UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

LABORATORNÍ ÚLOHA PLC MODICON

Tomáš Novák

Bakalářská práce
2016

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Novák**
Osobní číslo:	**I13069**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**Laboratorní úloha PLC Modicon**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cíl: Cílem je vytvořit laboratorní úlohu s PLC Modicon fy Schneider Electric.
Obsah teoretické části: Teorie logického řízení. Programovatelné logické automaty a inteligentní relé - hardware a programování. HMI interface.
Obsah implementační části: Instalace softwaru pro programování PLC, simulaci a vizualizaci úlohy logického řízení. Vytvoření laboratorní úlohy - jak po hardwarové tak i softwarové stránce.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

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

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

Datum zadání bakalářské práce: **20. listopadu 2015**
Termín odevzdání bakalářské práce: **13. května 2016**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2016

Prohlášení

Prohlašuji:

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 13. 5. 2016

Tomáš Novák

ANOTACE

Bakalářská práce je věnována tvorbě laboratorní úlohy logického řízení pomocí programovatelného logického automatu Modicon TM241. Je uveden obecný popis hardwaru a softwaru a jsou navrženy laboratorní úlohy pro studenty včetně zadání a vypracování. Čtenář získá základní zkušenosti s vývojovým prostředím SoMachine. Pomocí řešených příkladů si osvojí nejen psaní programů ve všech poskytovaných jazycích, ale také tvorbu aplikací pro grafický panel Magelis.

KLÍČOVÁ SLOVA

programovatelný logický automat, grafický panel, programovací jazyky, SoMachine

TITLE

LABORATORY EXCERCISE PLC MODICON

ANNOTATION

Bachelor thesis is dedictated to the creation of laboratory excercise with programmable logic controller Modicon TM241. It presents general description of the hardware and software and laboratory excercises for students including the assignment and elaboration are prepared. The reader obtains the basic skills with development enviroment SoMachine. He learns not only how to write programs in all provided languages but also how to create aplications to the graphic panel Magelis.

KEYWORDS

Programmable logic controller, Graphic panel, Programming languages, SoMachine

Obsah

Seznam zkratek

DRAM	Dynamická RAM
HMI	Human Machine Interface
POU	Programmable Organization Unit
PLC	Programmable Logic Controller
RAM	Random Access Memory
SRAM	Statická RAM
USB	Universal Serial Bus

Seznam značek

f	frekvence, Hz
I	elektrický proud, A
U	elektrické napětí, V

Seznam obrázků

Seznam tabulek

ÚVOD

Cílem této bakalářské práce je vytvořit laboratorní úlohu pro studenty pomocí programovatelného logického automatu (PLC) a grafického panelu (HMI). Studenti se naučí pracovat s vývojovým prostředím, vyřeší úlohu pomocí dostupných programovacích jazyků, naprogramují PLC a vytvoří program pro grafický panel.

Zařízení, které byly vybrány, jsou od firmy Schneider Electric. Konkrétně se jedná PLC Modicon M241 TM241CEC24T a grafický panel Magelis HMISTU HMIS5T. K zařízením jsou dále k dispozici dvě tlačítka a dvě kontrolky.

1 LOGICKÉ ŘÍZENÍ

Řízení technologických procesů představuje zpětnovazební systém. Vlastní technologický proces má na vstupu vstupní zdroje a generuje výstupní produkty na výstupu. Na proces mohou samozřejmě působit poruchy dané jak rozdílnou kvalitou vstupních zdrojů, tak i rozdílnými podmínkami funkce vlastního procesu způsobených vnějším působením prostředí. Vlastním cílem řízení technologického procesu je na základě řídicích veličin (programu) a stavu technologického procesu (stavové veličiny) řídit akčními veličinami technologický proces tak, aby výstup odpovídal požadovanému zadání (Bayer, 2008).

V automatizační technice existují takové technologické procesy, jejichž řízení lze realizovat logickým řídícím obvodem. Logický řídicí obvod je takový obvod, u kterého vstupní a výstupní proměnné nabývají pouze hodnot 0 a1.

Podle chování se logické obvody dělí na kombinační logické obvody a sekvenční logické obvody.

1.1 FORMY POPISU LOGICKÉ FUNKCE

Logickou funkci lze popsat 5. následujícími způsoby:

- Pravdivostní tabulka
- Logický výraz
- Seznam indexů vstupních vektorů
- Mapa logické funkce
- Vícerozměrná krychle

Pravdivostní tabulka

Pravdivostní tabulka je nejpřirozenější forma zápisu logické funkce. Obsahuje výčet všech vstupních proměnných a jim odpovídajícím výstupů.

Je to tedy 2^n řádků a n sloupců pro n vstupních proměnných a m sloupců pro m výstupních proměnných.

Logický výraz

Logický výraz je algebraickým vyjádřením logické funkce. Tvoří model chování a i model struktury.

Term je člen logického výrazu neboli booleovský výraz. Rozeznáváme term:

- **Minterm** – součinový term, kombinace proměnných v přímé nebo negované formě
- **Maxterm** – součtový term, kombinace proměnných v přímé nebo negované formě

Každou úplnou logickou funkci lze vyjádřit výrazem ve tvaru úplné **normální disjunktivní** nebo úplné **normální konjunktivní formy**.

- Disjunktivní – součet všech mintermů, které nabývají hodnoty 1
- Konjunktivní – součin všech maxtermů, které nabývají hodnoty 0

Seznam indexů vstupních vektorů

Seznam indexů představuje zjednodušený zápis pravdivostní tabulky. Uvádí seznam indexů, pro které logická funkce:

- nabývá hodnoty 1 $$f(x_1 x_2 x_3) = \Sigma(2,3,5,7) \quad (1.1)$$
- nabývá hodnoty 0 $$f(x_1 x_2 x_3) = \Pi(2,3,5,7) \quad (1.2)$$

Mapa logických funkcí

Vznikne tak, že se proměnné logické funkce rozdělí na dvě skupiny, přičemž se každé kombinaci jedné skupiny přiřadí sloupec a každé kombinaci druhé skupiny se přiřadí řádek. Průnik řádků a sloupců odpovídá kombinaci všech proměnných a odpovídá jednomu řádků v pravdivostní tabulce. Do čtverečku se vpisuje hodnota logické funkce pro danou kombinaci vstupních proměnných. Logická funkce může nabývat hodnot:

- logická „1“,
- logická „0“,
- neurčitý stav „-“.

Podle kódování proměnných se dělí mapy na:

Tab. 1.1 – Karnaughova mapa

$x_3\ x_2 \setminus x_1\ x_0$	00	01	11	10
00	0	1	3	2
01	4	5	6	7
11	12	13	15	14
10	8	9	11	10

Tab. 1.2 – Svobodova mapa

$x_3\ x_2 \setminus x_1\ x_0$	00	01	10	11
00	0	1	2	3
01	4	5	6	7
10	8	9	10	11
11	12	13	14	15

Vícerozměrné krychle

Pro n-proměnných je to n-rozměrná krychle, která vznikne vytvořením 2^n vektorů, přičemž souřadnice jsou dány hodnotami vstupních proměnných. Koncové body těchto vektorů leží na vrcholech jednotkové n-rozměrné krychle.

Toto vyjádření je poměrně složité a používá se zejména pro demonstraci bezpečnostních kódů.

Více informací o popisu logických se nachází v (Bayer, 2008).

1.2 MINIMALIZACE LOGICKÝCH VÝRAZŮ

Účelem minimalizace je minimalizovat logický výraz s cílem získat co nejjednodušší výraz pro původní logickou funkci a tím i tvar logické funkce pro minimální realizační strukturu. Podrobněji se touto problematikou zabývá (Bayer, 2008).

Minimalizační metody

1. Minimalizace s využitím Booleovy algebry

Pro minimalizaci se využívá jednotlivých zákonů Booleovy algebry.

Zákony:

$a + a = 0$	$a \cdot a = 0$	tautologie
$a + b = b + a$	$a \cdot b = b \cdot a$	komutativní zákon
$(a + b) + a = a + (b + c)$	$(a \cdot b) \cdot a = a \cdot (b \cdot c)$	asociativní zákon
$a + (a \cdot b) = a$	$a \cdot (a + b) = a$	zákon absorpce
$a \cdot 0 = 0$	$a + 1 = 1$	agresivnost 0 a 1
$a + 0 = 0$	$a \cdot 1 = a$	neutrálnost 0 a 1
$a \cdot (b + c) = ab + ac$	$a + (b \cdot c) = (a + b) \cdot (a + c)$	distributivní zákon
$a \cdot \bar{a} = 0$	$a + \bar{a} = 1$	zákon vyloučeného třetího
$\overline{abc} = \bar{a} + \bar{b} + \bar{c}$	$\overline{a + b + c} = \bar{a} \cdot \bar{b} \cdot \bar{c}$	De Morganovy zákony

2. Minimalizace výrazů s využitím map

Princip minimalizace spočívá v hledání smyček. To je taková oblast, která obsahuje 2^k políček. Snaha je získat co nejmenší počet co největších minimalizačních smyček, které zahrnují veškeré jedničky v mapě.

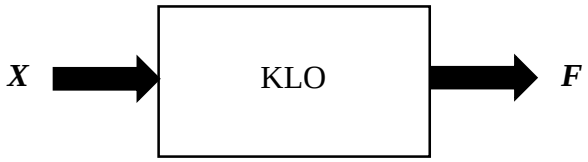
Tab. 1.3 – Minimalizace Karnaughovy mapa

$x_3\ x_2 \setminus x_1\ x_0$	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	1	1	1	1
10	0	1	1	0

$$f = x_2 + x_1x_2 + x_3x_1 \tag{1.3}$$

1.3 KOMBINAČNÍ LOGICKÉ OBVODY

Kombinační logické obvody jsou takové obvody, ve kterých stavy na výstupu závisí pouze na okamžitých kombinacích vstupních proměnní. Nezáleží zde na předchozích stavech vstupních proměnných. Takže jedné kombinaci vstupních proměnných odpovídá jeden stav na výstupu. Blokové schéma kombinačního logického obvodu je uvedeno na obr. 1.1 (Bayer, 2008).



Obr. 1.1 – Kombinační logický obvod

Kde ***X*** – Množina vstupních proměnných

F – Množina výstupních proměnných

KLO – Kombinační logický proměnných

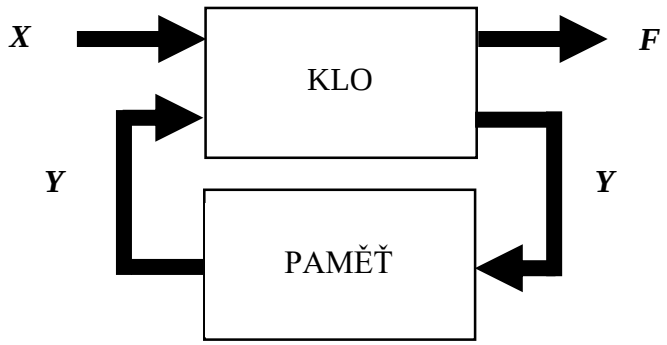
1.3.1 Návrh kombinačního logického obvodu

Z pravdivostní tabulky se vytvoří Karnaughova mapa, ze které vznikne minimalizovaný výraz pro výstupní logickou funkci.

1.4 SEKVENČNÍ LOGICKÉ OBVODY

Sekvenční logické obvody jsou takové obvody, ve kterých stavy na výstupu závisí na okamžitých kombinacích vstupní proměnných a na jejich předchozích stavech – vnitřních proměnných. To znamená, že si musí předchozí stavy pamatovat a musí mít tedy **paměť**.

Sekvenční logický obvod tedy tvoří obvod složený z kombinačního logického obvodu a paměti. Blokové schéma sekvenčního logického obvodu je uvedeno na obr. 1.2 (Bayer, 2008).



Obr. 1.2 – Sekvenční logický obvod

Kde ***X*** – Množina vstupních proměnných

F – Množina výstupních proměnných

Y – Množina vnitřních proměnných

KLO – Kombinační logický obvod

Chování sekvenčního logického obvodu popisují dvě soustavy (kombinačních) logických funkcí. První z nich je **soustava výstupních funkcí**, druhou je **soustava přechodových funkcí**, která určuje vstupy pro paměťové obvody – přechody mezi vnitřními stavy sekvenčního logického obvodu.

Podle tvaru výstupních funkcí se obvody rozlišují na typ **Mealy** a **Moore**.

- Mealy – výstupní funkce jsou funkcemi vnitřních i vstupních stavů sekvenčního logického obodu
- Moore – výstupní funkce jsou funkcemi pouze vnitřních stavů sekvenčního logického obvodu

Další dělení je na asynchronní a synchronní obvody. V asynchronním režimu není takt určován vnějším taktovacím signálem, ale ke změnám vnitřních stavů dochází bezprostředně po změnách vstupních stavů. V synchronním režimu se mohou kombinační funkce ustálit a ke změnám vnitřních stavů dochází dle taktu vnějšího taktovacího (hodinového) signálu.

1.4.1 Návrh sekvenčního logického obvodu

V tomto pododdílu budou uvedeny jednotlivé kroky návrhu sekvenčního logického obvodu. Podrobnější informace o návrhu jsou uvedeny v (Bayer, 2008).

1. Tabulka přechodů

Tabulka přechodů umožňuje zadat chování sekvenčního logického obvodu, což znamená zobrazení příštího stavu a výstupu v tabulkové formě. Počet sloupců této tabulky odpovídá počtu kombinací všech vstupních proměnných. Počet řádků odpovídá počtu kombinací všech vnitřních proměnných. Stabilní stavy jsou zapisovány tučně, přechody se zapisují bez úprav.

Tab. 1.4 – Tabulka přechodů

Stav \ x_2 x_1	00	01	11	10	Z
1	**1**	5	6	2	0
2	1	**-**	-	**2**	1
3	**-**	5	**3**	2	0
4	**4**	5	3	-	0
5	4	**5**	-	-	1
6	-	5	**6**	2	1

2. Redukovaná tabulka přechodů

Cílem redukce stavů je vyloučit nadbytečné stavy a tím zjednodušit realizaci obvodu. Slučují se ty řádky, pokud mají stabilní stav pro stejný vstupní vektor a všechny přechody pro ostatní vstupní vektory jdou do stejných stavů, ekvivalentních stavů nebo neurčitých stavů.

Podle pravidel lze v tab. 1.4 do jednoho řádku sloučit řádky 1, 2, 3 a 3, 4, 5, viz tab. 1.5.

Tab. 1.5 – Redukované stavy

Stav \ x_2 x_1	00	01	11	10
1, 2, 6	**1**	5	**6**	**2**
3, 4, 5	**4**	**5**	**3**	2

3. Kódování vnitřních stavů

Nejdříve se podle počtu vnitřních stavů z redukované tabulky přechodů zvolí počet vnitřních proměnných tak, aby jejich kombinace pokryly všechny vnitřní stavy sekvenčního logického obvodu. Pro počet vnitřních proměnných tedy platí:

$$s \leq 2^{P_{\min}} \tag{1.1}$$

kde **s** je počet stavů a $\boldsymbol{P_{min}}$ je minimální počet vnitřních proměnných. V tab. 1.5 jsou po úpravě pouze dva vnitřní stavy, takže podle rovnice 1.1 postačí jedna vnitřní proměnná.

Tab. 1.6 – Mapa přiřazení

Stav	1, 2, 6	3, 4, 5
Q	0	1

4. Mapa vnitřních proměnných

Mapa vnitřních proměnných vzniká tak, že se v redukované tabulce přechodů nahradí všechny stavy a přechody kódovými kombinacemi vnitřních proměnných podle mapy přiřazení.

Tab. 1.7 – Mapa vnitřních proměnných

$Q \setminus x_2\, x_1$	00	01	11	10
0	**0**	1	**0**	**0**
1	**1**	**1**	**1**	0

Z mapy vnitřních proměnných se vytváří obvody pro paměťové prvky. Obvykle se jedná o klopné obvody typu RS, JK nebo D. Každý má své pravidla na tvoření smyček v Karnaughových mapách. Níže jsou uvedena pravidla jednotlivých klopných obvodů:

a) RS
   - R – berou se všechny **0**, některé 0 a neurčité stavy
   - S – berou se všechny **1**, některé 1 a neurčité stavy

b) JK
   - J – berou se všechny **1**, některé 1, **0** a neurčité stavy
   - K – berou se všechny **0**, některé 0, **1** a neurčité stavy

c) D
   - berou se všechny **1**, 1 a může obsahovat neurčité stavy

5. Mapa výstupní funkce

Pro každou výstupní funkci zvlášť se vytvoří mapa výstupní funkce. Mapa vychází z redukované tabulky přechodů. Za stabilní stavy se dosadí hodnota výstupu, která je stavu přiřazena. Z přechodů vznikne neurčitý stav.

Tab. 1.8 – Mapa výstupní funkce Z

Z				
$Q \setminus x_2\, x_1$	00	01	11	10
0	0	-	1	1
1	0	1	0	-

2 PLC

PLC (Programable Logic Controller) je digitální elektronické zařízení, které využívá programovatelnou paměť pro ukládání instrukcí a implementaci specifických funkcí (např. logických, sekvenčních, časovacích, čítacích nebo aritmetických), pomocí kterých je díky digitálním či analogovým vstupům/výstupům schopno ovládat a kontrolovat různé typy zařízení či různé procesy.

PLC jsou určeny pro nasazení do tvrdých podmínek průmyslového prostředí, tomu odpovídá jejich konstrukce, především robustnost a odolnost proti rušení (Šmejkal, 1999; Zemanová, 2010).

2.1 PROČ PRÁVĚ PLC?

- **Rychlá realizace**

Hlavní předností PLC je možnost rychlé realizace systému. Technické vybavení nemusí uživatel vyvíjet. Stačí navrhnout a objednat vhodnou sestavu modulů do programovatelného automatu pro danou aplikaci, vytvořit projekt, napsat a odladit uživatelský program – a pak to vše realizovat.

- **Spolehlivost, odolnost, diagnostika**

Technické vybavení programovatelných automatů je navrženo tak, že jsou extrémně spolehlivé i v drsných průmyslových podmínkách, jsou odolné proti rušení i poruchám, vyznačují se robustností a spolehlivostí. PLC bývají vybaveny i vnitřními diagnostickými funkcemi, které průběžně kontrolují činnost systému a včas dokáží zjistit případnou závadu.

- **Nekončící změny v zadání**

Jen výjimečně se podaří, že první varianta řešení zůstane tou poslední. Mnohé nedostatky se právě zjistí ve fázi finalizace projektu, kde do styku s projektem přijdou lidi, kteří neměli přístup k vývoji a všimnou si nedostatečných detailů či funkcí.

Při použití programovatelného automatu obvykle stačí pouze opravit, změnit nebo rozšířit uživatelský program. Pokud požadavky vyžadují více nových vstupů a výstupů, není problém rozšíření o modul, který vyžadujeme. Případně lze doplnit další PLC jako podsystém (Šmejkal, 1999).

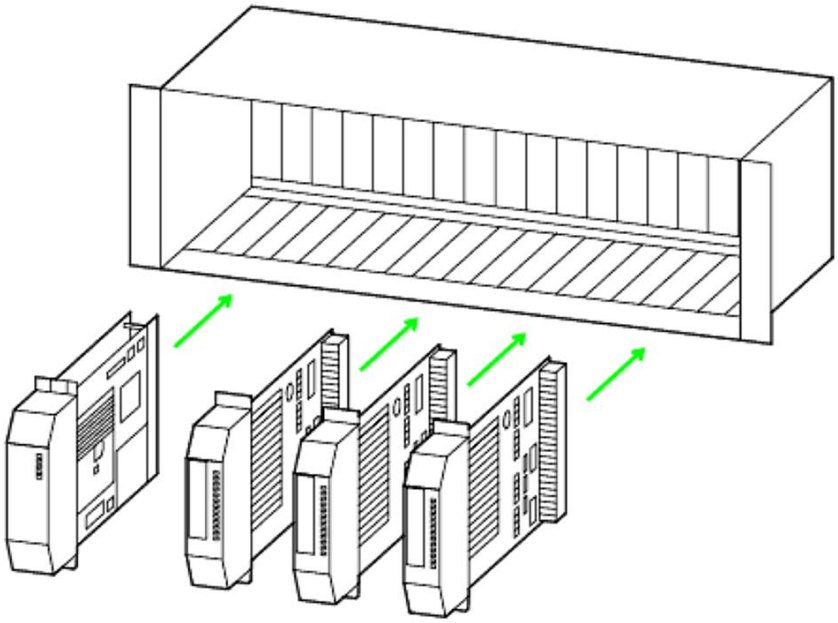
2.2 TYPY PLC

Sestava PLC se může skládat z několika modulů v případě **modulárního** typu PLC, nebo se může jednat i o **kompaktní** provedení PLC, které má již integrovány potřebné komponenty.

2.2.1 Modulární PLC

Modulární PLC jsou tvořeny několika moduly sestavenými do jednoho celku, kde jeden modul musí být CPU a k němu musí být připojené periferní moduly. Periférie jsou k CPU připojeny v jedné řadě za sebou pomocí směrnice. Pokud již není místo na potřebné moduly, lze PLC doplnit o rozšiřující rám, který umožní připojení dalších modulů.

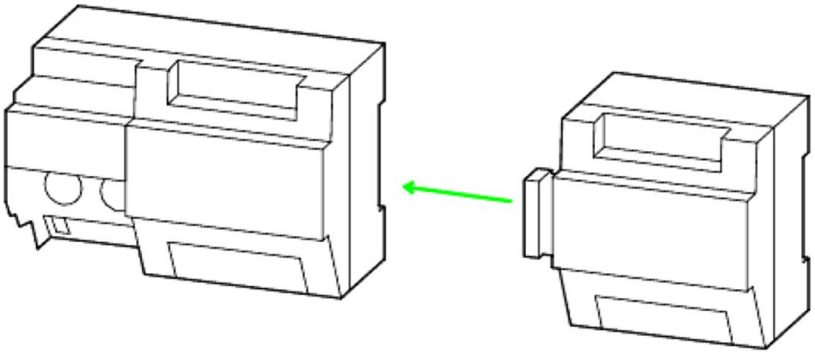
Tuto volbu PLC lze rozšířit o více modulů než kompaktní PLC.



Obr. 2.1 – Ukázka modulárního PLC (Zemanová, 2010, s. 22)

2.2.2 Kompaktní PLC

Kompaktní PLC je takové, že v jednom modulu je obsaženo nejen CPU, ale i určitý počet digitálních vstupů/výstupů případně i analogových vstupů/výstupů. PLC lze dále rozšířit pomocí několika dalších funkčních modulů.



Obr. 2.2 – Ukázka kompaktního PLC (Zemanová, 2010, s. 19)

2.3 PROGRAMOVÁNÍ PLC

Základem každého PLC je centrální jednotka, která poskytuje zařízení inteligenci. Centrální jednotka obsahuje mikroprocesor, mikrořadič, nebo jiný specializovaný řadič. Jeho programem jsou realizovány všechny funkce, které má uživatel k dispozici, tj. kompletní soubor instrukcí PLC, jeho systémové služby, časové a komunikační funkce.

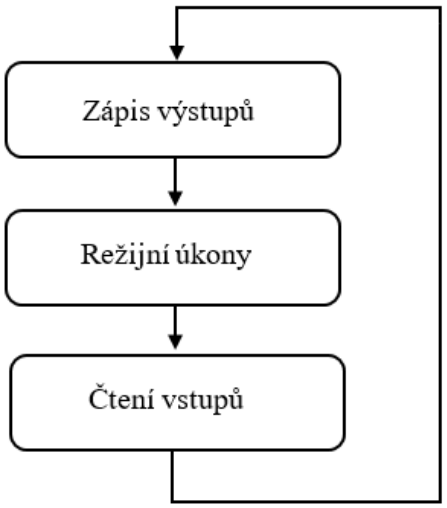
Paměťový prostor, který PLC poskytuje, je rozdělen na části. První část je určena pro uložení PLC programu, datových bloků a tabulek. Její obsah se obvykle během vykonávání programu nemění a její obsah je možné měnit v editačním režimu, když program „neběží“. Druhá část paměti je operační. Obsah operační paměti se dynamicky mění působením uživatelského a systémového programu. Jsou v ní lokalizovány uživatelské registry, čítače a časovače, obrazy vstupů a výstupů a jiné další systémové proměnné.

Protože byly PLC původně k realizaci logických úloh a k náhradě pevné logiky, nechybějí v žádném PLC instrukce pro základní logické operace s bitovými operandy. V dnešní době nabízejí PLC mnohem více instrukcí. Obvykle nechybí ani instrukce pro aritmetiku a operace s čísly. Některé PLC poskytují i velmi výkonné instrukce například pro komplexní operace např. pro fuzzy logiku a fuzzy regulaci (Šmejkal, 1999).

2.4 VYKONÁVÁNÍ PROGRAMU

Program PLC je posloupnost instrukcí a příkazů jazyka. Typickým režimem je cyklické vykonávání v programové smyčce. Programátor PLC se nemusí starat o to, aby se po konci programu vrátilo jeho vykonávání opět na začátek. To zajišťuje systémový program. Naopak každé dlouhé setrvání programu v programové smyčce je vyhodnoceno jako „fatální chyba“ a systém tento stav označuje jako překročení doby cyklu. Tuto dobu určuje programátor.

Cyklické vykonávání programu je znázorněno na obr. 2.3. Vždy po vykonání poslední instrukce uživatelského programu je předáno řízení systémovému programu, který provede tzv. otočku cyklu. V ní nejprve aktualizuje obrazy hodnot výstupů a vstupů. Po otočce cyklu je opět předáno řízení prvé instrukci uživatelského programu.



Obr. 2.3 – Vykonávání programu PLC

PLC program je vykonáván v cyklu. Po spuštění programu jsou nejprve provedeny režijní operace systému, aktualizace systémových a časových proměnných, naplánovaná aktivace procesů pro další cyklus, apod. Poté jsou sejmuty aktuální hodnoty vstupů, které jsou pro celý následující cyklus uchovány, jako obrazy vstupů. Následně se vykoná požadovaný program a na závěr jsou na výstupy vyslány aktuálně vyčíslené hodnoty obrazů výstupů. Celý cyklus se stále opakuje (Šmejkal, 1999).

2.5 PROGRAMOVACÍ JAZYKY PLC

K programování nabízejí PLC systémy specializované jazyky. Jazyky systémů různých výrobců si jsou podobné, ovšem ne úplně stejné. Programy tedy nelze mezi PLC jiných výrobců přenášet. PLC také nemusí podporovat všechny programovací jazyky.

V tomto oddílu budou postupně popsány jednotlivé programovací jazyky, které jsou nabízeny výrobci.

Rozdělení programovacích jazyků:

- **Textové**
  - IL – Instruction List (posloupnost instrukcí)
  - ST – Structured Text (vyšší programovací jazyk – obdoba Pascalu)

- **Grafické**
  - LD – Ladder Diagram (liniové či reléové schéma)
  - FBD – Function Block Diagram (schéma funkčních bloků)
  - CFC – Countinous Function Chart
  - SFC – Sequential Function Chart (nástroj sekvenčního programování)

2.5.1 IL – Instruction List

Zápis programu označovaný jako Instruction List je nejzákladnějším popisem programu. Program se skládá z posloupnosti jednotlivých operací tvořených základními instrukcemi (např. MOV pro přesun dat mezi registry, ADD pro sečtení dvou hodnot apod.) a operandy, které reprezentují jednotlivé registry a paměťová místa. Tento jazyk se více méně podobá jazyku Assembler, který je slouží pro programování mikrokontrolérů.

Tento způsob programování nám umožňuje mít dokonalý přehled o tom, co se v programu děje. Umožňuje teda maximální optimalizaci programu jak na jeho velikost, i rychlost vykonání.

Každá instrukce začíná na novém řádku a obsahuje operátor a v závislosti na typu operace, jeden nebo více operandů oddělených od sebe čárkou (Vojáček, 2011).

Výhody

- Přesná definice chování programu
- Paměťově i na rychlost úsporný program

Nevýhody

- Nutná znalost a orientace v příkazech a registrech
- Zdlouhavé kódy
- Horší přehlednost programu a orientace v něm

2.5.2 ST – Structured Text

Jazyk strukturovaného textu je obdobou vyšších programovacích jazyků pro PC či mikrokontroléry. Programy jsou tvořeny posloupností instrukcí, kde každá instrukce reprezentuje v IL jazyku celou posloupnost základních instrukcí. Jde tedy o vyšší programovací jazyk. Se svojí syntaxí připomíná programovací jazyk Pascal.

Tento jazyk je velice vhodný pro práci s daty, řetězci, databázemi a pro naprogramování složitých výpočetních algoritmů. Používá se tedy například pro náročné zpracování analogových signálů (Vojáček, 2011).

Výhody

- Snadné programování složitých vzorců a operací
- Ideální pro složité zpracování analogových signálů
- Ideální pro práci s velkými bloky dat
- Vhodný pro práci s textovými řetězci
- Vhodný pro realizaci/zpracování datové komunikace

Nevýhody

- Je nutné znát příkazy a přesnou syntaxi zápisu
- Horší přehlednost logických operací
- Méně vhodný pro přehledné zpracování velkého množství logických signálů

2.5.3 LD – Ladder Diagram

Ladder Diagram patří mezi základní programovací jazyk pro PLC a spolu s IL jsou nejrozšířenější. Zápis programu pomocí LD vychází z dob reléové techniky, kdy se místo procesorů či mikrokontrolérů nebo integrovaných logických obvodů používala soustava vzájemně propojených relátek. S jejich pomocí lze jednoduše realizovat základní logické operace typu AND, OR, NAND, NOR. Svou strukturou je tento zápis programu velice vhodný pro přehledné zpracování velkého množství logických vstupů/výstupů a jejich čítání, časování.

Hlavní výhodou tohoto programu je jasně definovaná posloupnost programu a jeho přehlednost. Je vhodný spíše pro programování jednodušších logických operací. Pokud v programu převažují složitější operace, je pak program velice obsáhlý a ztrácí svou přehlednost (Vojáček, 2011).

Výhody

- Jasně definovaná posloupnost zápisu, kterou nelze porušit
- Přehlednost programu (u menších programů)
- Velmi rychlé programování logických operací s funkcemi čítání a časování
- Ideální pro zpracování velkého počtu logických signálů

Nevýhody

- Nevhodný pro aritmetické operace a práce s daty
- U složitějších programů narůstá jeho délka

2.5.4 FBD – Function Block Diagram

Zápis programu pomocí FBD spočívá posloupnost programu v soustavě funkčních bloků, které realizují různé funkce. V podstatě je to obdoba LD jazyku, celý program je také organizován do řádků a u logických operací jsou použity přímo funkční bloky AND, NAND, OR, NOR, namísto sério-paralelní kombinace kontaktů. Tento zápis je tedy vhodný pro ty, kteří potřebují vlastnosti reléového schéma, tedy přehlednost a pevně dané členění, ale více mu vyhovuje zápis pomocí funkčních bloků (Vojáček, 2011).

Výhody

- Logické operace v podobě hradel
- Přehledný zápis programu
- Ideální pro zpracování velkého počtu logických signálů

Nevýhody

- Méně vhodný pro složitější zpracování analogových signálů
- Nevhodný pro programování složitých algoritmů (vzorců)
- Nevhodný pro hromadnou manipulaci s velkým množstvím dat

2.5.5 CFC – Continous Function Block

Zápis programu pomocí CFC je podobný jazyku FBD. Na rozdíl od FBD nemá pevně dané umístění komponent, takže je programátor může umisťovat kdekoli v POU. Vzhledem k tomu, že komponenty mohou být umístěny kdekoliv, pořadí provedení je určeno podle přidání komponenty. Jinak řečeno, co se dřív přidá, to se dřív provede. Pořadové číslo je umístěno v pravé horní části komponenty. Pokud pořadí provádění není správně, může být upraveno uživatelem (Vojáček, 2011).

Výhody

- Volné umístění komponent
- Kontrola nad pořadím provádění
- Grafické boxy pro komplexní funkce

Nevýhody

- Méně vhodný pro složitější zpracování analogových signálů
- Nevhodný pro hromadnou manipulaci s velkým množstvím dat
- Nevhodný pro realizaci datové komunikace

2.5.6 SFC – Sequential Function Chart

Programovací jazyk SFC je graficky orientovaný jazyk. Struktura programu připomíná vývojový diagram, který je tvořen posloupností kroků (Steps), oddělených jednotlivými přechodovými podmínkami (Transitions). Ke krokům mohou být připojeny akce (Actions).

Tento režim je ideální pro definování posloupnosti volání jednotlivých částí programu (podprogramů) napsaných některým z jazyků pro PLC (Vojáček, 2011).

Výhody

- Velmi přehledný zápis chování programu
- Přehledné definování a ošetření různých stavů programů
- Ideální pro realizaci sekvenční logiky
- Vhodný pro práci s ASCII řetězci

Nevýhody

- Nevhodný pro přímou realizaci zpracování analogových signál
- Nevhodný pro zpracování velkého počtu logických signálů
- Nevhodný pro programování složitých algoritmů

3 POUŽITÁ ZAŘÍZENÍ

V této práci jsou použity zařízení od firmy Schneider Electric. Konkrétně se jedná o PLC Modicon M241 typu TM241CEC24T a zadní modul Magelis HMIS5T s dotykovým displejem Magelis HMI S85.

3.1 MODICON M241 TM241CEC24T

Programovací jazyky pro PLC Modicon M241 jsou definovány podle normy IEC 61131-3. V normě jsou obsaženy tyto jazyky: IL, LD, ST, FBD a SFC. Nicméně, SoMachine umožňuje programování ještě dalším programovacím jazykem a to CFC.

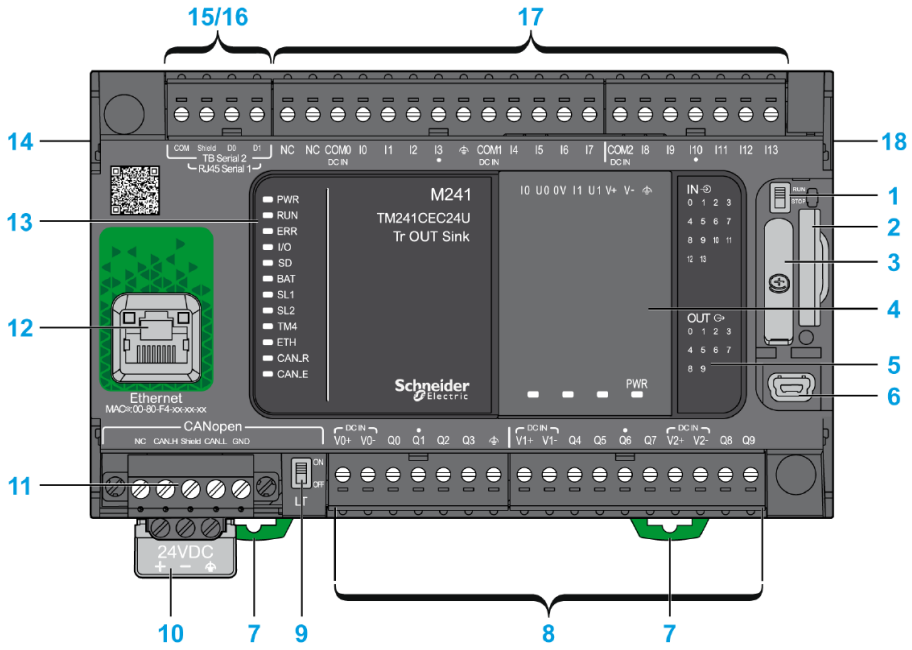
PLC je napájeno stejnosměrným napětím 24 V. Pro uložení aplikace a konfiguračních souborů se v zařízení nachází paměť typu Flash o velikosti 128 Mbytes. Na data aplikace, které jsou generovány během běhu aplikace, je v zařízení paměť typu RAM o velikosti 64 Mbytes.

Na PLC je také vestavěná sada digitálních vstupů a výstupů. Napěťový rozsah na vstupech a výstupech zařízení činí stejnosměrných 24 V. PLC také obsahuje několik komunikačních portů (Schneider Electric, 2014a; Schneider Electric, 2014b).

Níže je uveden podrobnější seznam vstupů, výstupů a komunikačních portů. Na obr. 3.1 je zobrazeno rozmístění jednotlivých částí PLC.

Popis zařízení

- 14 digitálních vstupů
    - 8 rychlých (fast inputs)
    - 6 běžných (regular inputs)
- 10 digitálních výstupů
    - 4 rychlé (fast outputs)
    - 6 běžných (regular outputs)
- Komunikační porty
    - 2 porty pro sériovou komunikaci
    - Ethernet port
    - CANopen port
    - Programovací USB mini-B port



Obr. 3.1 – Popis PLC Modicon TM241CEC24T (Schneider Electric, 2014a, s. 125)

Tab. 3.1 – Tabulka s rozmístěním komponent u PLC Modicon TM241CEC24T

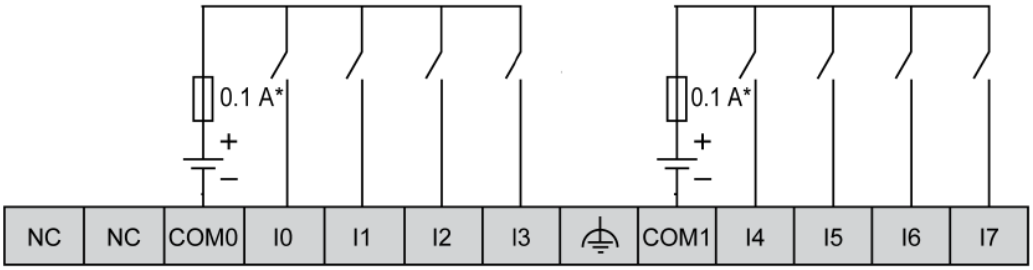
Označení	Popis
1	Run/Stop přepínač
2	Slot na SD kartu
3	Držák baterie
4	Cartridge slot
5	LED diody pro indikaci stavu vstupů a výstupů
6	Programovací USB mini-B port
7	Připínací zámek pro přichycení zařízení na lištu
8	Digitální výstupy
9	CANopen Line termination přepínač
10	Napájecí konektor: stejnosměrné napětí 24 V
11	CANopen port
12	Ethernet port – konektor typu RJ45
13	Stavové LED
14	Konektor sběrnice TM4
15	Sériový port 1
16	Sériový port 2
17	Digitální vstupy

Tab. 3.1 – Tabulka s rozmístěním komponent u PLC Modicon TM241CEC24T

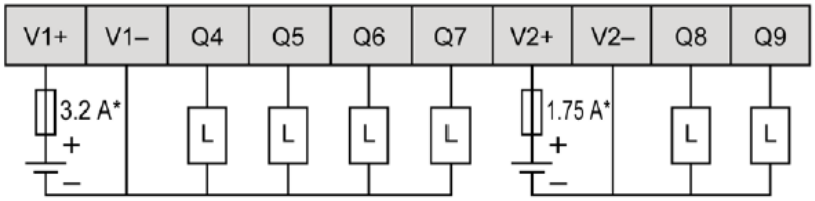
Označení	Popis
18	Konektor sběrnice TM3/TM2

3.1.1 Zapojení vstupů/výstupů

Digitální vstupy a výstupy vyžadují samostatné napájení. Využívají tzv. pozitivní logiku, kdy logické „1“ odpovídá vyšší napětí, než logické „0“. Porty COM0 a COM1 nejsou interně propojeny. To samé platí o portech V1+, V2+ a V1-, V2- (Schneider Electric, 2014a).



Obr. 3.2 – Zapojení vstupů PLC (Schneider Electric, 2014a, s. 179)

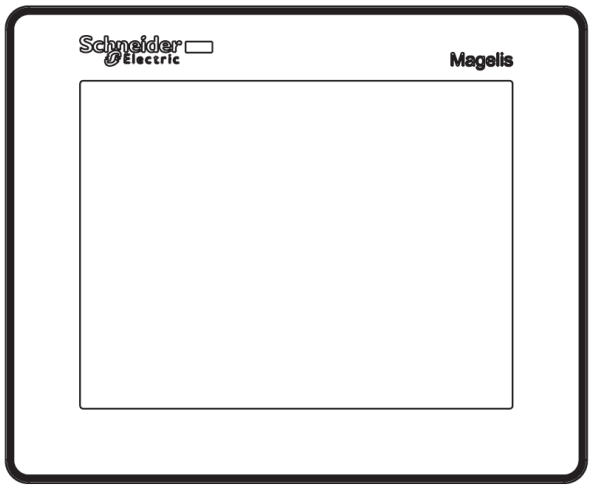


Obr. 3.3 – Zapojení výstupů PLC (Schneider Electric, 2014a, s. 189)

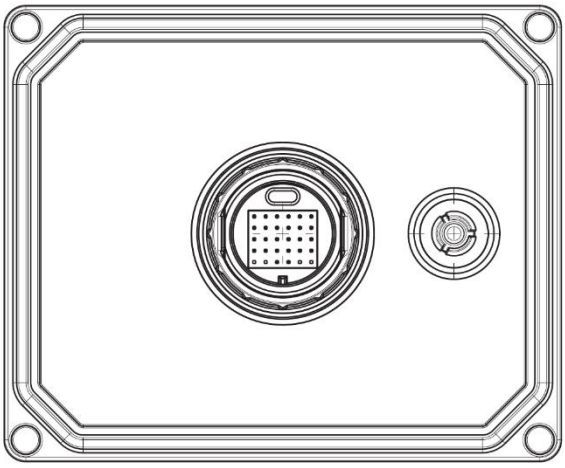
3.2 MAGELIS HMIS5T

Jedná se o zadní modul, ke kterému mohou být připojeny dva typy displejů a to 3,7 palce veliký HMI S65 nebo 5,7 palce veliký HMI S85, který je použit v této práci. Napájecí napětí modulu je stejnosměrných 24 V. Zadní modul obsahuje 3 typy pamětí. Pro aplikaci je k dispozici paměť Flash o velikosti 32 Mbytes. Pro data aplikace existuje paměť DRAM o velikosti 64 Mbytes a pro zálohu alarmových zpráv paměť SRAM o velikosti 64 kbytes. Toto zařízení komunikuje s PLC pomocí rozhraní Ethernet (Schneider Electric, 2012).

Displej:

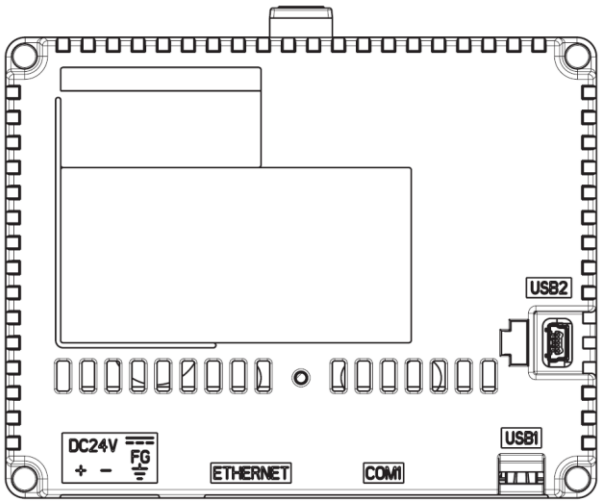


Obr. 3.4 – Přední strana displeje (Schneider Electric, 2012)

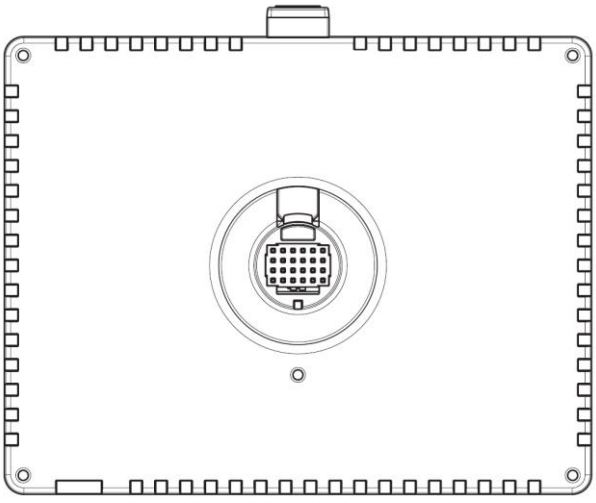


Obr. 3.5 – Zadní strana displeje (Schneider Electric, 2012)

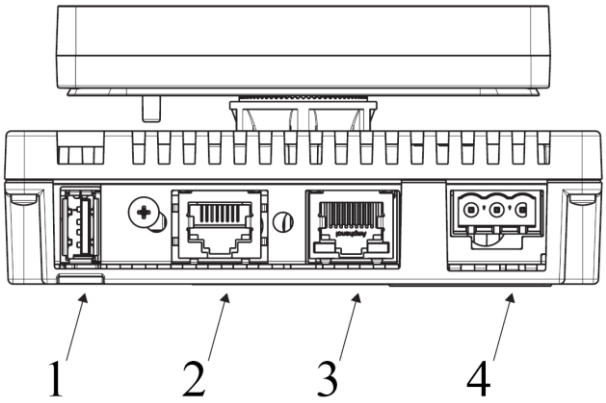
Zadní modul:



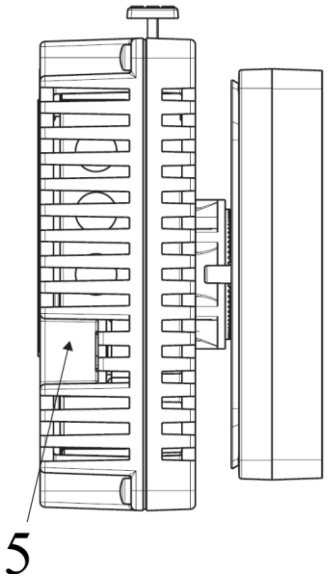
Obr. 3.6 – Přední strana modulu (Schneider Electric, 2012)



Obr. 3.7 – Zadní strana modulu (Schneider Electric, 2012)



Obr. 3.8 – Spodní strana modulu (Schneider Electric, 2012)



Obr. 3.9 – Boční strana modulu (Schneider Electric, 2012)

1 Standardní USB port

2 COM port

3 Ethernet port

4 Napájecí blok

5 Mini-B USB port

4 SOMACHINE

Programovací prostředí SoMachine umožňuje konfigurovat, programovat a dokonce zprovoznit celé stroje v jednom prostředí.

Prostředí je rozděleno do dvou specifických částí:

- Logic Builder – pro programování PLC
- Vijeo-Designer – pro programování grafických panelů

4.1 INSTALACE

Instalace SoMachine má následující hardwarové požadavky na počítač.

Tab. 4.1 – Hardwarové požadavky SoMachine

Zařízení	Minimum	Doporučené
Procesor	Pentium 4, 1,8 GHz Pentium M. 1,0 GHz (nebo podobné)	Intel Core™ i7, 2,7 GHz (nebo podobný)
RAM	3 GB	8 GB
Volný prostor na disku	5 GB	10 GB
Displej	1280 x 1024 pixelů	1680 x 1050 pixelů
Periférie	Myš, USB	

Před instalací SoMachine, je potřeba mít nainstalovaný **SoMachine Configuration manager**. Ten poskytuje následující možnosti:

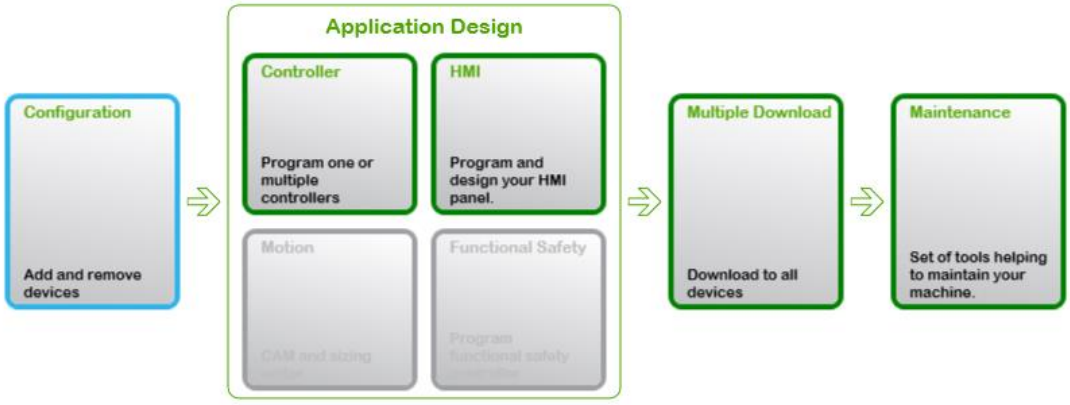
- Instalaci a odinstalaci SoMachine
- Možnost přizpůsobení instalace
- Správa licencí

Použité zařízení v této práci vyžadují update SoMachine. Je zapotřebí mít nainstalovaný minimálně SoMachine SP 2.1 a Vijeo-Designer SP 3.1.

4.2 SPRÁVA PROJEKTU

Po vytvoření nového nebo otevření již existujícího projektu je zobrazen **Workflow** (pracovní postup). Ten graficky znázorňuje postup realizace projektu a obsahuje následující 4 části:

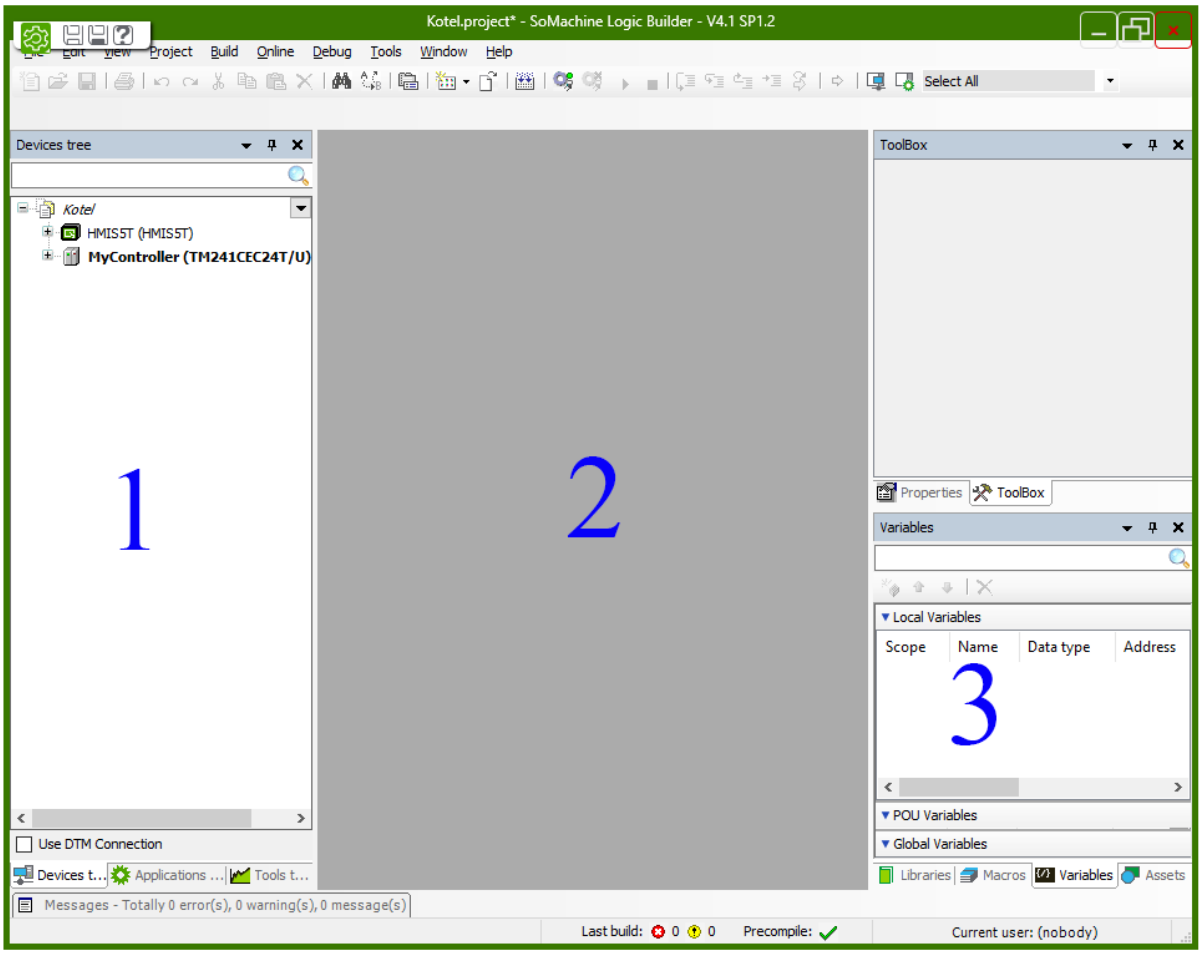
- Configuration – výběr zařízení
  - v tomto kroku lze přidávat, odebírat zařízení či je konfigurovat
- Application Desing – vytvoření aplikace
  - Controller: pro programování kontroléru je spuštěno prostředí Logic Builder
  - HMI: pro programování a návrh grafické aplikace je spuštěno prostředí Vijeo Desinger
  - Motion a Functional Safety nejsou v této verzi SoMachine 4.1 podporovány
- Multiple Download – nahrání programu do použitých zařízení
  - lze nahrát program do všech zařízení současně nebo do každého zařízení zvlášť
- Maintance – diagnostika, správa zařízení
  - nástroje pro diagnostiku projektu



Obr. 4.1 – Workflow projektu (pracovní postup)

4.3 LOGIC BUILDER

Logic Builder se spustí kliknutím na tlačítko **Controller** na obr. 4.1. Ten umožňuje uživateli tvořit aplikace na jedno či více PLC najednou. Na obr. 4.2 je zobrazeno výchozí okno Logic Builderu.



Obr. 4.2 – Výchozí okno Logic Builderu

1 Záložky: Device tree, Applications tree a Tools tree

2 Pracovní plocha

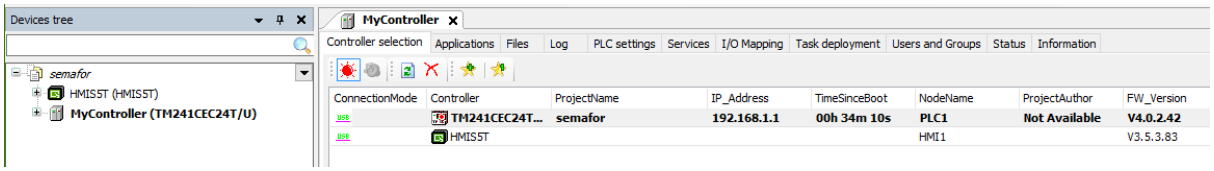
3 Záložky: Libraries, Macros, Variables, Assets

4.3.1 Zkouška komunikace PC s PLC

Po přidání konkrétního PLC do projektu a propojení s počítačem přes USB či Ethernet lze provést zkoušku komunikace, zda zařízení správně komunikuje s počítačem.

Pro aktivaci připojení je zapotřebí dvojitým kliknutím vybrat právě používaný kontrolér. Po dvojkliku se kontrolér označí tučným písmem. Na označení konkrétního kontroléru záleží především v případech, kde se pracuje s více zařízeními.

Zkouška komunikace se provede kliknutím na ikonu žárovky nebo po výběru možnosti ***Signal Optical*** po kliku pravým tlačítkem myši na daný kontrolér. Při správném připojení se rozblikají stavové LED diody.



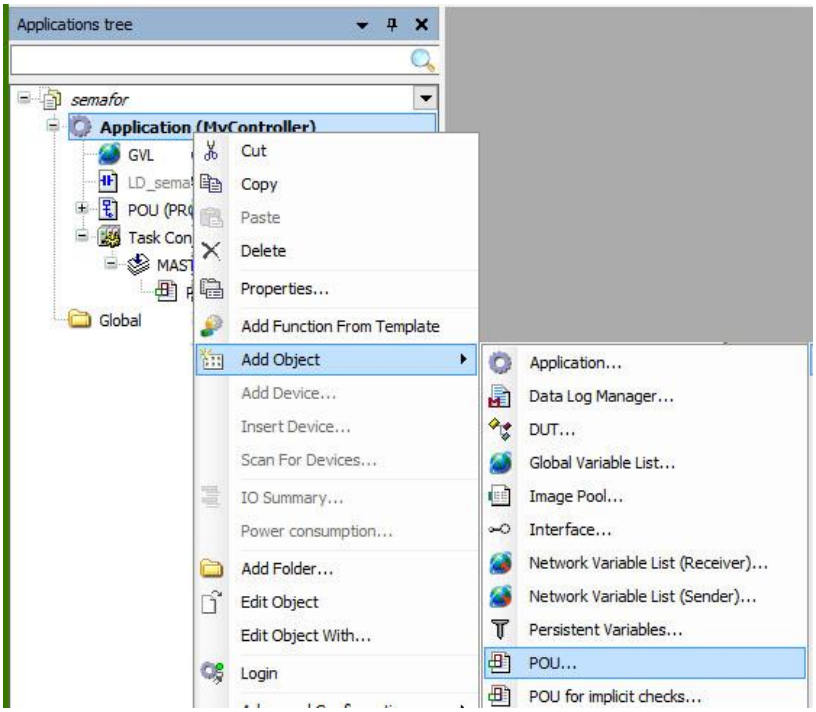
Obr. 4.3 – Zkouška komunikace s PLC

4.3.2 Vytvoření POU

POU, neboli Program Organization Unit je objekt v SoMachine, do kterého je psán kód programu. Podle potřeby použití se POU dělí na:

- Program POU,
- Function Block POU,
- Function POU.

POU vytvoříme pravým klikem na **Application**, poté se vybere možnost **Add Object** a z nabízeného menu **POU**.

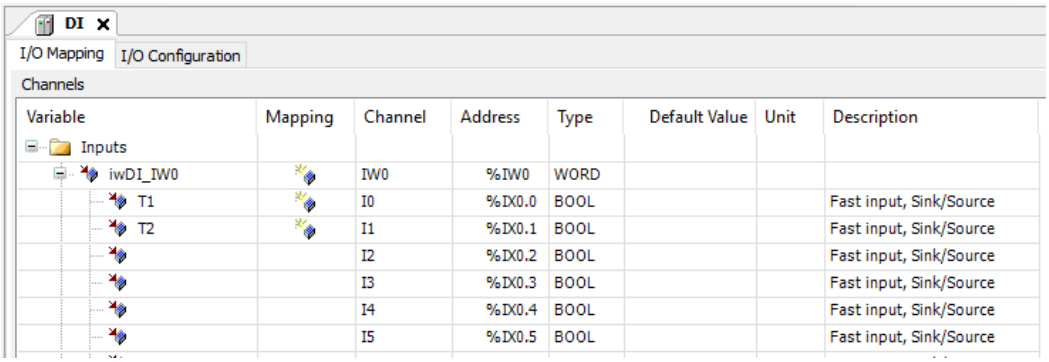


Obr. 4.3 – Vytvoření POU

4.3.3 Mapování proměnných na vstupy/výstupy PLC

Proměnné mohou být přiřazeny přímo na vstupy nebo výstupy PLC. To se provede tak, že se v záložce **Device tree** otevře konfigurace pro **Digital inputs/Digital outputs** a jednoduše

se pojmenují jednotlivé vstupy/výstupy podle potřeby uživatele. Dále se tyto proměnné nemusí v projektu deklarovat a jsou dostupné pro aplikace s tímto PLC.



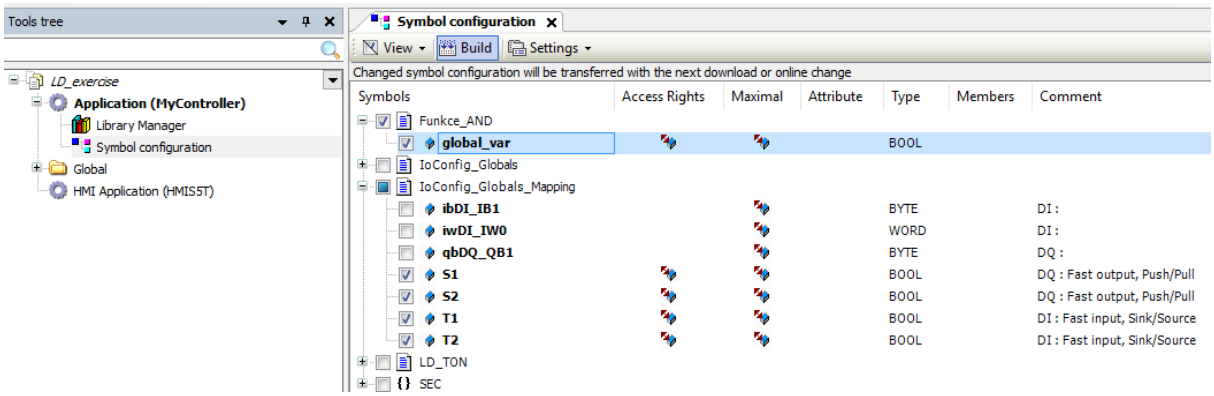
Obr. 4.4 – Mapování proměnných na vstupy/výstupy PLC

4.3.4 Sdílení proměnných

Sdílení proměnných je využíváno zejména v případech, kdy je potřeba sdílet informace mezi jednotlivými zařízeními, např. mezi PLC a HMI panelem.

Aby bylo možné sdílet proměnné, je zapotřebí mít v aplikaci v záložce **Tools tree** přidaný objekt **Symbol configuration**. V tomto objektu stačí pouze najít danou proměnou, která je potřeba sdílet, označit ji a nakonec kliknout na tlačítko **Build**. Proměnné se tímto promítnou do celého projektu ke všem zařízením.

Proměnným lze také spravovat přístupová práva kliknutím do políčka **Access Rights**.

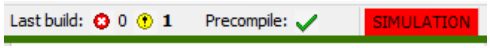


Obr. 4.5 – Sdílení proměnných

4.3.5 Simulace

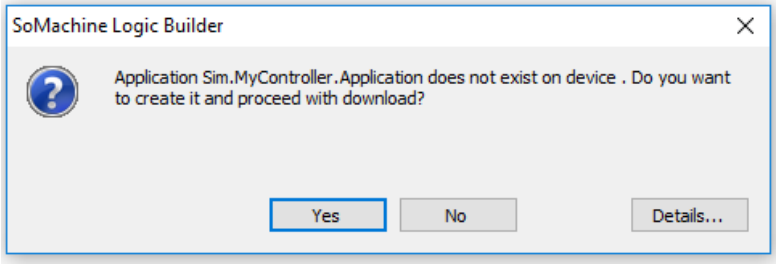
SoMachine poskytuje funkci **offline simulace**, Simulaci programu lze využít k vyzkoušení programu nebo k jeho ladění.

Simulaci programu spustíme kliknutím na možnost **Simulation** v záložce **Online**. Jakmile je program v simulačním módu, v dolní části Logic Builderu se objeví v červeném poli nápis **SIMULATION**.



Obr. 4.6 – Indikace simulačního módu

Dále je třeba přihlášení do simulátoru. To se provede vybráním možnosti **Login** ve stejné záložce. Po dotazu, kterým se připojíme k simulátoru, se může zobrazit dialogové okno, pokud v simulátoru doposud nebyl nahrán program. To je zobrazeno na obr. 4.7.

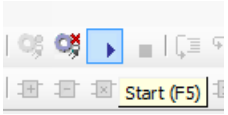


Obr. 4.7 – Potvrzení nahrání aplikace

Jakmile je uživatel přihlášen, ocitne se ve **STOP** módu simulace. To je znázorněno tak, že vedle červeného pole s nápisem **SIMULATION** objeví v červeném poli nápis **STOP**.

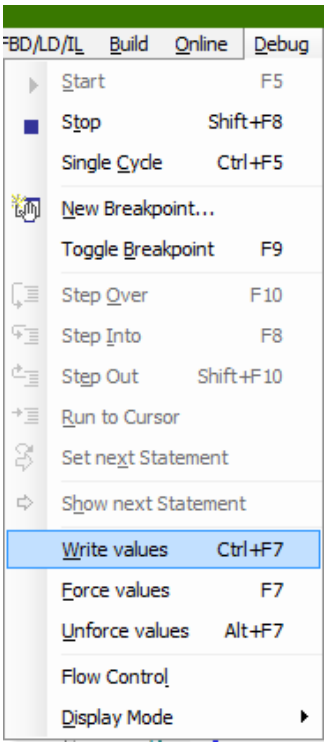
Obr. 4.8 – Indikace STOP módu simulace

Dalším krokem je třeba spustit simulaci. To se provede kliknutím na ikonu **Start**, která se nachází v hlavním panelu nástrojů nebo v záložce **Debug**. To, že program „běží“, je znázorněno zeleným políčkem s nápisem **RUN** místo nápisu **STOP** v dolní časti okna.

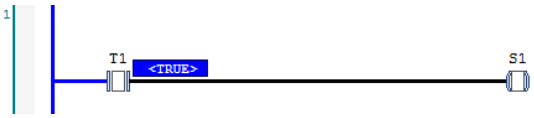


Obr. 4.9 – Tlačítko START pro spuštění aplikace

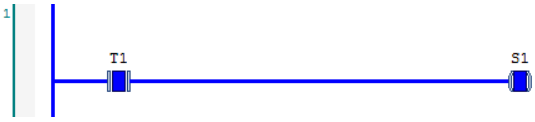
Posledním krokem je nastavení hodnot proměnných a následně jejich zapsání, aby se provedené změny promítly na výstup. Např. pokud se v programovacím jazyku LD simuluje funkce tlačítka, tak se stav tlačítka změní dvojitým klikem na kontakt reprezentující tlačítko. Zapsání stavu tlačítka se provede kliknutím na možnost **Write values** v záložce **Debug.**



Obr. 4.10 – Funkce pro zápis nastavených hodnot



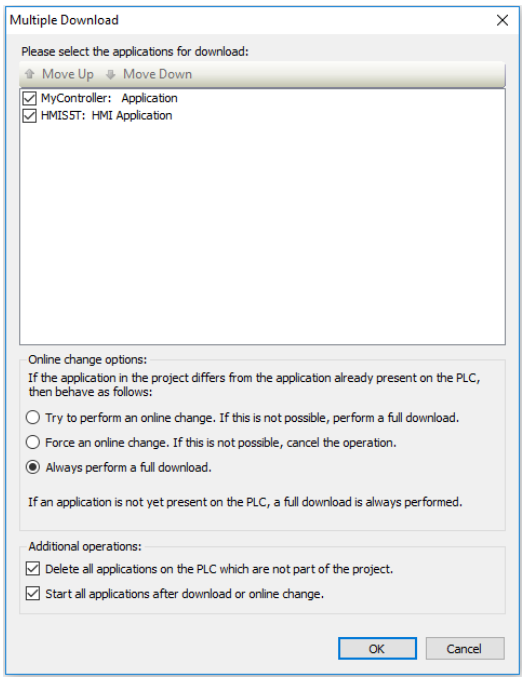
Obr. 4.11 – Nastavení hodnoty proměnné T1



Obr. 4.12 – Promítnutí hodnoty T1 na výstup S1

4.3.6 Odeslání programu do zařízení

Příkaz k odeslání programu do zařízení se nachází v záložce hlavního menu, **Online**. Zvolením možnosti **Multiple Download** se otevře průvodce, pomocí kterého se vyberou zařízení a způsob nahrávání programu.



Obr. 4.13 – Multiple Download

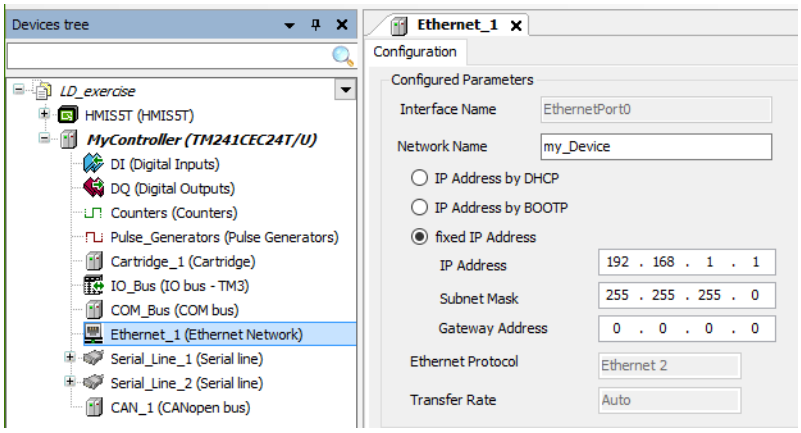
Aplikace se do zařízení nahrávají postupně podle pořadí v okně výběru zařízení. SoMachine nabízí 3 způsoby nahrání programu:

1) Pouze modifikované části programu budou změny a nově přidané části programu budou nahrány do zařízení.
2) Pokud nelze provést online změnu, nahrání programu do zařízení nebude vykonáno.
3) Vždy nahraje celý program.

Dále si uživatel může vybrat ze dvou doplňujících možností. První možnost nabízí vymazání všech aplikací, které nejsou součástí projektu. Druhou možností je, zda uživatel chce po dokončení nahrávání aplikaci spustit.

4.3.7 Nastavení komunikace s HMI panelem

Zařízení spolu komunikují pomocí rozhraní Ethernet. Proto je třeba zařízením nastavit IP adresu ve stejné síti. Dvojitým klikem na rozhraní **Ethernet** v záložce **Devices tree** se otevře jeho nastavení. Zde je možné nastavit zařízení příslušnou IP adresu, např. 192.168.1.1 s maskou sítě 255.255.255.0.

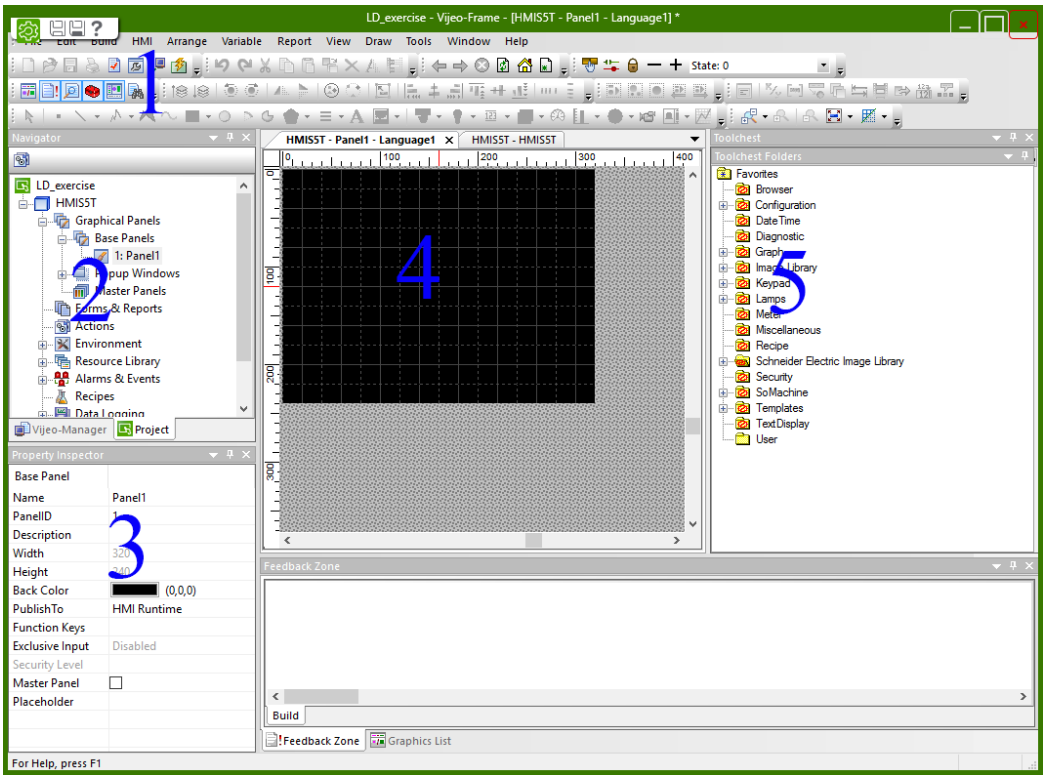


Obr. 4.14 – Nastavení IP adresy na PLC

4.4 VIJEO-DESIGNER

Jedná se o grafické prostředí, které slouží pro tvorbu aplikací na HMI panely. Toto prostředí je součástí programu SoMachine. Pokud grafická aplikace využívá proměnné, které jsou adresovány v PLC, je potřeba mít používané proměnné sdílené, jak je popsáno v kapitole 4.2.4. Komunikace mezi HMI panelem a PLC probíhá pomocí rozhraní Ethernet.

Na obr. 4.15 je zobrazeno výchozí okno Vijeo-Designeru s popisem jeho částí.



Obr. 4.15 – Výchozí okno Vijeo-Designeru

1 Panel nástrojů s prvky pro práci s grafickými objekty

2 Strukturální strom projektu

3 Bližší informace

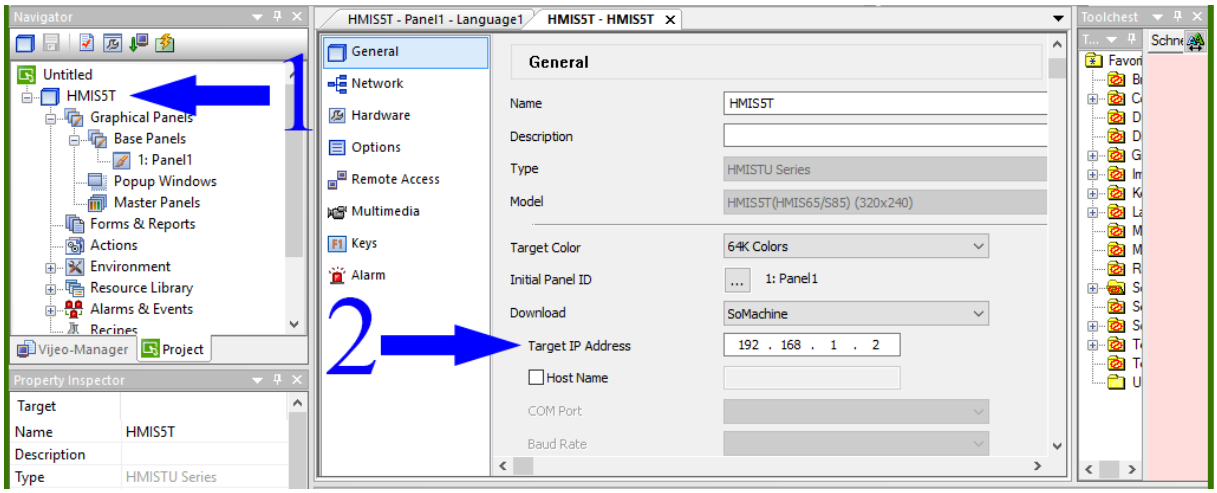
4 Pracovní plocha

5 Nabídka komponent

4.4.1 Nastavení komunikace s PLC

Aby mohl grafický panel komunikovat s PLC, musí se mu také nastavit IP adresa. Ta musí být ve stejné síti, jako adresa PLC.

IP adresa zařízení se nastaví tak, že se otevře konfigurace kliknutím na název zařízení. Po otevření konfiguračního okna, viz obr. 4.16, se adresa zapíše do políčka **Target IP Adress** (např. 192.168.1.2)

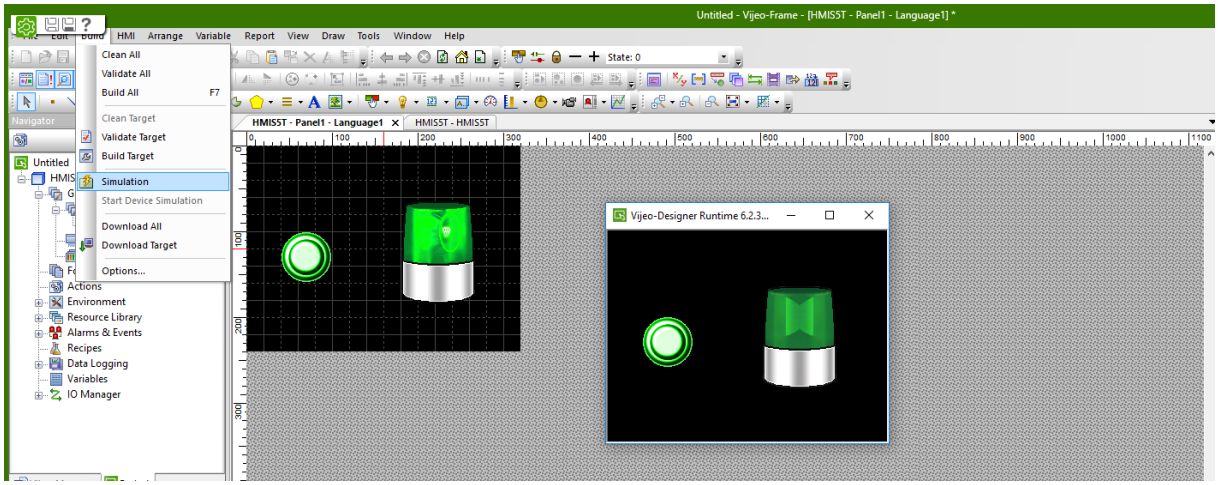


Obr. 4.16 – Nastavení IP adresy na HMI

4.4.2 Simulace

Simulace grafické aplikace se provede velice jednoduše. Stačí kliknout na možnost **Simulation** v záložce **Build**. Simulovaný grafický panel se otevře v novém okně.

Pokud je nějaký grafický prvek špatně nakonfigurovaný, např. chybí vstupní nebo výstupní proměnná, simulace se neprovede a vypíše se chybová hláška.



Obr. 4.17 – Spuštění simulace HMI aplikace

5 KONSTRUKCE

Veškerá zařízení jsou zasazena na lištu, která je upevněna v hliníkovém rámu. Pro zajištění bezpečnosti napájení byly pro PLC a HMI panel použity pojistky o velikosti 1 A. Pro napájení vstupů a výstupů byli použity pojistky o velikosti 0,1 A.

K zařízení je připojen blok se dvěma tlačítky a blok se dvěma kontrolkami. Jak jsou propojeny tlačítka s PLC a zdrojem je ukázáno na obr. 3.2. Kontrolky jsou zapojeny podle obr. 3.3. Ke kterým pinům PLC jsou tlačítka a kontrolky připojeny popisuje následující tabulka.

Tab. 5.1 – Rozmístění připojených tlačítek a kontrolek k PLC

Tlačítko 1	I0
Tlačítko 2	I1
Zelená kontrolka	Q0
Červená kontrolka	Q1

6 ZÁDÁNÍ LABORATORNÍCH ÚLOH

V této kapitole jsou uvedena jednotlivá zadání laboratorních úloh. K dispozici je PLC, grafický panel, dvě tlačítka a dvě kontrolky. Hardwarové zapojení všech komponent je popsáno v kapitole 5.

6.1 ÚLOHA Č. 1

Mějme dvě tlačítka T1, T2 a dvě kontrolky S1, S2. Napište program, podle kterého každé tlačítko bude ovládat jednu kontrolku. Program napište v programovacím jazyku ST.

6.2 ÚLOHA Č. 2

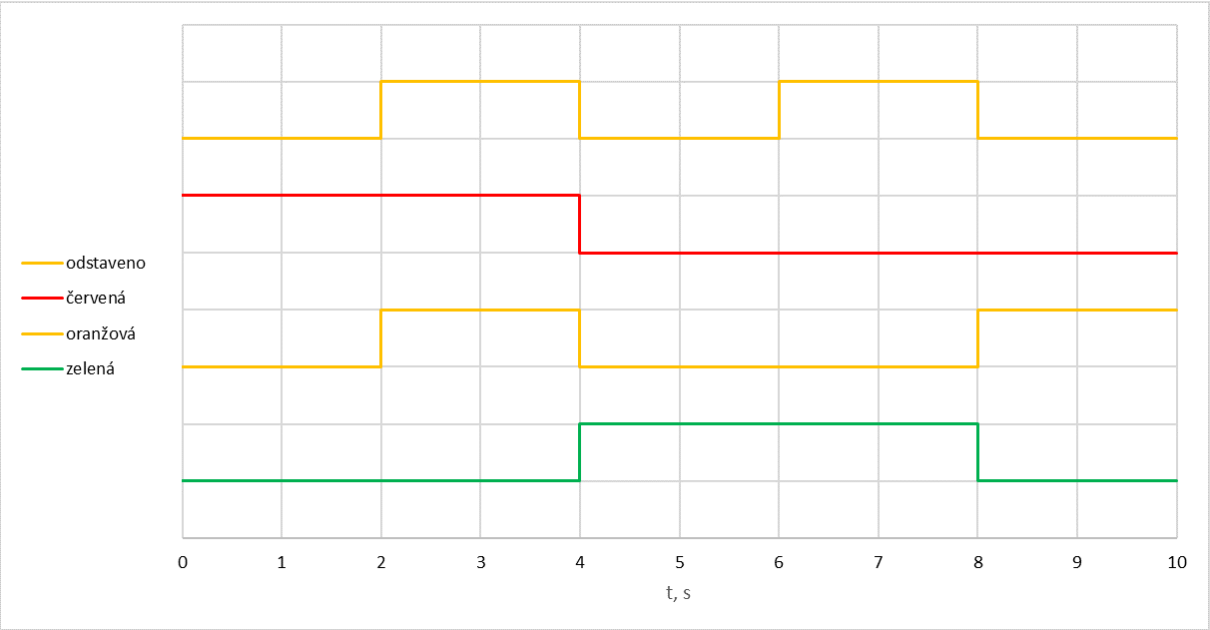
Mějme dvě tlačítka T1, T2 a dvě kontrolky S1, S2. Pomocí všech dostupných programovacích jazyků si vyzkoušejte psaní programu na následujících logických funkcí pro ovládání kontrolek.

$$S1 = T1 \text{ or } T2 \tag{6.1}$$

$$S2 = \text{not}(T1 \text{ or } T2) \tag{6.2}$$

6.3 ÚLOHA Č. 3

Dle časového diagramu, který je na obr. 6.1, naprogramujte semafor s možností jeho odstavení. Program napište v jazyce SFC. Vizualizaci semaforu s tlačítkem udělejte na grafickém panelu.

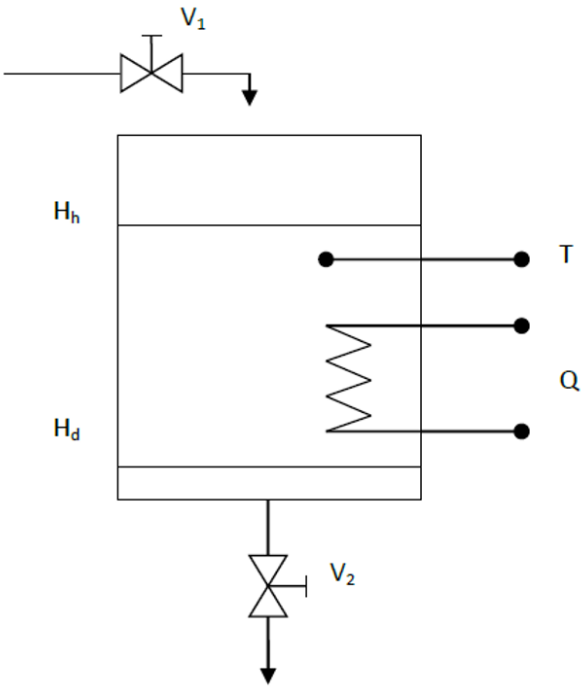


Obr. 6.1 – Časový diagram semaforu

6.4 ÚLOHA Č. 4

Navrhněte logický obvod zajišťující správnou funkci vsádkového reaktoru. Při provozu reaktoru se neustále opakují tyto operace: napuštění reaktoru, ohřev obsahu reaktoru na zadanou teplotu a vypuštění reaktoru. Jako paměťový prvek použijte klopný obvod RS.

Program napište v programovacím jazyku FBD. Simulaci výstupů a vnitřních proměnných vytvořte na grafickém panelu. Výstupy jako objekty měnící barvu a vnitřní proměnné jako tlačítka.



Obr. 6.2 – Funkční schéma vsádkového reaktoru

Tab. 6.1 – Definice logických proměnných

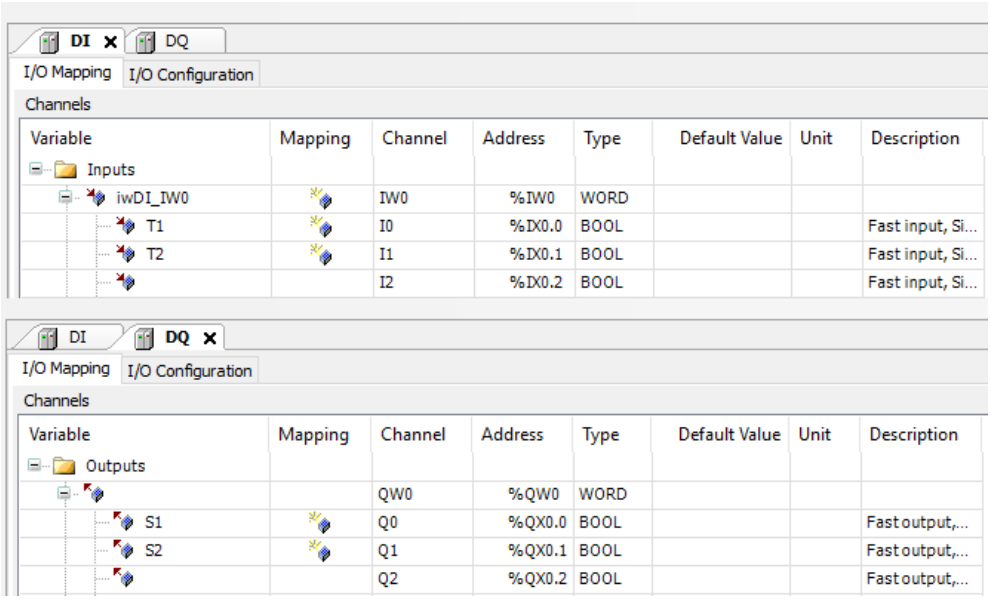
Ventil	V_1	V_2	Požadovaná teplota	T
Je otevřen	1	1	Je dosažena	1
Je zavřen	0	0	Není dosažena	0
Požadovaná výška hladiny	H_d	H_h	Topení	Q
Je dosažena	1	1	Topí	1
Není dosažena	0	0	Netopí	0

7 ŘEŠENÍ LABORATORNÍCH ÚLOH

V této kapitole jsou podrobně popsány řešení všech zadaných laboratorních úloh z kapitoly 6.

7.1 ÚLOHA Č. 1

Nejdříve se provede přiřazení proměnných na vstupy a výstupy, kde jsou připojeny tlačítka a kontrolky, viz obr. 7.1.



Obr. 7.1 – Úloha č. 1: pojmenování vstupů/výstupů

Po vytvoření POU se v programu jednotlivé tlačítka přiřadí kontrolkám. Jak bude vypadat program je ukázáno na obr. 7.2.

```
S1 := T1;
S2 := T2;
```

Obr. 7.2 – Úloha č. 1: řešení v jazyce ST

7.2 ÚLOHA Č. 2

V této úloze se vychází ze stejného zapojení tlačítek a kontrolek jako v úloze č. 1.

a) Řešení v jazyce IL

Tento programovací jazyk pracuje s pamětí **stack**. Výsledek operace na jednom řádku se uloží do stacku pro operaci na dalším řádku. Nový řádek se vloží výběrem možnosti **Insert IL line below** z menu nabízeného po kliku pravým tlačítkem. Kód v jazyce IL je uveden na obr. 7.3.

```
LD      T1          – uloží stav tlačítka T1
OR      T2          – provede logický součet T1 a T2
ST      S1          – zapíše výsledek na výstup S1
STN     S2          – zapíše negovaný výsledek na výstup S2
```

Obr. 7.3 – Úloha č. 2: řešení v jazyce IL

b) Řešení v jazyce ST

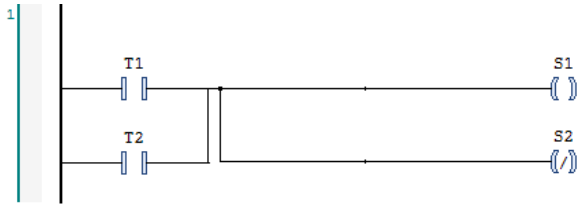
Tento programovací jazyk spadá do kategorie vyšších programovacích jazyků. Každá z rovnic lze jazykem ST zapsat jednoduše na jeden řádek.

```
S1 := T1 OR T2;          – zápis rovnice 6.1
S2 := NOT(T1 OR T2);     – zápis rovnice 6.2
```

Obr. 7.4 – Úloha č. 2: řešení v jazyce ST

c) Řešení v jazyce LD

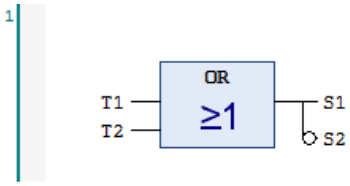
Logická funkce OR se v jazyce LD vytvoří pomocí dvou paralelně zapojených kontaktů. Poté stačí pouze připojit výstup S1 (Coil) a negovaný výstup S2 (Negated Coil)



Obr. 7.5 – Úloha č. 2: řešení v jazyce LD

d) Řešení v jazyce FBD

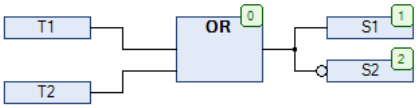
Funkční blok OR je vytvořen z prázdného boxu, který je vložen do projektu. Přepsáním tří otazníků uvnitř boxu na „OR“ se inicializuje funkční blok OR. Poté se přiřadí tlačítka vstupům bloku a na výstup bloku dva výstupy na kontrolky pomocí nástroje **Assignment** z nabídky **ToolBoxu**. Nakonec se provede negace výstupu S2.



Obr. 7.6 – Úloha č. 2: řešení v jazyce FBD

e) Řešení v jazyce CFC

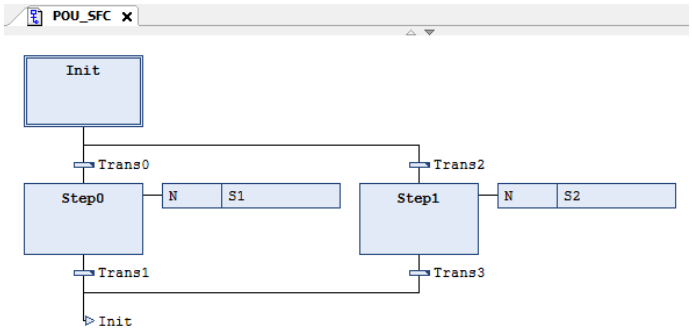
Funkční blok OR se vytvoří stejně jako v jazyce FBD. Z nabídky **Toolboxu** se vyberou vstupy a výstupy a připojí se k funkčnímu bloku OR.



Obr. 7.7 – Úloha č. 2: řešení v jazyce CFC

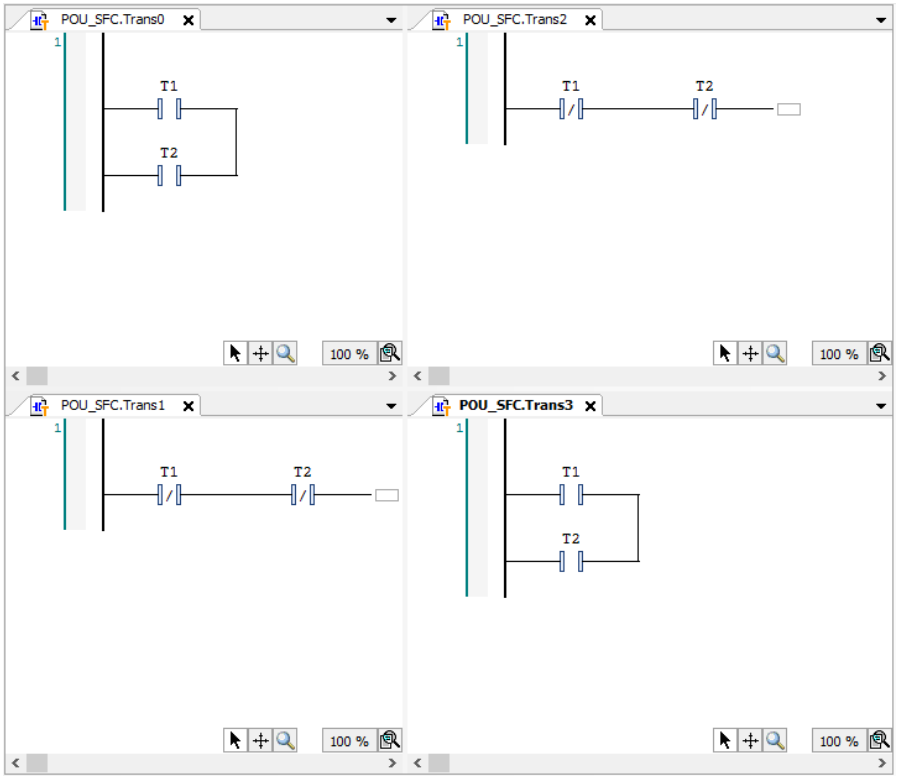
f) Řešení v jazyce SFC

V tomto jazyce se rovnice musí přepsat do vývojového diagramu. Je zapotřebí zajistit vstupní a výstupní podmínky (Trans0, Trans1, …) mezi jednotlivými kroky (Step0, Step1). Ke každému kroku (Step0 a Step1) byla přidána akce s identifikátorem **N**, která kontrolku nechá svítit tak dlouho, dokud se vykonávání programu nachází v daném kroku. Vývojový diagram je zobrazen na obr. 7.8.



Obr. 7.8 – Úloha č. 2: řešení v jazyce SFC – diagram

Přechody, i samotné kroky, mohou být napsány v jakémkoliv z uvedených jazyků. Vzhledem k jednoduchosti byl vybrán jazyk LD. Rovnice 6.1 je popsána přechody **Trans0** a **Trans1**. **Trans0** realizuje logickou funkci OR, pokud je jedno z tlačítek sepnuto, pokračuj dál. **Trans1** dělá opak, protože je potřeba podržet vykonávaní programu v kroku **Step0**. Pokud je některé z tlačítek sepnuto, program nepokračuje dál. Rovnice 6.2 je přesným opakem. Program dojde k vykonávání kroku **Step1** přes přechod **Trans2**, pokud není sepnuto ani jedno z tlačítek. Přechod **Trans3** při sepnutí jednoho z tlačítek pošle vykonávání programu dále.



Obr. 7.9 – Úloha č. 2: řešení v jazyce SFC – přechody

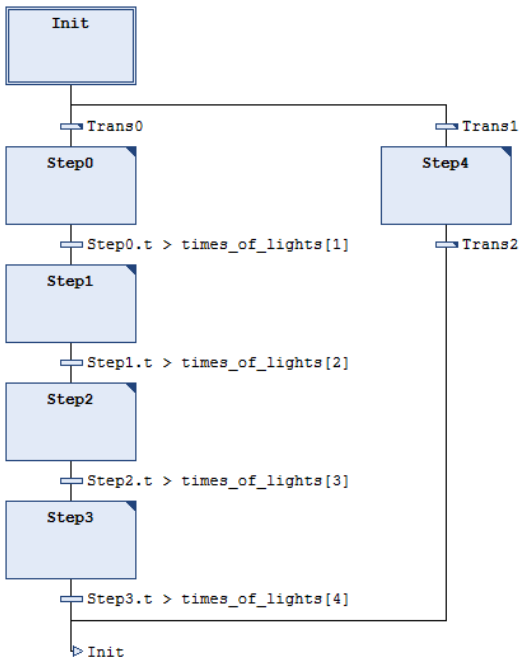
7.3 ÚLOHA Č. 3

Nejdříve je nutné založit potřebné proměnné pro ovládání světel, stav semaforu a pole s definovanými časy. Vše je uvedeno na obr. 7.10.

```
PROGRAM POU
VAR
    times_of_lights: ARRAY [1..4] OF TIME := [T#2S, T#2S, T#4S, T#2S];  //pole časů dle časového diagramu
    red: BOOL;                                                          //červené světlo
    orange: BOOL;                                                       //oranžové světlo
    green: BOOL;                                                        //zelené světlo
    state: BOOL;                                                        //stav semaforu
    blinker: BLINK;                                                     //blinker - instance bloku BLINK
END_VAR
```

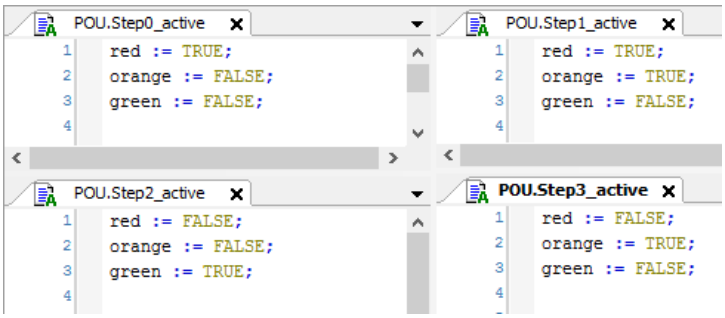
Obr. 7.10 – Úloha č. 3: použité proměnné

Funkce semaforu se rozdělí do jednotlivých kroků. Diagram semaforu je uveden na obr. 7.11.



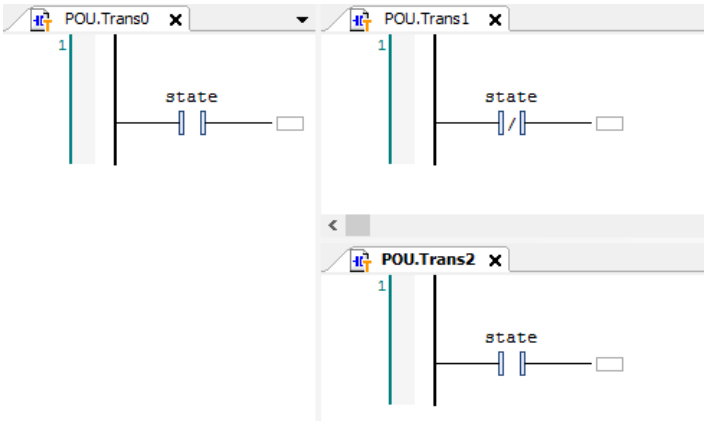
Obr. 7.11 – Úloha č. 3: vývojový diagram semaforu

V každém z kroků (**Step0-Step3**) se provede akce napsaná v jazyku ST, která rozsvítí jednotlivé kombinace světel semaforu, viz obr. 7.12. Přechody jsou dělány tak, že se měří čas setrvání v každém kroku a porovnává se s předdefinovaným časem z pole časů *times_of_lights*. Po uplynutí příslušné doby program přejde k dalšímu kroku.

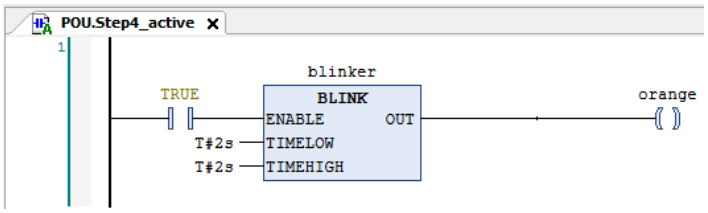


Obr. 7.12 – Úloha č. 3: kódy jednotlivých kroků

O odstavení semaforu se stará tlačítko na grafickém panelu. Přechody **Trans0**, **Trans1** a **Trans2** určují vykonávání programu podle tlačítka. Je-li sepnuto (*state* = true), semafor běží podle časového diagramu. Je-li rozepnuto (*state* = false), na semaforu bliká pouze oranžové světlo. Přechody jsou napsány v jazyce LD, viz obr. 7.13. Blikání při odstavení je realizováno v jazyce LD funkčním blokem BLINK v kroku **Step4**, viz obr. 7.14.



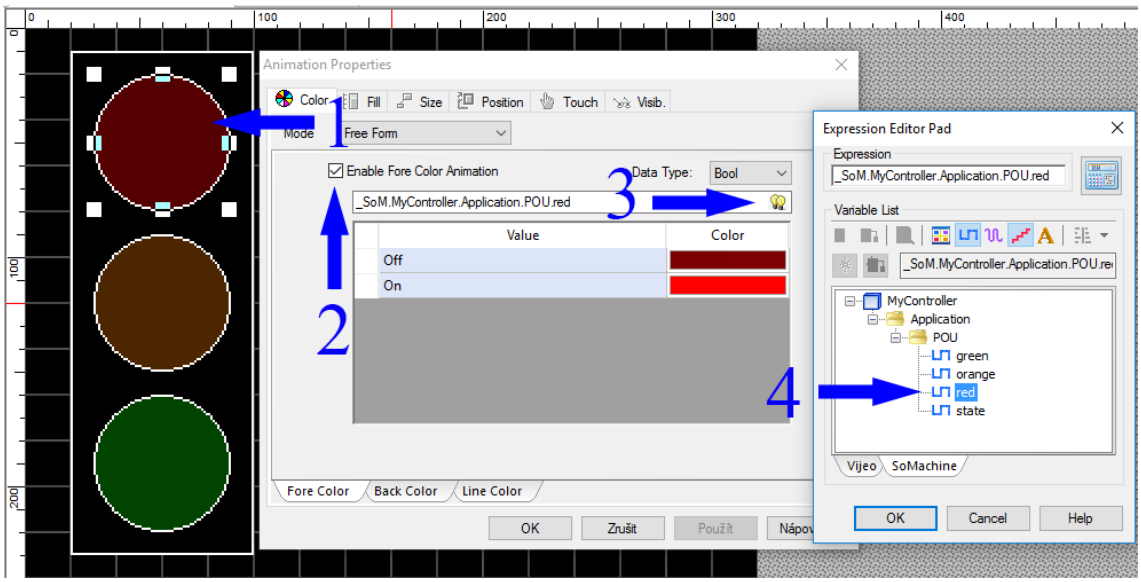
Obr. 7.13 – Úloha č. 3: kódy jednotlivých přechodů



Obr. 7.14 – Úloha č. 3: realizace blikání

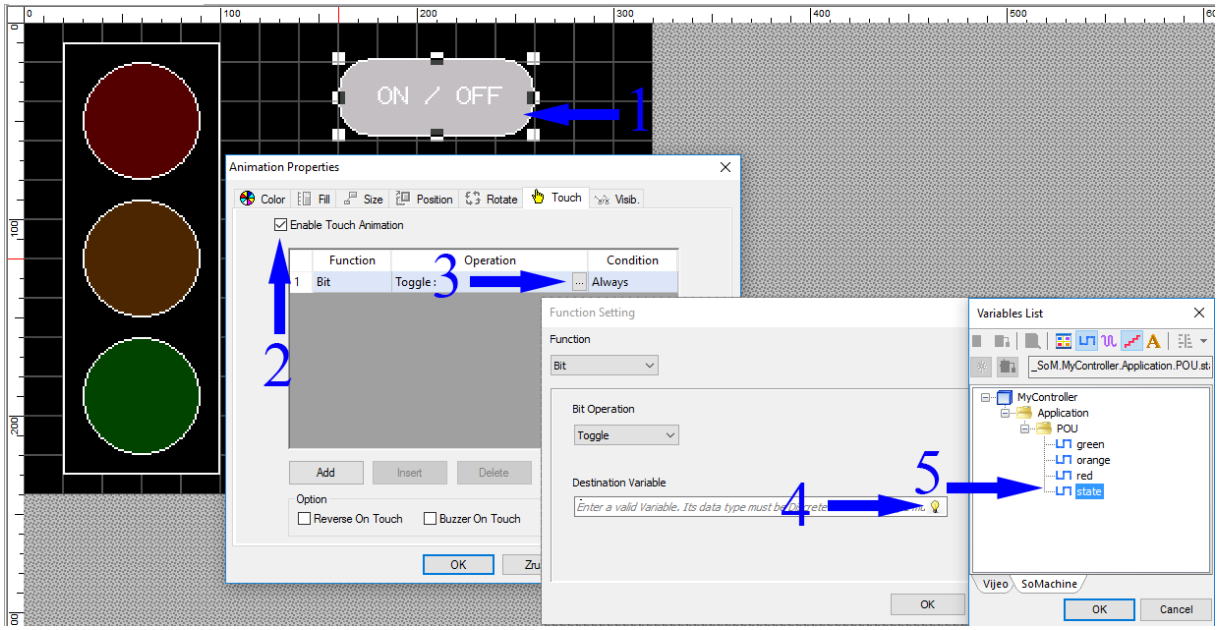
Pro vizualizaci se v prostředí Vijeo-Designer sestaví semafor pomocí nabízených grafických prvků. Je zapotřebí mít sdíleny následující proměnné: *red*, *orange*, *green* a *state*.

Aby objekty reprezentující světla semaforu měnily barvy, je potřeba jim tuto možnost aktivovat a nastavit v **Animation Properties**. Do tohoto nastavení lze vstoupit dvojklikem na daný objekt. Postup je uveden na obr. 7.15.



Obr. 7.15 – Úloha č. 3: Color Animation objektu

U objektu reprezentující tlačítko je potřeba nastavit reakci na dotyk. Je to stejný postup jako u změny barvy, akorát v záložce **Touch**. Postup je uveden na obr. 7.16.



Obr. 7.16 – Úloha č. 3: Touch Animation objektu

7.4 ÚLOHA Č. 4

V této úloze je úkolem navrhnout řídící obvod vsádkového reaktoru. Při provozu reaktoru se neustále opakují tyto tři fáze: napuštění, ohřev na požadovanou teplotu, vypuštění reaktoru.

V prvním stavu (napouštění) se reaktor nachází tehdy, když není dosažena horní úroveň hladiny. Jakmile je reaktor naplněn, přejde se do druhého stavu (ohřívání). Po ohřevu na požadovanou teplotu se reaktor vypustí. Jako paměťový prvek pro přechody mezi stavy je použit klopný obvod RS.

Tab. 7.1 – Tabulka přechodů vsádkového reaktoru

Stav \ H_h H_d T	000	001	011	010	110	111	101	100	V1	V2	Q
1	**1**	**1**	**1**	**1**	2	3	–	–	1	0	0
2	–	–	–	–	**2**	3	–	–	0	0	1
3	1	1	**3**	**3**	–	**3**	–	–	0	1	0

Sloučením prvního a druhého řádku tabulky vznikne redukovaná tabulka přechodů.

Tab. 7.2 – Redukovaná tabulka přechodů

Stav \ H_h H_d T	000	001	011	010	110	111	101	100
1,2	**1,2**	**1,2**	**1,2**	**1,2**	**2**	3	–	–
3	1,2	1,2	**3**	**3**	–	**3**	–	–

Dalším krokem je zavedení vnitřní proměnné. Po redukci stavů zůstaly dva stavy. Tyto dva stavy se dají popsat jednou vnitřní proměnnou, viz tab. 7.3.

Tab. 7.3 – Zavedení vnitřní proměnné

Stav	1,2	3
Q	0	1

Po zakódování vnitřních stavů vnitřními proměnnými se vytvoří mapa vnitřních proměnných, kde se stavy a přechody nahradí hodnotou vnitřní proměnné a sestaví se rovnice pro klopný obvod RS.

Tab. 7.4 – Mapa vnitřních proměnných

Q \ H_h H_d T	000	001	011	010	110	111	101	100
0	0	0	0	0	0	**1**	–	–
1	**0**	**0**	1	1	–	1	–	–

$$R = \overline{H_d} \tag{7.1}$$

$$S = H_h \cdot T \tag{7.2}$$

Nakonec se sestaví mapy výstupních funkcí pro V_1, V_2 a Q, ze kterých se získají minimalizované tvary výstupních funkcí.

Tab. 7.5 – Mapa výstupní funkce V_1

V_1								
A \ H_h H_d T	000	001	011	010	110	111	101	100
0	1	1	1	1	0	–	–	–
1	–	–	0	0	–	0	–	–

$$V_1 = \overline{A} \cdot \overline{H_h} \tag{7.3}$$

Tab. 7.6 – Mapa výstupní funkce V_2

V_2								
A \ H_h H_d T	000	001	011	010	110	111	101	100
0	0	0	0	0	0	–	–	–
1	–	–	1	1	–	1	–	–

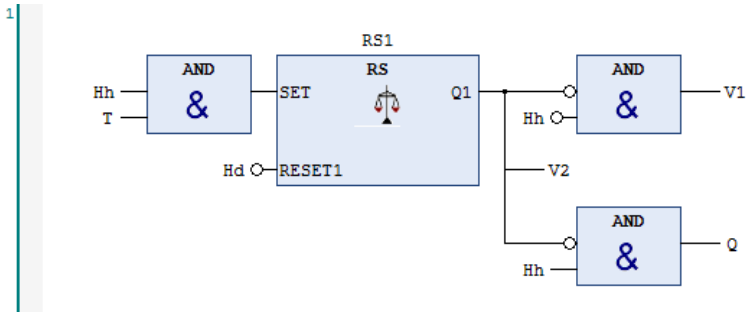
$$V_2 = A \tag{7.4}$$

Tab. 7.7 – Mapa výstupní funkce Q

Q								
A \ H_h H_d T	000	001	011	010	110	111	101	100
0	0	0	0	0	1	–	–	–
1	–	–	0	0	–	0	–	–

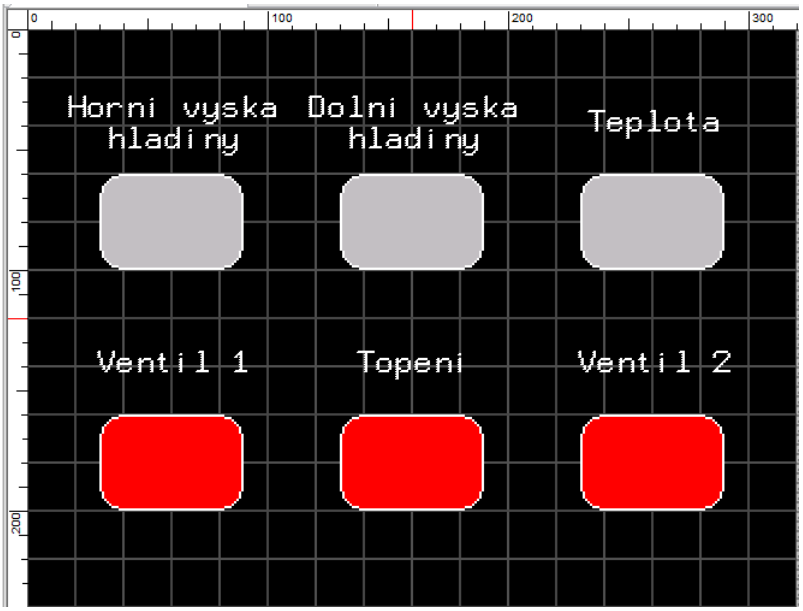
$$Q = \overline{A} \cdot H_h \tag{7.5}$$

Dalším krokem je zapsat rovnice pomocí jazyku FBD. Za jednotlivé operace se dosadí funkční bloky. Program je uveden na obr. 7.17.



Obr. 7.17 – Úloha č. 4: program vsádkového reaktoru

Poté se nasdílí potřebné proměnné pro Vijeo-Designer a vytvoří se simulace vnitřních a výstupních proměnných. Objektům realizující výstupní proměnné se nastaví **Color Animation** a objektům realizujícím vnitřní proměnné se nastaví **Color Animation** i **Touch Animation** na příslušné proměnné. Aplikace pro grafický panel může vypadat např. jako na obr. 7.18.



Obr. 7.18 – Úloha č. 4: simulace vsádkového reaktoru

ZÁVĚR

Cílem této práce bylo vytvořit laboratorní úlohy s PLC Modicon. Byly vytvořeny 4 úlohy logického řízení s různou obtížností.

V první úloze bylo úkolem napsat program, kde každé tlačítko ovládá jednu kontrolku. Jedná se o nejzákladnější program, na kterém lze vyzkoušet funkci systému a že jsou tlačítka a kontrolky správně zapojeny.

V druhé úloze bylo úkolem napsat jednoduché logické funkce ve všech dostupných programovacích jazycích. Pro zápis logických funkcí je nejméně vhodný programovací jazyk SFC z důvodu složitosti zápisu. V případě složitých funkcí není moc vhodný ani jazyk IL, jelikož se program stává dlouhým. Pro logické funkce jsou nejvíce vhodné jazyky LD, FBD, CFC a ST.

Třetí úloha již pracuje s grafickým panelem. Úkolem bylo napsat program a vytvořit simulaci semaforu dle zadaného časového diagramu s možností jeho odstavení. Student si vyzkouší sekvenční programování a tvorbu aplikace pro grafický panel.

Čtvrtá úloha se věnuje návrhu a realizaci sekvenčního logického obvodu. Úkolem je navrhnout logický obvod pro správnou funkci vsádkového reaktoru. Logický obvod je realizován programovacím jazykem FBD. Simulace výstupních a vnitřních proměnných byla vytvořena pomocí grafického panelu.

Všechny vytvořené laboratorní úlohy byly odzkoušeny na PLC Modicon M241 a HMI panelu Magelis HMIS5T.

LITERATURA

BAYER, J.; HANZÁLEK, Z.; ŠUSTA, R. 2008. *Logické řízení*. Vyd. 2. V Praze: České vysoké učení technické. ISBN 978-80-01-04106-2.

SCHNEIDER ELECTRIC. 2012. *Magelis HMI STU 655/855: User Manual*. [online]. [cit. 2016-05-05]. Dostupné z: http://www.farnell.com/datasheets/1975768.pdf

SCHNEIDER ELECTRIC. 2014a. *Modicon M241 Logic Controller: Hardware Guide*. [online]. [cit. 2016-04-25]. Dostupné z: http://www.eschneider.pl/download/04%20Automatyka%20przemyslowa/PLC%20sterowniki/M241_PLC_Modicon-Hardware_Guide_2014ENG.pdf

SCHNEIDER ELECTRIC. 2014b. *Modicon M241 Logic Controller: Programming Guide*. [online]. [cit. 2016-04-25]. Dostupné z: http://www.rakurs.su/wp-content/uploads/2015/01/Rukovodstvo-po-programmirovaniya-Modicon-M241-eng.pdf

ŠMEJKAL, L.; MARTINÁSKOVÁ M. 1999. *PLC a automatizace*. 1. vyd. Praha: BEN - technická literatura,. ISBN 80-860-5658-9.

VOJÁČEK, A. 2011. Programovací režimy pro PLC dle IEC 61131-3 (CoDeSys). *Automatizace.hw.cz: rady a poslední novinky z oboru*. [online]. [cit. 2016-04-25]. Dostupné z: http://automatizace.hw.cz/programovaci-rezimy-pro-plc-dle-iec-611313-codesys

ZEMANOVÁ, B. 2010. *PLC -programovatelné automaty: Automatizační technika*. [online]. [cit. 2016-04-25]. Dostupné z: http://slideplayer.cz/slide/2535859/#

PŘÍLOHY

A - CD

Příloha A

Příloha k bakalářské práci

Laboratorní úloha PLC Modicon

Tomáš Novák

CD

Obsah