
4					• Located in or on front of secondary or local panel or console. • Visible on front of panel or on video display. • Normally operator accessible at panel front or console.
5					• Located in rear of secondary or local panel. • Located in field cabinet. • Not visible on front of panel or on video display. • Not normally operator accessible at panel or console.

Tabulka 3 Identifikační písmena

	First letters (1)		Succeeding letters (15)		
	Column 1	**Column 2**	**Column 3**	**Column 4**	**Column 5**
	Measured/Initiating Variable	**Variable Modifier (10)**	**Readout/Passive Function**	**Output/Active Function**	**Function Modifier**
A	Analysis (2)(3)(4)		Alarm		
B	Burner, Combustion (2)		User's Choice (5)	User's Choice (5)	User's Choice (5)
C	User's Choice (3a)(5)			Control (23a)(23e)	Close (27b)
D	User's Choice (3a)(5)	Difference, Differential, (11a)(12a)			Deviation (28)
E	Voltage (2)		Sensor, Primary Element		
F	Flow, Flow Rate (2)	Ratio (12b)			
G	User's Choice		Glass, Gauge, Viewing Device (16)		
H	Hand (2)				High (27a)(28a)(29)
I	Current (2)		Indicate (17)		
J	Power (2)		Scan (18)		
K	Time, Schedule (2)	Time Rate of Change (12c)(13)		Control Station (24)	
L	Level (2)		Light (19)		Low (27b)(28)(29)
M	User's Choice (3a)(5)				Middle, Intermediate (27c)(28) (29)
N	User's Choice (5)		User's Choice (5)	User's Choice (5)	User's Choice (5)
O	User's Choice (5)		Orifice, Restriction		Open (27a)
P	Pressure (2)		Point (Test Connection)		
Q	Quantity (2)	Integrate, Totalize (11b)	Integrate, Totalize		
R	Radiation (2)		Record (20)		Run
S	Speed, Frequency (2)	Safety(14)		Switch (23b)	Stop
T	Temperature (2)			Transmit	
U	Multivariable (2)(6)		Multifunction (21)	Multifunction (21)	
V	Vibration, Mechanical Analysis (2)(4)(7)			Valve, Damper, Louver (23c)(23e)	
W	Weight, Force (2)		Well, Probe		
X	Unclassified (8)	X-axis (11c)	Accessory Devices (22), Unclassified (8)	Unclassified (8)	Unclassified (8)
Y	Event, State, Presence (2)(9)	Y-axis (11c)		Auxiliary Devices (23d)(25)(26)	
Z	Position, Dimension (2)	Z-axis (11c), Safety Instrumented System (30)		Driver, Actuator, Unclassified final control element	

6. Úkol 2: Tvorba P+I diagramu v programu QElectroTech

Cílem cvičení je seznámit se s návrhovým systémem QElectroTech a vytvořit digitální podobu P+I diagramu z předchozího úkolu.

6.1. Pokyny k vypracování:

1. Seznamte se s programem QElectroTech. K seznámení využijte tutoriál nacházející se v této kapitole.
2. Nakreslete P+I diagram technologie pasterizace mléka, navrhněte si prvek „Tank“ podle vlastních potřeb
3. Použijte v P+I diagramu legendu nespecifikovaných funkcí měřících bodů.

6.2. Výstup:

Požadovaným výstupem, který bude odevzdán do IS, je:

1. PDF soubor obsahující Vámi vytvořený P+I diagram a legendu.

6.3. Tutoriál pro práci s programem QElectroTech

Tato kapitola popisuje základy práce v programu QElectroTech v 0.90.

6.3.1. Horní lišta

V horní části okna programu se nachází lišta (Obrázek 14) se základními funkcemi pro práci s projektem. Lišta obsahuje tlačítka pro tisk výkresu, nebo export výkresu do formátu pdf.

Obrázek 14 Horní lišta nástrojů

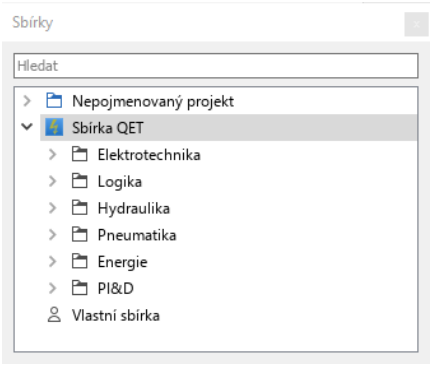
6.3.2. Pracovní plocha

Po vytvoření projektu je v záložce Projekty znázorněna struktura tohoto projektu a je vytvořen jeden výkres (list). Po otevření tohoto listu vidíme, že pracovní plocha je ohraničena rámečkem a razítkem. Vzhled listu je možné pomocí drag&drop ze seznamu **Záhlaví výkresů QET** změnit, případně lze záhlaví (v seznamu) po rozkliknutí libovolně upravovat. Pro realizaci projektu doporučujeme použít vzor „Default“. Velikost listu je možné upravovat klepnutím pravým tlačítkem na pracovní plochu a výběrem **Přidat/odstranit řádek/sloupec**. Mějte však na paměti, že je lepší schémata dělit do více listů jednak kvůli přehlednosti a také kvůli možnosti pozdějšího tisku, např. na formát papíru A4.

Každý list projektu by měl mít vyplněné jméno autora, datum a název projektu. To je možné vyplnit buď ve vlastnostech listu, případně globálně ve vlastnostech projektu (pravé tlačítko na jméno projektu v záložce **Projekty**).

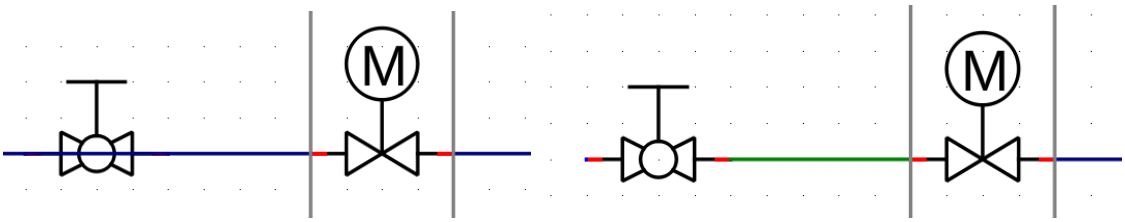
6.3.3. **Knihovny**

Přístup ke všem knihovnám, je možný pomocí okna „Sbírky“ (Obrázek 15). Knihovny obsahují elektrické komponenty pro tvorbu elektro-schémat i pro P+I diagramy. Ve složkách „Hydraulika“ a „Pneumatika“ jsou komponenty pro tvorbu hydraulických a pneumatických schémat. Pro první cvičení pro Vás bude důležitá právě knihovna PI&D. Ve složce „Energie“ jsou dodatečné komponenty, které lze případně použít pro doplnění P+I schématu.



Obrázek 15 Okno Sbírky – knihovny všech komponentů

Symbol lze vložit na pracovní plochu pomocí drag&drop, přičemž program při pohybu symbolem nad plochou zobrazuje modré čáry horizontálně a vertikálně od svorek – pomocí nich je možné sesouladit nově přidávaný prvek s již existujícími. Pokud je aktivní volba  na liště ikon, pak v případě, že se horizontálně, či vertikálně od svorek nového prvku nacházejí prázdné svorky jiného prvku, zbarví se příslušná čára do zelena a po upuštění prvku dojde k automatickému zadrátování (Obrázek 16).

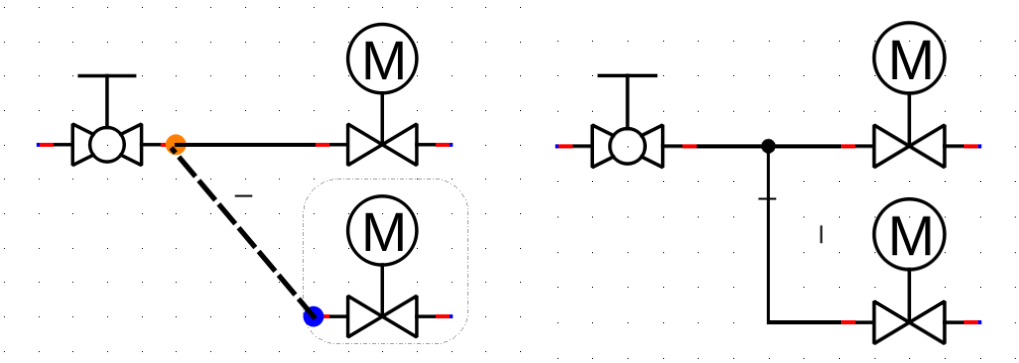


Obrázek 16: Vložení symbolu na plochu

Při propojování více symbolů (větvení tras) je nutné propojovat vždy svorky prvků (Obrázek 17 vlevo). Není možné připojit svorku nového prvku někam doprostřed existujícího propoje. Program však dovede poměrně inteligentně vést nový propoj v zákrytu se stávajícím propojem a tam, kde se tyto rozdělí, namalovat puntík (Obrázek 17 vpravo).

Automaticky vygenerovanou dráhu je možné následně upravovat – klikněte na propoj, v každé hraně se zobrazí modrá tečka, za kterou je možné propoj potáhnout do požadovaného směru. Klikem pravého tlačítka a výběrem **Upravit vodič** je možné upravit vzhled propoje a zejména jej pojmenovat a např. přiřadit velikost napětí na vodiči, či zadat typ média, který je potrubím transportováno. Stejně tak je možné volit barvu a styl čáry propoje.

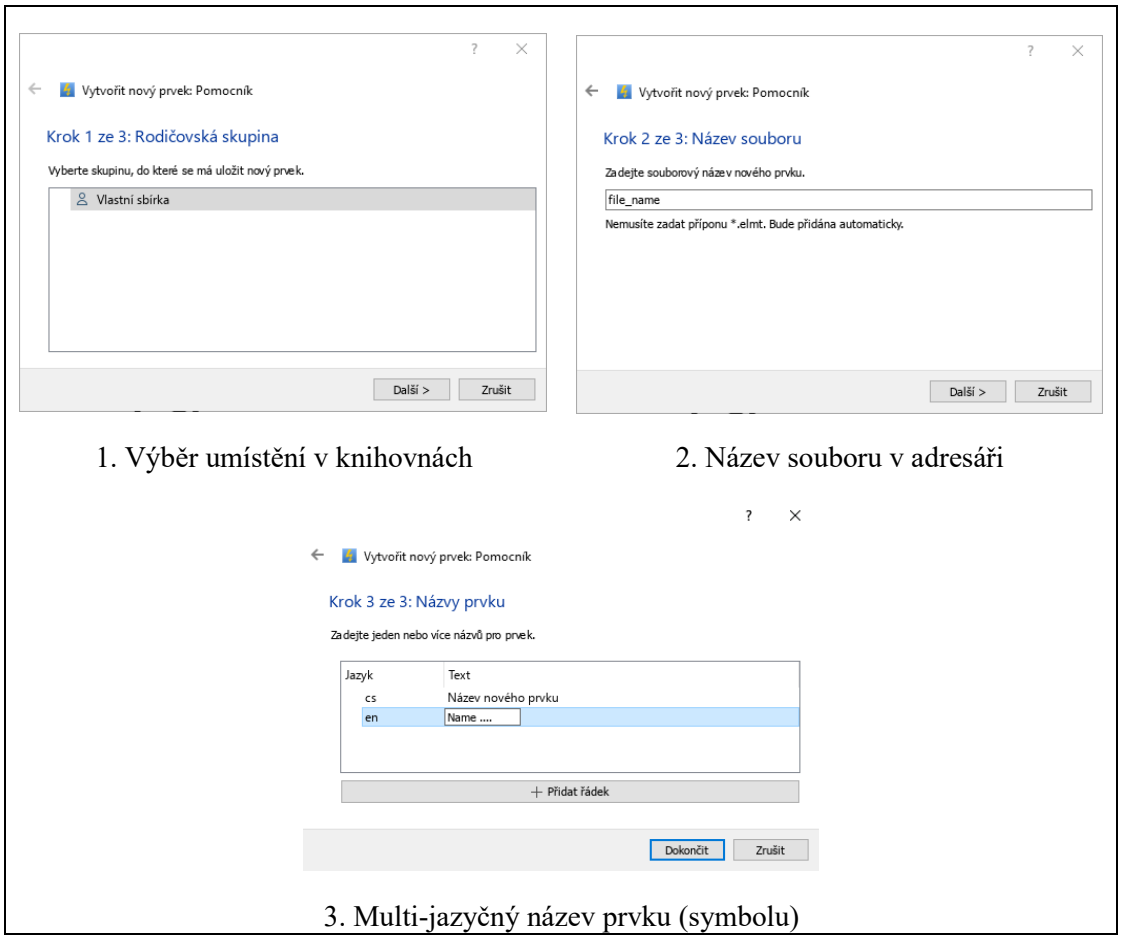
Vyplněný text propoje je důležitý zejména v případě, kdy budete potřebovat na tento propoj vytvořit referenci na jiném listu!



Obrázek 17: Propojení tří prvků

6.3.4. Vytvoření nového symbolu

V případě, že v knihovně neexistuje požadovaný symbol, je možné vytvořit symbol nový, nebo klonovat a upravit existující. Kliknutím pravým tlačítkem na „Vlastní sbírka“ se vyvolá seznam možností, který nabízí vytvoření nového prvku. Po rozkliknutí je nutné zadat informace celkem do tří oken (Obrázek 18). Zvolí se umístění prvku v knihovně, název souboru, pod kterým je poté soubor dohledatelný v adresáři a název symbolu, pod kterým se nový symbol bude zobrazovat v knihovně. Název je možné zadat pro každý jazyk zvlášť. Přepínání mezi jazyky se děje automaticky v závislosti na globálním nastavení jazyku v programu.

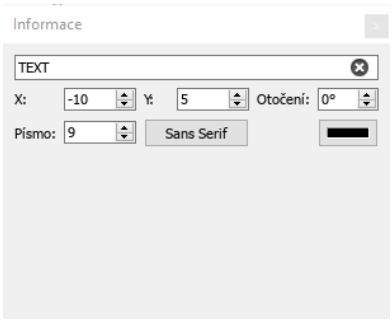


Obrázek 18 Založení nového prvku (symbolu)

Po rozkliknutí symbolu dvojklikem, nebo přidáním nového symbolu, se otevře okno, kde lze symbol upravovat (Kvůli umístění souborů knihoven v systémovém adresáři je „Sbírka QET“ pouze pro čtení). V levé části je lišta grafických nástrojů (Obrázek 19). Lze kreslit samostatné úsečky, obdélníky, elipsy, mnohoúhelníky, textové pole a části elips.

Obrázek 19 Lišta grafických nástrojů

Pro každý aktuálně vybraný grafický prvek lze nastavovat jeho vlastnosti (Obrázek 20). Nastavit lze umístění prvku a různé jiné parametry, které jsou individuální pro každý druh prvku.



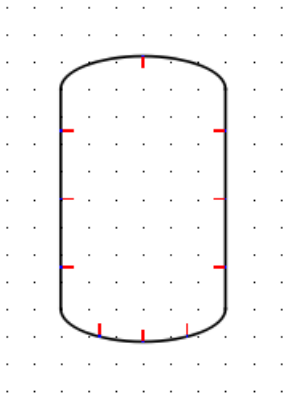
Obrázek 20 Nastavení vlastností grafických prvků

Obyčejný text je pevně zakotvený na své pozici a jeho obsah je neměnný. Naproti tomu dynamický text lze nastavit tak, aby zobrazoval některou z informací o prvku. Na výběr je ze tří typů zdrojů textu:

- „Vlastní text“ lze upravovat i během kreslení na pracovní ploše. Zobrazuje předdefinovaný text, který lze upravit uživatelem.
- „Informace o prvku“ vybírá pouze jeden parametr symbolu, který zobrazuje v textovém poli tak jak je. Uživatel má možnost změnit tento text pouze v nastavení parametrů součástky na pracovní ploše (Ctrl+E). Nejčastěji je zobrazován parametr „Štítek“.
- „Smíšený text“ nabízí variabilitu kombinace vlastního textu a parametrů symbolu. Lze zkombinovat několik parametrů dohromady.

Asi nejefektivnější způsob, jak nastavit dynamický text, je nastavení „Informace o prvku“ pro parametr štítek. Po přidání symbolu na pracovní plochu je poté nutné vyplnit tento parametr ve vlastnostech konkrétního symbolu.

Aby bylo možné symbol propojit s ostatními symboly, je nutné přidat alespoň jednu svorku. Modré body na svorkách udávají místo připojení. Po exportu schématu do pdf souboru se červené části svorek nevykreslují. Estetiku svorek při návrhu lze tedy zanedbat, slouží pouze pro určení směru vedení. Obrázek 21 například zobrazuje umístění svorek tak, aby bylo možné táhnout vedení směrem ven z tanku.

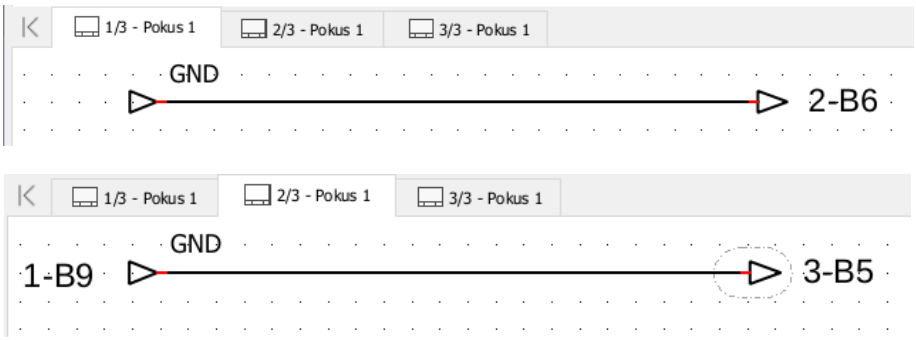


Obrázek 21: Symbol v knihovně s naznačenými svorkami

6.3.5. Odkazování mezi listy

Často se stává, že potřebujeme roztáhnout napříč všemi (nebo skupinu) listů vodič s napětím, signálový vodič, či potrubí a potřebujeme, aby jednak existovala vnitřní kontrola propojení (např. GND na GND) a zejména, aby se zobrazovala reference (viz Obrázek 22).

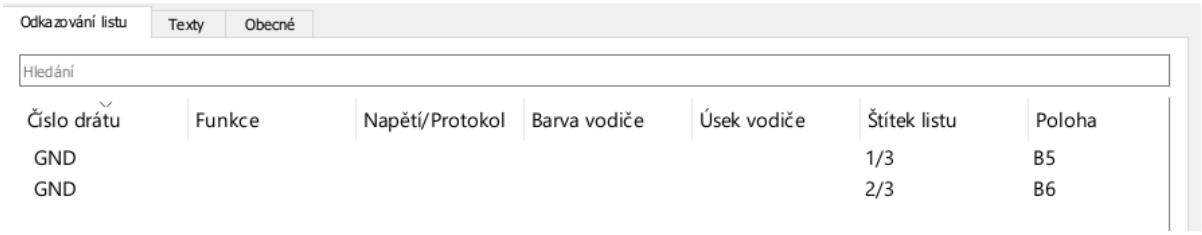
Toho docílíme vložením dvou speciálních prvků z knihovny Elektrotechnika/Odkazování listů – **Další list** a **Předchozí list**. Pokud tedy chceme propojit referencovat např. signál GND na listech 1 a 2, musíme na list 1 vložit prvek **Další list** a na list 2 vložit prvek **Předchozí list** a někam je připojit. Na obrázku je znázorněno také propojení na list 1 (zatím nevíme odkud, možná by se tam hodilo vložit pouze jednoduchou svorku) a propojení na list 3.



Obrázek 22: Referencování signálů

Dále je velice vhodné pojmenovat stejně oba vodiče (zde GND). Není to podmínkou, ale pokud chceme vodiče vzájemně propojit, pak různé pojmenování nedává smysl.

Po vytvoření prvků dvojklikem na jeden z nich otevřeme okno vlastností a vybereme kartu odkazování listu (Obrázek 23). Zde klikneme na požadovaný signál. Pokud jsme tedy otevřeli **Další list** na listu 1, vybereme signál GND na listu 2/3, klikneme pravým tlačítkem, vybereme **Spojit prvek** (signál změní barvu na zelenou) a potvrdíme pomocí **Použít**.

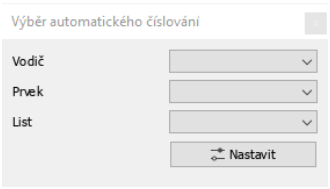


Obrázek 23 Seznam dostupných spojů pro spojení

Následně se objeví textová reference 2-B6 na listu 1 a 1-B9 na listu 2. Ta obsahuje vždy číslo listu a „souřadnici“ v rámci listu, na které prvek nalezneme.

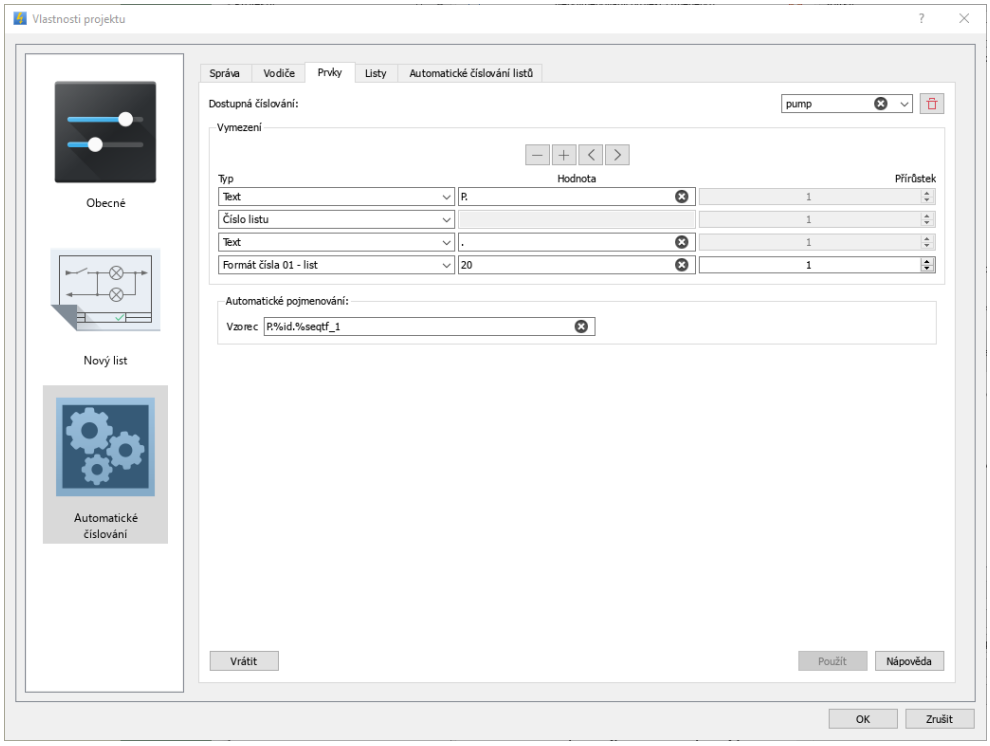
6.3.6. Číslování položek

Automatické číslování prvků lze nastavit pomocí okna „Výběr automatického číslování“ (Obrázek 24). Okno se ve výchozím rozložení nachází jako karta na pravém panelu. Číslování se provádí zvlášť podle kategorií. Tlačítko „**Nastavit“** otevírá okno, kde lze nastavit textový vzorec číslování.



Obrázek 24 Okno Výběr automatického číslování

V záložce „**Prvky**“ lze nastavit například vzorec číslování (Obrázek 25), který začíná písmenem P, první číslo značí číslo stránky a druhé číslo značí individuální číslo symbolu, které začíná od čísla 1 pro každou stránku.



Obrázek 25 Nastavení číslování symbolů

Pod zadávacími políčky se zobrazuje automaticky vygenerovaný textový vzorec, který udává, které proměnné symbolu se vypíšou do štítku symbolu. **Automatické číslování prvků však někdy přináší problémy, domníváme se proto, že v projektu naší velikosti není nutné a ani povinné jej používat.** Alternativní možností je rozkliknout každý vložený symbol zvlášť a zadat číslo do parametru „Štítek“ ručně.

7. Úkol 3: Volba vhodné instrumentace

Cílem cvičení je samostatně v jednotlivých skupinách vybrat vhodné akční členy a snímače fyzikálních veličin pro pasterizační jednotku.

7.1. Pokyny k vypracování:

1. Vytvořte seznam požadovaných akčních členů a snímačů na základě Vámi vytvořeného P+I diagramu.
2. Pro veškeré zvolené prvky shromážděte dokumentaci (Datasheety).
3. Pro všechny zvolené snímače shromážděte následující parametry:
    a. Rozsah
    b. Citlivost
    c. Rozlišení
    d. Provozní podmínky (mezní teploty, typ média, typ použití atd.)
    e. Chyba měření
    f. Mechanické vlastnosti
    g. Rozhraní
4. Pro všechny akční členy typu ventil shromážděte následující parametry:
    a. Mechanické vlastnosti
    b. Provozní podmínky (mezní teploty, typ média, typ použití atd.)
    c. Maximální průtok
    d. Doba přestavení (otevírání/zavírání)
    e. Způsob otevírání (NC, NO, servopohon, elektromotor)
5. Pro všechny akční členy typu elektromotor shromážděte následující parametry:
    a. Krouticí moment
    b. Jmenovité napětí
    c. Jmenovitý proud
    d. Jmenovité otáčky
    e. Provozní podmínky (mezní teploty, typ média, typ použití atd.)

7.2. Výstupy:

Požadovaným výstupem, který bude odevzdán do IS, je doplněná technická dokumentace o

1. Informace o výběru komponent pro realizaci hlavního toku materiálu (Vstupní, výstupní ventil s ovládáním, Teplotní, tlakový snímač, Spojitý a limitní (horní) snímač hladiny, motor míchadla s řízením a převodovku) - vypište požadavky, které jsou na každou komponentu kladené.
2. Pro každou vybranou komponentu uveďte přehledně v tabulce parametry, zejména ty, které považujete důležité pro konkrétní aplikaci. Odůvodněte výběr každé komponenty, porovnejte požadované parametry technologie a parametry vybraných prvků.
3. K odevzdání přiložte katalogové listy jednotlivých komponent (nebo "vyrobte" katalogový list tiskem informací z webové stránky do PDF). Katalogové listy vhodně pojmenujte (dodržujte tuto konvenci: ventil_vstup.pdf, servo_vstup.pdf, snimac_teplota.pdf, snimac_limit_hladina.pdf, motor_mixer.pdf, prevodovka_mixer.pdf, apod...) a při odevzdávání všechny zazipujte společně s hlavním dokumentem.

8. Úkol 4: Elektrotechnické schéma

Cílem cvičení je samostatně v jednotlivých skupinách vytvořit elektrické schéma pro řešenou technologii. ***Výstupy tohoto úkolu budou použity pro pozdější programování.***

8.1. Pokyny k vypracování:

1. Seznamte se s návrhem elektrotechnické dokumentace v nástroji QElectroTech. K seznámení využijte tutoriál, který je součástí této kapitoly.
2. Prostřednictvím návrhového software QElectroTech vytvořte elektrotechnickou dokumentaci k navrhované technologii.
3. Jako řídicí systém použijte programovatelné automaty, které se nacházejí v laboratořích.

8.2. Výstupy:

Požadovaným výstupem, který bude odevzdán do IS, je:

1. PDF soubor s elektrotechnickou dokumentací.
2. PDF s vyexportovanou tabulkou PLC vstupů/výstupů.

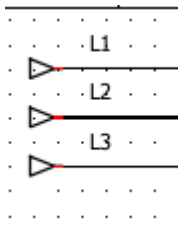
8.3. Tutoriál pro tvorbu elektrotechnických schémat v programu QElectroTech

Tato část navazuje na úvod do práce s QElectroTech popisovaný v kapitole 6.3. Pro tvorbu elektrotechnických schémat.

8.3.1. Knihovna pro elektrotechnické schéma a tvorba propojů

Po vytvoření nového listu v programu QElectroTech je možné vkládat symboly ze složky „Elektrotechnika“ v okně Sbírek. Prostředí pro kreslení P+I diagramů a elektrotechnických schémat je stejné. Tažení propojů mezi symboly je také obdobné.

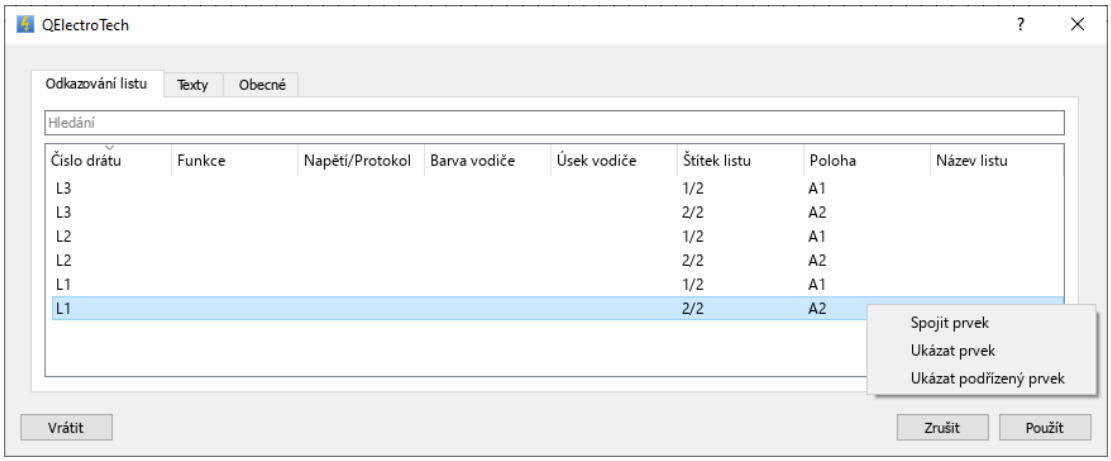
Při tvorbě silového elektrotechnického schématu je dobré začít od hlavního trojfázového přívodu. Pro lepší orientaci v projektu je vhodné vložit Symboly Odkazování listů (Obrázek 26). Bez nich by také nebylo možné zakreslit propoje.



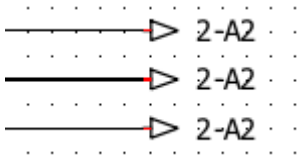
Obrázek 26 Odkaz na předchozí list

Pozor při párování odkazů, proces může být trochu neintuitivní. Nejdříve je nutné mít vytvořené alespoň dva listy na kterých jsou odkazy na předchozí a další list. Dvojklikem na odkaz se otevře okno se seznamem odkazů. Dvojklikem na položku seznamu se položka pouze zvýrazní ve schématu. Pro skutečné připojení je nutné kliknout pravým tlačítkem myši a vybrat možnost „spojit prvek“. Správné

propojení odkazů je na obrázku Obrázek 28. Všimněte si, že první číslo odkazuje na číslo listu a následuje kód za pomlčkou, který udává pozici.

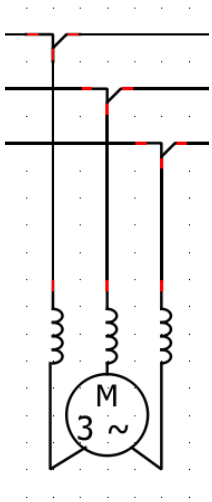


Obrázek 27 Párování odkazů na listy



Obrázek 28 Spárované odkazy na další list

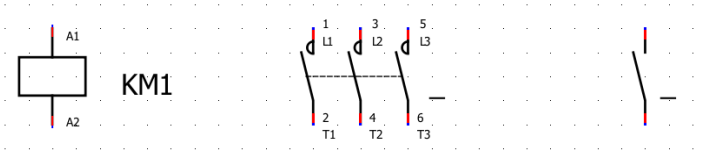
Připojení jednotlivých obvodů z hlavního přívodu realizujte pomocí komponentů z knihovny „Elektrotechnika/Víceżilový/Spojení“. Knihovna nabízí několik způsobů navazování/větvení spojů. Obrázek 29 ukazuje jednoduchý příklad jednoho takového spojení. Elektrotechnické schéma uvažuje i tyto spojení jako fyzický prvek, protože odráží nutnost použití fyzického komponentu ke spojení silových kabelů při instalaci (tímto se liší například od schématu plošného spoje).



Obrázek 29 Příklad propojení vodičů

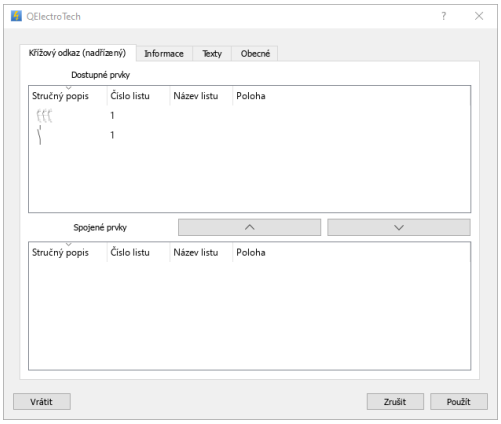
8.3.2. **Provázané symboly**

Elektro schéma obsahuje symboly, které lze seskupit pod jednu komponentu. Například stykač lze rozdělit na silové kontakty, pomocný kontakt a cívku (Obrázek 30).



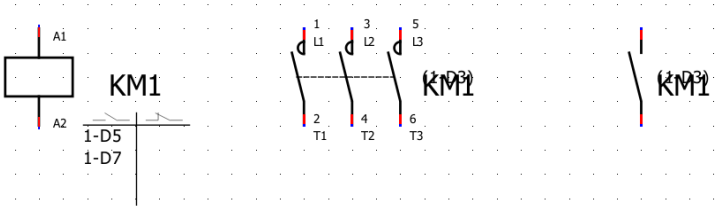
Obrázek 30 Sestavení stykače ze symbolů cívky, silových kontaktů a pomocného kontaktu

Symbol cívky je nadřízený a lze ho provázat s několika podřízenými symboly kontaktů stykače. Po rozkliknutí cívky dvojklikem se otevře okno (Obrázek 31), kde lze křížovými odkazy provázat symbol cívky provázat s podřízenými symboly. Odkaz je vytvořen klikem pravým tlačítkem na příslušný symbol a vybrat možnost „Spojit prvek“. Všimněte si karty „Informace“, kde se nachází parametry symbolu. Zde lze změnit parametr „Štítek“, který se propíše do provázaných symbolů.



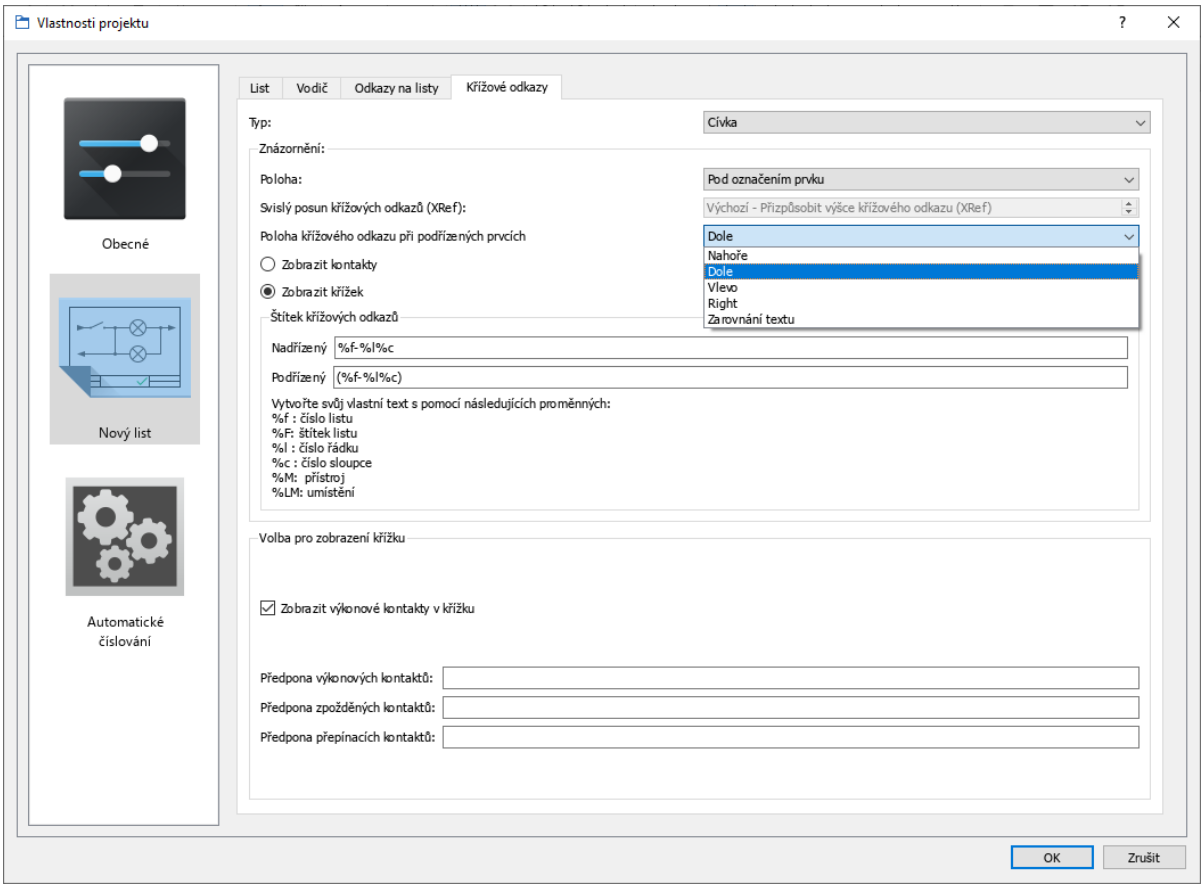
Obrázek 31 Provázání symbolů křížovým odkazem

Provázané symboly, které dohromady tvoří komponentu stykače, zobrazují informace o provázání (Obrázek 32).



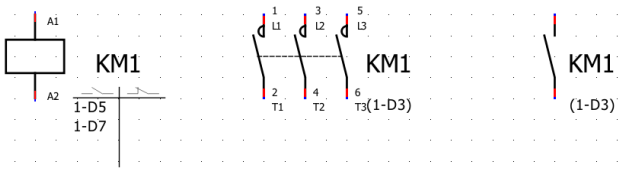
Obrázek 32 Provázané symboly

Lze si všimnout překrývání textů. Problém je možné vyřešit změnou nastavení. V horní levé části programu klikněte na „Projekt“, poté „Vlastnosti projektu“, čímž se otevře okno nastavení. Klikněte na symbol „Nový list“ a poté na kartu „Křížové odkazy“. Změňte nastavení „Poloha křížového odkazu při podřízených prvcích“ na polohu „Dole“ (Obrázek 33).



Obrázek 33 Změna nastavení polohy odkazů

Grafické vykreslení se tím opraví (Obrázek 34).



Obrázek 34 Opravená poloha křížových odkazů

9. Úkol 5: UML diagramy

Cílem cvičení je samostatně v jednotlivých skupinách vytvořit UML diagramy popisující funkci řídicího software. ***Výstupy tohoto úkolu budou použity pro pozdější programování.***

9.1. Pokyny k vypracování:

1. Prostřednictvím libovolného návrhového SW (např. volně dostupného draw.io) vytvořte tyto UML diagramy:
    a. Use-case diagram popisující, jaké funkce budou mít k dispozici uživatele ve SCADA vizualizaci (neregistrovaný, přihlášený, technolog), která bude sloužit pro ovládání procesu (manuální režim, automatický režim, konfigurace systému). Pro jeden z případů užití vytvořte specifikaci (viz přednáška).
    b. Sekvenční diagram, který popíše činnosti a výměnu informací mezi jednotlivými moduly (Batch kontrolér, řídicí moduly, moduly zařízení) při vykonávání fází procedury pasterizace (napouštění, vypouštění, ohřev s mícháním). Při modelování využijte také podmínky a cykly.
    c. Stavový diagram, který popíše činnost fáze napouštění. Uvažujte pouze stavy Idle, Running, Held
    ~~d.~~
    ~~technologického procesu – mixovacího tanku. Specifikujte a nakreslete třídy (názvy, atributy, metody) pro následující objekty:~~
        ~~i. Ventily~~
        ~~ii. Čerpadla~~
        ~~iii. Analogový snímač (např. teplota v tanku)~~
        ~~iv. Digitální snímač (např. hladina v tanku plovákem)~~
        ~~v. Mixovací tank (zahrnující uvedené snímače)~~
        ~~vi. Procesní buňka (zahrnující několik mixovacích tanků)~~
        ~~vii. Nezapomeňte na~~
            ~~1. agregace,~~
            ~~2. generalizace,~~
            ~~3. asociace, případně další typy vztahů mezi těmito třídami~~

9.2. Výstupy:

Požadovaným výstupem, který bude odevzdán do IS, je:

1. PDF soubor s uvedenými UML diagramy

10. Úkol 6: Implementace řídicího SW pro PLC

Cílem úkolu je samostatně v jednotlivých skupinách vytvořit PLC program, který je modulárně strukturován v souladu s normou S88 pro dávkovou výrobu.

10.1. Pokyny k vypracování

1. V inženýrském rámci TwinCAT vytvořte obslužný program pro navrhovanou technologii tak, aby splňovaly požadavky pro připojení k systému SkuBatch a dávkového řízení. Těmito požadavky jsou:
    a. Možnost řídit jednotlivá řízená zařízení (ventily, míchadla, ohřev) manuálně prostřednictvím vizualizace.
    b. Možnost aktivovat automatické řízení jednotlivých zařízení prostřednictvím fází.
    c. Možnost řídit fáze v PLC (napouštění, vypouštění, míchání, ohřev) v manuálním režimu prostřednictvím vizualizace.
    d. Možnost aktivovat automatické řízení jednotlivých fází v PLC prostřednictvím fází v MES systému **SkuBatch.**
2. Pro tvorbu programu využijte předpřipravenou šablonu, která se nachází v kartě předmětu.
3. Jako vstupy a výstupy použijte vstupy a výstupy softwarového simulátoru
4. Detailní požadavky na SW jsou umístěny v následujícím textu.

10.2. Požadovaný výstup

Požadovaným výstupem, který bude předveden vyučujícímu je:

1. Funkčnost jednotlivých řídicích modulů
2. Funkčnost jednotlivých modulů zařízení
3. Funkčnost rozhraní pro systém SkuBatch

10.3. Programová kostra

Cílem cvičení je ukázat studentům, jakým způsobem vytvořit PLC program tak, aby byl srozumitelný, udržovatelný a v souladu s danými požadavky. Za tímto účelem je v informačním systému v kartě předmětu připravena kostra programu, která tyto požadavky splňuje. Není bezpodmínečně nutné z této kostry vycházet, nicméně pokud se rozhodnete pro vlastní implementaci, ***musíte zachovat všechny výše uvedené požadavky.***

Předpřipravená kostra programu obsahuje HW konfiguraci, seznam obecných a vstupně výstupních proměnných (mapování vstupů a výstupů na simulátor) a strukturu objektů (funkce, datové typy, datové objekty).

10.3.1. Přehled kostry

Před začátkem práce je vhodné projít a zkontrolovat si jednotlivé části kostry programu:

- Povšimněte si, že každý z objektů je určitého datového typu a podívejte se, jak jsou pro jednotlivé skupiny objektu tyto typy opakovaně využívány.

- Rozbalte položku DUTs v PLC projektu a projděte si jednotlivé vytvořené datové typy. Všimněte si, jak jsou do sebe jednotlivé struktury zanořeny. Zorientujte se, které datové typy popisují řídicí moduly, a které pak fáze.

- Rozbalte položku POUs a otevřete si postupně bloky Main_Batch, FB_Tank, bloky řídicích modulů (pojmenovány podstatnými jmény - Heater), bloky fází (pojmenovány také podstatnými jmény, ale označujícími činnosti - Heating). Všechny bloky si pořádně prohlédněte. Pokud nebudete něčemu rozumět, zeptejte se!

- Věnujte alespoň půl hodiny tomu, že si projdete strukturu programu tak, abyste při následujícím čtení nemuseli vždy dlouho hledat požadované entity. Je doporučeno psát si poznámky.

10.3.2. Základní stavební bloky programu

Následující kapitola jednotlivě popisuje základní stavební bloky kostry programu.

10.3.2.1. MB_Connector

Programový blok, ve kterém je implementována výměna dat se simulátorem. Vyčtená data ze simulátoru jsou vkládána do globálních proměnných s prefixy **di_** a **ai_**, data, která budou zapisována do simulátoru jsou brána z globálních proměnných s prefixy **do_** a **ao_**.

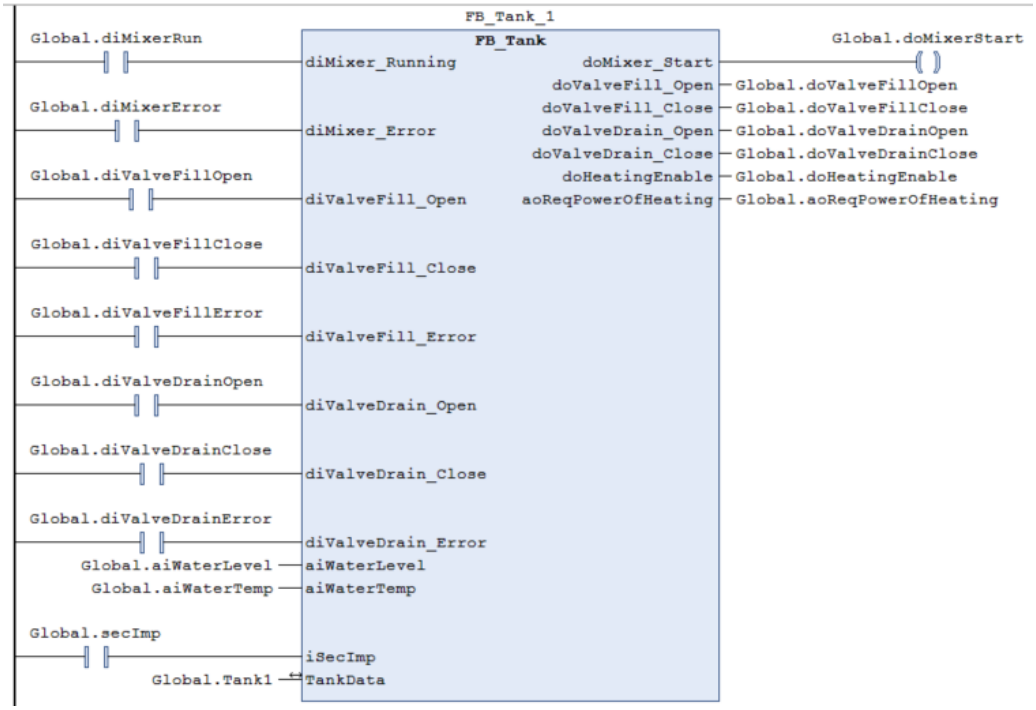
10.3.2.2. Main_Batch

Programový blok **Main_Batch**, jehož volání je zajištěno pomocí PLC Tasku **Default** volá cyklicky funkční blok typu ***FB_Tank*** instanciovaný jako **FB_Tank_1**. Tato funkce obsahuje ostatní potřebné funkce a funkční bloky pro kompletní funkcionalitu řízení jednoho tanku.

Povšimněte si, že vstup/výstupy do/z tanku jsou přímo binární či analogové proměnné z technologického procesu (mohly by to být přímo fyzické vstup/výstupy, nicméně ty my nemáme, místo toho jsou jimi bity získané parsováním komunikačního rámce).

Všimněte si také velmi důležité vstupně/výstupní proměnné **Tank1** (poslední na levé straně, označená dvojšipkou), která předává kompletní instanci globálního datového bloku Tank1 typu **TTank** do řídicí struktury.

```
PROGRAM Main_Batch
VAR
    FB_Tank_1 : FB_Tank;
END_VAR
```



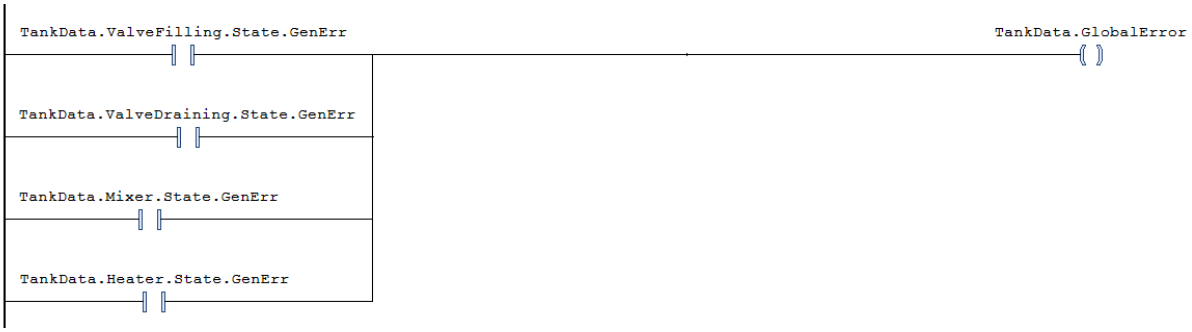
Obrázek 36 Volání funkčního bloku tank v programu Main_Batch

Instanciací funkčního bloku FB_Tank (např. FB_Tank_2) s novou instancí datového bloku a namapováním jiných vstupně/výstupních proměnných na nožičky této funkce (z další jednotky – fyzického tanku) je pak jednoduché vytvořit řízení většího množství tanků. Pro každou instanci funkčního bloku by bylo potřeba vyrobit a konfigurovat také instanci datového bloku.

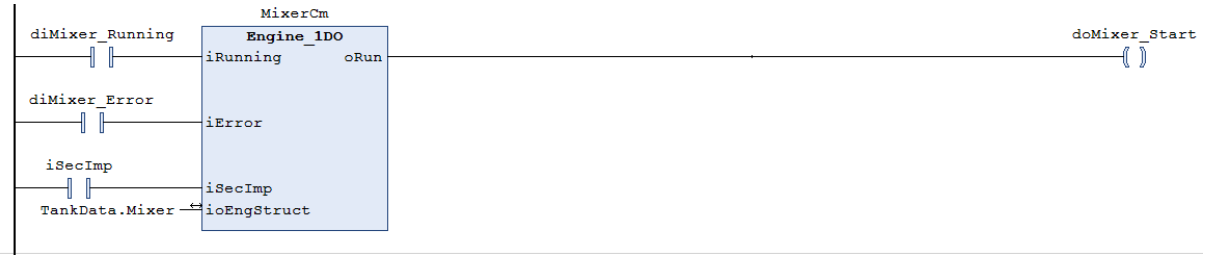
Funkční blok typu FB_Tank tedy obsahuje veškeré funkcionality pro řízení jednoho tanku, těmi jsou:

- řízení akčních členů prostřednictvím volání bloků modulů zařízení - 2x ventil, motor a topení,
- řízení fází - funkce pro napouštění, vypouštění, míchání a zahřívání,
- různé obslužné rutiny triviálního rázu.

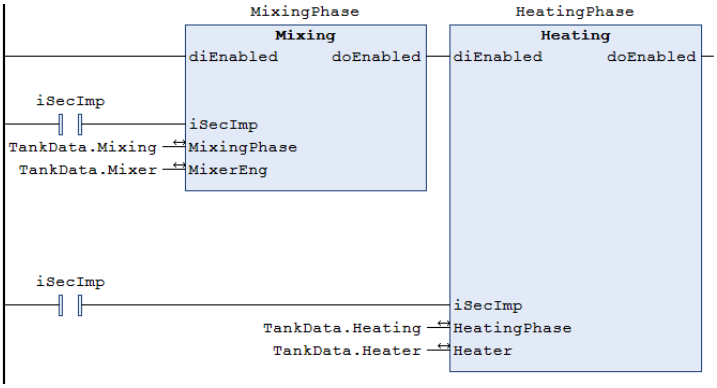
Ze vstupních proměnných ***FB_Tank*** se uvnitř předávají jednotlivé proměnné do daných funkcí a výstupy se přiřazují na výstup funkčního bloku ***FB_Tank***.



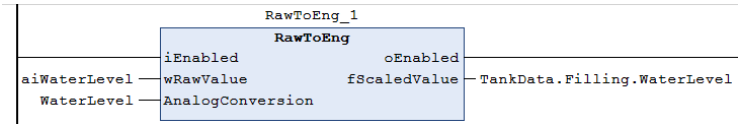
Obrázek 37 Logický součet dílčích chybových příznaků jednotlivých modulů zařízení udávající příznak GlobalError využívaný v rámci vizualizace i komunikace s nadřazeným systémem



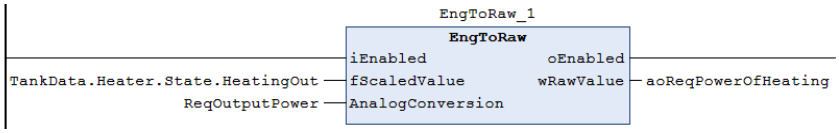
Obrázek 38 Volání funkčního bloku řídicího modulu mísení – vidíme zde přenesení vstupních parametrů bloku FB_Tank (jako je diMixer_Running) dovnitř bloku Mixer (jehož konkrétní instance je součástí datového bloku TankData předaného do FB_Tank zvnějšku). Stejným způsobem se volají další řídicí moduly



Obrázek 39 Volání funkčního bloku fáze míchání – opět vidíme jak propojení vstupních parametrů bloku FB_Tank s parametry instance funkčního bloku MixingPhase. Těmi jsou datový blok fáze a datový blok konkrétního vypouštěcího ventilu. Vstupní sekundový impulz slouží k realizaci různých funkcí v rámci funkčního bloku. Stejným způsobem jsou volány všechny fáze

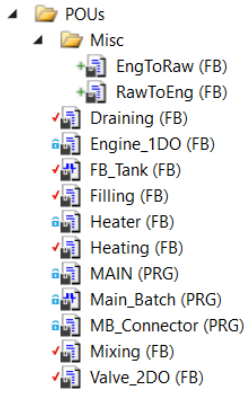


Obrázek 40 Funkční blok pro přepočet dat z převodníku (komunikace) na inženýrské hodnoty. Povšimněte si, že parametry převodu jsou schovány ve struktuře WaterLevel, která je součástí definice funkčního bloku tanku. V případě více tanků s různými potřebnými parametry bychom museli nastavení parametrů vyčlenit z FB_Tank na jiné místo.



Obrázek 41 Funkční blok pro přepočet dat z inženýrské hodnoty na data pro převodník (komunikaci). Opět platí to stejné, co bylo napsáno o opačné funkci RawToEng.

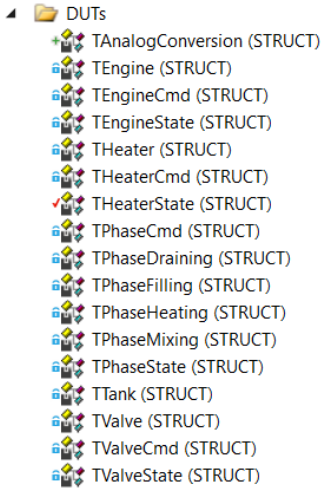
Všechny výše zmíněné funkční bloky, jak je možné po otevření vidět, jsou prázdné, nebo obsahují pouze komentáře, které popisují funkcionalitu jednotlivých částí kódu, které mají být Vámi dopsány.



Obrázek 42 Funkce pro jednotlivé celky

10.3.3. Uživatelsky definované datové typy kostry

Uživatelsky definované struktury - typy (DUT - user data types) jsou již v kostře programu předdefinovány. Vaše inovace je možná v DUT pro zařízení, nicméně pro splnění požadavků na implementaci plně dostačuje využití navrhnutých v předpřipraveném projektu. ***Z důvodu propojení dat pomocí názvu mezi PLC a MES systémem (pomocí protokolu OPC UA), není možné zasahovat do struktur DUT fází (tedy těch s prefixem TPhase).***



Obrázek 43 Typy datových struktur

Hierarchie datových typů je následující:

- Datové typy pro řídící moduly jsou Engine, Heater a Valve a obsahují struktury State (tato struktura kompletně popisuje stav řídicího modulu), Cmd (tato struktura kompletně specifikuje příkazy, pomocí kterých může být modul ovládán ve všech režimech) a triviální datové typy, popisující vstupní/výstupní parametry.

  Příkladem může být datový blok Heater, jehož vstupem je požadovaná aktuální hodnota teploty (iWaterTemp), výstupem pak akční zásah (oHeaterPower) a binární informace o spuštění topného elementu. Setpoint teploty (nastavuje se "ručně") a výkon topení v manuálním režimu (opět ruční nastavení) jsou schovány ve struktuře ioHeatStruct, která je do funkčního bloku předána z nadřazeného funkčního bloku FB_Tank. Tato struktura je vstupně/výstupním parametrem, tedy jakékoli změny budou udělány, promítnou se do "hlavní" instance (Tank1 v

programu Main_Batch). Vnitřní parametry, jako je např. pidMode slouží pro využití výhradně během volání funkčního bloku.

```
FUNCTION_BLOCK Heater
VAR_INPUT
    iWaterTemp : REAL;
END_VAR
VAR_OUTPUT
    oHeaterEnable : BOOL;
    oHeaterPower : REAL;
END_VAR
VAR_IN_OUT
    ioHeatStruct : THeater;
END_VAR

VAR
    pidParameters : ST_CTRL_PID_PARAMS;
END_VAR
```

- Datové typy pro fáze jsou označeny Phase… a obsahují rovněž struktury State, Cmd (Commands) a triviální datové typy sloužící pro nastavení parametrů.
- Datový typ tanku obsahuje 2 x UDT Valve (napouštěcí, vypouštěcí ventil), 1 x Engine (míchadlo) a 1 x Heater (ohřev). Dále obsahuje datový blok pro každou fázi (PhaseFilling, PhaseDraining, PhaseHeating, PhaseMixing).
- Odlišný **typ** tanku by mohl obsahovat jiné akční elementy a případně i fáze
- Stejný typ tanku - druhý tank - může využít celý datový typ bez modifikace

10.3.4. Popis koster řídicích modulů

Implementace řídicího modulu bude popsána na modulu Engine_1DO, který je využit pro řízení míchadla (a může být v budoucnu použit pro řízení jakéhokoli řízeného zařízení, které se ovládá přepínáním jedné své vstupní hodnoty - Zapnuto/Vypnuto a svůj chod signalizuje také jedním svým výstupem). V příslušném funkčním bloku se lze podívat na jeho vstupy a výstupy.

```
FUNCTION_BLOCK Engine_1DO
VAR_INPUT
    iRunning : BOOL;
    iError : BOOL;
    iSecImp : BOOL;
END_VAR
VAR_OUTPUT
    oRun : BOOL;
END_VAR
VAR_IN_OUT
    ioEngStruct : TEngine;
END_VAR
VAR
    secImpEdge: BOOL;
    tempStartupFail: BOOL;
END_VAR
```

Obrázek 44 Vstupy a výstupy funkčního bloku

Vstupními signály jsou:

- iRunning - zpětná vazba o tom, zda pohon běží (běžně se používá pouze zpětná vazba od stykače, proudové/napěťové relé, snímač otáček nebo zpětná vazba z FM).
- iError – signalizace chyby pohonu. Signalizace bývá z FM, tepelné ochrany pohonu, zařízení hlídající prosak vody do pohonu/oleje apod.

- iSecImp - vstup sekundového impulzu pro počítání motohodin. V demonstrační úloze budeme zobrazovat motominuty/motosekundy (záleží na Vás), v běžné praxi stačí pro obsluhu zobrazovat pouze motohodiny.

Výstup je zde jediný a slouží pro ovládání digitálního výstupu, kterým se spíná pohon (většinou stykač (pro malá zařízení relé), signál pro frekvenční měnič atp.).

Jako jediný InOut (data, která jsou definována jako výstup, ale je možné je číst a zapisovat/měnit je) je struktura ioEngStruct datového typu Engine. Jak bylo již zmíněno výše, datové struktury jsou předdefinovány a je vhodné se na ně podívat a ujasnit si význam jednotlivých proměnných. Uvnitř této struktury jsou mezi jednotlivými voláními bloku udržovány hodnoty všech jeho vnitřních stavů (proměnných)

Expression	Type	Value
iRunning	BOOL	FALSE
iError	BOOL	FALSE
iSecImp	BOOL	FALSE
oRun	BOOL	FALSE
ioEngStruct	REFERENCE TO...	
Cmd	TEngineCmd	
SetRT	DINT	<Dereferenc...
AutCtrl	BOOL	<Dereferenc...
ManCtrl	BOOL	<Dereferenc...
TurnOnMan	BOOL	<Dereferenc...
TurnOffMan	BOOL	<Dereferenc...
Req2RunAutCtrl	BOOL	<Dereferenc...
ResetSUF	BOOL	<Dereferenc...
State	TEngineState	
RT	DINT	<Dereferenc...
Run	BOOL	<Dereferenc...
RunOut	BOOL	<Dereferenc...
Err	BOOL	<Dereferenc...
StartUpFail	BOOL	<Dereferenc...
AutCtrl	BOOL	<Dereferenc...
ManCtrl	BOOL	<Dereferenc...
MayRun	BOOL	<Dereferenc...
MayAutoRun	BOOL	<Dereferenc...
GenErr	BOOL	<Dereferenc...
StateInt	INT	<Dereferenc...
StartupFailTime	INT	<Dereferenc...
Time2Start	INT	<Dereferenc...
secImpEdge	BOOL	FALSE
tempStartupFail	BOOL	FALSE

Obrázek 45 Paměťová struktura motoru

Každá struktura pro ovládání v kostře programu vypadá podobně jako ***DUT Engine***. Skládá se ze tří základních částí: příkazy (Cmd), stavy (State) a parametry (Param).

Jednotlivými stavy jsou:

- ***RT (Running Time)*** - motohodiny, datový typ Double Integer dostačuje pro počítání motohodin, z vlastní zkušenosti (TB) integer nestačí)
- ***Run*** - bool, signalizace chodu pohonu, pokud vstupní signál signalizuje chod, tento bit má hodnotu true.
- ***RunOut*** - bool, signál slouží pro detekci toho, zda se funkční blok snaží sepnout výstup. Pokud bychom ve funkci chtěli číst výstup, bude to možné, ale taková část kódu bude označena jako nesyntakticky správně (program fungovat bude, ale tento řádek bude podtrhnutý žlutě). Hrozí

zde nebezpečí, že hodnotu přečteme špatně - sepneme na jednom řádku výstup a v následujícím ho čteme, bohužel zde nebude výstup ještě sepnutý, záleží na tom, kdy PLC své výstupy aktualizuje. Zpravidla jako defaultní nastavení to bývá na konci cyklu smyčky, je možné jej dělat i periodicky, ale také (při použití distribuovaných I/O) to může být synchronně v závislosti na použité komunikaci nebo vzorkovací periodě sběrnice I/O karet.

- ***Err*** - Stejně jako signál Run, jedná se o přepis vstupní proměnné iErr.
- ***StartUpFail*** - viz požadavky na řízení zařízení *Nerozběh*. Pokud sepneme pohon do určité doby, očekáváme informaci o chodu. S tím souvisí parametr Time2Start v UDT Engine a také nadefinovaný časovač SUF_Timer ve funkčním bloku.
- ***AutCtrl/ManCtrl*** (automatické/ manuální ovládání), pohon musí být v jednom z těchto dvou stavů, pokud je v režimu manuálním, je možné ho ovládat povely z HMI/SCADA, pomocí Cmd TurnOnMan, TurnOffMan, pokud je v automatickém režimu, pohon je ovládán z PLC pomocí signálu Req2RunAutCtrl.
- ***MayRun/MayAutoRun*** - signalizace vhodná pro prvotní kontrolu zařízení. Pokud je sepnutý MayRun, pak je pohon možné ovládat v manuálním režimu a nemá žádnou poruchu, pokud je aktivní MayAutoRun, pohon nemá žádnou poruchu a je možné jej ovládat
- ***GenErr - (General error)*** většina zařízení má více typů poruchy, signály Err a StartUpFail jsou vhodné pro zápis do alarmového deníku, ale pro signalizaci pohonu červeně ve vizualizaci je vhodné mít sdružený bit poruchy. Poruchu GenErr není potřeba kvitovat zvlášť, zmizí, pokud nebude aktivní žádná z "podřízených" poruch.

Následující seznam popisuje jednotlivé příkazy. Tyto příkazy by měly být vždy po zpracování navráceny do klidové úrovně.

- ***SetRT*** - nastavení motohodin. Jak bylo zmíněno výše, ne vždy je pohon nový, nebo s výměnou řízení přichází i výměna pohonů, je tedy nutné mít možnost nastavovat motohodiny.
- ***AutCtrl/ManCtrl*** - přepínání mezi automatickým a manuálním režimem. V případě přijetí obou signálů zároveň by měl být, z hlediska bezpečnosti upřednostněn manuální režim.
- ***TurnOnMan/TurnOffMan*** - jak bylo již zmíněno, signály slouží pro ovládání pohonu v manuálním režimu, pokud je pohon v automatu, i přesto se signál musí vyhodnotit, akce se nesmí vykonat a signál se vrátí do klidového stavu.
- ***Req2RunAutCtrl*** - signál, kterým PLC ovládá modul zařízení, pokud je v automatu. Tzn. řízení přichází z algoritmu ovládající tento modul zařízení nebo z nadřazeného řízení.
- ***ResetSUF*** - resetování poruchy nerozběhu. Pokud se porucha nerozběhu vygeneruje, zařízení bude v poruchovém stavu. Je na obsluze/operátorovi/údržbě, aby zařízení zkontrolovala a pak tuto poruchu resetovala. (Volitelně) je možné přidání počtu automatických pokusů a intervalu pokusů opakovat pokus rozběhnout pohon automaticky, a až následně bude porucha trvalá a bude třeba vyčkat na obsluhu. Toto je vhodné implementovat např. u ventilů nebo čerpadel apod., kdy je možné, že při delším nepoužívání nebo u starších pohonů kde mohou být mechanické části zatuhlé. Pak se zařízení může chovat následovně: čerpadlo je přicpané nečistotami, modul zařízení se pokusí pohon spustit, po 10s nepřijde zpětná vazba o chodu a pohon se vypne, nečistoty však postoupily o kousek dál. Čerpadlo využívá protékající vodu jako vlastní chlazení motoru, ale na sucho nějaký minimální čas vydrží běžet. Následuje interval, kdy se čeká na automatické opakování (motor chladne - neprotékala voda) a následně se modul zařízení (pokud je stále požadavek na chod) pokusí pohon opětovně sepnout.

10.3.5. Implementace funkcionalit modulů zařízení

Ve funkčním bloku Eng_1DO je připraveno několik částí (regionů). Každá z nich je pojmenovaná a u každé je napsán komentář. Do každé z těchto částí napište pouze kód, který jí přísluší podle konkrétního komentáře.

Tabulka 2 Popis jednotlivých regionů modulu zařízení Eng_1DO

Region	Popis
RunningTime	Použijte vstup iSecImp k navyšování počtu motosekund. Tento vstup má periodu 1 sekundu se střídou 50:50. Buď využijte funkční blok R_TRIG k detekci náběžné hrany, nebo vymyslete mechanizmus jinak. V každém případě budete muset doplnit nějakou "paměťovou buňku" do struktury TEngine. Pokud je hodnota SetRT větší nebo rovna 0, nastavte počet motosekund na tuto hodnotu a hodnotu SetRT nastavte na hodnotu -1. Takto je realizováno uživatelské nastavení hodnoty motosekund.
RunState	Zda zařízení skutečně běží, není možné určit pouze tím, že jsme zadali příkaz k jeho sepnutí. Do stavu Run tedy nastavte hodnotu vyčtenou ze zpětné vazby od zařízení (iRunning).
Manual/Automat	Na základě požadavku na automatický (příkaz Cmd.AutCtrl) nebo manuální režim se změní stavové bity automatického (State.AutCtrl) a manuálního režimu. Nezapomeňte vždy požadavek vyresetovat! Zde rovněž proveďte inizializaci – pokud nebude aktivní automatický ani manuální režim, preferujte manuální.
MR/MAR	Nastavení pomocných příznaků MayRun a MayAutoRun. MayRun je aktivní v případě, že je aktivní manuální režim a není nastaven příznak poruchy. MayAutoRun funguje analogicky pro automatický režim.
ManCtrl	Zpracování příkazů Cmd.TurnOnMan a Cmd.TurnOffMan. Pokud je aktivní příkaz TurnOnMan a současně je aktivní pomocný příznak pro manuální režim, aktivujte stav RunOut. Analogicky postupujte pro příkaz TurnOffMan. Nezapomeňte oba příkazy vynulovat!
AutCtrl	Pokud je aktivní automatický režim a je aktivní příznak MayAutoRun, otestujte příkaz Req2RunAutCtrl. Na základě jeho hodnoty nastavte opět stav RunOut.
ErrStates	1. Vytvořte funkční blok TON časovače (opět je potřeba přidat jeho definici do datového typu TEngine - časovač potřebuje paměť, která se bude nacházet v instanci struktury celého tanku). Po sepnutí výstupu počítá dobu Tim2Start. Pokud do této doby není aktivní zpětná vazba od zařízení, vygeneruje se příznak (stav) StartUpFail.

	2. Dále pokud zařízení aktivovalo svůj výstup iError (vstup do PLC), nastavte příznak (stav) Err. 3. Logickým součtem obou výše popsaných chybových příznaků je příznak GenErr. 4. Příkaz ResetSUF resetuje příznak StartUpFail. Nezapomeňte vyresetovat i tento příkaz! 5. Pokud je aktivní příznak GenErr, je nutné vypnout výstup modulu (deaktivovat zařízení - přepnout do bezpečného stavu) - RunOut. **Za blok ErrStates vložte příkaz, který zrcadlí hodnotu stavu RunOut z datového bloku na výstup #oRun. Hodnoty jsou duplikovány, protože zatímco RunOut je možné i číst a je využívána ve vizualizaci, výstup oRun je připojen přes výstup funkčního bloku tanku přímo na fyzický výstup PLC a ovládá zařízení.**
StatusInt	Pro potřeby vizualizace (přebarvování stavových symbolů) je nutné nastavit hodnotu int proměnné State.StateInt následovně: 1. Pokud není GenErr a modul je spuštěn (ve stavu Run), přiřaďte konstantu #Running (0) 2. Pokud není GenErr a modul je zastaven, přiřaďte konstantu #Stop (1) 3. Pokud je GenErr, nastavte na základě stavu Run konstanty #RunningAndErr (2) nebo pouze #Err (3).

Tabulka 3 Popis jednotlivých regionů modulu zařízení ventilu

Region	Popis
RunningTime	Tento region se zde nevyskytuje, tedy nepočítáme motohodiny servomotoru manipulujícího s ventilem.
RunState	Zda je ventil otevřen či uzavřen není možné určit pouze tím, že jsme zadali příslušný příkaz. Do stavu Open/Closed tedy nastavte hodnotu vyčtenou ze zpětné vazby od zařízení (iOpen, ...).
Manual/Automat	Funkcionalita je totožná s modulem ENG_1DO
MR/MAR	Funkcionalita je totožná s modulem ENG_1DO
ManCtrl	Zpracování příkazů Cmd.OpenMan, Cmd.CloseMan a Cmd.StopMan. Pokud je aktivní příkaz OpenMan a současně je aktivní pomocný příznak pro manuální režim, aktivujte stav OpenOut a deaktivujte stav CloseOut.

	Analogicky postupujte pro příkaz CloseMan. Příkaz StopMan pak deaktivuje oba stavy OpenOut i CloseOut. Nezapomeňte všechny příkazy nakonec vynulovat!
AutCtrl	Pokud je aktivní automatický režim a je aktivní příznak MayAutoRun: 1. Otestujte, zda je nastaven příkaz Req2OpenAut a deaktivován příkaz Req2CloseAut. Pokud je podmínka splněna, aktivujte OpenOut a deaktivujte CloseOut. 2. Pokud jsou příkazy Req2OpenAut a Req2CloseAut prohozeny, přepněte stavy OpenOut a CloseOut obráceně. 3. Pokud není aktivní ani jeden z příznaků Req2OpenAut a Req2CloseAut, deaktivujte oba stavy.
ErrStates	1. Definujte dva TON časovače analogicky, jako v bloku Engine_1DO (a typu TEngine). První: Pokud je sepnut OpenOut a rozepnut CloseOut, počítá dobu Time2Open. Druhý: při obráceném smyslu stavů počítá dobu Time2Close. 2. Pokud do této doby není aktivní zpětná vazba od zařízení, vygeneruje se příznak (stav) StartUpFail. 3. Logický součet výstupů (Q) těchto časovačů dává hodnotu StartUpFail. 4. Dále pokud zařízení aktivovalo svůj výstup iError (vstup do PLC), nastavte příznak (stav) Err. 5. Logickým součtem obou výše popsaných chybových příznaků je příznak GenErr. 6. Příkaz ResetSUF resetuje příznak StartUpFail. Nezapomeňte vyresetovat i tento příkaz! 7. Pokud je aktivní příznak GenErr, je nutné vypnout výstupy modulu (deaktivovat zařízení - přepnout do bezpečného stavu) - OpenOut a CloseOut. Za blok ErrStates vložte příkaz, který zrcadlí hodnotu stavů OpenOut a CloseOut z datového bloku na výstup oOpen a oClose. Hodnoty jsou duplikovány, protože zatímco OpenOut a CloseOut je možné i číst a je využívána ve vizualizaci, výstupy oOpen a oClose jsou připojeny přes výstup funkčního bloku tanku přímo na fyzické výstupy PLC a ovládají zařízení.
StatusInt	Opět pro potřeby vizualizace je nutné nastavit int hodnotu StColor. Vzhledem k množství možných hodnot máte tento blok předvyplněn.

Pro ventily (Val_2DO) budou existovat dva výstupy - pokyn pro zavření a pokyn pro otevření. Tyto dva výstupy by neměly nikdy být sepnuty zároveň. Struktura funkčního bloku je velice podobná, nezapomeňte vždy změnit zejména název interního datového bloku na iVal_Struct.

Pro ohřev (Heater) budeme mít opět pouze jeden číslicový výstup a jednu zpětnou vazbu ze zařízení (můžete tedy vycházet z bloku Eng_1DO). Navíc však přibude analogový výstup nastavovaný PID regulátorem. V bloku Heater máte veškeré použití PID regulátoru předvyplněno, seznamte se se syntaxí.

10.3.6. Test funkčnosti modulu

Po implementaci kódu funkčního bloku Engine_1DO, je nutné otestovat funkčnost. V první fázi je vhodné otevřít si datový blok, monitorovat stavy a nastavovat příkazy. Vepsáním nové hodnoty do sloupce **Prepared value** a stiskem ikony  dojde k jejímu zapsání

TwinCAT_Project1.Untitled1.Main_Batch.FB_Tank_1.MixerCm

Expression	Type	Value	Prepared value
iRunning	BOOL	FALSE	
iError	BOOL	FALSE	
iSecImp	BOOL	TRUE	
oRun	BOOL	FALSE	
ioEngStruct	REFERENCE TO...		
Cmd	TEngineCmd		
SetRT	DINT	-1	
AutCtrl	BOOL	FALSE	
ManCtrl	BOOL	FALSE	TRUE
TurnOnMan	BOOL	FALSE	
TurnOffMan	BOOL	FALSE	
Req2RunAutCtrl	BOOL	FALSE	
ResetSUF	BOOL	FALSE	
State	TEngineState		
RT	DINT	315	
Run	BOOL	FALSE	
RunOut	BOOL	FALSE	
Err	BOOL	FALSE	
StartUpFail	BOOL	FALSE	
AutCtrl	BOOL	TRUE	
ManCtrl	BOOL	FALSE	
MayRun	BOOL	FALSE	
MayAutoRun	BOOL	TRUE	
GenErr	BOOL	FALSE	
StateInt	INT	1	
StartupFailTime	INT	0	
Time2Start	INT	0	
secImpEdge	BOOL	FALSE	
tempStartupFail	BOOL	FALSE	

Obrázek 46 Testování modulů zařízení

Během testování samozřejmě musíte mít spuštěn simulátor technologie. Dejte pozor, jestli komunikace se simulátorem funguje (zjistíte prozkoumáním hodnot stavů Modbus TCP komunikace).

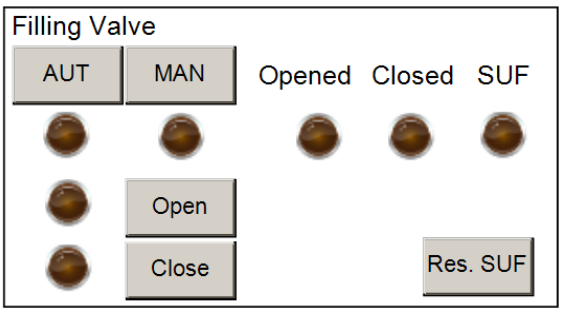
10.3.7. Připojení vizualizace k modulu

Nejsnáze lze zkontrolovat funkčnost modulu ve chvíli, kdy je připojen k vizualizaci. Tu si však napřed musíme vytvořit.

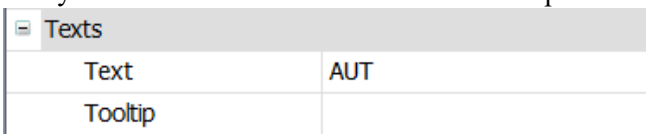
1. Klepneme pravým tlačítkem na záložku **VISUs** PLC projektu a zvolíme Add - Visualisation. Pokud jsme právě vytvořili naši první obrazovku vizualizace, vloží se projektu také položka

Visualisation Manager, pomocí které je možné nastavovat parametry vizualizace (např. startovací obrazovku).

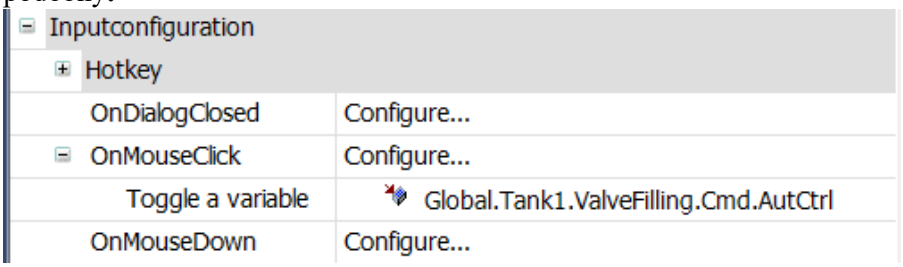
2. Na pracovní plochu naskládáme potřebné vizualizační prvky výběrem z toolboxu. Můžeme použít např. Rectangle - rámeček, Label - statický text, Lamp - kontrolku, Button - tlačítko. Návrh vizualizace pro jeden modul řízení může vypadat např. takto:



3. Texty tlačítek můžeme změnit v panelu **Properties** ve skupině **Texts:**

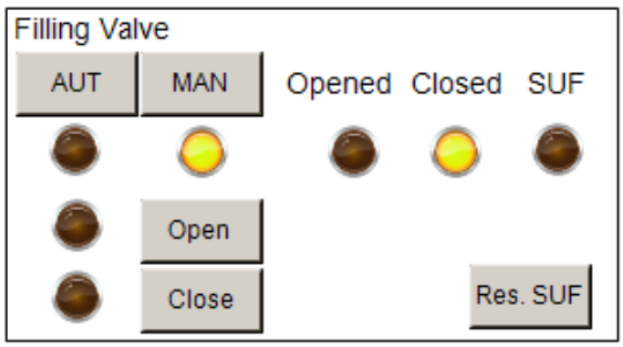


4. Chování tlačítek pak ve skupině **Inputconfiguration**, kde pro událost OnMouseClick zvolíme chování **Toggle Variable** a vybereme příslušnou proměnnou ze seznamu. Výsledek bude podobný**:**



5. Proměnnou navázanou na indikátor pak změníme pomocí vlastnosti **Variable**:

6. Po spuštění programu pak můžeme přejít na stránku s vizualizací a můžeme nyní ovládat chování řídicího modulu napouštěcího ventilu pomocí aktivních prvků:



7. Můžete vizualizovat také průběh komunikace s technologií přes Modbus TCP. Vyhnete se tím během ladění nepříjemnostem…

Kvalita vizualizace nebude předmětem hodnocení, je tedy zcela na Vás, jak ji pojmete. Pomocí vizualizace však budete při obhajobě prezentovat funkčnost Vašeho programu, tedy pokuste se vizualizaci udělat tak, aby toto bylo možné a nepřipravili jste se zbytečně o body.

10.3.8. Implementace Fáze

K tomuto bodu nepřistupujte, pokud nemáte dokončeny, otestovány a připojeny k vizualizaci všechny moduly zařízení.

Technologický proces probíhající v tanku se rozdělí do čtyř fází:

- Fáze napouštění
- Fáze vypouštění
- Fáze míchání
- Fáze ohřevu

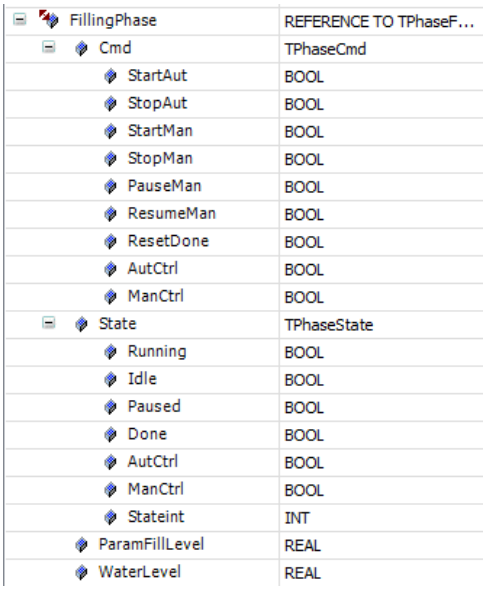
Stejně jako pro moduly zařízení, tak i pro fáze jsou již přichystány DUT a funkční bloky, které je potřeba doprogramovat.

Je nutné si uvědomit hierarchii řízení řízeného zařízení - řízené zařízení je možno, v našem případě, ovládat na nejnižší úrovni pomocí modulu zařízení - pokud je zařízení v manuálním režimu, je možné jej ovládat pomocí povelů zapnout a vypnout, přepne-li se modul zařízení do automatického režimu, znamená to, že řízené zařízení je možné ovládat nadřazeným řízením, v našem případě se jedná o fázi. Fáze v tomto režimu získává pozici subfunkce, která má své povely a stavy. Každou fázi (jak již bylo zmíněno výše) lze také přepínat mezi automatickým a manuálním režimem. Tzn., že pokud je fáze v manuálním režimu, je možné ji zapnout nebo vypnout z operátorského panelu pomocí příkazů zapnout nebo vypnout fázi. Fáze může obsahovat alespoň jeden modul zařízení, ale pro složitější fáze (ne v této demonstrační úloze) může obsahovat i několik modulů zařízení.

Rozdíl mezi spouštěním modulu zařízení a fáze vychází z principu požadavků na fázi - každá fáze má definované ukončení - pokud máme míchadlo, může se např. jednat o vypršení doby chodu míchadla, pro zahřívání dosažení požadovaného setpointu teploty apod. Naopak při manuálním ovládání modulu zařízení je vypnutí vždy na operátorovi či obsluze, která je v tomto režimu za chování systému zodpovědná.

Stejně jako u modulů zařízení je nutné podívat se na vstupy/výstupy jednotlivých funkčních bloků a zjistit význam jednotlivých proměnných v předdefinovaném DUT.

DUT pro fáze není možné měnit ani nijak upravovat - jakýkoli zásah způsobí nekompatibilitu s MES systémem SkuMES.



FillingPhase	REFERENCE TO TPhaseF...
Cmd	TPhaseCmd
StartAut	BOOL
StopAut	BOOL
StartMan	BOOL
StopMan	BOOL
PauseMan	BOOL
ResumeMan	BOOL
ResetDone	BOOL
AutCtrl	BOOL
ManCtrl	BOOL
State	TPhaseState
Running	BOOL
Idle	BOOL
Paused	BOOL
Done	BOOL
AutCtrl	BOOL
ManCtrl	BOOL
Stateint	INT
ParamFillLevel	REAL
WaterLevel	REAL

Obrázek 50 DUT fáze napouštění

Na Obr. 50 je definována struktura pro fázi napouštění. Význam jednotlivých proměnných je následující:

Příkazy (Cmd):

- ***StartAut*** – pokud je fáze v automatickém režimu MES systém pomocí této proměnné odstartuje fázi (přepne ji do stavu Running).
- ***StopAut*** – v případě poruchy systému, nebo příkazu procesního inženýra na zastavení fáze (dávky) se tímto signálem fáze ukončí (přepne se do stavu Idle).
- ***Start/StopMan*** - stejně jako u předchozích dvou signálu, ale funguje pouze, pokud je fáze v režimu manuálním.
- Nádstavba proti automatickému režimu je, že v manuálním režimu se objevují povely PauseMan a ResumeMan, z důvodu testování fází pro operátora/procesního inženýra. V důsledku to znamená např., že se spustí fáze míchání, po několika vteřinách se proces míchání pozastaví (přepne do stavu Paused), míchadlo je vypnuté, je zkontrolován obsah v tanku a poté se stiskne ResumeMan a fáze pokračuje dále po dobu zbývajícího času.
- ManCtrl/AutCtrl - přepínání mezi režimy manuál a automat (stejně jako u modulu zařízení).
- Reset Done - signál, který používá MES systém pro vyresetování stavu Done.

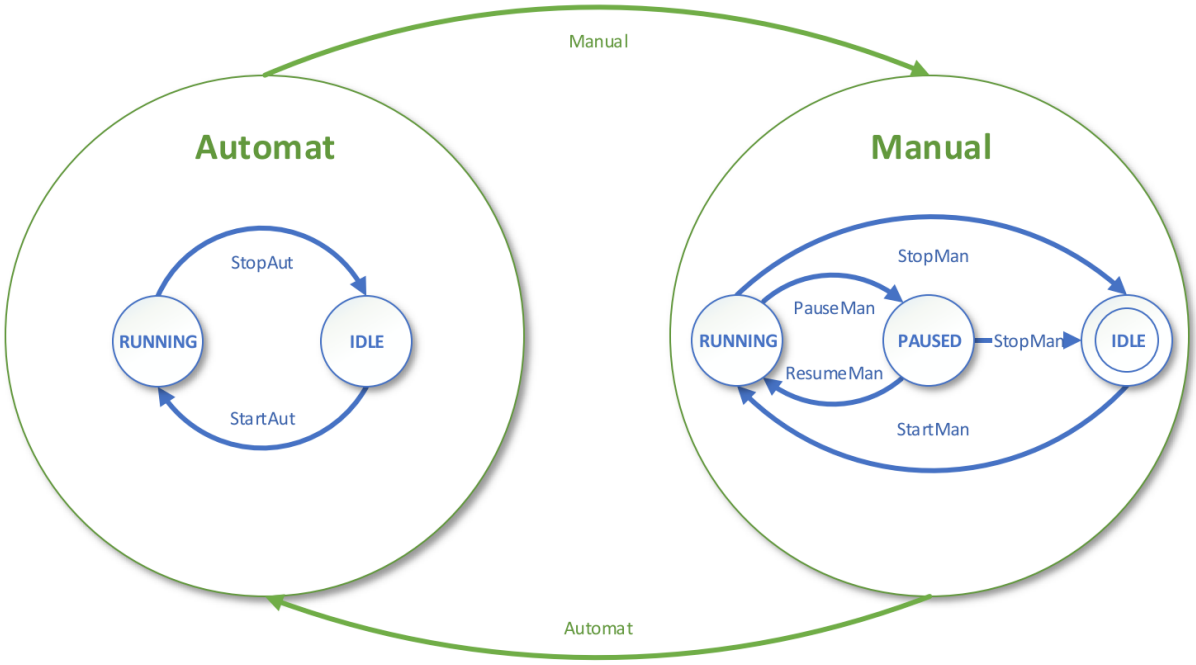
Stavy (State):

- Running/Idle/Paused - fáze je v daném režimu.
- Done - signál který se vygeneruje po dokončení fáze. Signalizace pro MES o tom, že byla fáze dokončena.
- AutCtrl/ManCtrl - signalizace o tom v jakém režimu je fáze.
- StateInt - číselná hodnota stavu, pro jednodušší zobrazení ve vizualizaci.

Parametry (Param) :

- FillLevel - slouží pro definování ukončovací podmínky fáze. Napouštění probíhá, dokud hladina nedosáhne požadovaného setpointu.

Fáze také musí znát aktuální hodnotu hladiny, která se fázi vkládá pomocí proměnné WaterLevel – tato hodnota je porovnávána s FillLevel a na základě splnění podmínky se fáze ukončí.



Obrázek 51 Stavy a povely fáze

Nyní přichází čas na programování. Dejte se do něj.

Region	Popis
Aut/Man	V rámci tohoto regionu se na základě příkazů AutCtrl a ManCtrl nastaví příslušný stav fáze napouštění. Dále pak zde ošetříme případ, kdy není aktivní ani jeden stavový příznak (AutCtrl či ManCtrl). V takovém případě nastavíme manuální režim. Všimněte si, že příkaz pro nastavení automatického režimu je Cmd.AutCtrl, kdežto příznak (stav) aktivity je State.AutCtrl.
ManCtrl	Pokud je fáze v manuálním režimu, zpracovávají se v tomto regionu příkazy pro změnu stavu – StartMan, StopMan, PauseMan, ResumeMan. Na základě těchto příkazů se přepíná stav automatu fáze mezi stavy Running, Idle, Paused. Tato sekce pouze definuje přechody automatu, nezabývá se činnostmi, které má fáze vykonávat v jednotlivých stavech.

	Pozn.: Při zpracování každého příkazy proveďte reset příznaku Done. Po zpracování konkrétního příkazu jej resetujte.
AutCtrl	Pokud je fáze v automatickém režimu, zpracovávají se v tomto regionu příkazy StartAut a StopAut. Příkazy opět realizují přechody stavového automatu fáze. Opět je po zpracování resetujte.
Done	Zde budeme realizovat 1) Testování, zda je splněna ukončovací podmínka fáze (v případě napouštění tedy, že je hladina vody vyšší nebo rovna žádané hladině). Pokud ano, nastavíme příznaky StopAut a StopMan, které zabezpečí korektní ukončení fáze. Dále pak nastavíme příkaz Done, který indikuje dokončení fáze nadřazenému systému. 2) Obsluhu příkazu ResetDone. Pokud bude nastaven, vyresetujeme příznak Done. Nezapomeneme opět resetovat i příkaz ResetDone.
States	V tomto regionu budeme realizovat vlastní činnost fáze. 1) Pokud fáze běží, nastavíme modulu zařízení příkaz Req2OpenAut (a pro pořádek mu vymažeme příkaz Req2CloseAut). 2) Pokud fáze neběží, provedeme opak.
„“	V poslední části, která není (ale může být) obalena regionem provedeme 1) Nastavení defaultního stavu (Idle) v případě, že není aktivní ani jeden ze stavových příznaků 2) Nastavení proměnné Stateint na hodnoty #Idle, #Running či #Pause podle aktuálního stavu. Tato proměnná existuje kvůli efektivnějšímu zpracování vizualizace.

10.3.9. Testujeme fázi napouštění

Stejně jako u modulů zařízení, tak i fází je třeba otestovat funkčnosti. Pro začátek je možné testovat z DB posíláním povelů a sledováním chování řízených zařízení.

Funkčnost fáze demonstrujte vyučujícímu.

10.3.10. Vytvoříme vizualizaci

Tagy potřebné na propojení PLC s HMI jsou opět předdefinované, je však třeba přidat funkcionalitu tlačítkám, zobrazovacím okénkům apod.

Funkčnost vizualizace demonstrujte vyučujícímu.

10.3.11. Doprogramujeme ostatní fáze

Fáze pro demonstrační úlohu jsou 4.

Pro fázi zahřívání je zajímavý region Done:

Done	V tomto regionu je nutno testovat jako ukončovací podmínku nejen dosažení žádané hodnoty, ale také dle zadání, zda hodnota za posledních 10 sekund nevystoupila z intervalu ±1 °C. K tomu je vhodné využít sekundový vstup, který je přiveden na vstup každé fáze (i každého modulu zařízení). V případě, kdy je fáze spuštěná, je nutné testovat, zda se skutečná hodnota pohybuje v žádaném rozmezí. V případě, že ano a současně je aktivní sekundový vstup (vždy pouze 1x za sekundu), přičte se do interní proměnné fTime hodnota 1. Pokud tato proměnná nabyde hodnoty >= 10, je možné fázi ukončit. Pokud ale v průběhu počítání žádaná hodnota opustí definovaný interval, je nutné fTime vynulovat. Stejně tak ji vynulujte vždy, pokud fáze neběží.

10.3.12. Testujeme

…

10.3.13. Vytvoříme vizualizaci

Bude doplněno, nicméně vizualizace fází může vypadat obdobně, jako vizualizace řídicích modulů, jen trochu jinými tlačítky a indikátory, odpovídající Cmd a State, případně analogovými hodnotami (teplota, výška hladiny, výkon topení).

10.4. Automatické spouštění fází stavovým automatem v PLC

Než se vše spojí s MES systémem, je možné zkusit si naprogramovat výrobní proces v PLC. V důsledku to znamená, že vytvoříte stavový automat ve strukturovaném textu, který bude jednotlivé fáze parametrizovat spouštět v definovaném pořadí.

```
CASE State OF
    1:  Global.Tank1.Filling.ParamFillLevel := 1.8;
        Global.Tank1.Filling.Cmd.ManCtrl := TRUE;
        Global.Tank1.Filling.Cmd.StartMan:= TRUE;
        IF Global.Tank1.Filling.State.Done = TRUE THEN
            State := 2;
        END_IF
    2: //...
END_CASE
```

Napustit, míchat + topit, vypustit. Pro pokročilejší – mělo by být možné definovat množství dávek a pak celý proces spustit a vyrobit předdefinovaný počet dávek.

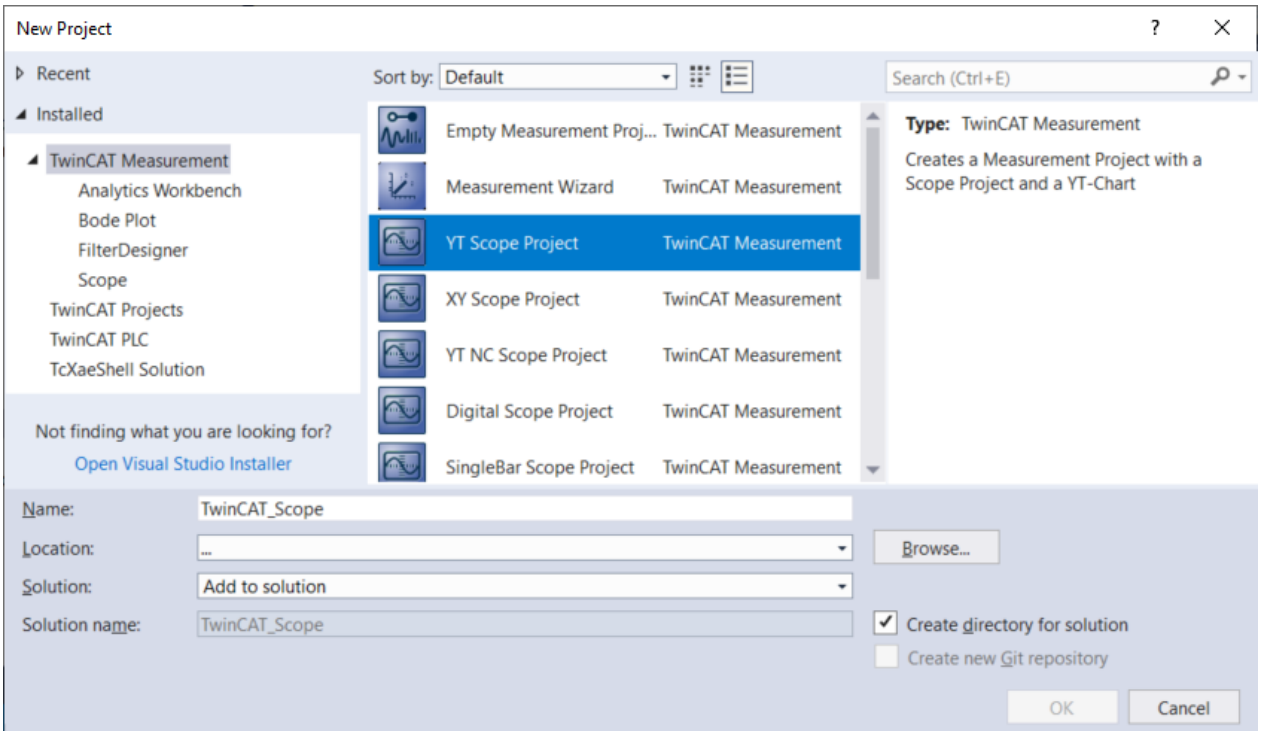
11. Optimalizace regulačního děje – návrh regulátoru pomocí nástroje MATLAB

Pro návrh regulátoru je třeba mít představu o dynamickém chování řízené soustavy. Pro tyto účely je třeba provést proces identifikace a následně syntézu (návrh) regulátoru. V následujícím testu budou popsány tyto procesy využitím nástrojů, které poskytuje prostředí MATLAB.

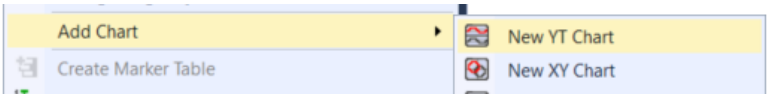
11.1. Tvorba dat pro MATLAB

Data pro MATLAB můžeme získat několika způsoby. Jedním z nich je např. využití knihovního funkčního bloku **FB_CTRL_LOG_DATA**, který umí zapisovat do CSV datového souboru. My si ale ukážeme jinou cestu, která se více hodí pro ladění a zahrnuje vzorkování dat, zpracování a vizualizaci přímo v systému TwinCAT.

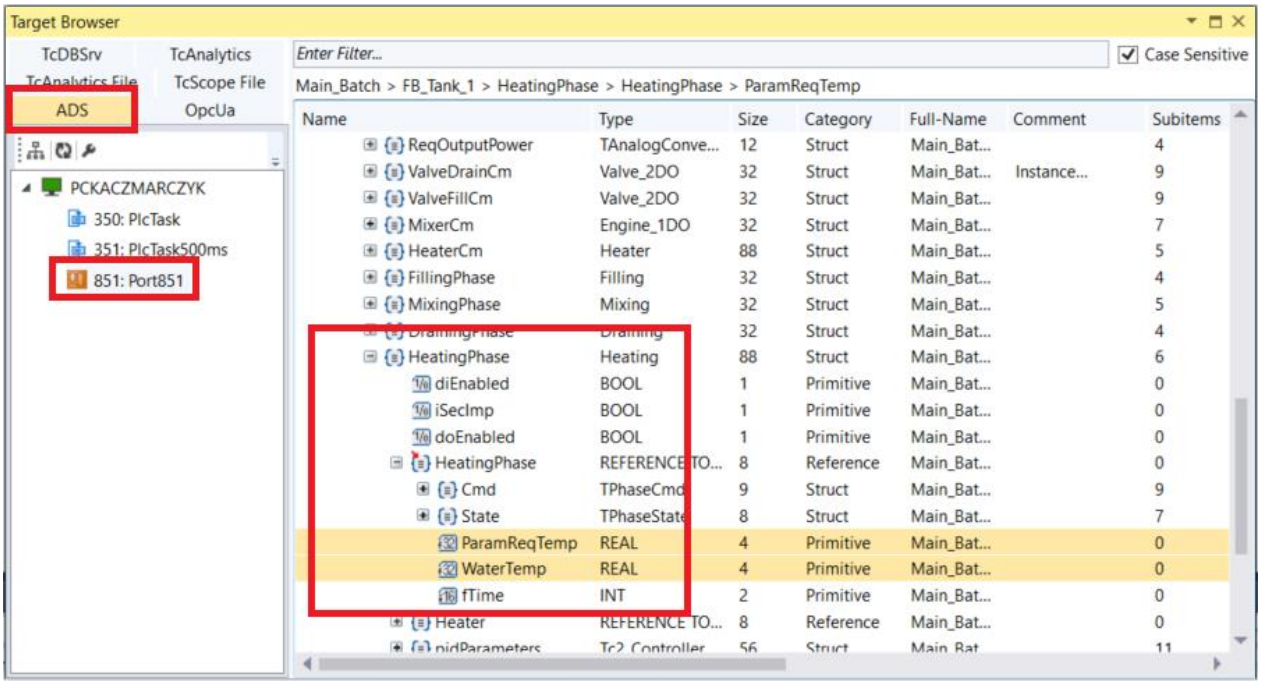
Vytvoříme si ladicí projekt, prostřednictvím kterého budeme schopni vizualizovat data. Přidejme si tedy do našeho Solution projekt **YT Scope Project** skupiny **TwinCAT Measurement – Scope**. V případě, že skupinu TwinCAT Measurement nemáme instalovánu, je nutno doinstalovat balíček **TF3300 | TwinCAT 3 Scope Server** ze stránek BECKHOFF.



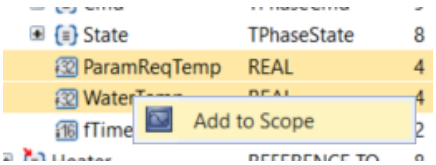
Pole **Location** ponechejte stejné, jako je umístění vašeho projektu **Tanky**. Po vytvoření projektu dále klikem pravým tlačítkem myší na položku projektu v **Solution Explorer** zadejte **Add Chart – New YT Chart**. Tím dojde k vytvoření nového grafu.



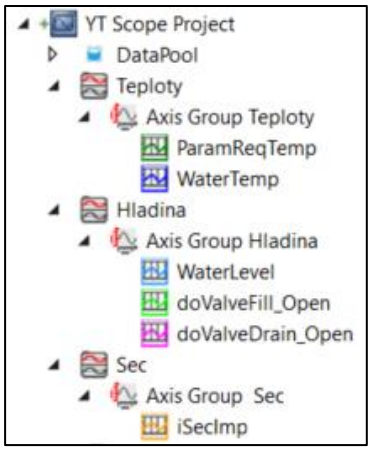
Nyní klikněte pravým tlačítkem opět na položku **YT Chart** ve stromu projektu a vyberte **Add – New Axes** a následně na položku **Axis Group** pravým tlačítkem a zadejte **Target Browser**. Ujistěte se, že máte vybrán paměťový prostor ADS, dále vyberte v lokálním PC (Soft PLC) položku Port851 a pak v hlavní části okna rozbalte strukturu dle obrázku níže.



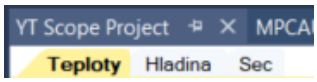
Pomocí Ctrl + klik myší vyberte dvě hodnoty – žádanou teplotu vody a skutečnou teplotu vody a přes pravé tlačítko myši vyberte **Add to Scope**. Na dotaz, zda se má vytvořit separátní kanál pro každý ze symbol dejte **Ne**. Tato odpověď zabezpečí vložení obou proměnných do jednoho souřadného systému.



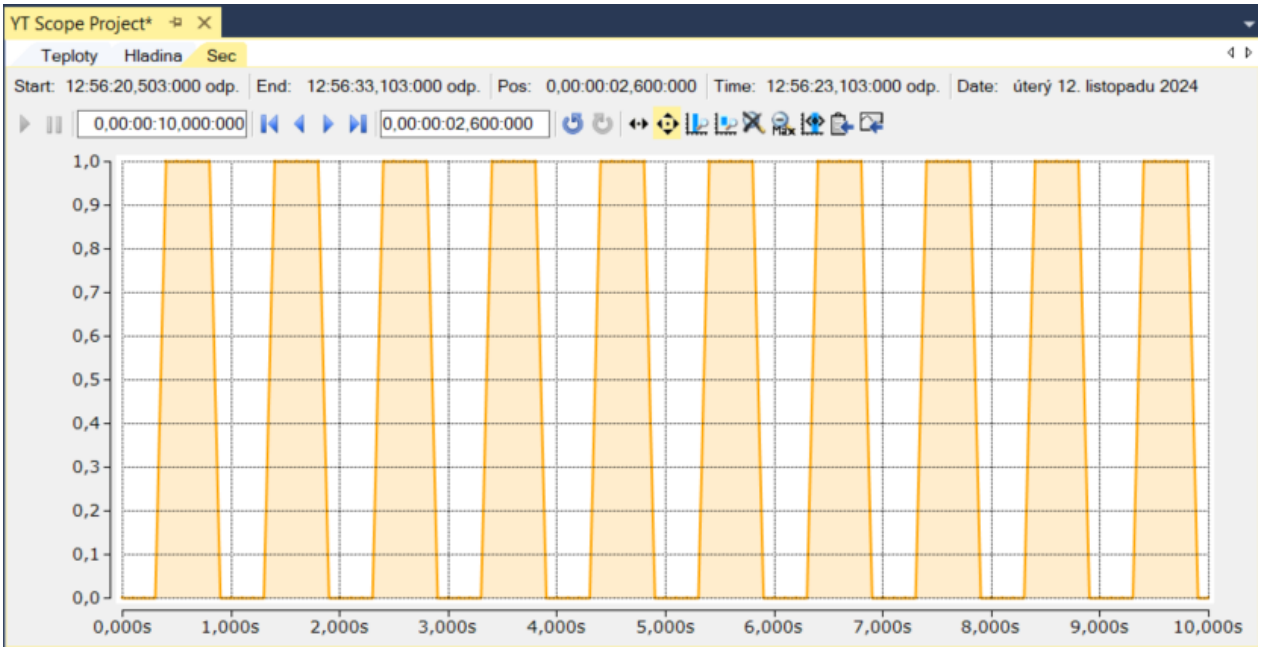
Do projektu můžeme vložit více grafů. Na obrázku níže vidíme vložený ještě graf pro zobrazení výšky hladiny a stavu napouštěcího a vypouštěcího ventilu a dále graf, ve kterém bude vizualizována funkčnost sekundového vstupu – to jen pro kontrolu funkce komunikace.



Po dokončení aktivujeme konfiguraci a spustíme program v SoftPLC. Nyní (pokud není) otevřeme kartu YT Scope Project dvojklikem na některý z definovaných grafů. Vidíme, že v této kartě je možno přepínat mezi zobrazením jednotlivých grafů.



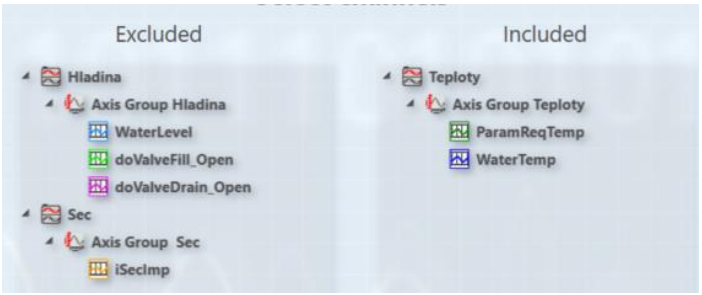
Pro start záznamu stiskneme ikonu **Start Record (F5)** z panelu  . Po nachytání dostatečného množství dat můžeme stiskem **Stop Record (F6)** záznam opět zastavit (první dvě ikony na následujícím obrázku).



Pro zpracování v programu Matlab potřebujeme data exportovat do souboru .CSV. Z hlavní nabídky vybereme volbu **Scope – Export** a postupujeme po krocích:

- Vybereme volbu **Character Separated Values**

- Vybereme pouze teploty



- Ponecháme doporučenou časovou periodu (uloží se celý záznam)
- Vybereme parametry pro export
  - Cull Timestamp
- Nastavíme CSV parametry
  - CSV-Separator: Semicolon
  - Decimal mark: Point
  - Header configuration: None
- Vybereme název umístění souboru
  - Název: sys_data.csv

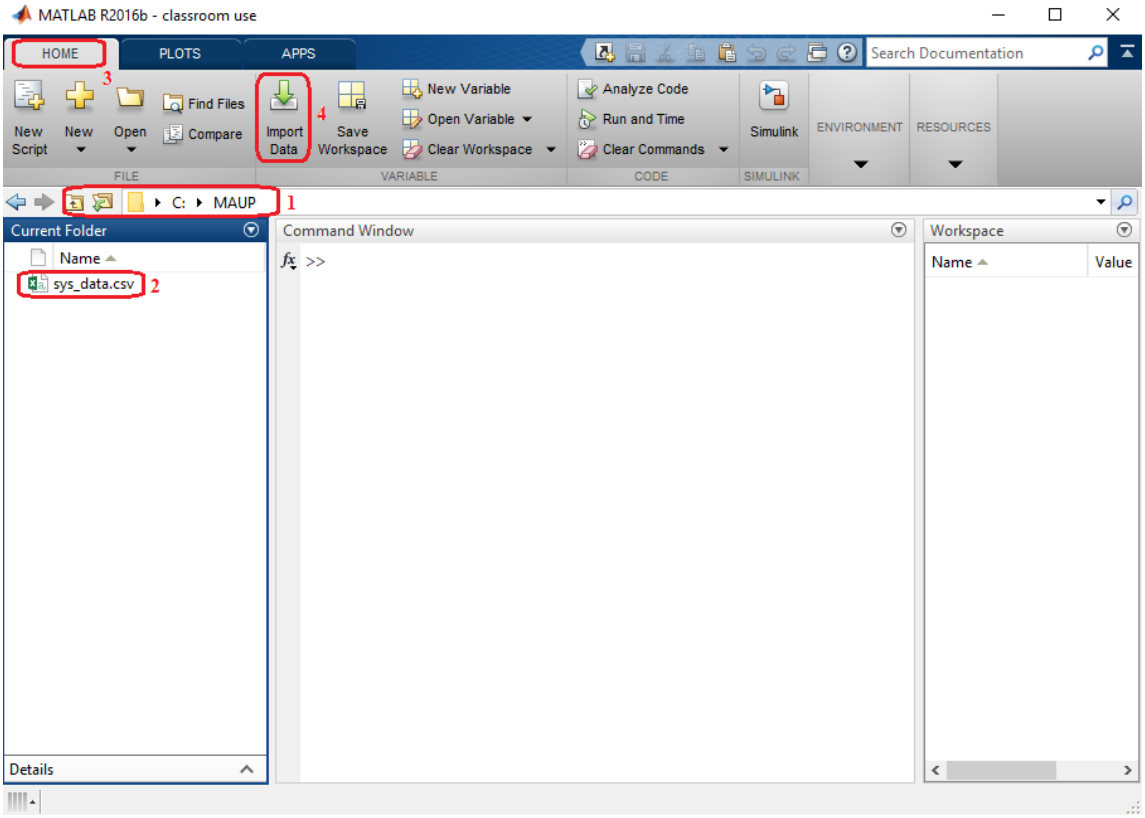
Pokud si následně soubor zobrazíme, uvidíme následující:

```
133758861805030000;0;133758861805030000;0
133758861806030000;0;133758861806030000;0
133758861807030000;0;133758861807030000;0
133758861808030000;0;133758861808030000;0
133758861809030000;0;133758861809030000;0
133758861810030000;0;133758861810030000;0
133758861811030000;0;133758861811030000;0
```

První a třetí hodnota udává časovou značku měření ve formátu **8-digit LDAP/FILETIME** (pro detail viz https://www.epochconverter.com/ldap). Druhá a čtvrtá hodnota pak udávají žádanou, resp. skutečnou hodnotu. Tyto budeme potřebovat pro identifikaci.

11.1.1. Načtení dat

V předchozím textu byl popsán proces získání (měření) vstupně/výstupních dat z PLC reprezentujících vstupní a výstupní signály z řízené soustavy ve formě změřené přechodové charakteristiky. Tato data byla uložena do souboru sys_data.csv. Naměřené hodnoty z tohoto souboru je nejprve nutné importovat do prostředí MATLAB, viz následující obrázek.

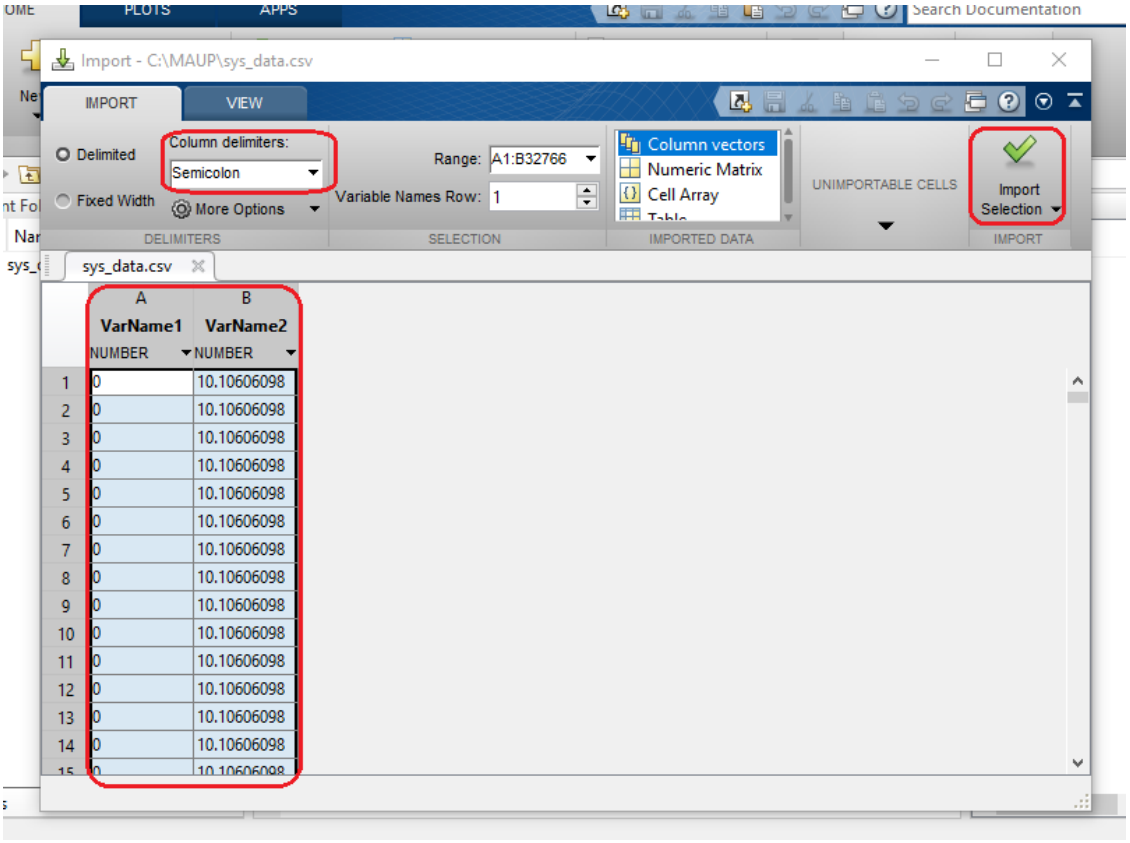


1. Nastavit pracovní adresář (1) – zvolte vlastní, případně dle pokynů vyučujícího.
2. Zkontrolovat, zda je v pracovním adresáři přítomen soubor sys_data.csv (2)
3. V záložce ***Home*** (3) kliknout na ***Import Data*** (4) a v dialogovém okně vybrat *.csv* soubor.

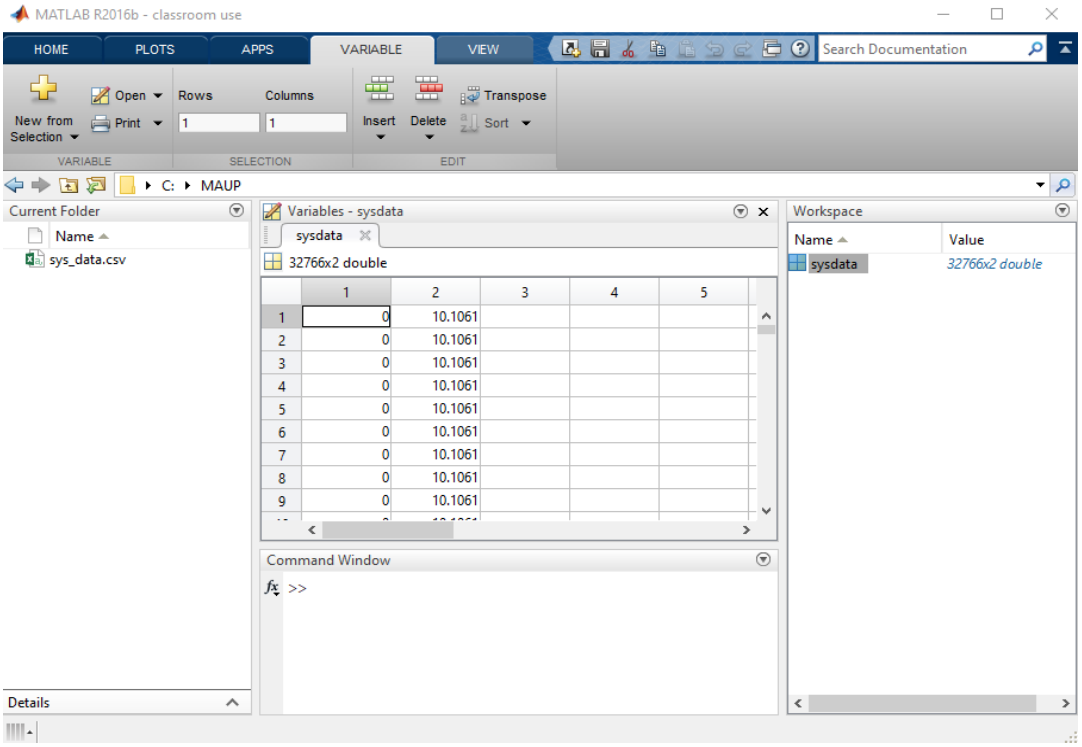
Otevře se nové okno, viz následující obrázek, a zde zkontrolujeme, zda jsou data správně rozparsována. Výsledek by měl vypadat jako na následujícím obrázku, tj. jsou k dispozici **2 oddělené sloupce** se vstupním a výstupním signálem. Pokud se sloupce nezobrazují správně, je třeba změnit typ oddělovače v menu ***Column delimiters***.

Pozn.: Pokud jsou v původním souboru data s desetinnou čárkou, je třeba změnit na desetinnou tečku!

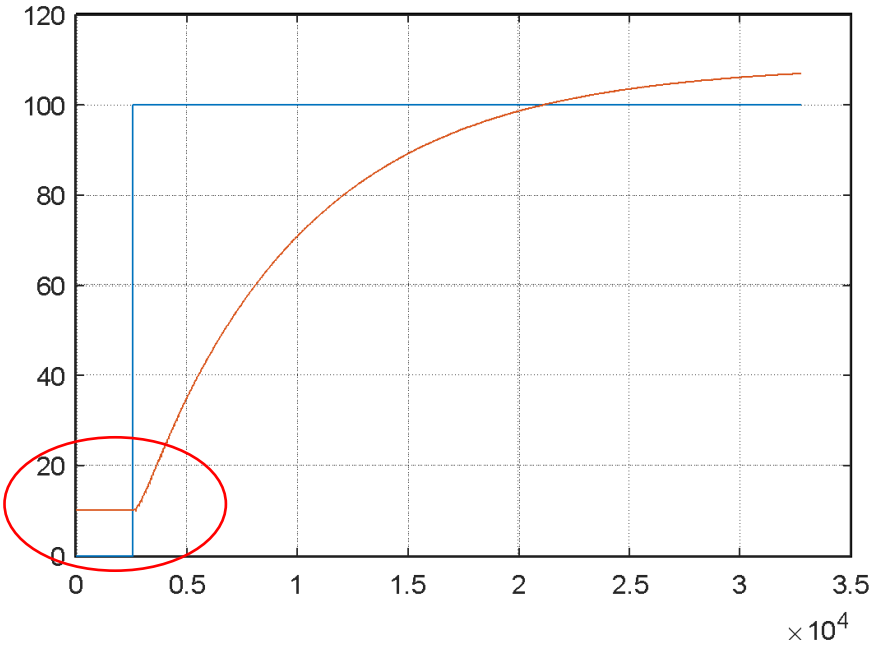
4. Poté je třeba vybrat datový typ, do jakého se budou data importovat. Vhodná je např. matice, tj. ***Numeric Matrix***.
5. Nakonec kliknout na Import selection.



Po chvíli by se měla zobrazit ve ***Workspace*** matice dat obsahující importovaná data z *.csv* souboru. Viz následující obrázek.



Zde je vhodné si data zobrazit v grafu (příkaz ***plot***), aby bylo vidět, s čím budeme dále pracovat.



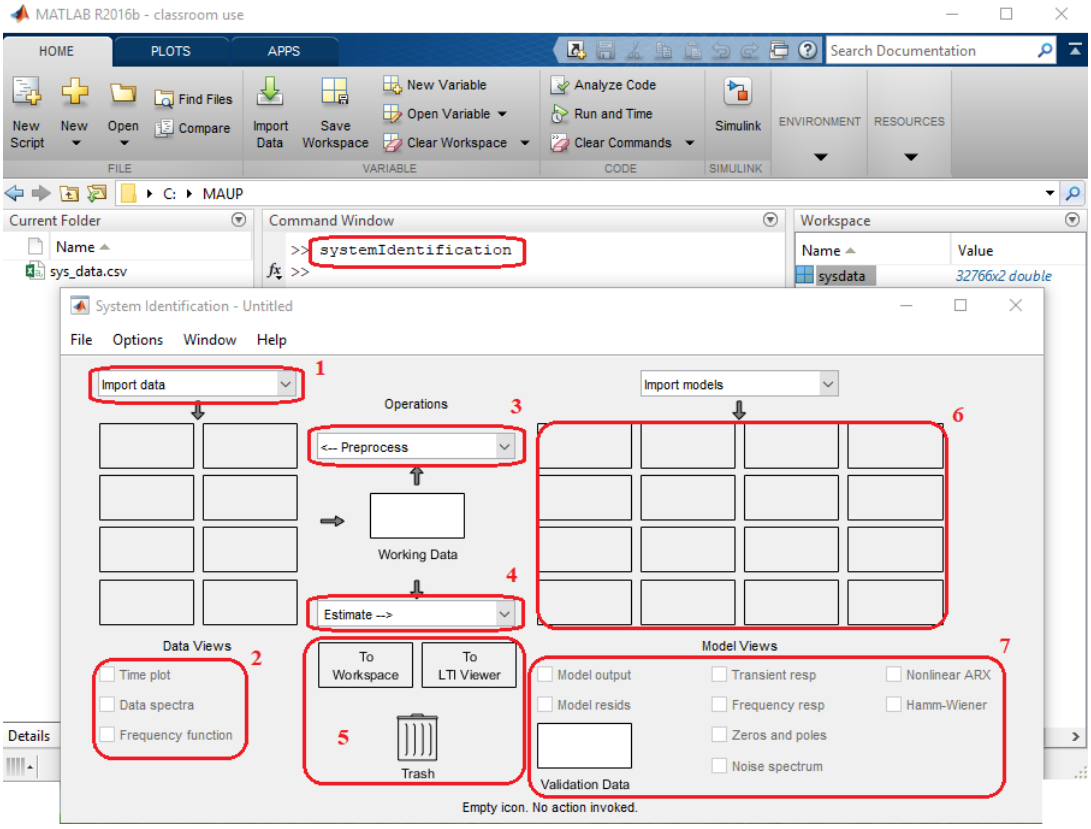
Uvedená data reprezentují přechodovou charakteristiku dynamického systému. Z této charakteristiky lze zjistit (stanovit) několik informací, a sice:

- typ soustavy: statická/astatická,
- zesílení soustavy,
- přibližně určit řád systému,
- počáteční podmínky.

Chystáme se dále identifikovat přenos systému, který je definovaný pro **nulové počáteční podmínky**. Je tedy třeba zajistit to, aby oba průběhy (vstupní i výstupní signál) splňovaly podmínku $f(0) = 0$.

11.1.2. Identifikace

Pro identifikaci parametrů soustavy je možné využít nástroj ***System Identification Toolbox***. Ten lze spustit např. zápisem příkazu ***systemIdentification*** (nebo ***ident***) v ***Command Window***. Otevře se následující okno.

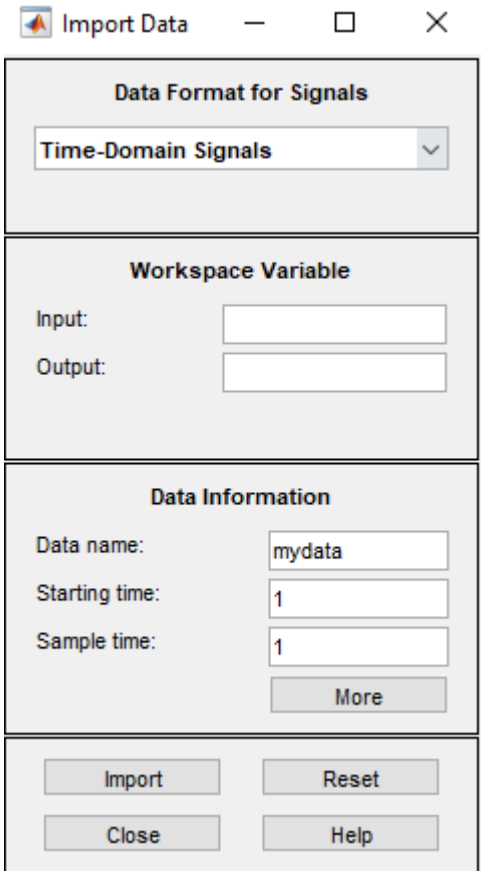


Popis sekcí v okně System Identification:

1. **Import data** – zde je možné našíst data v časové či frekvenční doméně, apod.
2. **Data Views** – zobrazení importovaných dat, případně zobrazení spektra, apod.
3. **Preprocess** – zde je možné provést preprocesing vstupních dat, tj. např. filtrovat apod.
4. **Estimate** – vlastní výběr způsobu identifikace
5. **Okno pro práci s identifikovaným modelem** – zde je možné výsledný model uložit do Workspace nebo jej smazat. Provádí se přetažením modelu ze sekce 6.
6. **Identifikované (resp. importované) modely –** zde se zobrazí výsledek identifikace. Pokud již máme např. uložený model ve Workspace, lze jej zde importovat
7. **Model Views –** zobrazení charakteristik identifikovaného modelu

Postup identifikace dat je v našem případě následující:

1. Rozbalit nabídku ***Import data*** a vybrat ***Time domain data…***
2. Zobrazí se vyskakovací okno, kde je třeba vybrat vstupní data a uvést další parametry, viz následující obrázek.



3. Nyní zadáme vstupní (***Input***)a výstupní (***Output***) data, pojmenovat datový soubor (***Data name***) a vybrat začátek (***Starting time***) a vzorkovací periodu (***Sample time***). V našem případě to bude:

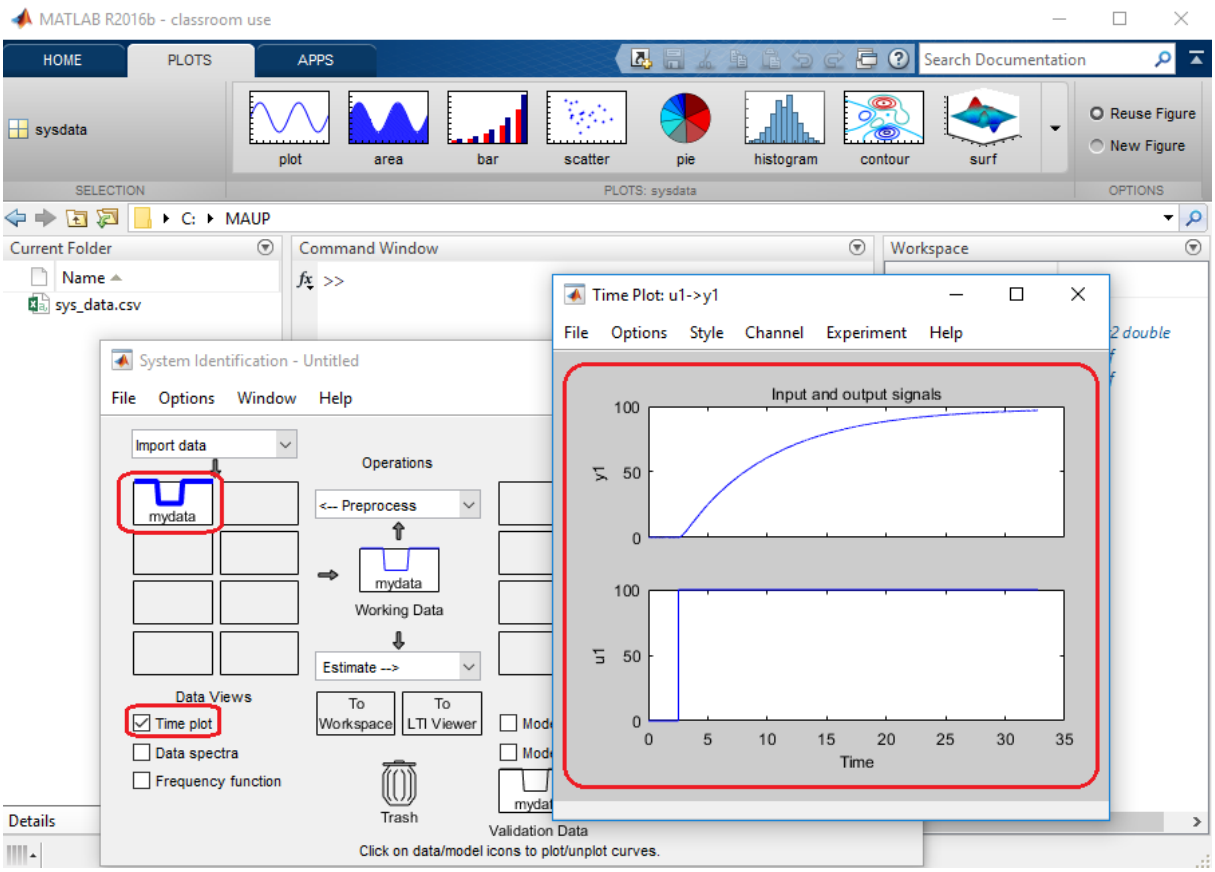
Input: sysdata(:,1) všechny řádky, 1. sloupec

Output: sysdata(:,2) všechny řádky, 2. sloupec

Starting time: 0

Sample time: 0.1 (pokud vzorkovací perioda, s jakou jste ukládali data v PLC byla 100 ms)

4. Kliknout na ***Import***
5. V sekci Import data se zobrazí importovaná data, která lze zobrazit zaškrtnutím ***Time plot***

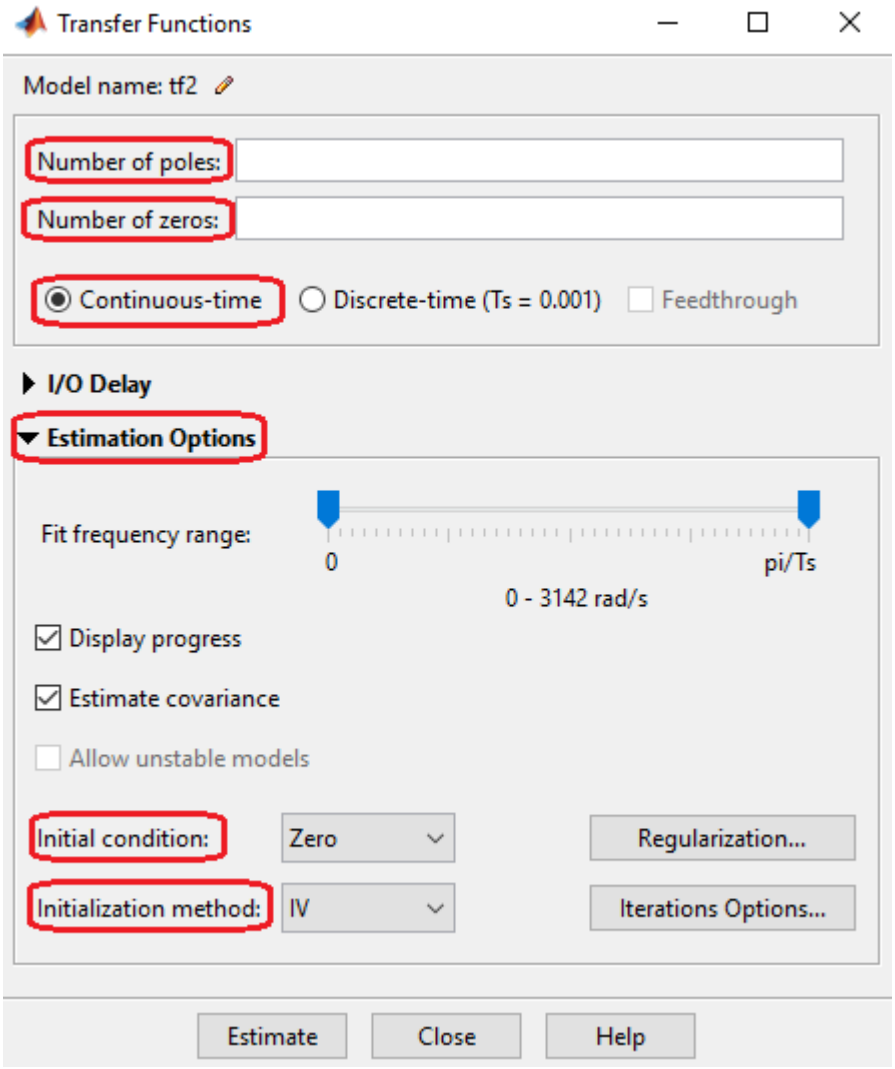


6. Vstupní data nejsou výrazně zašuměná a jsou tak připravena přímo pro identifikaci, viz nabídka ***Estimate***. Zde máme na výběr několik možností. Nejjednodušší způsob je např. ***Transfer Function Models***.

7. Otevře se následující okno, kde je třeba nastavit:

   - **počet pólů** (Number of poles)
   - **počet nul** (Number of zeros)
   - **spojitý** (Continuous-time) **model**
   - v záložce Estimation Options pak **počáteční podmínky** (Initial condition) na nula (Zero) a **inicializační metodu** můžeme nechat IV (Instrument Variable)

Počet pólů a nul zvolte s ohledem na tvar naměřené přechodové charakteristiky.

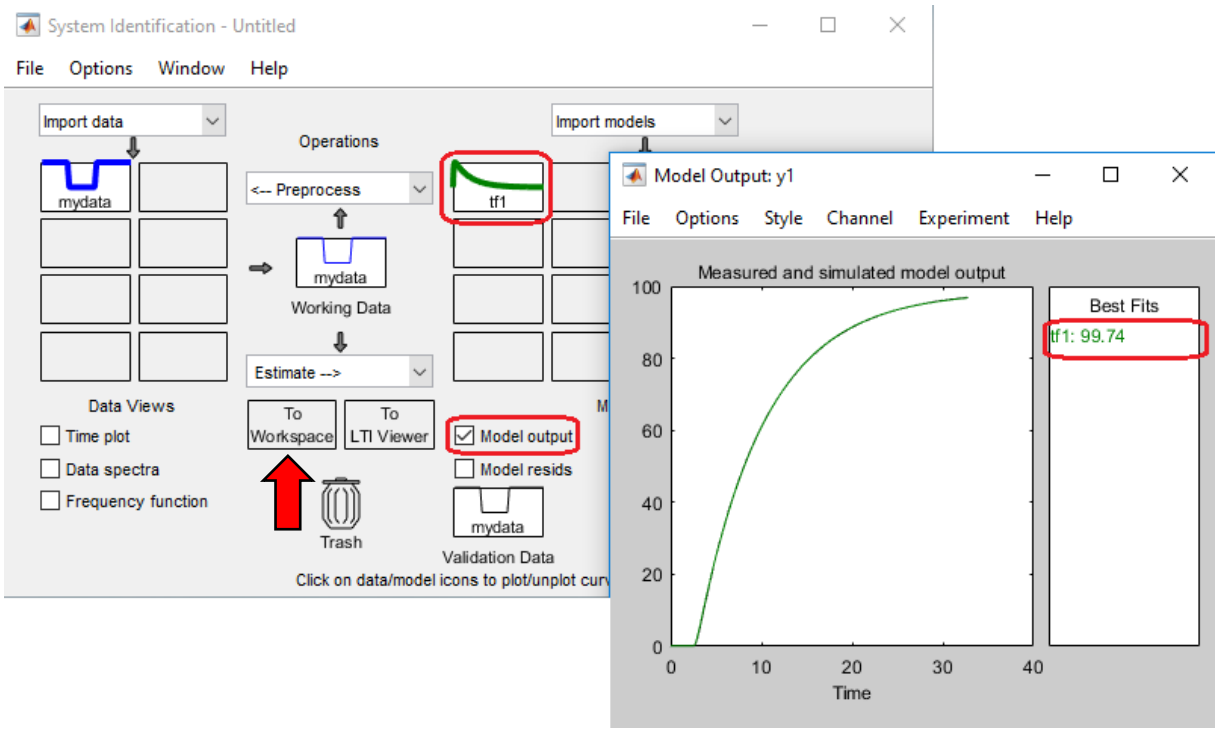


8. Poté stačí kliknout na ***Estimate***.

9. Po chvilce se zobrazí výsledek (identifikovaný model) a můžeme si ověřit úspěšnost identifikace např. kliknutím na ***Model output*** (v tomto případě je přesnost identifikace vyjádřená parametrem Best fit na hodnotě 99.74 %, což je velice dobrý výsledek), viz následující obrázek.

10. Nyní už můžeme importovat model do Workspace jeho přetažením do okna ***To Workspace***.

11. Ve Workspace už lze s modelem pracovat jako s klasickou přenosovou funkcí, tj. lze aplikovat standardní příkazy (pzmap, step, impulse, bode, nyquist, apod.)



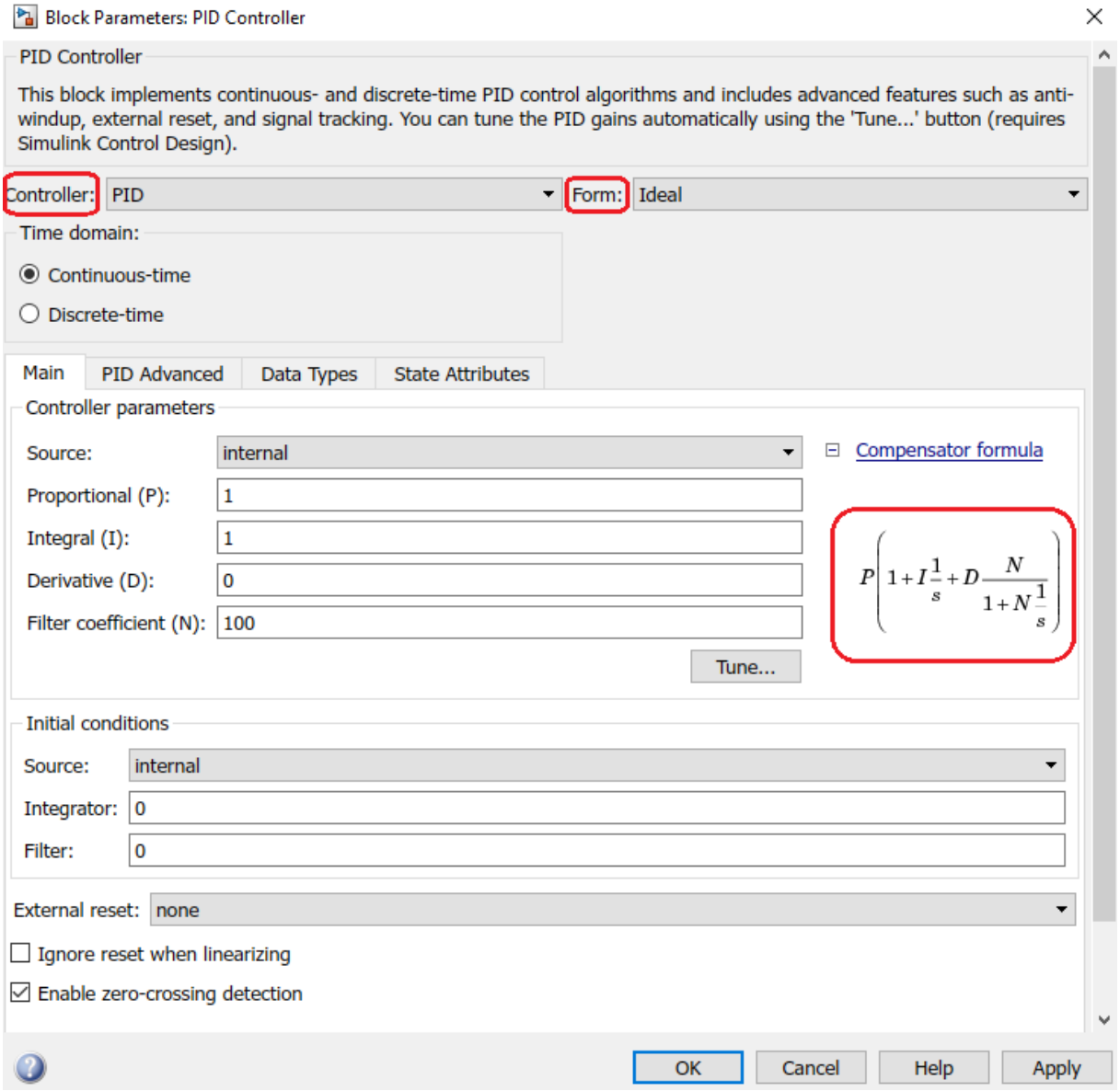
11.1.3. Návrh regulátoru

Pro návrh regulátoru je možné využít nástroje ***PID tune***. Ten lze spustit buď přímo z Command Window příkazem ***pidtool*** (zde jsou však omezené možnosti) nebo pracovat s tímto nástrojem v Simulinku.

Zde si v prvním kroku vytvoříme standardní regulační smyčku, kde jako soustavu vložíme identifikovaný model a jako regulátor blok ***PID controller*** z nabídky ***Continuous***.

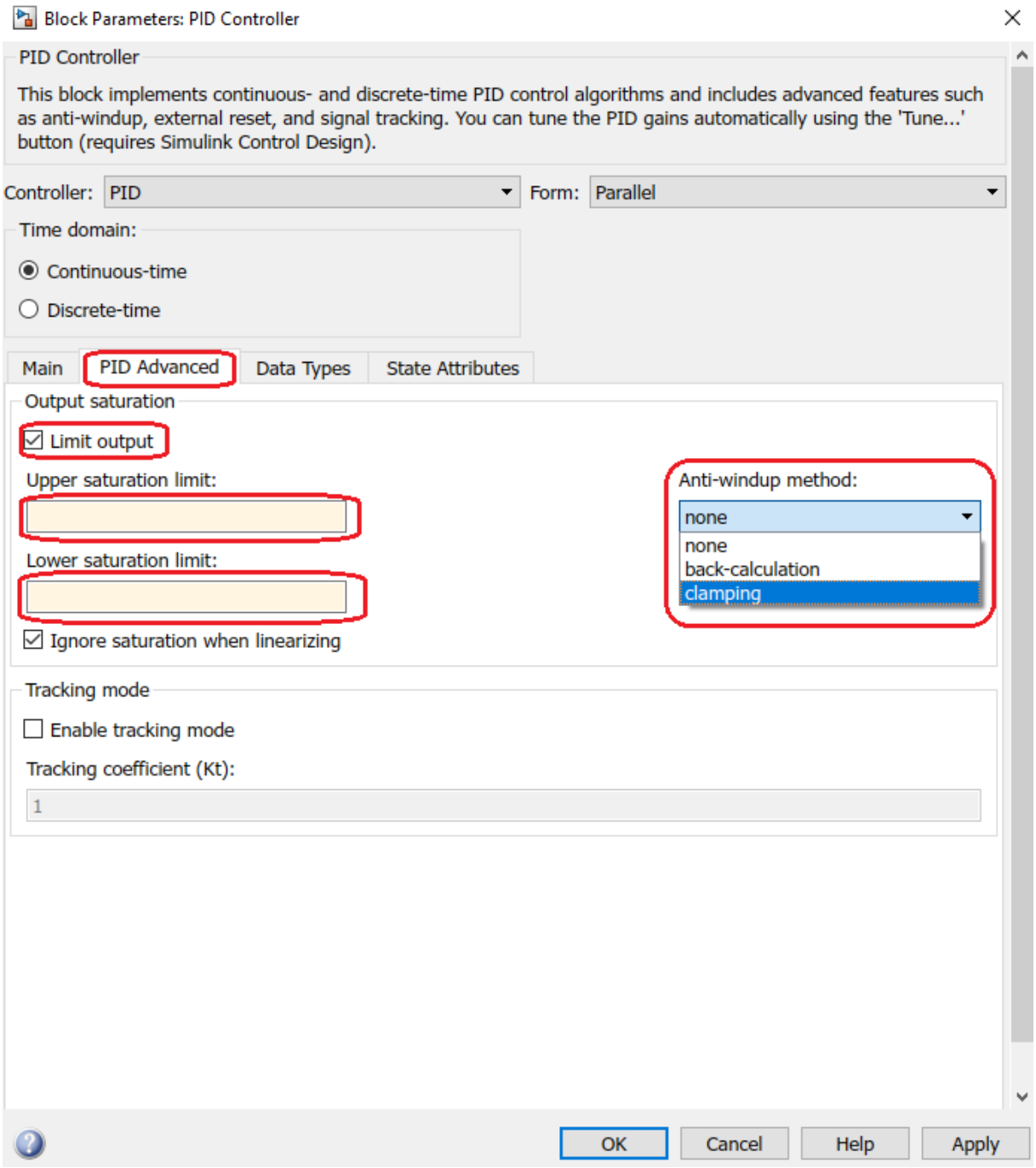
Neměli bychom zapomínat na omezení, které v obvodu máme, a sice omezený akční zásah. Vložíme tedy ještě blok ***Saturation*** z nabídky ***Discontinuities*** a nastavíme horní a dolní mez. Výsledné schéma by mělo vypadat následovně.

Nastavení regulátoru provedeme z nabídky PID tool, kterou otevřeme dvojklikem na PID Controller. Zde vybereme vhodný typ regulátoru (P, PI, PD, PID, apod.) a zvolíme formu zápisu – ***Ideal*** nebo ***Parallel***. Níže se nám tyto tvary zobrazí.



Vzhledem k tomu, že PID regulátor implementovaný v PLC Simatic využívá tvar odpovídající možnosti ***Ideal***, zvolíme tento tvar. Ladění si ukážeme např. na **PI regulátoru**.

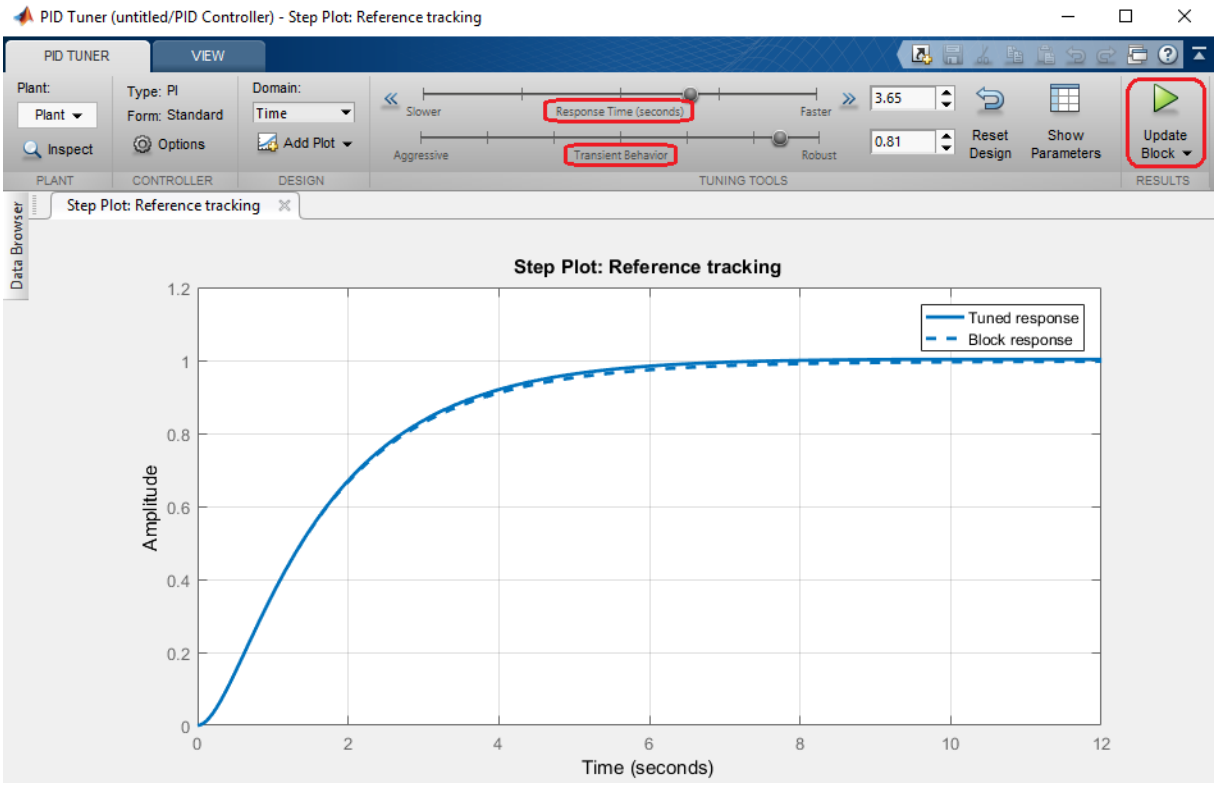
Výše bylo zmíněno omezení akčního zásahu. To je třeba zohlednit i při návrhu regulátoru a lze provést nastavením tohoto omezení (Limit output) v záložce ***PID Advanced***. Ve stejné záložce lze také nastavit metoda ***Anti-windup*** (např. ***Clamping***).



Vrátíme se do záložky ***Main*** a klikneme na tlačítko ***Tune***.

Otevře se okno, kde můžeme jednoduše pomocí posuvníků ***Response time*** a ***Transient Behavior*** měnit vlastnosti regulátoru. Výsledná odezva regulátoru se zobrazuje v grafu.

Až budeme s průběhem (odezvou) spokojeni, můžeme kliknout na tlačítko ***Update block***.



Následně můžeme toto okno zavřít a vrátit se do okna ***PID Parameters***. Zde už jsou aktuální hodnoty složek P a I podle provedeného „návrhu“.

Regulátor můžeme otestovat pomocí simulace v Simulinku a pokud je vše tak, jak má být, je možné výsledné parametry použít (**ve správném tvaru**) pro blok PID regulátoru v PLC Simatic.

12. Programování typu BATCH – systém pro řízení výroby

Zadání: zadefinujte stanici PLC, její adresu

1. Na stanici vytvořte jednotku tanku.
2. Pro jednotku vytvořte fáze jednotky, které použijete v receptuře.
3. Vytvořte testovací recepturu pro ověření fáze napouštění a vypouštění
4. Vytvořte recepturu pro řízení celé technologie.

12.1. Nastavení projektu PLC pro komunikaci OPC-UA

Pro spuštění OPC-UA serveru v PLC a možnost jeho konfigurace v konfiguračním PC je nutné nainstalovat následující balíčky:

- TF6100-OPC-UA-Server-V5.2.exe (nutno instalovat do PLC i do PC v případě použití SoftPLC)
- TF6100-OPC-UA-Client.exe (testovací aplikace)
- TF6100-OPC-UA-Configurator.exe (konfigurační aplikace)

Po nainstalování balíčků je nutné v aplikaci TwinCAT provést aktivaci licence stejně, jako jste aktivovali licenci pro Modbus TCP.

TF6100	TC3 OPC-UA	cpu lic...		expires ...	BDCC0070-42D5-49AE-ABF1-1D4...

V dalším kroku je nutné systému říci, které proměnné budou komunikovány prostřednictvím OPC-UA serveru. Toto se nastaví přímo v definici bloku globálních proměnných takto:

```
{attribute 'qualified_only'}
VAR_GLOBAL
    {attribute 'OPC.UA.DA' := '1'}
    {attribute 'OPC.UA.DA.StructuredType' := '1'}
    Tank1 : TTank;

    {attribute 'OPC.UA.DA' := '1'}
    test_opc : INT := 1;
```

První přidaný atribut ***qualified_only*** označuje, že k proměnným v následujícím datovém bloku bude možné přistupovat pouze prostřednictvím jejich plného názvu (např. tedy Global.Tank1, pokud se datový blok jmenuje Global).

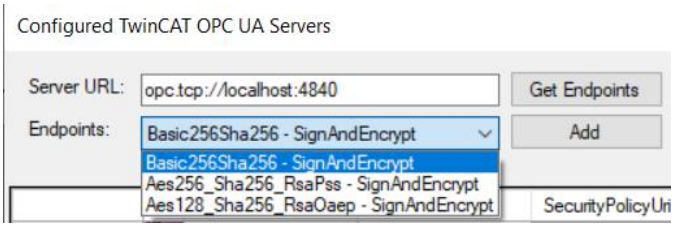
Další atribut ***OPC.UA.DA*** nastavený na hodnotu 1 označuje následující symbol (proměnnou) jako komunikovanou prostřednictvím OPC-UA (DA = Data Access). Atribut ***OPC.UA.DA.StructuredType*** pak označuje datový typ jako strukturovaný. Pro Tank1 je nutné jej použít, pro test_opc hodnotu typu INT zjevně nikoli.

Dále je potřebné vygenerovat v PLC projektu popisný OPC UA **TMC** soubor.

TBD: Kubo, zde se můžeš vypsat

Nyní máme projekt připraven pro spuštění s OPC-UA serverem. Provedeme aktivaci konfigurace a odstraníme případné chyby překladu. Dále provedeme spuštění v SoftPLC.

Dalším krokem je spuštění OPC UA Konfigurátoru a stisknutí tlačítka **Edit** na liště. Jako URL serveru zadáme adresu **opc.tcp://localhost:4840** a zmáčneme tlačítko **Get Endpoints**. Ze seznamu endpointů vybereme první a stiskneme tlačítko **Add**.



Zavřeme okno přidání UA serveru. Následně na liště vybereme tento server z rozbalovacího seznamu a stiskneme tlačítko **Connect**. Měl by se objevit dialog pro zadání přístupových údajů. Jedná se o údaje, které budou **při inicializaci serveru** (kterou nyní procházíme) **nastaveny** (tedy nikoli o údaje, které nastavil před Vámi někdo jiný!). Zadejte tedy

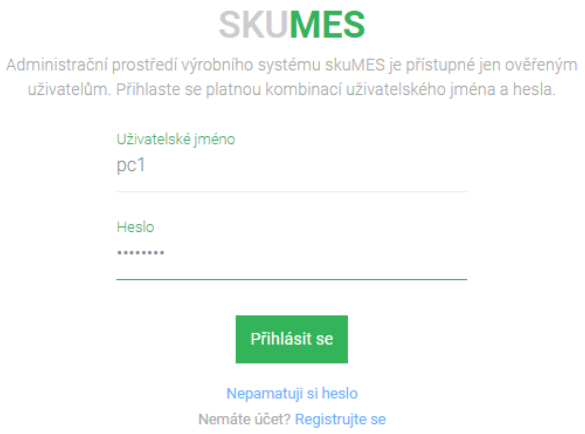
- Username: **testuser**
- Password: **mpcaup2030**

Po potvrzení se objeví otázka, zda chcete vyčíst aktuální konfiguraci serveru, zadejte **Ano**. Tím máte server nakonfigurován.

Nyní ověříme funkčnost sdílení dat. Spusťte

12.2. Přihlášení do MES systému a popis rozhraní

K webové aplikaci pro správu činnosti MES systému je možné se připojit prostřednictvím prohlížeče na adrese **http://skumes.skupra.cz**. Jako uživatelské jméno zadejte text „pcX“, heslo je pak „studentX“, kde X je číslo vašeho pracoviště.

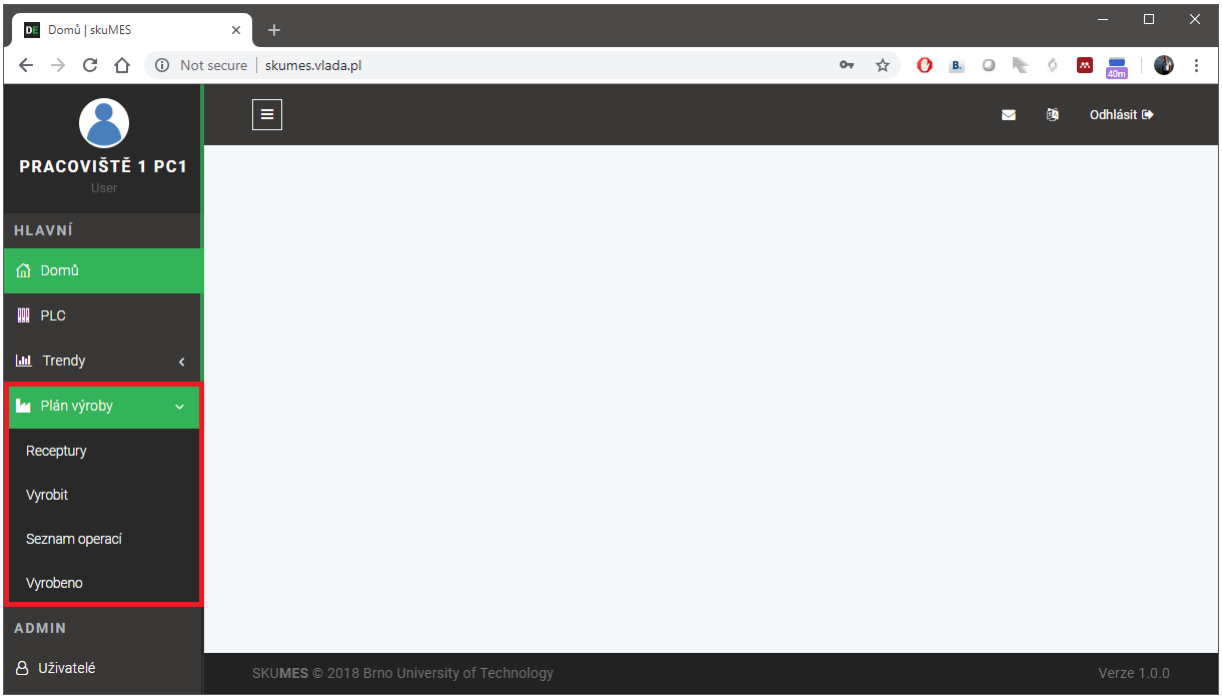


Obr. 7: Přihlášení do aplikace SkuMES

Na Obr. 7 je zobrazeno přihlašovací okno aplikace. Po vyplnění uživatelského jména a hesla stiskněte tlačítko „Přihlásit se“ pro přihlášení. Je možné rovněž vytvoření nového uživatele prostřednictvím odkazu „Registrujte se“. Takový uživatel však má práva pouze pro nahlížení do systému. Nemá možnost tedy nic měnit.

Po přihlášení se dostanete do prostředí SkuMES. SkuMES je modulární systém. V současné době obsahuje pouze dva moduly, z nichž pouze jeden je možné plnohodnotně ovládat prostřednictvím webu.

- SkuMES Plán výroby – jedná se o BATCH modul sloužící k řízení výrobních procesů
- SkuMES Trendy – jedná se o modul sloužící pro vizualizaci trendů sledovaných procesních hodnot (v současné době je implementováno pouze výkonné jádro modulu logující trendy do databáze, komplexní zobrazení trendů není podporováno).



Obrázek 53 Prostředí aplikace SkuMES

12.3. Definice komunikace s PLC

Jádro systému SkuMES komunikuje s PLC (v našem případě Simatic S7-1500) prostřednictvím OPC-UA. Komunikace spočívá v tom, že dochází periodicky k obousměrné výměně definované datové struktury. Tato struktura je tvořena z následujících hierarchicky uspořádaných entit:

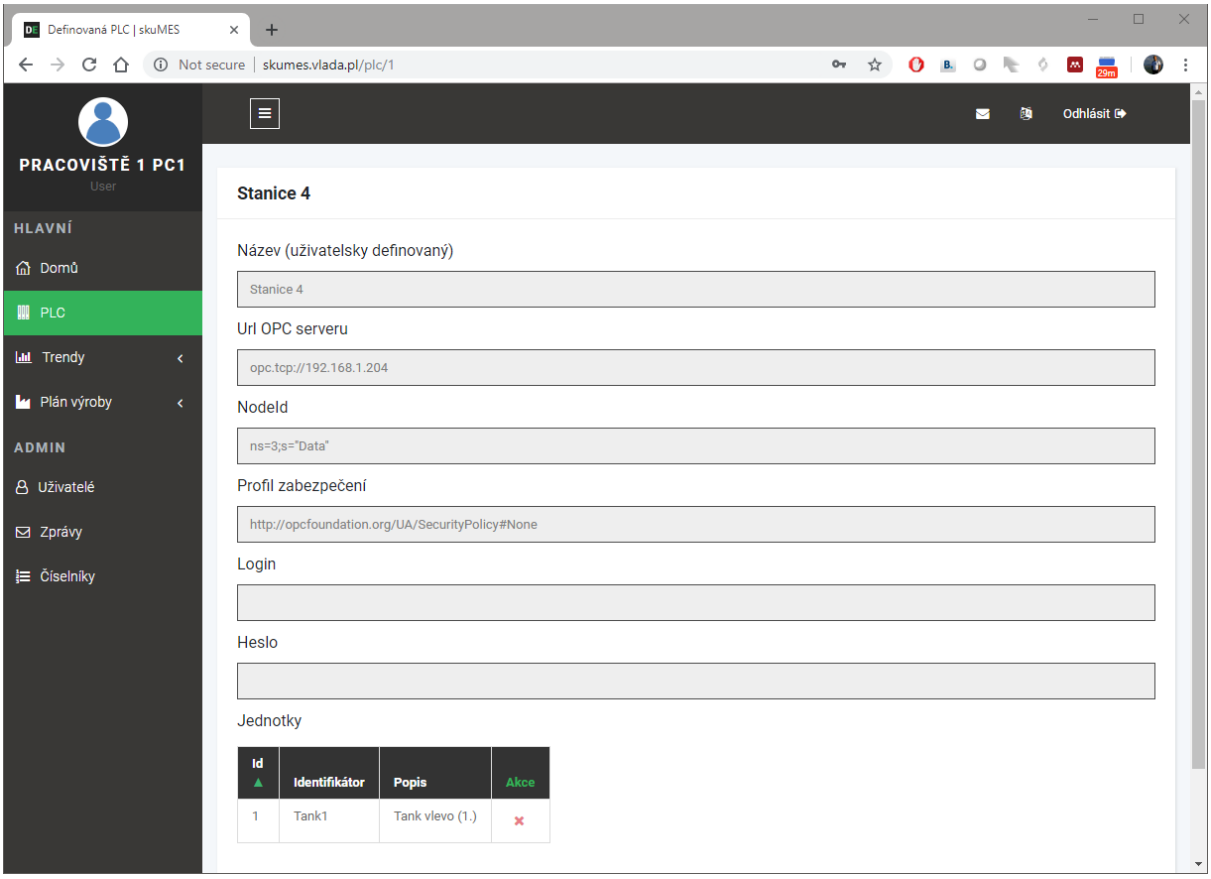
- PLC (Procesní buňka)
    - Jednotka
        - Modul zařízení („fáze“)

- Cmd
- State
- Param

Datová struktura vám je jistě povědomá. Jedná se samozřejmě o strukturu, kterou znáte z PLC programu Tanky, a kterou jsme pro vás připravili. Nyní víte, proč jsme kladli zvláštní důraz na to, že strukturu nelze modifikovat.

V systému SkuMES je musí být rovněž definována podobná struktura, která pro SkuMES vytváří abstrakci fyzického modelu dle S88. Tuto strukturu jsme pro vás už také připravili. Vy ji můžete zkontrolovat prostřednictvím webového rozhraní výběrem položky „PLC“ z nabídky „Hlavní“, následně zvolíte příslušnou stanici (PLC) (viz Obr. 9).

Na tomto obrázku si můžete všimnout nastavení přístupu k OPC serveru stanice, tedy jeho URL, NodeId (což je prefix, který určuje konkrétní část objektové hierarchie v serveru – všimněte si zejména textu „Data“, tedy názvu datového bloku v PLC programu). Dále se zde nachází konfigurace profilu zabezpečení dle specifikace OPC Foundation, a v případě požadavku na zabezpečené připojení i login a heslo k PLC.

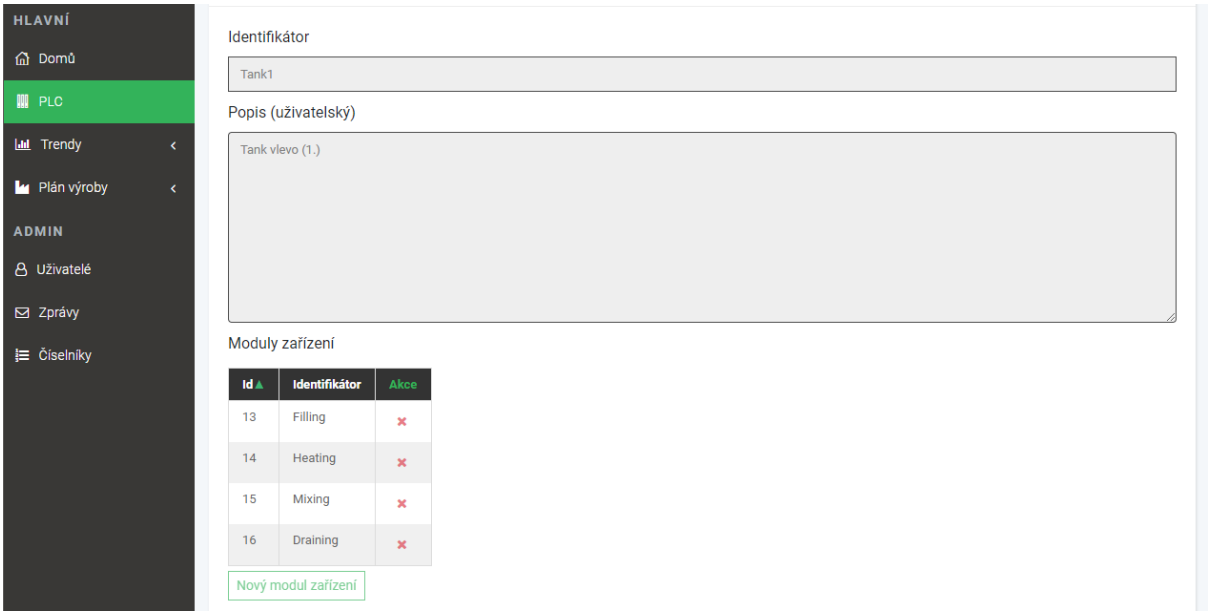


Obr. 9: Nastavení PLC komunikace

V tabulce „Jednotky“ se pak nachází seznam všech definovaných jednotek v procesní buňce. V našem případě se jedná pouze o jediný tank (Tank 1).

Kliknutím na příslušný řádek jednotky se otevře pohled s detailními informacemi o jednotce (Obr. 10). Zde lze definovat popis jednotky a dále její moduly zařízení[1]. Tento krok tedy definuje strukturu modulů zařízení ve fyzickém zařízení (viz Obr. 2). Jména modulů zařízení v jednotce musí přesně korespondovat se jmény modulů zařízení definovanými v datovém bloku PLC.

Kromě jména však není pro moduly zařízení nutno nic nastavovat, neboť struktura datového bloku v PLC programu, která slouží pro výměnu (synchronizaci) dat se předpokládá známá a neměnná.



Obr. 10: Nastavení konkrétní jednotky

Mapování řízení na zařízení určuje hranice mezi řízeními. Jedno z možných řešení, které je využito právě v systému SkuMES, je mapování fází na řídících modulech. Základní řízení se tedy vykonává v PLC a procedurální řízení zajišťuje jádro systému BATCH. Při návrhu základního řízení v PLC musíme respektovat požadavky výrobce BATCH systému. Ve vašem projektu se tak stalo právě proto, že jste použili předpřipravné datové bloky a implementovali pouze výkonný kód.

12.4. Tvorba receptur a jejich životní cyklus

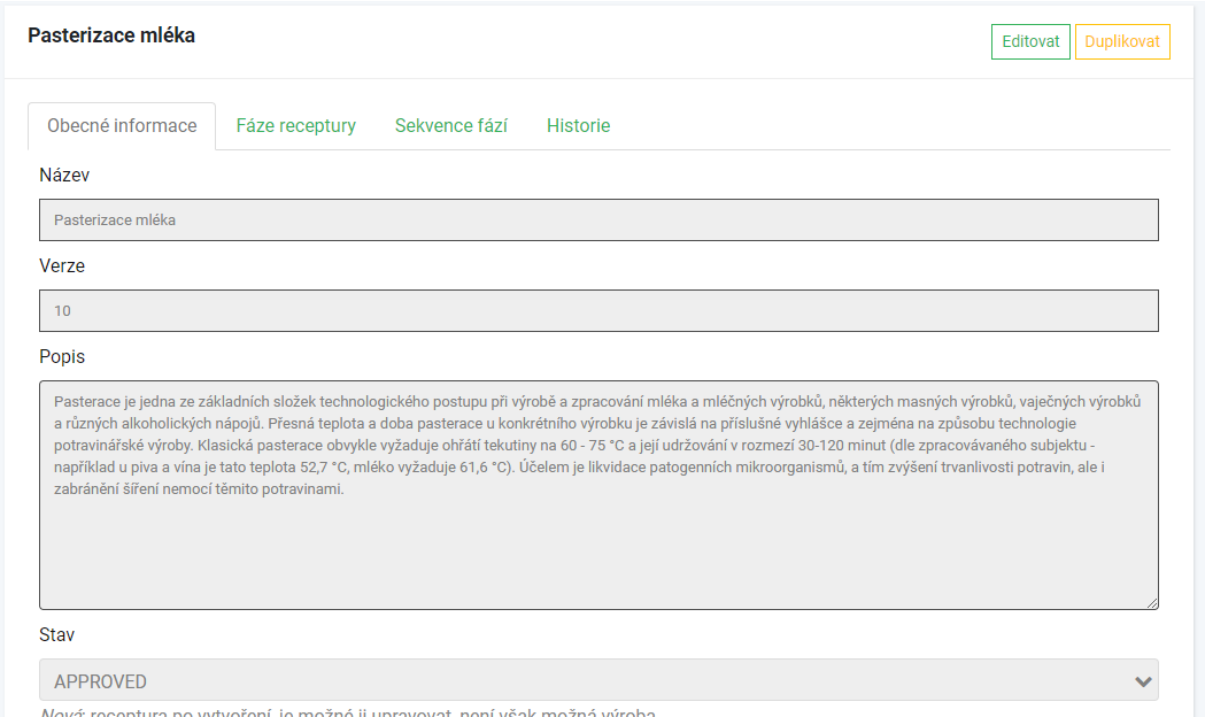
Základním stavebním prvkem procedurálního řízení je, jak už víte, procedura jako součást receptury. Prakticky je možné, aby byly receptury definovány buď v rámci MES systému nebo v jiném systému (například systému PLM, pokud jej firma využívá). V PLM systému bude mít firma nad recepturami lepší kontrolu, bude možné evidovat nejen receptury jako takové, ale rovněž jejich vztah k jednotlivým aspektům obchodního řetězce apod.

V našem případě však receptury definujeme v systému MES. V záložce Plán výroby – Receptury je možné zobrazit seznam definovaných receptur. Každá z receptur v seznamu je identifikována svým názvem, popisem a stavem.

[1] Někdy je nesprávně označujeme jako „fáze“ právě proto, že v našem způsobu implementace figurují na stejné úrovni ve fyzickém modelu, jako fáze v modelu procedurálním.

Po vytvoření se receptura nachází ve stavu **Nová**. V tomto stavu je možné ji parametrizovat. Po schválení receptury se její stav změní na **Schválená**. Teprve nyní je možné na základě této receptury vyrábět dávky. Naopak, recepturu v tomto stavu nelze již nijak modifikovat. To má samozřejmě svůj smysl – ve chvíli, kdy jsou na základě této receptury vyrobeny nějaké dávky, chceme jako výrobce navždy evidovat, jakým způsobem dávky vznikly.

Potřebujeme-li provést úpravu schválené receptury, jedinou možností je recepturu duplikovat. V takovém případě se vytvoří nová receptura ve stavu **Nová**. Aby bylo možné snáze odlišit takto duplikované receptury, má každá receptura po svém vytvoření verzi 1. Každou duplikací se verze receptury zvyšuje. U každé vyrobené dávky je samozřejmě evidováno, na základě které receptury a které verze byl produkt vyroben. Podíváme-li se na Obr. 11, uvidíme detaily receptury pasterizace mléka, kterou budete využívat v rámci cvičení. Vidíte nyní, jakým způsobem lze asi recepturu duplikovat.



Obr. 11: Obecné informace o receptuře

Jedinou možností změny schválené receptury je její převod do stavu **Archivovaná**. U takových receptur se již nepředpokládá výroba – není možné je pro výrobu zvolit. Je samozřejmě možné vyrobit duplikát takové receptury stejně, jako bylo popsáno výše.

Pozn.: Ne všichni uživatelé systému mají možnost vytvářet-mazat-duplikovat-schvalovat receptury. Jednotlivé akce se nastavují pro konkrétní uživatelé separátně.

Každá receptura má, mimo svou hlavičku, kterou jsme viděli na Obr. 11 také ještě *Formuli*, která v rámci tohoto MESu není explicitně definována[2] a samozřejmě *proceduru*. Procedura receptury se rozpadá na jednotkové procedury, operace a fáze. Vzhledem k tomu, že náš software je vyvinut pro řízení výroby na jednoduchých technologiích, vymezují se zatím naše možnosti pouze na definování sekvence nejnižších procedurálních entit - fází. Předpokládáme zde, že během výroby zvolíme procesní

[2] Neřešíme zde proces nakládání se zásobami ani další záležitosti, se kterými by definování formule dávalo smysl.

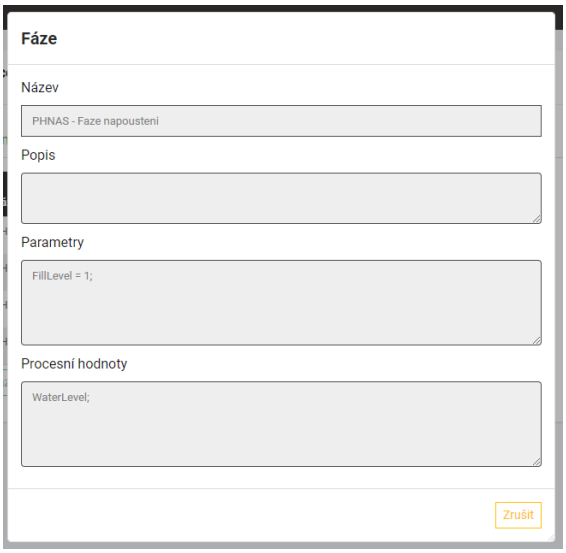
buňku, a dále jednotku, na které bude probíhat výroba. Ta bude realizována v rámci jediné operace (protože je jediná, není nijak explicitně specifikována) pomocí sekvence fází.

Na kartě Fáze receptury se tedy definuje seznam fází (zatím bez jakékoli návaznosti na sebe). Každá z fází receptury má svůj název, volitelný popis a dále seznam parametrů a procesních hodnot. Seznam parametrů se vkládá ve formátu **Název=Hodnota;<nový řádek>**, přičemž dovolené jsou pouze parametry definované v datovém objektu modulu zařízení v podsložce **Param**. Procesní hodnoty se zapisují jako **Název1;<nový řádek>Název2;** atd… Dovolené jsou pouze takové názvy, které jsou definované přímo v rootu datového objektu modulu zařízení. Pro lepší názornost se podívejte na Obr. 12 a

Nastavené parametry jsou před spuštěním konkrétní fáze kopírovány do modulu zařízení. Ve chvíli, kdy je konkrétní fáze spuštěna, jsou periodicky vyčítány a ukládány do databáze definované procesní hodnoty. Tyto hodnoty mohou být později využity pro generování reportů z výroby.

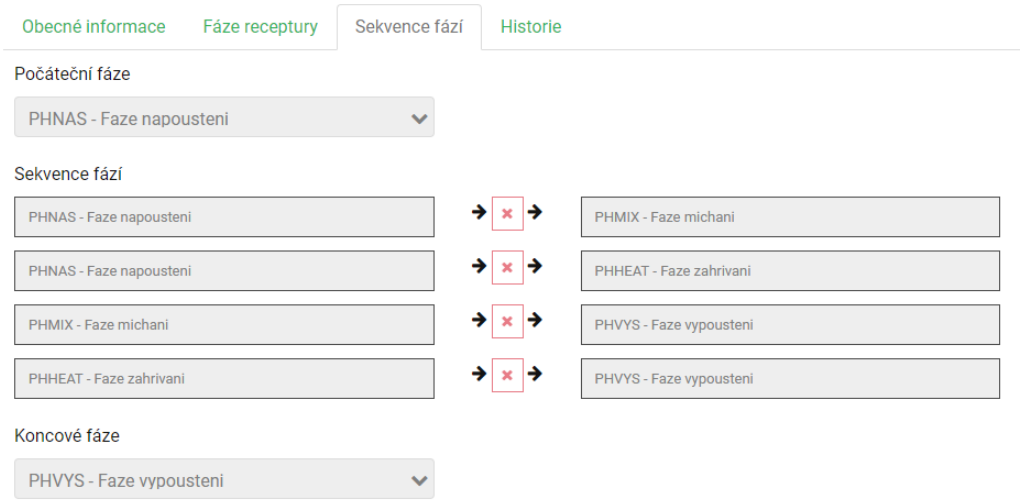
8		Filling	"PhaseFilling"		☐	☑	☑	☑	☐
9		Cmd	Struct		☐	☑	☑	☑	☐
10		State	Struct		☐	☑	☑	☑	☐
11		Param	Struct		☐	☑	☑	☑	☐
12		FillLevel	Real	10.0	☐	☑	☑	☑	☐
13		WaterLevel	Real	0.0	☐	☑	☑	☑	☐

Obr. 12: Parametry a procesní hodnoty fáze napouštění v datovém bloku modulu zařízení



Obr. 13: Konfigurace názvu, parametrů a procesních hodnot ve fázi procedury

Víme tedy nyní, jakým způsobem definujeme jednotlivé fáze nějaké receptury. Nyní je na místě, abychom určili, v jakém pořadí se tyto fáze budou v rámci receptury vyskytovat, potažmo vykonávat. Na záložce Sekvence fází můžete vidět totéž, co na Obr. 14.



Obr. 14: Sekvence fází procedury receptury pasterizace mléka

Způsob definování se jeví jako poněkud chaotický. Je to z důvodu jednoduchosti implementace konfigurační aplikace. Popsaný obrázek tedy říká, že

1) Počáteční fází je fáze napouštění. Tuto fázi bude BATCH spouštět jako první při startu výroby dávky.

2) Po dokončení fáze napouštění se tok řízení rozdělí a budou se vykonávat dvě další fáze souběžně – ZAHŘÍVÁNÍ a MÍCHÁNÍ.

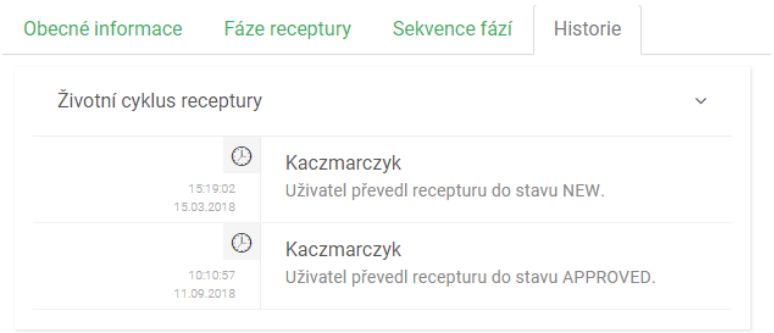
3) Jakmile budou ukončeny obě dvě tyto fáze, tok řízení se opět spojí do fáze VYPOUŠTĚNÍ.

4) Fáze vypouštění bude fází, kterou vykonávání dávky končí.

Pozn.: Je nutné vyplnit počáteční a koncovou fázi!

Pozn.: Při definování vlastní sekvence fází buďte opatrní, program neřeší nekorektní zadání, zacyklení toku apod!

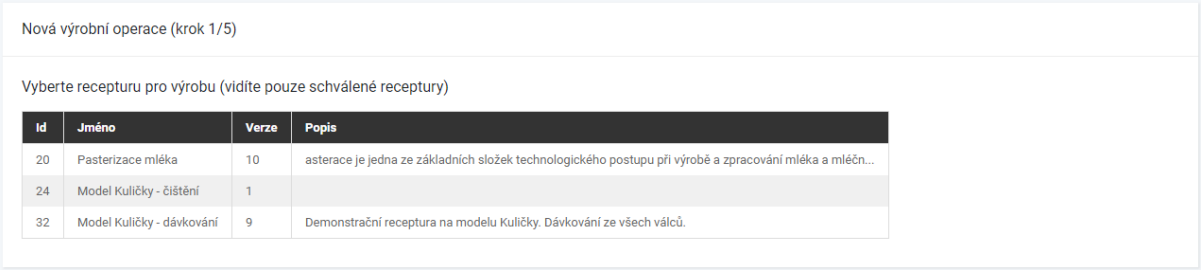
Systém SkuMES sleduje na velice jednoduché bázi životní cyklus každé z receptur. Ve chvíli, kdy je konkrétní receptura převedena uživatelem do některého z výše popsaných stavů, je tato skutečnost zaznamenána v databázi. V záložce Historie příslušné receptury lze následně dohledat, kdo, kdy a do jakého stavu recepturu přepnul.



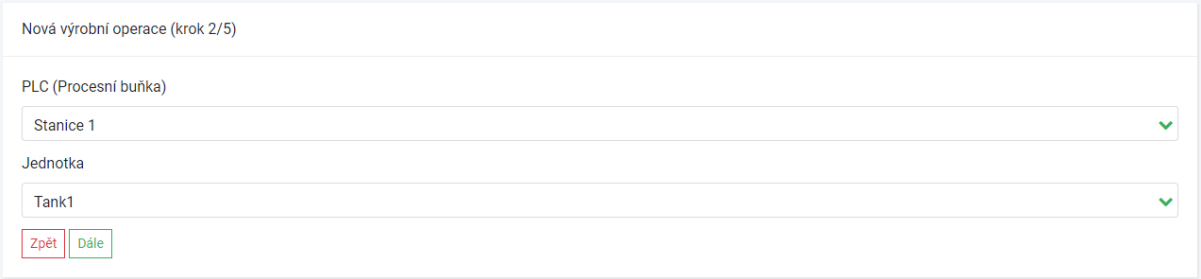
Tím jsme vyčerpali možnosti definování receptur v systému SkuMES. Jen podotkněme, že takto definovaná receptura je **hlavní recepturou**. Z ní budou před výrobou každé dávky vytvářeny klony – **řídicí receptury** specifické pro každou dávku.

12.5. Naplánování výroby

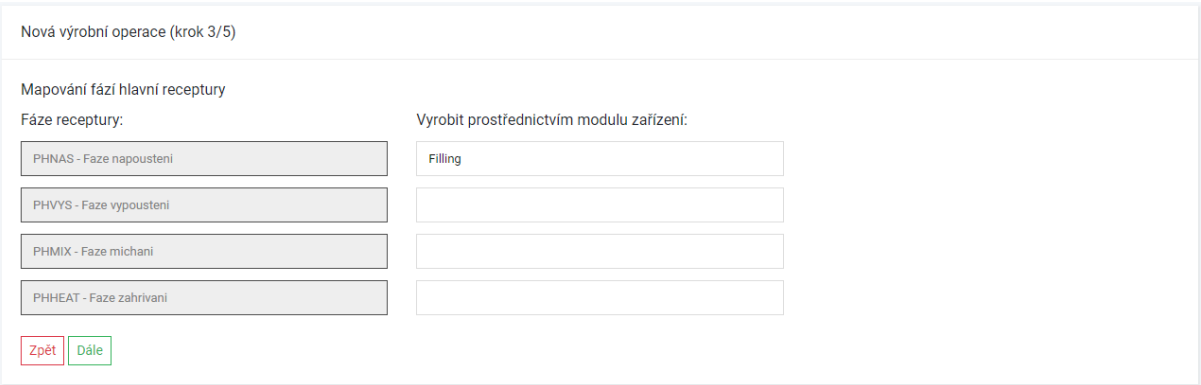
Nyní je na čase vyrobit podle naší receptury dávku. Volbou Plán výroby – Vyrobit z levé nabídky se dostanete do sekce, ve které definujete parametry této dávky. V prvním kroku je potřeba vybrat konkrétní recepturu pro výrobu. Vyberte nyní tedy recepturu pasterizace mléka.



V dalším kroku je potřeba vybrat procesní buňku a konkrétní jednotku, na které bude výroba dávky probíhat. Vyberte tedy své PLC a jednotku Tank1. Stiskněte tlačítko **Dále**.

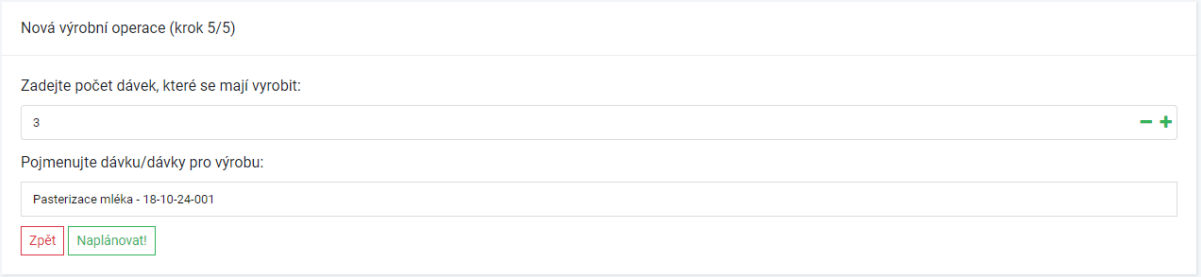


Nyní přicházíme ke kroku, ve kterém přiřadíme jednotlivé fáze procedury receptury ke konkrétním modulům zařízení. Zde dejte pozor, program vás nijak neupozorní, pokud na fázi napouštění namapujete například modul zařízení míchání – nemá jak – BATCH systém netuší, jaká konkrétní funkcionalita se za jednotlivými moduly zařízení skrývá. Všechny mají stejné rozhraní a BATCH je tedy od sebe nemůže nijak rozlišit (což je mimochodem také účelem). Po namapování všech fází stiskněte tlačítko **Dále**.

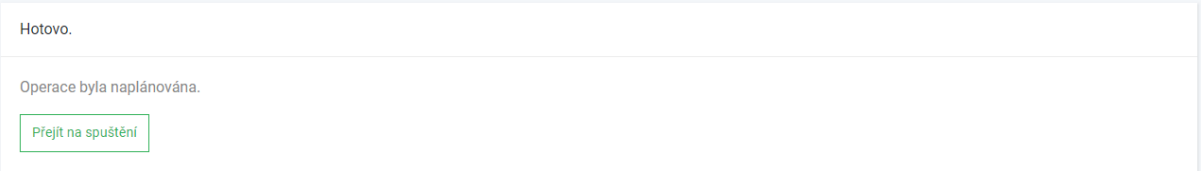


Výrobu lze parametrizovat jednak řazením jednotlivých fází procedury hlavní receptury, jednak nastavením parametrů pro tyto fáze. Existuje však možnost, jak parametry této procedury (mírně) modifikovat, například při kolísání nějaké vlastnosti některého ze vstupních materiálů. Proto je při výrobě konkrétní dávky vytvářen klon hlavní receptury – receptura řídicí. V této řídicí receptuře již nemůžeme ovlivnit pořadí fází, ale je možné provést změnu parametrů. V okně, které nyní vidíte před sebou, vidíte seznam všech parametrů všech fází procedury hlavní receptury. Můžete nyní provést změnu některého z parametrů. Po nastavení stiskněte tlačítko **Dále**.

Posledním krokem je specifikovat, kolik dávek má být vyrobeno s konfigurací, kterou jste nyní provedli. Zadejte tedy příslušný počet (v systému je nastaveno omezení na vytvoření max. 5 dávek). Svou dávku (či dávky) také smysluplně pojmenujte tak, abyste je dokázali odlišit od dávek vytvořených na jiném pracovišti. Nakonec stiskněte tlačítko **Naplánovat!**



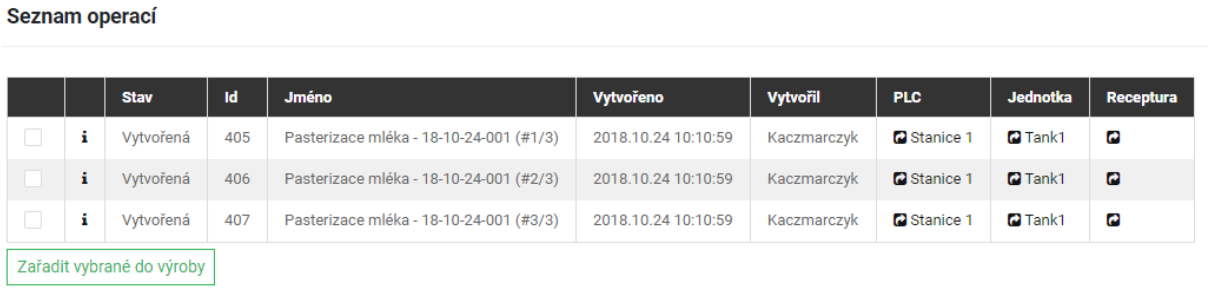
Tím je tedy plánování dokončeno. Stiskem tlačítka **Přejít na spuštění** se dostanete do formuláře **Seznam operací**.



12.6. Spuštění plánovaných operací

Na stránce **Vyrobit – Seznam operací** vidíte seznam všech dávek, které ještě nebyly vyrobeny a jejich aktuální stav. Můžete si všimnout, že ve chvíli, kdy vytvoříte více dávek hromadně, jsou jednotlivé dávky odlišeny přidáním indexu na konec jejich jména. Z každé dávky můžete odkazem přejít na specifikaci konkrétní procesní buňky, jednotky a příslušné receptury. Stiskem tlačítka se pak

dostanete k detailům výroby konkrétní dávky. Můžete vidět, že v případě dávky ve stavu Vytvořená nejsou k dispozici žádné detaily.



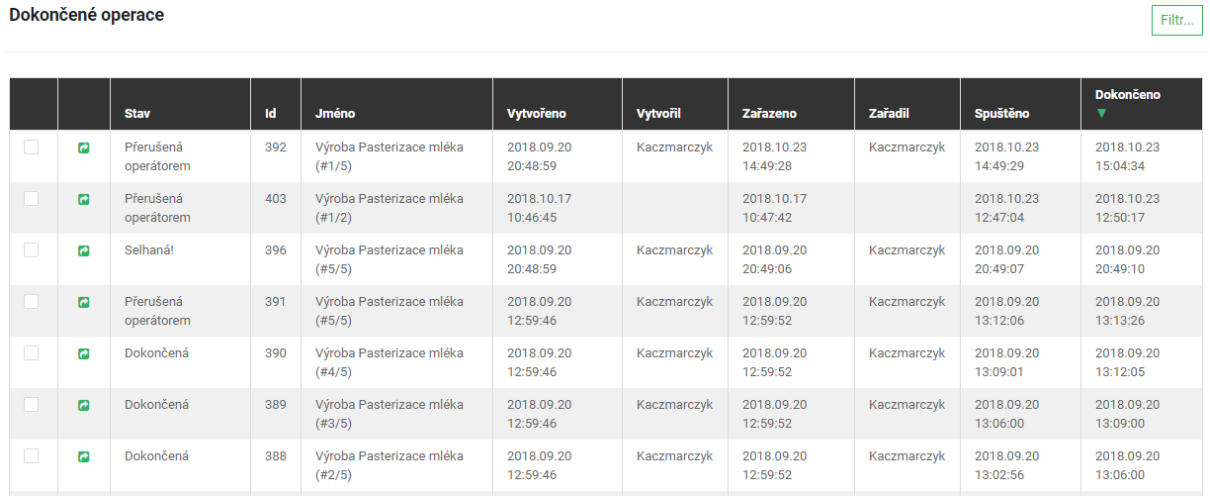
Výrobu konkrétní dávky/dávek spustíte tak, že změníte stav zaškrtávače na příslušném řádku na ☑ a stisknete tlačítko **Zařadit vybrané do výroby**.

12.7. Sledování průběhu výroby

Sledujte!

12.8. Report z výroby

Dokončené dávky zobrazíte výběrem **Plán výroby – Vyrobeno** z hlavní nabídky. V tabulce jsou uvedeny všechny dávky, které již byly vyrobeny, nebo jejichž výroba nějakým způsobem selhala. Klepnutím na ikonu u příslušné dávky můžete přejít na report z výroby.



Report následně obsahuje několik sekcí:

- Obecné informace – zejména časové údaje a osoby zodpovědné za plánování a výrobu, dále pak detailně identifikuje použitou hlavní recepturu
- Parametry řídicí receptury – které mohly být nastaveny jinak, než tomu je u receptury hlavní

- Podrobný průběh – seznam startu a ukončení jednotlivých fází výroby na modulech zařízení. V této sekci je rovněž tlačítko, pomocí kterého lze exportovat do souboru .csv časový průběh sledovaných procesních hodnot.
- Podpis
- Štítky s čárovými kódy – lze jimi označit např. zásobník, do kterého byl hotový produkt přemístěn.

Vytiskněte si report z výroby vaší dávky. Bude sloužit jako příloha k dokumentaci.